台灣二ＯＯ五年國際科學展覽會

科　　別：數學

作品名稱：停車就是彈硬幣

得獎獎項：大會獎第一名

　　美國正選代表:參加美國第 56 屆國際科技展覽會

學　　校：臺北市立建國高級中學

作　　者：賴俊儒

評語與建議事項：

停車場問題與彈硬幣遊戲由表面上看來毫無關係。作者採用新穎的方法說明兩者的組合關係，並且對於更細節的計數問題做出創新的結果。

我叫賴俊儒，目前就讀建國中學數理資優班三年級。

我對於競賽和科展兩方面都有一些涉獵，在競賽方面曾選上 2004 年國際數學奧林匹亞候補國手，因此我對於組合問題有強烈的興趣，而且常嘗試用不同的角度來看問題。

去年我參加了 2004 年國際科展，並獲選赴加拿大代表，其中經過科教館教授們數個月的指導，稍微窺見了「科展」的門徑。比賽結束後在游森棚教授(教官)的指導下對停車場問題(parking function)做了一些研究，在這幾個月中得到了意料之外的好結果，在教授的鼓勵下報名了 2005 國際科展。

做研究的過程是辛苦但是有趣的，很高興有這個機會體會做研究的樂趣，最後僅在此感謝教官，感謝科教館的輔助計畫，也感謝繆友勇老師的鼓勵支持。

(作者爲右二者。)

停車就是彈硬幣

一、 前言

在這個科展中我們要研究兩個非常有趣的問題：

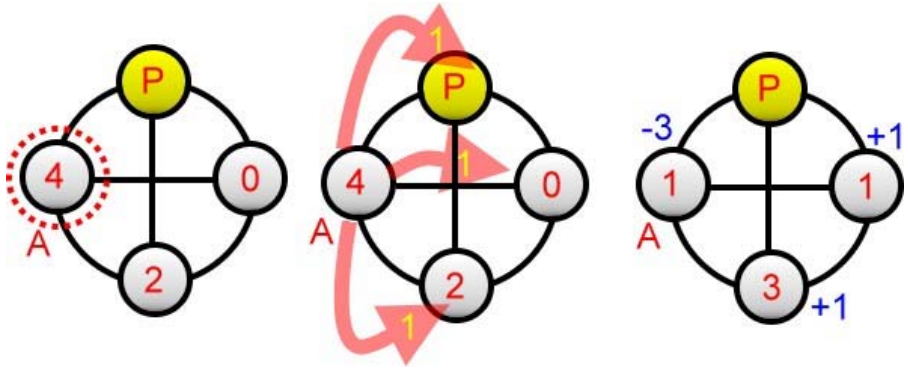
停車場問題 與 彈硬幣遊戲.

停車場問題是這樣的：在一條單行道上有n個車位，編號從 1 到n。現在有n個司機排成一排要進入停車。但是每個司機都有怪癖，各自有最想要停的位子。他們依序將車子開進單行道，如果想要停的位子是空的，當然停在這個位子。但是如果不巧那個位子已經被停了，不得已只好找下一個空位，姑且停之。但是如果往下找都沒有空位，由於是單行道，司機就只好開走不停了。

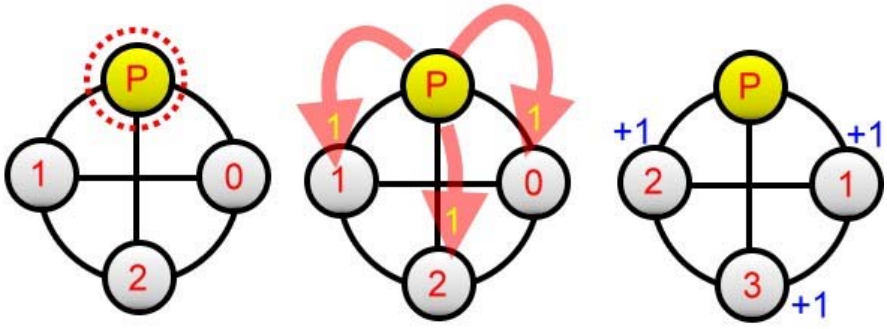
比如說，如果現在有五輛車，司機的喜好分別是(3,1,2,5,2)。則五輛車都可以順利停車。但是司機的喜好如果是(3,1,4,5,4)，有些車就無法停車了。

彈硬幣遊戲是這樣的：考慮圓內接正 n+1 邊形，任意兩點都連線。這正 n+1 邊形中有一個頂點 P 是特殊的，每個頂點上一開始都放有一些硬幣(各點硬幣數可以不同)。如果 P 以外的某個頂點上的硬幣數≥n 個，我們可以對這個頂點進行操作：一次操作是指將這個頂點上的硬幣各分一個給每個其他頂點。點 P 只在其他點都無法操作時操作。我們不理會頂點 P 上的錢數，因此這個遊戲可以無限地玩下去。

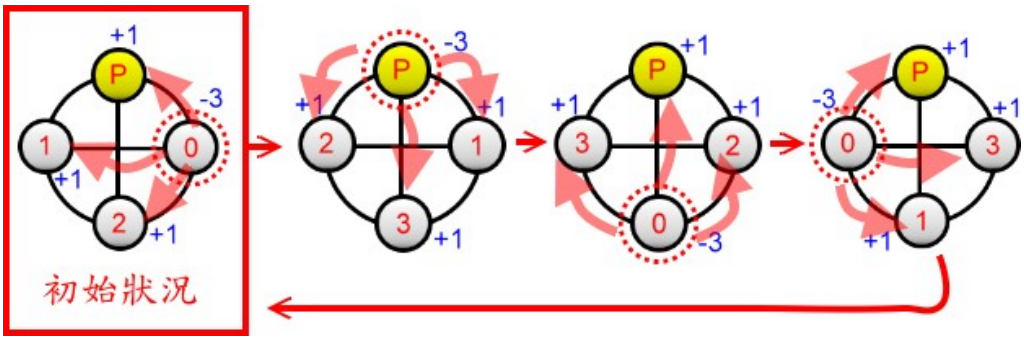
比如說，下面的左圖中對頂點 A 作操作，結果得到右邊的圖。



比如說，下面的左圖原本已經塞住(玩不下去)了，因此可對頂點 P 作操作，結果得到右邊的圖。



彈硬幣遊戲有時會操作回到原來的狀態。比如以下這個開始就塞住的例子，經過幾次操作後會回到原來狀態。我們稱這種「開始就塞住但經過幾次操作後會變回來的」叫做循環狀態。



停車場問題 與 **彈硬幣遊戲** 是兩個截然不同的遊戲，看起來一點關係也沒有。本件科展作品主要就在研究停車場問題和彈硬幣遊戲之間有趣的關連。我們的研究主要包含了以下幾個驚奇的結果:

1. 令 P_n 為能順利停車的總方法數；$P_{n,k}$ 為能順利停車，且第 1 個司機喜歡第 k 號停車位的組數。我們求出了 $P_{n,k}$，並且發現幾個奇妙的關係。

2. P_n 恰好就是彈硬幣遊戲中，總點數為n+1 的循環狀態總數。**因此停車場問題 與 彈硬幣遊戲根本是同一件事!**

3. 我們可以把上面的概念推廣到其他的情況。在”圓內接正 n 邊形只有相鄰兩點連線，特殊點 P 位於圓心，與每點都連線”的情況下玩彈硬幣遊戲，則結果居然和 盧卡斯 數列與 費氏數列有關!

二、 停車場問題

「在一條單行道上有n個車位，編號從 1 到n。現在有n個司機排成一排要進入停車。但是每個司機都有怪癖，各自有最想要停的位子。他們依序將車子開進單行道，如果想要停的位子是空的，當然停在這個位子。但是如果不巧那個位子已經被停了，不得已只好找下一個空位，姑且停之。但是如果往下找都沒有空位，由於是單行道，司機就只好開走不停了。」

而現在我們給出兩個基本定義：

1. P_n 代表 n 個司機都停好車的總方法數
2. $P_{n,k}$ 為 n 個司機都停好車且第 1 個司機喜歡第 k 個車位的方法數

而已知是 $P_n=(n+1)^{n-1}$

例子：n=3 的所有停車函數

例子	都能停車?	例子	都能停車?	例子	都能停車?
111	○	211	○	311	○
112	○	212	○	312	○
113	○	213	○	313	×
121	○	221	○	321	○
122	○	222	×	322	×
123	○	223	×	323	×
131	○	231	○	331	×
132	○	232	×	332	×
133	×	233	×	333	×
$P_{n,1}$	8	$P_{n,2}$	5	$P_{n,3}$	3

接著我們觀察 n 為定值時，$P_{n,k}$ 的數量

n	$P_{n,1}$	$P_{n,2}$	$P_{n,3}$	$P_{n,4}$	$P_{n,5}$	$P_{n,6}$
1	1					
2	2	1				
3	8	5	3			
4	50	34	25	16		
5	432	307	243	189	125	
6	4802	3506	2881	2401	1921	1296

乍看之下並沒有什麼規律

但是若我們將每一列的最右邊添一項 0，再將 $P_{n,k}$ 和 $P_{n,k+1}$ 兩兩相減，即會得出下表(紅字即為相減值)：

n=1						1	**1**	0				
n=2					2	**1**	1	**1**	0			
n=3				8	**3**	5	**2**	3	**3**	0		
n=4			50	**16**	34	**9**	25	**9**	16	**16**	0	
n=5		432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

這個差值表有很多奇妙的性質。

1. 首先，他是左右對稱的!但是這個表原本的意義($P_{n,k}$ 和 $P_{n,k+1}$ 相減)卻跟對稱一點關係都沒有!舉個例子：我們會確定「第 1 個司機停在第 1 個位子」的總方法數，減去「第 1 個司機停在第 2 個位子」的總方法數，會等於「第 1 個司機停在最後 1 個位子」的總方法數!

2. 再來我們觀察第 1 個左斜行，最前面的幾項是(1, 1, 3, 16, 125, 1296)，而這個東西根本就是(1^{-1}, 2^0, 3^1, 4^2, 5^3, 6^4)也就是 n^{n-2}!還記得 $P_n=(n+1)^{n-1}$ 嗎？所以我們又可以確定：在 n 個司機要停車的狀況中，「第 1 個司機停在第 1 個位子」的總方法數，減去「第 1 個司機停在第 2 個位子」的總方法數，會等於 n-1 個司機要停車的的總方法數。

3. 如果我們只看 n 是偶數(令 n=2k)的情形：

n=2					2	1	1	1	0			
n=4			50	16	34	9	25	9	16	16	0	
n=6	4802	1296	3506	625	2881	480	2401	480	1921	625	1296	1296	0

觀察黃色的格子，會發現左邊格子的值恰好是右邊格子的 2 倍。這代表說：「第 1 個司機停在第 1 個位子」的總方法數，恰為「第 1 個司機停在第 k+1 個位子」的總方法數的 2 倍。又是一個不直觀的性質。

而到目前為止**沒有人用組合的方法去解釋這些性質**，接下來我們要引進一套對應法，把所有不同的停車的情況對應成一個標號樹(即是在一個 n+1 個點的樹上面標示不重複的 0~n)，再用標號樹來解釋上面三個性質。

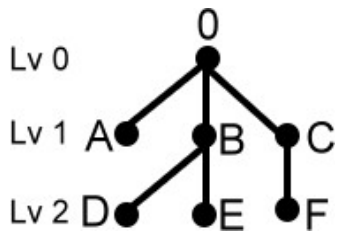
對應法(標號樹=>停車函數)

首先，對於每一個標號樹我們做下列動作：

1. 將標示 0 的點置於最上層(我們將之命名成第 0 層)

2. 將所有跟 0 點距離是 k 的點放置在第 k，並保持原本的連線。

3. 同一個點的子代的排列是由左到右從小到大排列
(如圖，其中 A<B<C，D<E，而 F 不必然大於 E)



接著，對於每個點都給 2 個賦值 F 和 W，以$\binom{F}{W}$表示在點右方。我們用 F(X)和 W(X)來代表 X 這個點的 F 和 W。可以直觀的想像 W 代表 Want，為每個司機喜歡停的車位。因為不是說司機想停哪裡就可以停哪裡，所以我們建立了 F(就是 Fact)，用來紀錄位置。而 F 和 W 的產生方式我們在下面詳述。

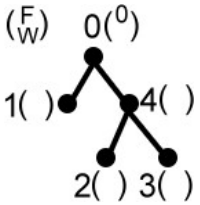
產生$\binom{F}{W}$的程序如下：

1. 將點 0 的賦值定義成$\binom{0}{\cdot}$，即是說 W(0)無意義。

2. 每次只處理一層，處理時由第一層依序處理到最下層。在處理同一層時，從每一層的最左邊的點依序處理到最右邊的點。

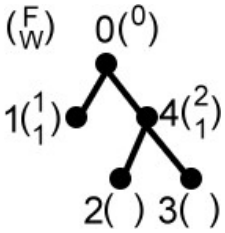
3. 此時對於第 k 個正在處理的點 X 來說，我們定義 F(X)=k。以$\hat{X}$代表 X 上層那個點，則我們定義 W(X)=F($\hat{X}$)+1。
舉一個例子:

第一步



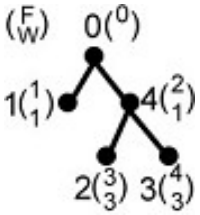
首先，給定 F(0)=0

第二步



先處理第 1 層。
在第 1 層中先處理左邊那點，也就是 1
F(1)=1，W(1)=F(0)+1=1。
再來處理 4
F(4)=2，W(4)=F(0)+1=1。

第三步



再來處理第 2 層。
在第 2 層中先處理左邊那點，也就是 2
F(2)=3，W(2)=F(4)+1=3。
再來處理 3
F(3)=4，W(3)=F(4)+1=3。

而每個點的 W 值寫下來就是一組停車函數。例如說上面那個標號樹就對應到(1,3,3,1)。

對應法(停車函數=>標號樹)

而給定一個停車函數($a_1, a_2, \cdots, a_n$)，可以由以下做法得出對應的標號樹：

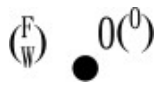
1. 將點 0 放置在第 0 層，賦值為$\binom{0}{.}$

2. 每次處理時只處理 1 層，層數的挑選由上到下，同一層的處理由左到右。

3. 每次處理時觀察已處理完的層數中最下面那層，設那層的點的 F 值為 X_1~X_k。此時我們將 a_1~a_n 中所有值為 X_1+1~X_k+1 的數挑出來，把其與相對應的點連線。使其滿足「對任何一個點 X 來說都有 W(X)=F($\hat{X}$)+1。」

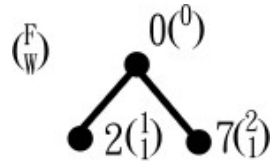
舉個例子，給定停車函數(6,1,4,2,3,3,1,4)

第一步



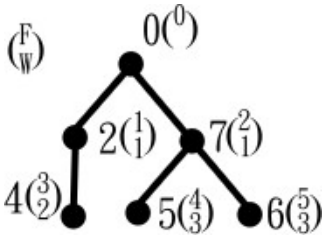
首先將點 0 放在第 0 層

第二步



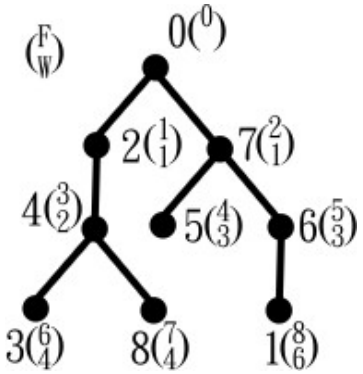
接著處理第 1 層，已知第 0 層的 F 值只有 0 一種，故挑出所有 W=1 的點(即 2,7)接上點 0。

第三步



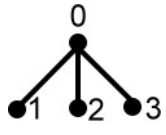
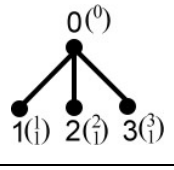
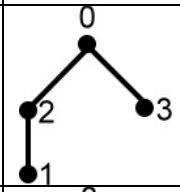
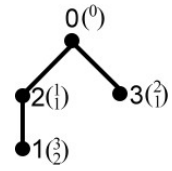
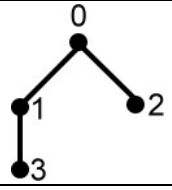
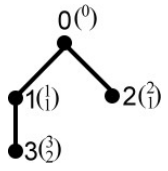
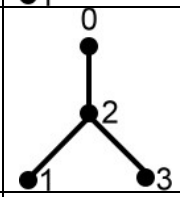
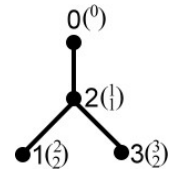
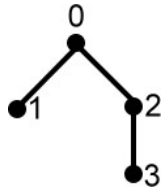
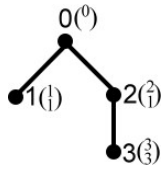
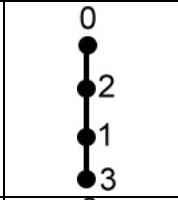
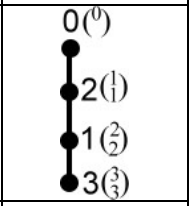
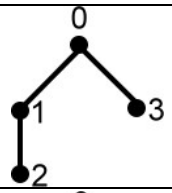
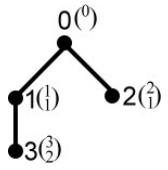
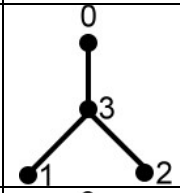
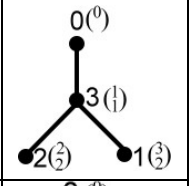
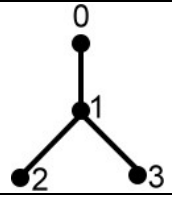
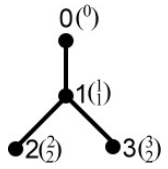
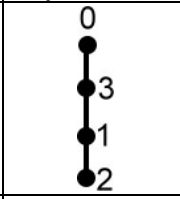
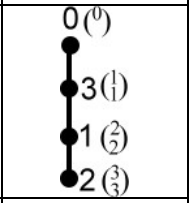
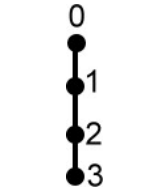
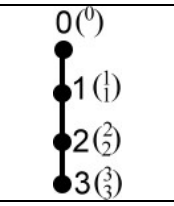
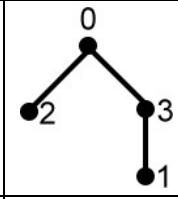
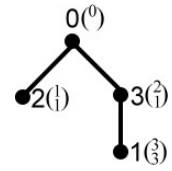
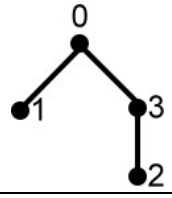
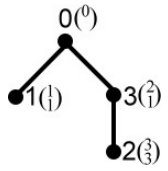
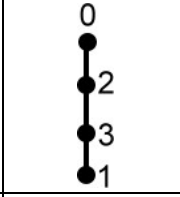
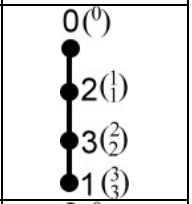
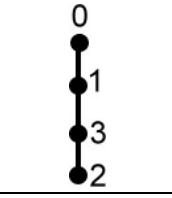
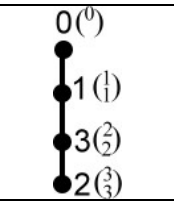
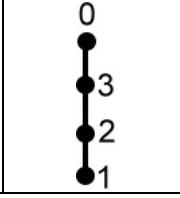
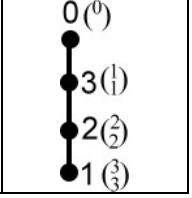
接著處理第 2 層，已知第 1 層的 F 值為 1~2，故挑出所有 W=2~3 的點(即 4,5,6)分別接上點 2、7。

第四步



接著處理第 3 層，已知第 2 層的 F 值為 3~5，故挑出所有 W=4~6 的點(即 1,3,8)分別接上點 4、6。

因此我們可以建立一個停車函數對到標號樹的對應表：如下(n=3)

停車函數	圖		停車函數	圖	
111			211		
112			212		
113			213		
121			221		
122			231		
123			311		
131			312		
132			321		

用這個對應法我們就可以證明本段最初所提到的定理，但是由於該證明篇幅過長，為保持連貫性，我們將其放到最後一段討論。接下來的段落我們先假設已經證明了下面這幾個定理。

定理 4.1 $P_{n,1} - P_{n,2} = n^{n-2}$

n=1					1	**1**	0					
n=2				2	**1**	1	**1**	0				
n=3			8	**3**	5	**2**	3	**3**	0			
n=4		50	**16**	34	**9**	25	**9**	16	**16**	0		
n=5	432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0	
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

即是說加底的部分是 n-1 個司機要停車的的總方法數。

定理 4.2 $P_{n,2} - P_{n,3} = (n-1)^{n-2}$

n= 1						1	**1**	0				
n=2					2	**1**	1	**1**	0			
n=3				8	**3**	5	**2**	3	**3**	0		
n=4			50	**16**	34	**9**	25	**9**	16	**16**	0	
n=5		432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

即是說加底部分是 n-1 乘上 n-2 個司機要停車的的總方法數。

定理 4.3 $P_{n,k} - P_{n,k+1} = P_{n,n-k+1} - P_{n,n-k+2}$

這個等式說明了為什麼差值表是對稱的。

定理 4.4 當 n 為偶數時(令 n=2k)，$P_{n,1} = 2 \times P_{n,k+1}$

n=2					2	**1**	1	**1**	0			
n=4			50	**16**	34	**9**	25	**9**	16	**16**	0	
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

三、 彈硬幣遊戲

「考慮圓內接正 n+1 邊形，任意兩點都連線。這正 n+1 邊形中有一個頂點 P 是特殊的，每個頂點上一開始都放有一些硬幣(各點硬幣數可以不同)。如果 P 以外的某個頂點上的硬幣數≥n 個，我們可以對這個頂點進行操作：一次操作是指將這個頂點上的硬幣各分一個給每個其他頂點。點 P 只在其他點都無法操作時操作。我們不理會頂點 P 上的錢數，因此這個遊戲可以無限地玩下去。」

為了方便起見，我們給點P一個很貼切的名字，叫做政府。可以想像政府擁有很多錢，所以一定有足夠多的錢發給每個人。再來就是說政府很沒有良心，平常政府是不發錢的。在每一次彈硬幣操作(金錢交易)中政府都會獲得一塊錢(抽稅)，所以政府的錢會愈來愈多，外面的錢會愈來愈少，直到某一天大家都沒有錢了，政府才勉為其難發錢給所有人。

承前言，我們把『開始就塞住但經過幾次操作後會變回來的』叫做循環狀態。並且我們發現總點數為n+1 的循環狀態總數恰好就是P_n。這意味著這兩個乍看之下一點關係都沒有的問題其實是同一件事情！而我們可以在這兩者之間找到一組對應，並且提出下面幾個問題：

1. $P_{n,k}$透過對應法對應到彈硬幣遊戲後會變成什麼？
2. 更甚之，在第 2 段中提到的幾個奇妙的性質透過對應法對應到彈硬幣遊戲後會變成什麼？
3. 在彈硬幣遊戲中塞住的，和經過幾次操作會變回來的狀態，如果透過對應法對到停車問題，會變成什麼？

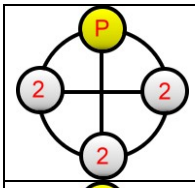
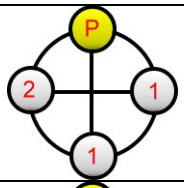
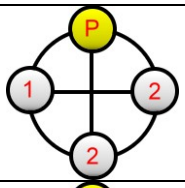
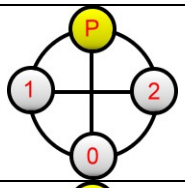
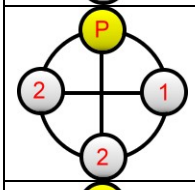
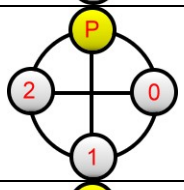
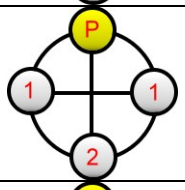
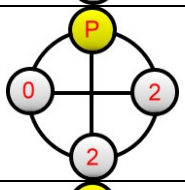
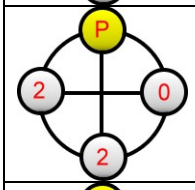
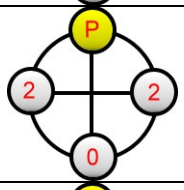
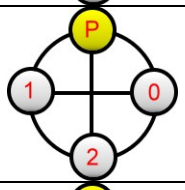
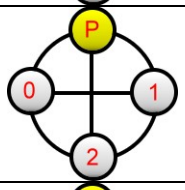
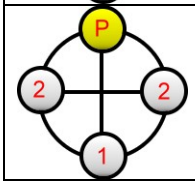
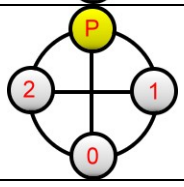
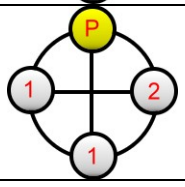
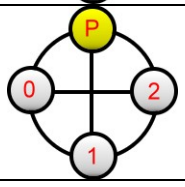
四、 彈硬幣遊戲和停車問題間的關係

若我們從政府開始，依照逆時針方向把循環狀態中，所有非政府的點的錢數紀錄下來，會得到一串 n 位的數列。因為我們知道循環狀態一開始是塞住的，所以這 n 個數都要小於 n。

而我們隨即發現，如果我們把 n 和這數列的每一項的差寫下來，所得到的數列會是一個停車問題的解。例如說錢數若是

(3,1,3,2)，則因為我們知道 n=4，就寫下(1,3,1,2)，而我們會發現(1,3,1,2)是第 1 個司機喜歡第 1 個車位，第 2 個司機喜歡第 3 個車位，第 3 個司機喜歡第 1 個車位，第 4 個司機喜歡第 2 個車位的情況，恰好是一組全部都可以停好車的解。

例子：n=4 時的循環狀態和停車函數

	111		122		211		231
	112		123		212		311
	113		131		213		312
	121		132		221		321

所以說一個在彈硬幣遊戲中，一開始塞住，但是經過幾次操作就會回來的錢幣的分布情形；換一種敘述就變成了數個正要排隊停車的司機他們腦袋裡想要停的車位編號。這又是一個非常神奇的對應!

定理 1　　循環狀態和停車函數之間存在一組對應

對應法：

將圖上的 n 個點標上 1 到 n，可以寫下$(m_1, m_2, \cdots, m_n)$，其中m_k為第 k 號點上面的錢數。則我們把他們和 n 相減，得到$(n - m_1, n - m_2, \cdots, n - m_n)$，此數列即為停車函數的一個解；另一方面，一組停車函數$(a_1, a_2, \cdots, a_n)$也可以經由和 n 相減後得到$(n - a_1, n - a_2, \cdots, n - a_n)$，必為一組循環狀態的錢數。

而我們先證明($n - a_1, n - a_2, \cdots, n - a_n$)會是一組循環狀態，即證明他對應到的圖會塞住且經過若干次操作後會回來。

證明會塞住：

該點可以操作 <=> 該點錢數≥該點連線數= n。

而由停車函數的定義，我們對於每個 i 都有$a_i \geq 1$，故可得到 n >n-1≥$n - a_i$ (k = 1~n)，所以此圖中每一點都不能操作=>此圖塞住。

證明經過若干次操作會回來：

引用[2]裡面的定理「*不管先處理哪一點都不影響結果*」，所以我們以下列方式選定操作的點：每次操作的點為所有錢數大於等於 n 的點中編號最小的點。

因為此圖塞住，所以第一次操作的點必為政府，第一次動作完了以後其他所有點錢數都+1，則我們可以得到 n≥$m'_k \geq n - k + 1$ (k = 1~n)，故第 2 次操作的點必為m'_1。

接下來要用歸納法證明：當第 1 次~第 i 次(i = 2~n)操作的點為政府和$m'_1 \sim m'_{i-1}$時，第 i+1 次操作的點必為m'_i。

因為$m'_1 \sim m'_{i-1}$都被操作過，我們知道當一個點操作時所有其他的點都會獲得 1 塊錢；自己操作時會減少 n 塊錢(給其他 n－1 個點和政府)，所以我們有$m'_k \leq$ (n－1)+i－n = i－1 ≤ n－1 (k = 1 ~ i－1) ，所以此時$m'_1 \sim m'_{i-1}$皆不可以操作。

又$m'_i \geq$ (n－i+1) + (i－1) = n =>m'_i可以操作，故m'_i必為所有錢數為不小於 n 的點當中編號最小的點=>第 i+1 次操作的點必為m'_i。歸納法成立。

又我們得到：n+1 次操作後政府和$m'_1 \sim m'_n$都過恰操作過一次，故每個點的值不變(被給錢 n 次，錢+ 1×n；操作 1 次，錢－n×1)。=> 此圖為經過 n+1 操作後會回到初始狀態。

而根據上面我們證出**循環狀態和停車函數之間存在一組對應**。

而我們透過這組對應法就會發現：

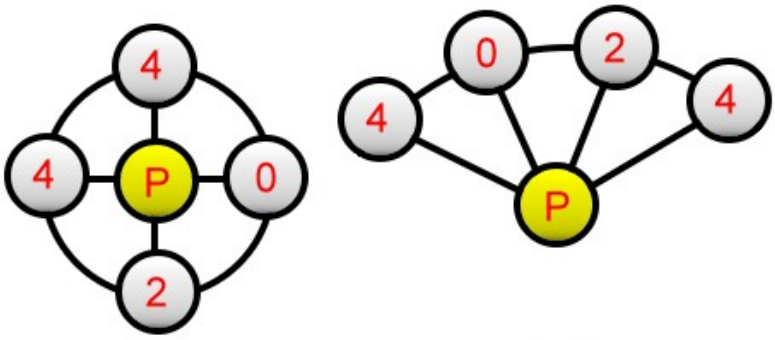
1. 對於任何一個循環狀態來說，每個點上面的錢數和 n 的差，恰好分別就是停車問題中 n 個司機分別想要停的車位編號。

2. 從另外一個角度敘述，n 個司機正在單行道上排隊停車，如果他們可以全部停到車，則把他們喜歡的車位編號跟 n 相減，就會得到一個彈硬幣遊戲的循環狀態!

五、 各種特殊圖形上的彈硬幣遊戲

而在其他特殊的圖形上，彈硬幣遊戲也有許多有趣的性質。為求把彈硬幣遊戲一般化，我們重新定義一次操作：「如果一個點上的錢數大於等於該點的連線數，則我們把他上面的錢各分一塊給所有和他相連的點。」

在本節中我們主要討論車輪和扇子上面的彈硬幣遊戲，車輪是說一個圓周上面有 n 個點，中心有個軸點與圓周上所有的點連線；扇子則是取車輪的一段弧(上面還是放 n 個點)及軸心所構成。(如下圖)



以車輪和扇子為討論主題的原因是因為他們先天上就有一個地位特殊的點，即是軸心。軸心和所有的點都相連，所以我們就取軸心當作政府來討論。而我們又發現了車輪跟盧卡斯數列有關；扇子跟費氏數列有關。

六、 車輪上的彈硬幣遊戲

下面先給三個基本定義：

1. W_n 代表點數為 n+1 的車輪的循環狀態的總數。
2. $W_{n,k}$ 代表循環狀態中，若我們將上面的錢數依序寫下來，第一個數恰好為 k 的總數。
3. 如果一個點上面的錢數為 0，則我們叫他作斷點。所以斷點可以將車輪分成若干段。(斷點為 0 或 1 時，將車輪視為一段)

定理 2　　車輪的循環狀態的充要條件為：

1. 所有點的錢數≤2
2. 任一段必存在一個錢數=2 的點

檢查一下 n=3 時的例子：

222	211	210	102
221	121	120	022
212	112	202	021
122	220	201	012

都會符合

證明充分性

因為所有點的錢數≤2，車輪上所有點的連線數都是 3，所以他一定會塞住，而我們接下來要證明他經過數次操作一定會回來。為了證明這個，我們根據斷點數的不同來分別討論。

1. 斷點數 =0 時

此時每個點的錢數皆在 1 和 2 之間(否則就會有斷點)，且存在至少一個錢數為 2 的點。而在政府做完第一次操作後，此時每個點的錢數皆在 2 和 3 之間，且存在至少一個錢數為 3 的點。則第二次一定是操作之中任何一個錢數為 3 的點。

對於任何一個未操作過的點來說，因為本身錢數≥2，所以當他的鄰點其中之一操作過後，該點的錢數 3，必可作一次操作

已知「*不管先處理哪一點都不影響結果*」[2]，所以我們必可以從第二次操作選定的那個點開始，依照順時針選擇下一點操作，使得每個圓圈上的點都恰操作過一次。此時每個點的錢數必跟最初一樣(操作過一次錢數-3，兩個臨點和政府都恰給過自己 1 塊錢)。

2. 斷點數 =1 時

同理，第一次操作的點必為政府。第二次我們任選一個錢數為 3 的點 X 來操作，此時我們命名 X 經由順時針方向到斷點(即錢數為 0 的那一點)途中經過的點為 $A_1, \cdots, A_i$；由 X 開始逆時針途中經過的點為 $B_1, \cdots, B_j$。

因為「*不管先處理哪一點都不影響結果*」，我們令第 3~第 i+2 次操作的點為 $A_1, \cdots, A_i$，再之後的 j 次為 $B_1, \cdots, B_j$。此時斷點錢數為 3（因為政府操作過一次，且斷點左右兩邊也都操作過一次），則取斷點為最後一次操作的點。

此時每個車輪上的點都恰操作過一次，故每個點的錢數必跟最初一樣(操作過一次錢數-3，兩個臨點和政府都恰給過自己 1 塊錢)。

3. 斷點數 ≥2 時

由 2.(**斷點數 = 1**) 的方法，當斷點數為 k 時(k≥2)，前 n+1 - k 次操作必可以取政府和車輪上所有不是斷點的點，因為充要條件第 2 點可以得知斷點間兩兩不會相鄰，所以每個斷點左右都一定作過一次操作。此時每的斷點的錢數必為 3，最後 k 次令所有斷點操作，即會符合條件：此時每個車輪上的點都恰操作過一次，每個點的錢數數必跟最初一樣。

證明必要性

證明必要性則分別對兩個條件用反證法來證

1. 反設存在一點錢數≥3

則該點可以操作，此圖不會塞住，故不會是循環狀態，矛盾。

2. 反設存在一個分段之中沒有錢數為 2 的點

則我們持續操作，直到此圖下一次塞住的時候停下來看。因為該分段(包括兩端的斷點)沒有錢數是 2 的點，故該分段中的每個點的錢數會從起始狀態的 011…110 變成塞住時的 222…222，且與該分段相鄰的兩個點錢數減 1，其他的點錢數不變。如果和這個分段相鄰的點錢數其中一個最初是 2，則變成 1 之後會把這兩個分段連在一起；如果最初都是 1 則會產生一個新的分段使得此分段之中有至少一個錢數為 2 的點。所以新的塞住的狀態必符合給定的兩項條件，為一循環狀態。

我們知道新的循環狀態車輪上的錢數比最初情形多，而容易知道在循環的過程中車輪上的錢數會先加 n 然後再慢慢減下來，所以永遠不會回到最初情形。矛盾，故得證。

七、 車輪上的循環狀態

我們先看看 W_n 大概長什麼樣子。例子：n=4 的車輪的 45 種情形

2222	2211	1122	2220	1210	1202	2021	1012	0211
2221	2121	2111	2210	1120	2101	2012	0222	0121
2212	2112	1211	2120	2202	1201	1022	0221	0112
2122	1221	1121	1220	2201	1102	2011	0212	2020
1222	1212	1112	2110	2102	2022	1021	0122	0202

只看這樣看不出所以然，接著我們嘗試以斷點的個數來觀察 n 在變化時，W_n 的個數。(下表是以程式跑出的結果)

斷點數	n=1	2	3	4	5	6	7	8	9
0	1	3	7	15	31	63	127	255	511
1		2	9	28	75	186	441	1016	2295
2				2	15	69	252	804	2349
3						2	21	128	594
4								2	27

總數	1	5	16	45	121	320	841	2205	5776
=	1^2	3^2-4	4^2	7^2-4	11^2	18^2-4	29^2	47^2-4	76^2

而我們知道1,3,4,7,11,18,29,…是著名的盧卡斯數列(盧卡斯數列即為，初始項L_1=1, L_2=3, 符合$L_n=L_{n-1}+L_{n-2}$的遞迴數列。)

我們再換個方法，觀察$W_{n,k}$的個數

K	n=1	2	3	4	5	6	7	8	9
0	0	1	3	8	21	55	144	377	987
1	0	1	5	16	45	121	320	841	2205
2	1	3	8	21	55	144	377	987	2584
Total	1	5	16	45	121	320	841	2205	5776

表格中第一列和第三列的1,3,8,21,55,144…，恰好是費氏數列(1,1,2,3,5,8,13,21,34,55,89,144…)的偶數項。(費氏數列即為，初始項F_1=1, F_2=1, 符合$F_n=F_{n-1}+F_{n-2}$的遞迴數列。)

我們又有以下的驚奇發現：

1. 當n為奇數時$W_n = L_n^2$；當n為偶數時$W_n=L_n^2-4$
2. $W_n = W_{n+1,1}$
3. $W_{n,2} = W_{n+1,0} = F_{2n}$(費氏數列)

接著我們要證明上面的發現

引理3.1 $W_{n,01}=W_{n-1,01}+W_{n-1,02}$

根據車輪的充要條件，$W_{n,01}$中不可能以010開頭，所以我們有$W_{n,01}=W_{n,011}+W_{n,012}$。而我們又可以藉由把第二位去掉(反過來是將第一和第二位之間插入一個1)，將$W_{n,011}$和$W_{n-1,01}$作一個對應，且經過如此的操作後不會改變數列是不是循環狀態，所以我們有$W_{n-1,01} = W_{n,011}$。同理有$W_{n-1,02} = W_{n,012}$。

=> $W_{n,01}=W_{n,011}+W_{n,012} =W_{n-1,01}+W_{n-1,02}$，得證。

引理3.2 若n=1~ k-1時有$W_{n,0}= F_{2n-2}$**，則** $W_{n,02}=W_{n,01}+F_{2n-5}$**成立**

若$W_{n,02}$的第一個分段裡有兩個以上的 2，則我們可以把他跟$W_{n,01}$作一個對應，方法是將第二位的 2 改成 1(反過來是將 1 改成 2)，因為該分段裡有兩個以上的 2，所以將 2 改成 1 以後該分段裡還是存在至少一個 2，故操作不會改變該數列是不是循環狀態。

若$W_{n,02}$的第一個分段裡僅有一個 2，則我們依照第一個分段的長度分類，因此我們有$W_{n,02} = W_{n,01}+W_{n,020}+W_{n,0210} +W_{n,02110}+\cdots+W_{n,021\ldots10}+(021\cdots1)$

對於任一個$W_{n,021\ldots10}$(共 k 個 1)來說，我們有$W_{n,021\ldots10}=W_{n-k-2,0}$，則我們有：$W_{n,02}=W_{n,01}+W_{n-2,0}+W_{n-3,0}+\cdots+W_{2,0}+1$

$=W_{n,01}+F_{2n-6}+F_{2n-8}+\cdots+ F_4+ F_2+1$

$=W_{n,01}+F_{2n-5}$，得證。

定理 3.1 $W_{n,0}= F_{2n-2}$

接下來用歸納法證明$W_{n,0} = F_{2n-2}$，我們知道 n=2 時有$W_{n,01}=F_{2n-4}$和$W_{n,02}=F_{2n-3}$，再來證若 n=2 ~ k-1 時都有$W_{n,01}=F_{2n-4}$和$W_{n,02}=F_{2n-3}$，則我們有$W_{k,01}=F_{2k-4}$和$W_{k,02}=F_{2k-3}$。

$W_{k,01}=W_{k-1,01}+W_{k-1,02}$ (引理 3.1)$=F_{2k-6}+F_{2k-5}= F_{2k-4}$

$W_{k,02}=W_{k,01}+F_{2k-5}$(引理 3.2) $=F_{2k-4}+ F_{2k-5}= F_{2k-3}$

故由歸納法得證$W_{n,0}=W_{n,01}+W_{n,02}=F_{2n-3}+F_{2n-4}=F_{2n-2}$

引理 3.3 $W_{n,20}=W_{n,02}$

我們在$W_{n,20}$中每個數列第二位起頭向左數皆會產生一個 02 開頭的新數列，且如果原本的數列是循環狀態，產生的新數列也是循環狀態，反之亦然；在$W_{n,02}$中每個數列第二位起頭向左數皆會產生一個 20 開頭的新數列，且如果原本的數列是循環狀態，產生的新數列也是循環狀態，反之亦然。故$W_{n,20}$和$W_{n,02}$有一個對應=>$W_{n,20}=W_{n,02}$。

引理 3.4 $W_{n,21}=W_{n,22}=W_{n-1,2}$

我們將$W_{n,21}$和$W_{n,22}$中每個數列第一位去掉之後即產生一個在$W_{n-1,2}$裡的數列，且如果原本的數列是循環狀態，產生的新數列也是循環狀態，反之亦然；將$W_{n-1,2}$裡每個數列的第一和第二位間插入 1 或 2 會產生在$W_{n,21}$或$W_{n,22}$裡的數列，如果原本的數列是循環狀態，產生的新數列也是循環狀態，反之亦然。故他們三者之間兩兩有對應，故$W_{n,21}=W_{n,22}=W_{n-1,2}$ 。

定理 3.2 $W_{n,2}= F_{2n}$

我們有$W_{n,2}=W_{n,20}+W_{n,21}+W_{n,22}=W_{n,02}$(引理 3.3)+2 $W_{n-1,2}$(引理 3.4)=F_{2n-3} (定理 3.1)+ 2 $W_{n-1,2}$

由歸納法，我們知道 n=1 時有$W_{n,2}$= 2n，再來要證 n = k-1 成立時 n = k 也成立。則我們有$W_{k,2}=F_{2k-3}+ 2W_{k-1,2} =F_{2k-3}+ 2 F_{2k-2} = F_{2k}$，得證。

定理 3.3 $W_{n,1}=W_{n-1}$

我們將$W_{n,1}$中每個數列的第一位去掉可以對應到W_{n-1}且不會改變他是不是個循環狀態；將W_{n-1}中每個數列的第一位前加上 1 可以對應到$W_{n,1}$且不會改變他是不是個循環狀態。故他們兩者間有一個對應，故$W_{n,1}=W_{n-1}$ 。

定理 3.4 $W_n=L_1+L_3+ \cdots +L_{2n-1}$，即為：

當 n 為奇數時$W_n = L_n^2$；當 n 為偶數時$W_n=L_n^2-4$

由歸納法，n=1 時成立，再來要證 n = k-1 成立時 n = k 也會成立。$W_k=W_{k,0}+W_{k,1}+W_{k,2}= F_{2k-2}+W_{k-1}+F_{2k}=L_{2k-1}+(L_1+L_3+ \cdots +L_{2k-3})=L_1+L_3+ \cdots +L_{2k-1}$，得證。

因此我們可以整理出下表：

項	$W_{n,01}$	$W_{n,02}$	$W_{n,10}$	$W_{n,11}$	$W_{n,12}$	$W_{n,20}$	$W_{n,21}$	$W_{n,22}$
值	F_{2n-4}	F_{2n-3}	F_{2n-4}	W_{n-2}	F_{2n-2}	F_{2n-3}	F_{2n-2}	F_{2n-2}

八、 扇子上的彈硬幣遊戲及循環狀態

在所有的循環狀態中我們從扇子的某一端開始，沿著扇緣將點上面的錢數寫下來，會得到一串 n 項的數列。為了避免和費氏數列混淆，我們將這個寫下來的數列定義為 N_n，而 $N_{n,k}$ 為之中第一個數為 k 的總數。再觀察 $N_{n,k}$ 的個數會得到

K	n=1	2	3	4	5	6	7	8	9
0	0	1	3	8	21	55	144	377	987
1	1	2	5	13	34	89	233	610	1597
Total	1	3	8	21	55	144	377	987	2584

發現扇子的循環狀態，及他的分項討論和費氏數列有密切的關係。我們可以看出

1. $N_n = F_{2n} = N_{n+1,0}$
2. $N_{n,1} = F_{2n-1}$

然後我們發現可以用一種說法把扇子和車輪的循環狀態連在一起，即是有一個對應：如果我們把扇子的兩個端點都加 1，接下來再添上一個虛擬的斷點(錢數為 0)把扇子的兩個端點接在一起，則會產生一個車輪的循環狀態。而如此一來我們就可以把扇子的充要條件寫出來：

充要條件：

若我們將錢數為 0 的點是為斷點，則扇子的充要條件則是

1. 任何一點的錢數都小於他的連線數
2. 對於每個分段來說，至少存在一個點，該點獲得一塊錢後就可以作一次操作。

證明充分性：

如本段開始所提到的，我們建立一個虛擬的點 S，其上的錢數為 0，使其和扇子兩端點相連，再把扇子兩端點錢數加 1，這樣會得到一個車輪，而這時充要條件所提到的兩個條件會變成：

1. 所有點的錢數≤2
2. 任一段必存在一個錢數=2 的點

所以新圖必定是一個車輪的循環狀態，又因為[2]裡面提到「無論先操作哪一點都不會影響結果」，所以我們把點 S 的操作留到最後面，而我們把那個車輪一次循環所作過的 n 次操作紀錄下來。容易發現如果我們把前 n-1 次操作套用到扇子上也會成立。

證明必要性：

跟第六段一樣我們使用反證法：第一個條件就是塞住的條件，一定要成立。否則會矛盾。對於第 2 個條件來說，我們一樣建立一個虛擬的點 S，其上的錢數為 0，使其和扇子兩端點相連，再把扇子兩端點錢數加 1，這樣會得到一個車輪。

根據第六段的證明，同理我們會得到一個總錢數比初始狀態要來的多的循環狀態的車輪，同理初使狀態對應到的車輪經過若干次操作以後一定不會回到原本狀態，所以如果不符合第二個條件，則該扇子經過若干次操作後還是不會回到原本的狀態。得證。

九、 扇子循環狀態的計數

觀察簡單的例子：(n=4)

0210	0111	1110	1101 1011
0120	0211	1120	1021 1201
0220	0121	1210	1211 1121
	0221	1220	1111 1221
	0201	1020	
端點 0 0	端點 0 1	端點 1 0	端點 1 1

會發現扇子的循環狀態的兩個端點只有可能是(00, 01, 10, 11)四種，而根據第八段，我們可以將他對應到某一段是(101, 102, 201, 202)的循環狀態的車輪，即是，以(101, 102, 201, 202)開頭的車輪總數。又我們知道不會有 00 相連的情況，所以我們有：

1. $N_{n,0} = N_{n,0\ldots1} + N_{n,0\ldots0} = W_{n+1,102} + W_{n+1,101} = W_{n+1,10} = F_{2n-2}$
2. $N_{n,1} = N_{n,1\ldots0} + N_{n,1\ldots1} = W_{n+1,201} + W_{n+1,202} = W_{n+1,20} = F_{2n-1}$

將這兩個式子相加即得：

$N_n = N_{n,0} + N_{n,1} = F_{2n-2} + F_{2n-1} = F_{2n}$

得證。

因此我們把剛剛的表推廣成

端點	n=1	2	3	4	5	6	7	8	9
00	0	0	1	3	8	21	55	144	377
01	0	1	2	5	13	34	89	233	610
10	0	1	2	5	13	34	89	233	610
11	1	1	3	8	21	55	144	377	987
Total	1	3	8	21	55	144	377	987	2584

十、 停車場函數的計數

在本段中我們將證明第二段所提到的性質。先複習一下第二段所提到的對應法：

對應法(停車函數=>標號樹)

給定一個停車函數($a_1, a_2, \cdots, a_n$)，可以由以下做法得出對應的標號樹：

1. 將點 0 放置在第 0 層，賦值為$\binom{0}{\cdot}$

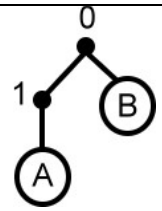
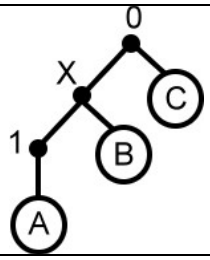
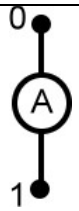
2. 每次處理時只處理 1 層，層數的挑選由上到下，同一層的處理由左到右。

3. 每次處理時觀察已處理完的層數中最下面那層，設那層的點的 F 值為 X_1~X_k。此時我們將 a_1~a_n 中所有值為 X_1+1~X_k+1 的數挑出來，把其與相對應的點連線。使其滿足「對任何一個點 X 來說都有 W(X)=F($\hat{X}$)+1。」

定理 4.1 $P_{n,1} - P_{n,2} = n^{n-2}$

依照此對應方法，我們把停車函數對應成標號樹：

(在這之後圖中的黑圓圈都代表一個可以是空集合的樹。)
(其中$P_{n,2}$中的Ⓧ代表第 1 層中最小的數)
(其中$P_{n,n}$中的①上面接的點為樹 A 中最下層的最大的數)

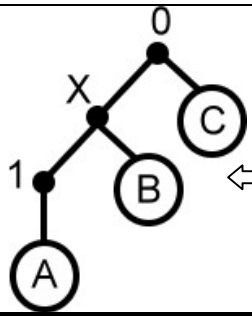
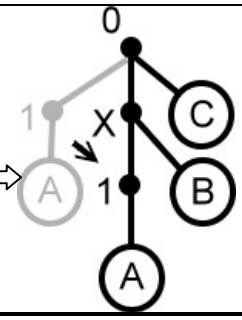
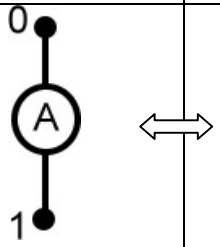
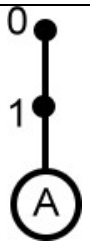
$P_{n,1}$	$P_{n,2}$	$P_{n,n}$
		

我們將$P_{n,1}$分成兩類:

第 1 類：第 1 層中有兩個以上的點(包括①)，令第 2 小的點為Ⓧ

第 2 類：第 1 層中恰有 1 點(即為①)

而接下來要證明第 1 類$P_{n,1}$和$P_{n,2}$、第 2 類$P_{n,1}$和$P_{n,n}$各有一個對應。

第 1 類$P_{n,1}$	$P_{n,2}$	第 2 類$P_{n,1}$	$P_{n,n}$
			

左=>右

第 1 類：令第 1 層中最小的數為Ⓧ，將樹①－Ⓐ接到Ⓧ下層即可。
第 2 類：將①接到樹 A 的最後一層中數字最大的一點即可。

右=>左

第 1 類：將樹①－Ⓐ接到⓪下層即可。
第 2 類：將①接到⓪的上方，再將⓪和①名字互換即可。

因此我們有 $P_{n,1} - P_{n,2} = n^{n-2}$，得證

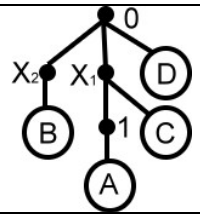
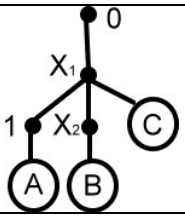
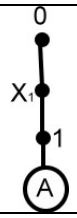
定理 4.2 $P_{n,2} - P_{n,3} = (n-1)^{n-2}$

我們將 $P_{n,2}$ 分成 3 類(如圖)

第 1 類 第 1 層有 2 個以上的點(最小和第 2 小分別為Ⓧ1, Ⓧ2)，①為Ⓧ1的子代。

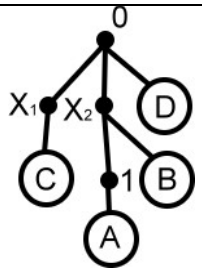
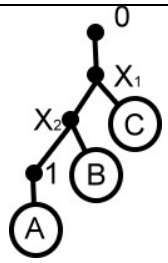
第 2 類 第 1 層恰有一點Ⓧ1，第 2 層有 2 個以上的點(最小的為Ⓧ2)。

第 3 類 第 1、2 層皆恰有一點，為Ⓧ1和①

第 1 類	第 2 類	第 3 類
		

再將 $P_{n,3}$ 分成 2 類(如圖)

第 1 類 第 1 層有 2 個點以上(最大和第 2 大分別為Ⓧ1, Ⓧ2) ①為Ⓧ2的子代。
第 2 類 第 1 層恰有 1 點Ⓧ1，第 2 層最大的為Ⓧ2，①為Ⓧ2的子代。

第 1 類	第 2 類	
		

而接下來要做的，就是證明第 1 類$P_{n,2}$和$P_{n,3}$、第 2. 類$P_{n,2}$和$P_{n,3}$各有一個對應。然後第 3 類$P_{n,2}$的量就是: $P_{n,2}$ - $P_{n,3}$的量，而第 3 類$P_{n,2}$的量即是$(n-1)(n-1)^{n-3}$，n-1 為Ⓧ1可填入的選擇種類，$(n-1)^{n-3}$為長為 n-2 的停車函數個數。而用乘法原理乘起來即為所求。

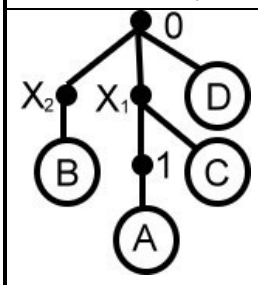
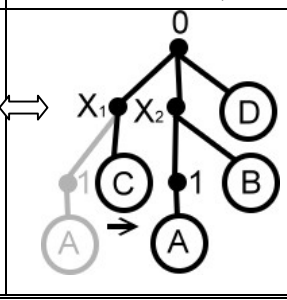
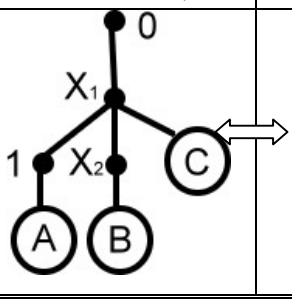
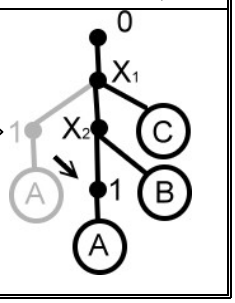
左=>右

第 1 類：將樹①－Ⓐ接到點Ⓧ2（第 1 層第 2 小)下層即可
第 2 類：將樹①－Ⓐ接到點Ⓧ2（第 2 層最小)下層即可

右=>左

第 1 類：將樹①－Ⓐ接到點Ⓧ1（第 1 層最小)下層即可
第 2 類：將樹①－Ⓐ接到點Ⓧ1（第 1 層唯一的一點)下層即可

如圖

第 1 類$P_{n,2}$	第 1 類$P_{n,3}$	第 2. 類$P_{n,2}$	第 2 類$P_{n,3}$
			

即有: $P_{n,2}-P_{n,3}$ = 第 3 類$P_{n,2}=(n-1)(n-1)^{n-3}=(n-1)^{n-2}$，得證

定理 4.3 $P_{n,k} - P_{n,k+1} = P_{n,n-k+1} - P_{n,n-k+2}$

先證$P_{n,k} - P_{n,k+1}$可以對應到所有形如下圖的標號樹：
其中 A 為有 k-1 個點的樹；B 為有 n-k 個點的樹
而①與 A 的接點為 A 中最後一層的最大的數。

即要證明：不是形如此圖的所有$P_{n,k}$都可以和$P_{n,k+1}$ 做對應。

令①在第 m 層，已知此圖為 k-leading=>第 0~第 m-1 層中必至少有 k 個點。如果：

1. 樹 A 中存在在第 m 層以下的點 or
2. 第 0~第 m-1 層中有 k+1 個以上個點

=>必可在樹 A 中找到一個點Ⓧ，滿足：「若只考慮由樹 A+⓪構成的標號樹，F(X)=k。」則我們將樹①－Ⓑ接到Ⓧ的下層即是一組對應。反之，如上圖的$P_{n,k}$中因為拔掉①－Ⓑ後，點數＜k，故找不到此對應。

當方向反過來時，則將樹①－Ⓑ接到圖中 F=k-1 的點的下方即可。而這樣的對應完畢不可能是形如上圖的標號樹。得證。

而如圖的標號樹的個數即為$C_{k-1}^{n-1} \times k^{k-2} \times (n-k+1)^{n-k-1}$，其中$C_{k-1}^{n-1}$是將 n-1 個點挑出屬於樹 A 的點的方法數，$k^{k-2}$是樹 A 本身$P_k$的總數，$(n-k+1)^{n-k-1}$是樹 B 本身$P_{n-k}$的總數，用乘法原理乘出來即得證。

定理 4.4 當 n 為偶數時(令 n=2k)，$P_{n,1} = 2 \times P_{n,k+1}$

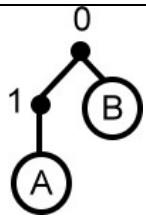


圖 1

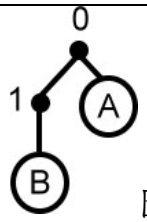


圖 2

圖 3

當 n=2k 時，1 開頭的標號樹的一般形狀必形如圖 1+圖 2，其中 A、B 為任意形狀的樹，且樹 A 的長度比樹 B 短。(因為此時標號樹上的點有 2k+1 個，扣掉①、⓪還剩 2n-1 個點，所以必定有一個樹比較短，不失一般性令其為 A)

因此我們可以得知形如圖 1 和圖 2 的兩類樹個數一樣且有對應而(k+1)開頭的樹必定形如圖 3，且符合：

1. A、B 為任意形狀的樹，且樹 B 的點數≥k(即，樹 B 比樹 A 長)
2. ①為樹 A 和樹 B 的切點(cut point)
3. ①的父代為樹 B 中某一個 F=k 的點

而我們可以把圖 1 和圖 3 作一個對應。

圖 1=>圖 3

把樹①－Ⓐ拔下來，此時剩餘部分至少有 k+1 個點，所以必存在一個 F=k 的點 X，再把①－Ⓐ的①接到點 X 下方即可。

圖 3=>圖 1

把樹①－Ⓐ拔下來接到⓪的下方即可。

則我們有 $P_{n,k+1}$ = |形如圖 3 的所有 $P_{n,1}$|

=> $P_{n,1} = 2 \times P_{n,k+1}$，得證

十一、 未來展望

1. 我們目前只做了有規律的圖形上面的彈硬幣遊戲，但是彈硬幣遊戲可以一般化，對於任一個給定的圖(graph)來說，我們可以在這個圖的正上方加一個人工的點(即政府)，所以該圖本身就可以彈硬幣，直到他塞住為止，而塞住了以後就全部加 1，在[2]裡面對於一般化時候的彈硬幣遊戲的總數有作出結果，但是完全沒有細分其中的狀況。

2. 關於停車函數和標號樹之間的對應，在[1]裡面作者提出了不同於我們這件的對應法，引起我們好奇的是這兩種對應法的比較。什麼樣的停車函數經由這兩種方法會對到同一個標號樹?而這類停車函數佔了總量的多少?而[1]裡面的對應法跟彈硬幣遊戲又有怎麼樣的關聯?

3. 在[3]裡面我們作了停車函數的一般化的計數，和一般化後根據第一個司機的喜好位子討論。而一般化後的停車函數放到彈硬幣遊戲來看的話應該會導出不少有意思的性質，或者有可能是說，一般化的彈硬幣和一般化的停車函數是同一件事。

4. 而我們在這件科展裡面主要討論 k 開頭的停車函數的關係，也可以著手去看「有 k 個人想要停第 l 個位子」這種情形。

5. 關於停車函數我們還想到數種延伸，例如說無限停車位或者是兩排車位的單行道，或者是設立殘障車位(總長度超過 n 個車位，但是某幾格不能停)…等等，有很多很多種推廣。

十二、 參考資料

[1] D. Foata and J. Riordan, Mappings of acyclic and parking functions, Aequationes math. 10 (1974), 10--22.

[2] NL Biggs, Chip-firing and the critical group of a graph J. Alg. Combin.9 (1999), 25-45.

[3] On the Enumeration of Parking Functions by Leading Numbers, Sen-Peng Eu, Tung-Shan Fu, and Chun-Ju Lai, preprint, 2004

Car Parking Made Hard

Introduction

In this project we focus on two combinatorial problems:
The Parking Function and ***the Chip-firing Games***.

Parking Function:

There are *n* parking spaces which are labeled from 1 to *n* along a one-way street. *n* drivers try to park their cars in turn with a favorite lot in their mind. A driver drives his car in front of his favorite lot. If it's vacant, then he parks in it. If it's occupied, then he drives along the road to park in the next unoccupied space. If no vacant spaces left, then he'll give up parking and drive away since it's a one-way street.

If all drivers park their cars successfully, we call the sequence $(a_1, a_2, \ldots, a_n)$ a ***parking function***, where a_i denotes the favorite space of the i^{th} driver.

Example: (2,4,1,1) is a ***parking function*** while (3,3,1,4) is not.

Chip-firing Games:

Considered a complete graph K_{n+1} with chips on the nodes. We define a ***firing*** as: " If there are more than *n*-1 chips on a node, then we distribute 1 chip from the chosen node to any other ones respectively".

There is a special node called the ***government*** which has infinite chips. It ***fires*** as long as all the other nodes cannot ***fire***.

For example, in the figure 1.1, we can fire node *A*.

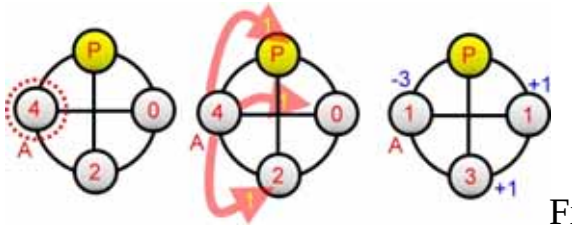


Figure 1.1

On the other hand, if the configuration is like figure 1.2, we have no

choice but to ***fire*** the ***government***.

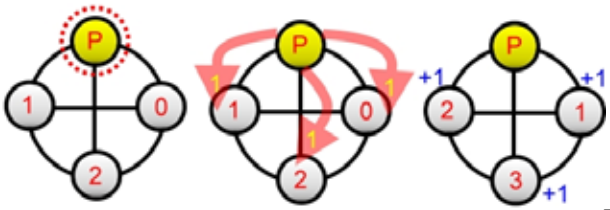


Figure 1.2

In some configurations we have to ***fire government*** at the beginning, and they will return to the initial states after several ***firings***. We call them the ***critical states***. As figure 1.3.

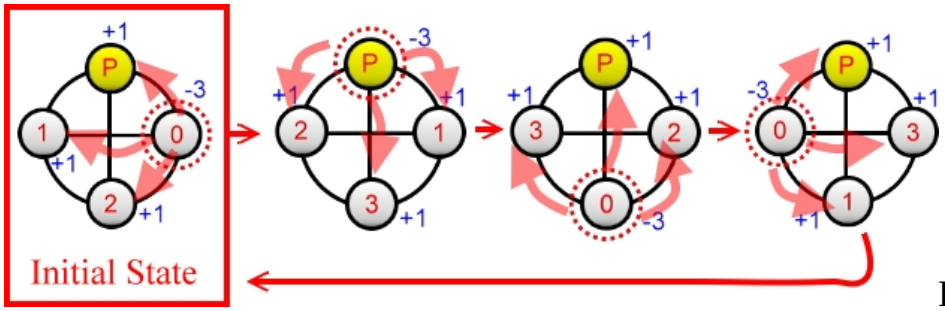


Figure 1.3

The ***parking function*** and the ***chip-firing games*** are two distinct problems, which appeared to be totally unrelated. In this project we focus on the interesting relations between them. We obtain the following results:

1. Let P_n denote the number of ways that n drivers park successfully. Let $P_{n,k}$ denote the number of ways that n drivers park successfully with $a_1 = k$. We figured out them and discovered several amazing properties.

2. We establish a combinatorial bijection between the ***parking functions*** and the ***labeled trees,*** which is equivalent to the second bijection of Riordan and Foata between all ***parking functions*** and all ***acyclics*** **[1].** However our approach is much more succinct and makes the combinatorial proof of k-leading counting possible.

3. Also, we exhibit a bijective correspondence between all ***parking functions*** and the seemingly unrelated "***critical states*** of the ***chip-firing game***"

4. The setting of the ***chip-firing game*** is then extended from a complete graph to a ***wheel*** graph and to a ***fan*** graph. To our surprise, the results are related to the classical ***Fibonacci*** and ***Lucas*** number.

Parking Functions

It is known that $P_n = (n+1)^{n-1}$. The table lists all the ***parking functions*** with n = 3 classified by the leading terms.

Example	Park able?	Example	Park able?	Example	Park able?
1 1 1	○	2 1 1	○	3 1 1	○
1 1 2	○	2 1 2	○	3 1 2	○
1 1 3	○	2 1 3	○	3 1 3	×
1 2 1	○	2 2 1	○	3 2 1	○
1 2 2	○	2 2 2	×	3 2 2	×
1 2 3	○	2 2 3	×	3 2 3	×
1 3 1	○	2 3 1	○	3 3 1	×
1 3 2	○	2 3 2	×	3 3 2	×
1 3 3	×	2 3 3	×	3 3 3	×
$P_{n,1}$	8	$P_{n,2}$	5	$P_{n,3}$	3

The first few $P_{n,k}$'s are:

n	$P_{n,1}$	$P_{n,2}$	$P_{n,3}$	$P_{n,4}$	$P_{n,5}$	$P_{n,6}$
1	1					
2	2	1				
3	8	5	3			
4	50	34	25	16		
5	432	307	243	189	125	
6	4802	3506	2881	2401	1921	1296

Table 2.1

By taking the differences between adjacent pairs of numbers in Table 2.1, we obtain an amazing regular pattern as in Table 2.2

$n = 1$						1	**1**	0					
$n = 2$					2	**1**	1	**1**	0				
$n = 3$				8	**3**	5	**2**	3	**3**	0			
$n = 4$			50	**16**	34	**9**	25	**9**	16	**16**	0		
$n = 5$		432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0	
$n = 6$	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

Table 2.2

Table 2.2 shows that:

1) The red part is **symmetric** though it has nothing to do with the original definition.

2) The values of the first diagonals columns are 1, 1, 3, 16, 125 and 1296, namely 1^{-1}, 2^{0}, 3^{1}, 4^{2}, 5^{3} and 6^{4}, which are exactly n^{n-2}! Recall that $P_n = (n+1)^{n-1}$, we found that $P_{n,1} - P_{n,2} = P_{n,n} = P_{n\text{-}1.}$

3) In the cases that n is even. Seeing the colored grids, we found that the 1st term is exactly as twice as the $k+1^{\text{th}}$ term (let $n = 2k$), which is another reasonless property!

$n = 2$					2	**1**	1	**1**	0				
$n = 4$			50	**16**	34	**9**	25	**9**	16	**16**	0		
$n = 6$	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

Bijection between Parking Functions and Labeled Trees

For any ***labeled tree*** with labels from 0 to n, we define a ***canonical form*****:** A tree is rooted at 0 and, for all nodes that share a same parent, they are in the ascending order from left to right.

(In a ***labeled tree*** in ***canonical form*** as Fig. 3.1. Because node A, B and C share a common parent, A < B < C must holds true; D < E is also deduced by the same reason. But F is not necessary greater than E since they don't share a common parent.)

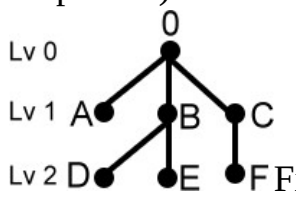


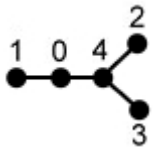
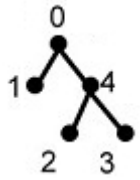
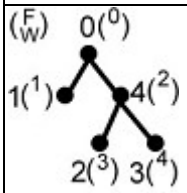
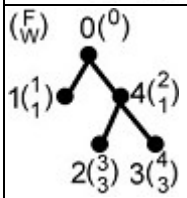
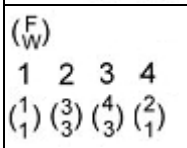
Fig 3.1

Next, for every node x we define two functions $w(x)$ and $f(x)$. Here w stands for "want", $w(x)$ represents the favorite lot of driver x. Therefore $(w(1), w(2), \ldots, w(n))$ is the desired ***parking function***. Due to the fact that it's uncertain for a driver whether he can park into his favorite lot, we establish $f(x)$, which stands for "fact", to keep track of the locations. We label them on the right side of node x as $\binom{f}{w}$.

The procedure to generate f and w in a given graph is the following:

1. Put the nodes of the tree in a ***canonical form***.
2. Traverse the labeled tree by a breadth first search and let $w(B_k) = k$, where B_k is the k^{th} node by BFS.
3. Let $\hat{x}$ denotes the parent of x. For all x, we have $w(x) = f(\hat{x}) + 1$.

For example:

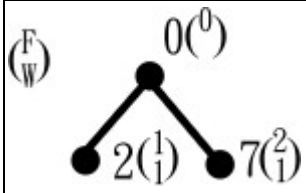
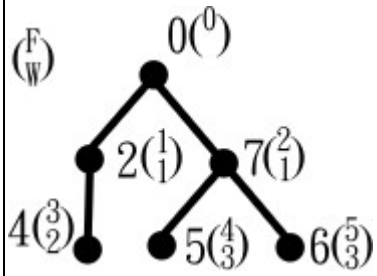
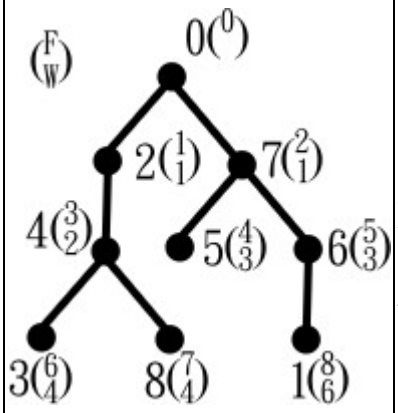
	This is the original ***labeled tree***.
	Step 1 We put the nodes in the ***canonical form***.
	Step 2 We determine f by the ***BFS***.
	Step 3 We determine w by " $w(x) = f(\hat{x}) + 1$"
	Thus, the desired ***parking function*** is $(w(1), w(2), w(3), w(4))$, which is (1,3,3,1)

On the other hand, we can build up a ***labeled tree*** for any given ***parking function*** $(a_1, a_2, \ldots, a_n)$by the following procedure (Satisfying the main idea " $w(x) = f(\hat{x}) + 1$ for all x"):

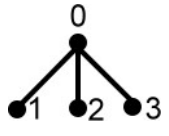
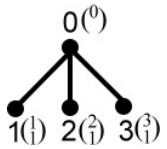
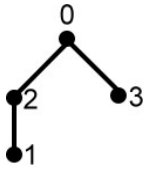
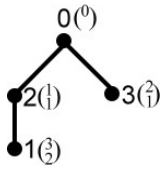
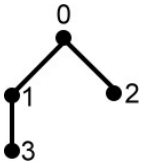
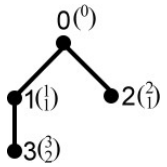
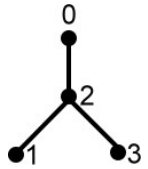
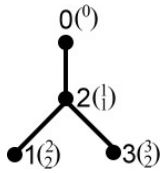
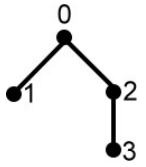
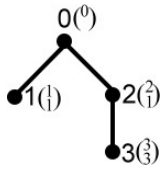
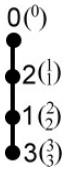
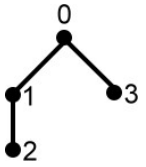
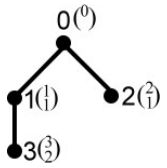
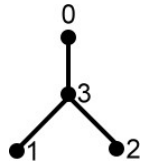
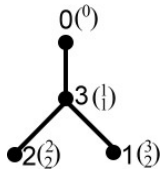
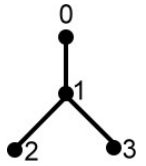
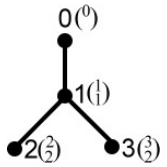
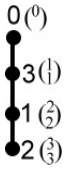
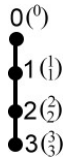
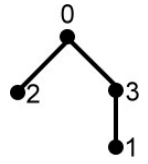
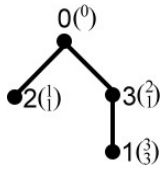
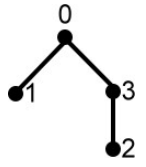
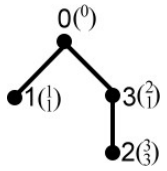
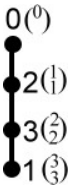
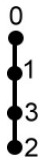
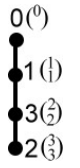
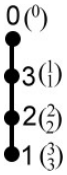
We use a recursive method to build up the ***labeled tree*** level by level.

1. Put node 0 on level 0 and let $f(0) = 0$.
2. For the lowest level, assume that labels of the nodes are $L_1, L_2, \ldots, L_s$. In order to satisfy "$w(x) = f(\hat{x}) + 1$ for all x", we pick every node p such that $w(p) = L_i + 1$ (i is one of 1~s) and let these nodes consist a new level beyond the lowest level and connect them to their corresponding parent.
3. For all children that share a common parent, we sort them in the ascending order from left to right.
4. Repeat step 2 and 3 till there are no nodes remain.

For Example, the given ***parking function*** is (6,1,4,2,3,3,1,4)

	Step 1 We put node 0 on level 0
	Step 2 Due to $f(0) = 0$, we pick up every node x with $w(x) = 1$ (namely node 2 and 7) and then connect them to node 0.
	Step 3 All available $f(x)$ on level 1 are 1 to 2, so we pick up every node x with $w(x) = 2$ to 3 (namely node 4, 5 and 6) and connect them to their responding parent.
	Step 4 All available $f(x)$ on level 2 are 3 to 5, so we pick up every node x with $w(x) = 4$ to 6 (namely node 1, 3 and 8) and connect them to their responding parent.
	There's no nodes left. Thus, we complete the procedure of building up the ***labeled tree*** in a ***canonical form***.

Thus we can establish a mapping table. This is the case that $n = 3$.

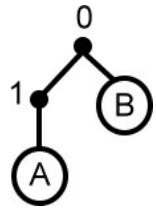
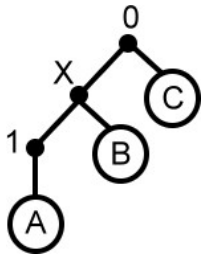
PF	Figure		PF	Figure	
111			211		
112			212		
113			213		
121			221		
122			231		
123			311		
131			312		
132			321		

In this section we will prove the theorems mentioned below by pure combinatorial way by way of ***labeled tree***. Here a black circle denote a tree which is allowed to be an empty set.

Theorem 1.1 $P_{n,1} - P_{n,2} = n^{n-2} = P_{n,n}$

n=1					1	**1**	0					
n=2				2	**1**	1	**1**	0				
n=3			8	**3**	5	**2**	3	**3**	0			
n=4		50	**16**	34	**9**	25	**9**	16	**16**	0		
n=5	432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0	
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

The mapping will turn $P_{n,1}, P_{n,2}$ and $P_{n,n}$ into the forms of the figures below (In figure 4.1b node *x* is the node with smallest label in level 1; in figure 4.1c the parent of node 1 is the node with largest label in the lowest level of subtree A.):

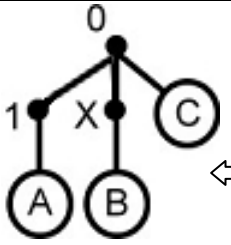
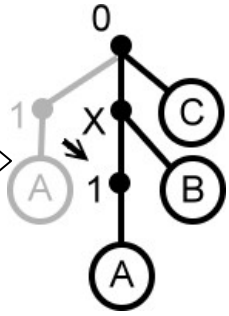
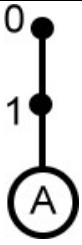
$P_{n,1}$	$P_{n,2}$	$P_{n,n}$
 Figure 4.1a	 Figure 4.1b	 Figure 4.1c

Then we classify $P_{n,1}$ into 2 types and going to prove that type-1 and type-2 $P_{n,1}$ are responding to $P_{n,2}$ and $P_{n,n}$ respectively.

Type-1: There are at least 1 node in level 1 include node 1. Then we name the node with second smallest label to be node *x*

Type-2: There is exactly 1 node namely the node 1 in the level 1.

Proof:

Type-1 $P_{n,1}$	$P_{n,2}$	Type-2 $P_{n,1}$	$P_{n,n}$
 Figure 4.2a	 Figure 4.2b	 Figure 4.2c	 Figure 4.2d

For a node *p* in a ***canonical form***, let *T*(*p*) denotes a subtree rooted at *p* containing all descendents of *p*. As the figures below, in type-1 $P_{n,1}$ we unplug *T*(1) and attach it to the node *x*, and then we'll obtain a standard form of $P_{n,2}$. On the other hand, we can obtain a type-1 $P_{n,1}$ by unplugging *T*(1) and then attaching it to the node 0 for any given $P_{n,2}$.

In the case of type-2 $P_{n,1}$, we unplug subtree A and connect node 1 to node 0, then we attach subtree A to node 1. Hence we obtain a $P_{n,n}$; for a given $P_{n,n}$, we unplug node 1 and then attach it to the node with largest label in the lowest level of A.

By Cayley's formula we know these are bijection, so the total number are the same, therefore we know $P_{n,1} - P_{n,2} = P_{n,n}$.

And it's easy to deduce that $P_{n,n} = n^{n-2}$ by map $P_{n,n}$ to P_{n-1} by simply deleting node 0 and renaming original node 1 to be node 0, vice versa. Therefore, $P_{n,1} - P_{n,2} = n^{n-2}$

Theorem 1.2 $P_{n,2} - P_{n,3} = (n-1)^{n-2}$

n=1					1	**1**	0					
n=2				2	**1**	1	**1**	0				
n=3			8	**3**	5	**2**	3	**3**	0			
n=4		50	**16**	34	**9**	25	**9**	16	**16**	0		
n=5	432	**125**	307	**64**	243	**54**	189	**64**	125	**125**	0	
n=6	4802	**1296**	3506	**625**	2881	**480**	2401	**480**	1921	**625**	1296	**1296**	0

As the same argument as Theorem 1.2, we classify $P_{n,2}$ into 3 types:

Type-1: There are at least 2 nodes in level 1, we name the node with smallest label to be X_1, which is the parent of node 1, then we name the second smallest to be X_2.

Type-2: There is exactly 1 node X_1 in level 1, and there are at least 2 nodes (include node 1) in level 2. We name the second smallest node to be X_2.

Type-3: There is exactly 1 node in level 1 and 2 respectively. We name the one in level 1 to be X_1 and the one in level 2 to be node 1.

Type-1	**Type-2**	**Type-3**

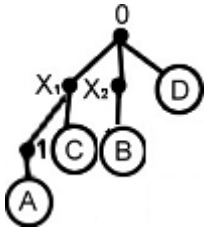
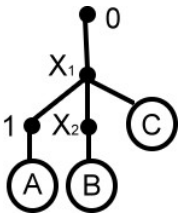
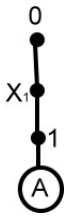

		

Figure 4.3a	Figure 4.3b	Figure 4.3c

Then we classify $P_{n,3}$ into 2 types and prove that type-1 $P_{n,2}$ and type-1 $P_{n,3}$ are bijective, so do type-2 $P_{n,2}$ and type-2 $P_{n,3}$. Thus, $P_{n,2} - P_{n,3}$ is equal to the total number of type-3 $P_{n,2}$, which can be easily enumerated.

Type-1: There are at least 2 nodes in level 1, we name the node with

smallest label to be X_1, then we name the second smallest to be X_2, which is the parent of node 1.

Type-2: There is exactly 1 node X_1 in level 1, and there are at least 2 nodes in level 2. We name the smallest node, which is the parent of node 1, to be X_2.

Type-1	**Type-2**	

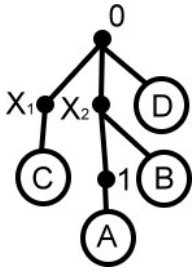
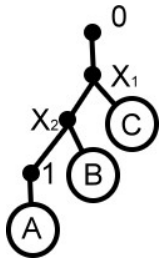

	

Figure 4.4a	Figure 4.4b	

And we can use the "Unplug and attach" method mentioned before to prove the bijection, just as the figures below:

Type-1 $P_{n,2}$	Type-1 $P_{n,3}$	Type-2 $P_{n,2}$	Type-2 $P_{n,3}$

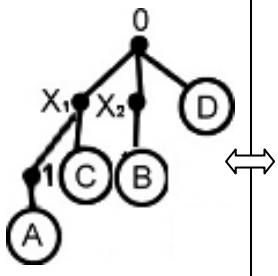
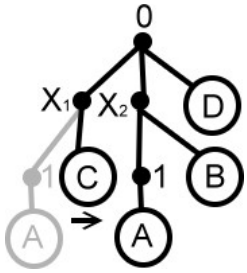
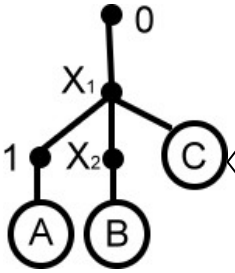
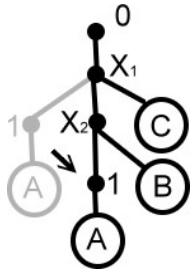

			

Figure 4.5a	Figure 4.5b	Figure 4.5c	Figure 4.5d

In these 2 types we just simply unplug and attach T(1) between X_1 and X_2, we'll find that they are indeed bijective.

Next, we're going to figure out the total number of type-3 $P_{n,2}$. There are $n-1$ ways to choose X_1, and there are $(n-1)^{n-3}$ ways to choose a labeled tree rooted at node 1. Consequently we obtain $P_{n,2} - P_{n,3} = (n-1)(n-1)^{n-3} = (n-1)^{n-2}$

Theorem 1.3 $P_{n,k}$ - $P_{n,k+1}$ $= P_{n,n-k+1}$ - $P_{n,n-k+2}$

This equation shows the reason why the table is symmetric. We're going to prove that there's a bijection between a sub set *S* of $P_{n,k}$ and $P_{n,k+1}$, *S* is the set of all k-leading ***parking functions*** that corresponding to ***labeled trees*** whose F(1)>*k*.

Proof:

For any given ***parking function***, we name *T* as the responding ***labeled tree***. *T*\{T(1)} contains more or equal to *k* nodes if and only if F(1)>*k*. If so, there exist a node X in *T*\{T(1)} such that F(X) = *k* .So we know that the sufficient and necessary condition that there exist a node X in *T*\{T(1)} such that F(X) = *k* is F(1)>*k*.

If such node X exist, we can attach T(1) on X to form a new ***labeled tree*** with W(1) = F(X) + 1 = $k+1$, which represents it's a $k+1$-leading ***parking function***. As the same argument we can unplug T(1) from a ***labeled tree*** with F(1) = $k+1$ and attach it on a node Y (Here F(Y) = *k*). By Cayley's formula we know there's a bijection between S and $P_{n,k+1}$. So $P_{n,k} - P_{n,k+1}$ is equal to the number of $P_{n,k}\backslash\{S\}$.

Recall the mapping we mentioned in the previous paragraph, we know that the responding ***labeled tree*** of $P_{n,k}\backslash\{S\}$ can be portrayed as figure 4.6.

Figure 4.6

A and B are rooted subtrees with $k-1$ and $n-k$ nodes respectively. The parent of the node 1 is the node with largest label in the lowest level of A.

The number of $P_{n,k}\backslash\{S\}$ is $\underbrace{C_{k-1}^{n-1}}_{A} \times \underbrace{k^{k-2}}_{B} \times \underbrace{(n-k+1)^{n-k-1}}_{C}$

A: Choose $k-1$ nodes to form subtree A.
B: The total ways to form a ***labeled tree*** A of length $k-1$.
C: The total ways to form a ***labeled tree*** B of length $n-k$.

Therefore we know $P_{n,k} - P_{n,k+1} = C_{k-1}^{n-1} \times k^{k-2} \times (n-k+1)^{n-k-1}$
$= C_{n-k}^{n-1} \times (n-k+1)^{n-k-1} \times k^{k-2} = P_{n,n-k} - P_{n,n-k+1}$

Theorem 1.4 In the case that $n = 2k$, $P_{n,1} = 2P_{n,k+1}$

n=2					2	1	1	1	0			
n=4			50	16	34	9	25	9	16	16	0	
n=6	4802	1296	3506	625	2881	480	2401	480	1921	625	1296	1296	0

In the case that $n = 2k$, the 1-leading parking function must can be portrayed as figure 4.7a union figure 4.7b, where subtree A is shorter than subtree B, due to the total number of nodes in A∪B is odd.

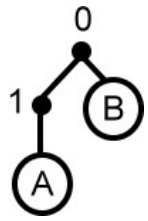


Figure 4.7a

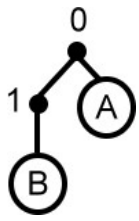


Figure 4.7b

Figure 4.7c

We're going to prove that there's a 2-to-1 mapping between $P_{n,1}$ and $P_{n,k+1}$. Firstly we know that the ***labeled trees*** responding to $k+1$-leading ***parking functions*** can be portrayed as figure 4.7c, where subtree B contains more or equal to $k+1$ nodes and node X, with F(X) = $k+1$, is the parent of the node 1.

As the same "unplug and attach" method in this paragraph, we know that there's a bijection between $P_{n,1}$ in the form of figure 4.7a and $P_{n,k+1}$; so do $P_{n,1}$ in the form of figure 4.7b. Therefore, it's a 2-to-1 mapping, which shows that $P_{n,1} = 2P_{n,k+1}$.

The Chip-firing Games

Considered a complete graph K_{n+1} with chips on the nodes. We denote a ***firing*** as: If there are more than n-1 chips on a node, then we distribute 1 chip from the chosen node to any other node . There is a special node P in these n+1 nodes, let it has infinite chips, and we fire node P as long as none of the other nodes can be fired. So the ***Chip-firing Games*** is a never-ending game since node P exists.

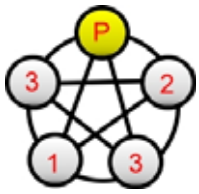
We give the node P a suitable name, the ***government***, which has a lot of money. It gains 1 dollar when any other node ***fires*** **(**as levying tax**);** and it ***fires*** as long as all the other nodes are poor (as a welfare system).

In some configurations we have to fire node P at the beginning, and they will return to the initial states after several firings. We call this kind of states as ***Critical States***.

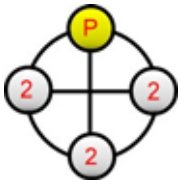
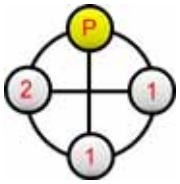
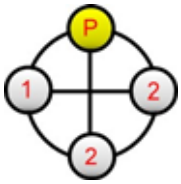
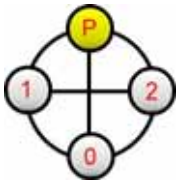
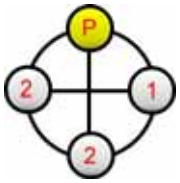
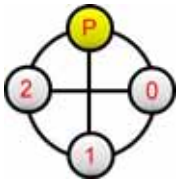
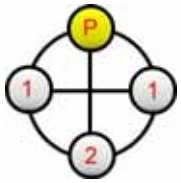
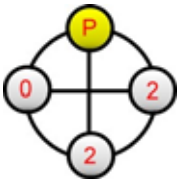
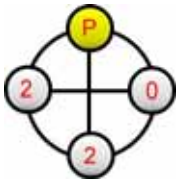
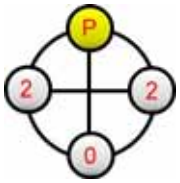
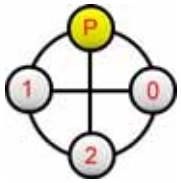
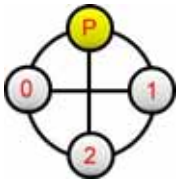
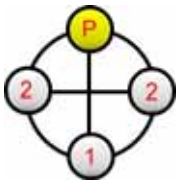
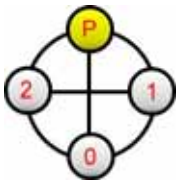
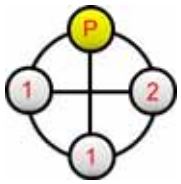
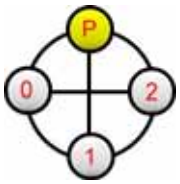
We soon found that the total number of ***Critical States*** in K_{n+1} is $(n+1)^{n-1}$, exactly equal to P_n, which means these two problems that seems to be totally different talk about the same thing.

We write down the number of chips starting from the ***government*** anticlockwise in a form of a sequence. Then for each term we subtract it from n and obtain another sequence. In the following we'll prove the latter ones are exactly ***parking functions***!

Take this figure for example, we write down (3132) in the first step. Then for each term we subtract it from 4 (This is a K_5, so n = 4), we'll obtain (1312), which is exactly a ***parking function***!



For instance, this is a table where n = 3.

	111		122		211		231
	112		123		212		311
	113		131		213		312
	121		132		221		321

Theorem 2: The mapping in the following is a bijection between ***Critical States*** and ***Parking Function***

The mapping ϕ:

The original ***parking function*** is $(a_1, a_2, ..., a_n)$. Then the responding ***Critical State*** is $(n-a_1, n-a_2, ..., n-a_n)$

The mapping ϕ^{-1}:

Let $(c_1, c_2, ..., c_n)$ denotes a structure of chips in a ***Critical State***, where c_i represents the chips on the i[th] node. And the responding ***parking function*** is $(n-c_1, n-c_2, ..., n-c_n)$

It's known that the technical definition of the ***parking function*** is that: "when sorting $(a_1, a_2, ..., a_n)$ into $(b_1, b_2, ..., b_n)$, where $b_1 \le b_2 \le ... \le b_n$, we'll have $b_i \le i \quad \forall i$"

We're going to prove the sufficient and necessary condition of a ***Critical State*** is "when sorting $(c_1, c_2, ..., c_n)$ into $(d_1, d_2, ..., d_n)$, where $d_1 \ge d_2 \ge ... \ge d_n$, we'll have $d_i \ge n-i \quad \forall i$" Then the mapping ϕ and ϕ^{-1} are apparently true.

In some cases there are more than one node can be fired. Here we introduce some lemma.

Lemma 2.1 Whatever order we choose to fire these nodes, we'll obtain the same ***Critical State*** in a common initial configuration. (Theorem 3.8, [2])

Lemma 2.2 If a configuration $(c_1, c_2, ..., c_n)$ is a ***Critical State*** then it will return to $(c_1, c_2, ..., c_n)$ after exactly n+1 firings. (Lemma 3.3, [2])

Lemma 2.3 A sufficient and necessary condition of the ***Critical State*** is "when sorting $(c_1, c_2, ..., c_n)$ into $(d_1, d_2, ..., d_n)$, where $d_1 \ge d_2 \ge ... \ge d_n$, we'll have $d_i \ge n-i \quad \forall i$"

Proof:

Necessary Condition:

If $d_i \geq n-i \quad \forall i$, then (government, $d_1, d_2, ..., d_n$) is a proper order which shows that it is a ***Critical State***.

Sufficient Condition:

If $(c_1, c_2, ..., c_n)$ is a configuration of a ***Critical State***. Firstly, we sort $(c_1, c_2, ..., c_n)$ into $(d_1, d_2, ..., d_n)$, where $d_1 \geq d_2 \geq ... \geq d_n$. According to the first part of the definition of the ***Critical State***, we have $n-1 \geq d_i \geq 0$ $\forall i$.

By Lemma 2.1 we simplify the procedure by defining a ***standard firing***:

"Each time we ***fire*** the node with the smallest label among if able." The labels mentioned here are the sorted ones (i.e. the *i* of d_i)

We know that the first node to ***fire*** must be the ***government*** due to the definition of the ***Critical States***. By the ***standard firing*** we assure that the second node to fire must be d_1 since it has the smallest label.

Next, we're going to prove "If we have ***fired*** the ***government*** and $d_1 \sim d_k$ $(k \leq n-1)$, then the next node to ***fire*** must be d_{k+1} by the ***standard firing***." by induction.

For $d_1 \sim d_k$, they gain *k* chips and lost *n* chips in the former ***firings***. So we have $k-1 \geq d_i \geq 0$ for *i* = 1 ~ *k*. Soon we deduce $n-2 \geq d_i$ for *i* = 1 ~ *k* since $k \leq n-1$. So the next node to ***fire*** can't be $d_1 \sim d_k$. In the rest nodes we know d_{k+1} is one of the non-government nodes with the most chips, and we also know that the next node can't be the government. Suppose not, then the total number of chips of non-government nodes won't hold constant after n+1 firings, which contradicts to Lemma 2.2. So the next node must be d_{k+1}.

Thus, we know that the node d_{k+1} can be fired after *k*+1 firings, which indicates that $d_{k+1} + k + 1 \geq n$. Moreover, $d_i \geq n-i \quad \forall i$.

By Lemma 2.3, the mapping ϕ and ϕ^{-1} are apparently true. So we proved Theorem 2.

By the bijection, we have the following corollary:

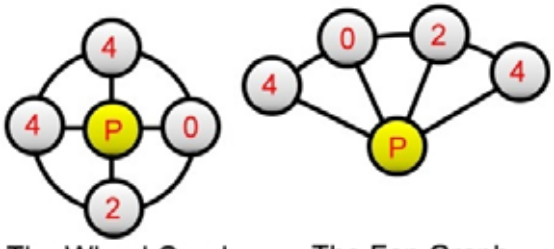
1. The enumeration by leading term holds constant through this bijection. So the results we deduced in the third paragraph can be interpreted in the forms of ***Chip-firing Games,*** which has **NOT** brought up by others yet.

2. We enumerate the leading term of generalized ***parking function*** in [3]. This shows that we may find a new expression of responding ***Chip-firing Games*** model.

The Generalized Chip-firing Games

In addition to the original ***Chip-firing Games***, we obtain some interesting results of the ***Chip-firing Games*** on special graphs. To generalize the problem, we redefine the ***firing*** to be: "If a node owns chips more or equal to its degree, then we distribute 1 chip from the it to any other node that connects it respectively."

In the following sections we'll discuss the Chip-firing Games on special graphs such as the ***Wheel Graph*** and the ***Fan Graph***.



The Wheel Graph The Fan Graph

The ***Wheel Graphs*** consists of a circle with *n* nodes and an axle that connects to any other node. Likewise, the ***Fan Graphs*** consists of a line with *n* nodes and an axle that connects to any other node.

We focus on these two graphs because they have an axle, which is suitable to be the government. Consequently we found that the Chip-firing Games on them are associated with ***Lucas sequence*** and ***Fibonacci sequence*** respectively.

Some basic definitions are given here:

1. W_n denotes the number of ***Critical States*** on ***Wheel*** with n+1 nodes.
2. If we write down the number of chips from the top node anti-clockwise, we'll obtain a sequence with length n. Let $W_{n,k}$ denotes the number of k-leading sequence.
3. If a node has no chips on it, we call it a ***break***. ***Breaks*** separate the ***Wheel*** into several parts.

In this paragraph we discuss the sufficient ad necessary condition of the ***Critical States*** on ***Wheel***. Furthermore, we enumerate its amount by leading term.

Theorem 3 The sufficient ad necessary condition of the ***Critical States*** on ***Wheel*** is that "For any part of ***Wheel***, there is at least one node with 2 chips, and the rest have less or equal to 1 chip."

For example:

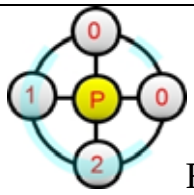


Figure 7.1

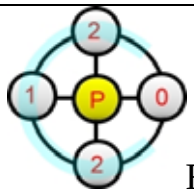


Figure 7.2

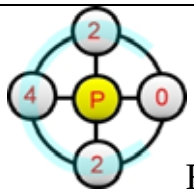


Figure 7.3

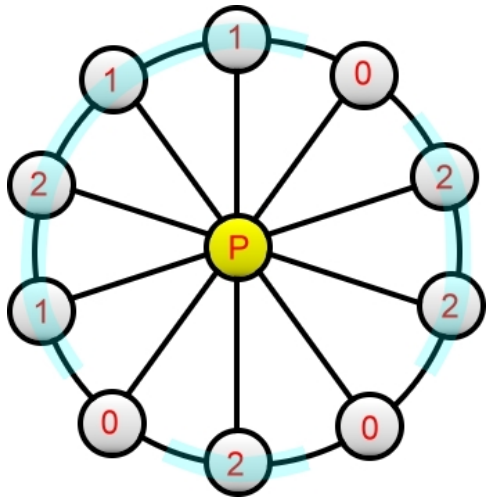


Figure 7.4

In figure 7.1, the ***breaks*** separate the ***wheel*** into 2 parts. There is no node with 2 chips in the smaller part, so this isn't a ***critical state***.

In figure 7.2, though there's a break on the ***wheel***, it still counts as 1 part. And there're 2 nodes with 2 chips, so this is a ***critical state***.

In figure 7.3, there is a node with more than 2 chips, so this isn't a ***critical state***.

In figure 7.4, the ***breaks*** separate the ***wheel*** into 3 parts. We check the parts one by one and find that for each part there is at least 1 node with 2 chips, so this is a ***critical state***.

Proof:

Sufficient condition:

If "For any part of ***Wheel***, there is at least one node with 2 chips, and the rest have less or equal to 1 chip." is true. Then we know that the first node to ***fire*** must be the ***government*** since the degrees of the nodes on the wheel are all 3, which satisfied the former condition of the ***critical states***.

Next, we're going to prove that it'll return to the initial configuration after several firings. To prove it, we discuss by the number of ***break***.

Case 1. **# *break* = 0**

In this case the number of chips of every node is either 1 or 2. Suppose not, there must be a ***break***. Consequently the number of chips of every node is either 2 or 3 after the first ***firing*** on the ***government***, and there is at least 1 node with more than 2 chips.

For any node that hasn't been fired yet, it will be able to ***fire*** after one of its neighboring node ***fired***.

By **Lemma 2.1**, we may arbitrarily determine an order to ***fire*** the rest nodes. So we make the order starts from any one node with 3 chips (after the first ***firing***) and go on fire them clockwise. By this method every node is ***fired*** exactly once. So the chips will remain constant because every node gains 3 chips from its neighboring nodes and lose 3 chips due to the ***firing***. So this is a ***critical state***.

Case 2. **# *break* = 1**

By the same reason, the first ***firing*** node is the ***government*** and the next one is a node *X* that has 3 chips on it. *X* goes clockwise to the only ***break*** *B* via $A_1 \sim A_i$, and goes anticlockwise to *B* via $B_1 \sim B_k$. Then we let the firing order to be $(X, A_1, \ldots, A_i, B_1, \ldots, B_k)$. After the procedure, all the neighboring nodes of *B* are ***fired***. So we can fire B. All nodes are ***fired*** after the procedure, which shows this is a ***critical state*** by the same argument as case 1.

Case 3. **# *break*** $\geq$ **2**

Just as the method mentioned in case 2, for each part of ***wheel*** we can ***fire*** all the ***non-break*** nodes in the beginning. By the time all the neighboring nodes of each ***break*** are ***fired*** and all the ***breaks*** are able to ***fire***. After ***firing*** of these breaks we obtain that all the nodes are ***fired*** exactly once and that it's a ***critical state*** by the same argument in case 1.

Necessary condition

If a state is ***critical***, we are going to prove that "For any part of ***Wheel***, there is at least one node with 2 chips, and the rest have less or equal to 1 chip." Suppose not, there exist a node with more than 2 chips or none of these nodes has more than 1 chip.

If there's a node with more than 2 chips, it's able to ***fire***, which contradicts to the definition of ***critical state***.

If none of these nodes has more than 1 chip, for any part in the form of (011…10). It'll be (*a*2…2*b*) when the ***government*** is about to ***fire*** again, where *a* and *b* are either 1 or 2 (judged by whether the neighboring nodes of the two ***breaks fire*** or not)**.**

Put an eye on this new ***state***, you'll find that all pair of ***breaks*** that form a part with entire 1 will turn into ***non-break*** nodes. On the other hand, as the same argument (just mentioned in the sufficient condition section) we know that the ***non-break*** nodes in a part with at least 1 node with 2 chips will hold constant, which shows that the new state satisfy the sufficient conditions. So the new state is a ***critical state***.

We know that the total number of chips on all the ***non-government*** nodes is periodical and there are the fewest chips in the ***critical state***. This shows that the original configuration is **NOT** a ***critical state*** since the chips on the ***non-government*** nodes will never decline to the initial configuration, which contradicts to our presupposition.

Therefore, we've proved the sufficient and necessary condition of the ***critical states*** on ***wheel***.

The Critical States on Wheel

We try to find the structure of critical states on wheel. To do this, we run program to analyze it by the number of ***breaks***. So we obtain the following table.

#***Break***	$n = 1$	2	3	4	5	6	7	8	9
0	1	3	7	15	31	63	127	255	511
1		2	9	28	75	186	441	1016	2295
2				2	15	69	252	804	2349
3						2	21	128	594
4								2	27
Total	1	5	16	45	121	320	841	2205	5776

It seems that we can hardly find the pattern of the table. But when we sum them up, we find that the total number is equal to L_n^2 when n is odd; and is equal to $L_n^2 - 4$ when n is even, where L_n denotes the n^{th} ***Lucas sequence***.

Total	1	5	16	45	121	320	841	2205	5776
=	1^2	3^2-4	4^2	7^2-4	11^2	18^2-4	29^2	47^2-4	76^2

Furthermore we observe it by the leading terms and the following table is obtained.

k	n = 1	2	3	4	5	6	7	8	9
0	0	1	3	8	21	55	144	377	987
1	0	1	5	16	45	121	320	841	2205
2	1	3	8	21	55	144	377	987	2584
Total	1	5	16	45	121	320	841	2205	5776

Then we found that the number of $W_{n,0}$ and $W_{n,2}$ are exactly the even terms of the ***Fibonacci sequence***. In this paragraph we're going to prove these properties.

Lemma 4.1 $W_{n,01} = W_{n-1,01} + W_{n-1,02}$

According to the sufficient and necessary condition of the ***critical states*** on ***Wheel***. The 3rd term of one of $W_{n,01}$ can't be 0. So we have $W_{n,01} = W_{n,011} + W_{n,012}$. And then we can make a bijection between $W_{n,011}$ and $W_{n-1,01}$ by simply deleting the 2nd term from $W_{n,011}$. So we have $W_{n,011} = W_{n-1,01}$; and also $W_{n,012} = W_{n-1,02}$. Then $W_{n,01} = W_{n-1,01} + W_{n-1,02}$ is soon deduced.

Lemma 4.2 If we have $W_{n,0} = F_{2n-2}$, then we can deduce $W_{n,02} = W_{n,01} + F_{2n-5}$.

If the 1st part of a state in $W_{n,02}$ contains more than 1 node with 2 chips, then we can make a bijection between this kind of $W_{n,02}$ and $W_{n,01}$ by take away 1 chip from the 2nd node, vice versa. After this move it'll still remain a ***critical state***.

If there's only 1 node with 2 chips in the 1st part, then we enumerate it by the length of the 1st part. So we obtain $W_{n,02} = W_{n,01} + W_{n,020} + W_{n,0210} + W_{n,02110} + \ldots + W_{n,021\ldots10} + W_{n,021\ldots11}$.

For $W_{n,021\ldots10}$ where there are k 1s in the 1st part. By simply deleting the 1st part, we deduce $W_{n,021\ldots10} = W_{n-k-2,0}$. Applying it to the equation we just mentioned, we have $W_{n,02} = W_{n,01} + W_{n-2,0} + W_{n-3,0} + \ldots + W_{2,0} + W_{1,0} = W_{n,01} + F_{2n-6} + F_{2n-8} + \ldots + F_2 + 1 = W_{n,01} + F_{2n-5}$.

Theorem 4.1 $W_{n,0} = F_{2n-2}$

We're going to prove this by induction. We know it's true when $n = 2$. Next, we need to prove: If $W_{n,01} = F_{2n-4}$ and $W_{n,02} = F_{2n-3}$ is true when $n = 2 \sim k-1$, then it'll still hold true when $n = k$.

$$W_{k,01} = W_{k-1,01} + W_{k-1,02} \text{ (\textbf{Lemma 4.1}) } = F_{2n-5} + F_{2n-6} = F_{2n-4}.$$
$$W_{k,02} = W_{k,01} + F_{2n-5} \text{ (\textbf{Lemma 4.2}) } = F_{2n-4} + F_{2n-5} = F_{2n-3}.$$

By induction, **theorem 4.1** is proved.

Lemma 4.3 $W_{n,20} = W_{n,02}$

Given a ***critical state*** whose two leading terms is (2,0). If we rewrite it by picking the 2nd node to be start node and record chips on the nodes clockwise(Recall that we recorded them ati-clockwise), and then a 02-leading sequence is produced and it also represents a ***critical state***. Thus, the action of recounting the sequence is a bijection between $W_{n,20}$ and $W_{n,02}$, so $W_{n,20} = W_{n,02}$.

Lemma 4.4 $W_{n,21} = W_{n,22} = W_{n-1,2}$

Deleting the 2nd term of a sequence in either $W_{n,21}$ or $W_{n,22}$, we obtain a sequence in $W_{n-1,2}$. If original sequence represents a ***critical state***, then the new sequence will still be a ***critical state*** due to each part still contains at least node with 2 chips, vice versa.

To map $W_{n-1,2}$ onto either $W_{n,21}$ or $W_{n,22}$, we simply add 1 or 2 respectively to be the new 2nd term. After this action it will still be a ***critical state***. Consequently we know that there's a bijection between $W_{n-1,2}$ and both $W_{n,21}$ and $W_{n,22}$. So $W_{n,21} = W_{n,22} = W_{n-1,2}$

Theorem 4.2 $W_{n,2} = F_{2n}$

We have $W_{n,2} = W_{n,20} + W_{n,21} + W_{n,22}$
$= W_{n,02} + W_{n-1,2} + W_{n-1,2}$ (Lemma 4.3 and 4.4)

$= F_{2n-3} + 2W_{n-1,2}$

By induction we know that it's true that $W_{n,2} = F_{2n}$ when $n = 1$, we're going to prove that as long as it's true when $n = k-1$, it'll still hold true when $n = k$.

Thus, we have $W_{n,2} = F_{2n-3} + 2W_{n-1,2} = F_{2n-3} + 2F_{2k-2} = F_{2n}$

Theorem 4.3 $W_{n,1} = W_{n-1}$

We map $W_{n,1}$ to W_{n-1} by deleting the 1st term of $W_{n,1}$. On the other hand we add a 1 to be the 1st term of the new sequence. As the same reason, this bijection does not change whether it's a ***critical state***. Due to there's a bijection between them, we have $W_{n,1} = W_{n-1}$.

Theorem 4.4 $W_n = L_1 + L_3 + ... + L_{2n-1}$, which is equivalent to: "$W_n = L_n^2$ when n is odd; "$W_n = L_n^2 - 4$ when n is even"

By induction, it's true when $n = 1$, we're going to prove that as long as it's true when $n = k-1$, it'll still hold true when $n = k$.

We know $W_k = W_{k,0} + W_{k,1} + W_{k,2} = F_{2k-2} + W_{k-1} + F_{2k} = L_{2k-1} + (L_1 + L_3 + ... + L_{2k-3}) = L_1 + L_3 + ... + L_{2k-1}$

The Critical States on Fan

As the previous paragraph, we enumerate the number of ***critical states*** on the ***Fan Graph***. In the following we give some definitions:

1. N_n denotes the number of n-term ***critical states*** on ***Fan***.
2. $N_{n,k}$ denotes the number of k-leading n-term ***critical states*** on ***Fan***.
3. $N_{n,ab}$ denotes the number of a-leading n-term ***critical states*** whose last term is b.

The reason of using N instead of F is to distinguish it from the ***Fibonacci sequence***.

As we did in the previous paragraphs, we enumerate the total number by leading term. Then the table is obtained.

k	$n = 1$	2	3	4	5	6	7	8	9
0	0	1	3	8	21	55	144	377	987
1	1	2	5	13	34	89	233	610	1597
Total	1	3	8	21	55	144	377	987	2584

We notice that the table is highly related to ***Fibonacci sequence***. In the following we prove the sufficient and necessary condition of the ***critical states*** on ***Fan***, which is similar to what on ***Wheel***.

Theorem 5 The sufficient and necessary condition of the ***critical states*** on ***Fan*** is "For any part, there exist a point X such that $c(X) = \deg(X) - 1$, and the rest have less or equal to 1 chip."

We establish a bijection between N_n and $W_{n+1,0}$, the mappings are described below:

The mapping $\varphi(N_n \to W_{n+1,0})$:

We establish a virtual point S which connects to both 2 ***ends*** and the government. Then we let $c(S) = 0$ and the number of chips on both 2 ends increased by 1.

The mapping $\varphi^{-1}(W_{n+1,0} \to N_n)$:

We delete the point stands for the 1st term so that the Wheel Graph will then turned into a Fan Graph. Next, we respectively take away 1 chip from 2 ***ends***.

If these 2 mappings don't change whether it's a ***critical state***. Then Theorem 4 is true because that we proved the sufficient and necessary condition in the previous paragraph.

Proof:

For a ***critical state*** on ***Fan***, there's a ***firing*** order (***government***, $p_1, p_2, ..., p_n$). With an eye on the ***Wheel*** Graph through mapping φ, we can establish a new ***firing*** order (***government***, $p_1, p_2, ..., p_n, S$) due to the mapping doesn't change the number of chips of every ***non-end*** point. Observing point *S* after p_n is ***fired***, we find that the number of chips is increased by 3 (from the ***government*** and both 2 ***ends***) and *S* is able to ***fire***. So the new configuration is a ***critical state***.

For a ***critical state*** on ***Wheel***, there's a ***firing*** order (***government***, $q_1, ..., q_i, 1, r_1, ..., r_k$). After mapping φ^{-1}, we'll obtain a configuration on ***Fan***. The new ***firing*** order can be (***government***, $q_1, ..., q_i, r_1, ..., r_k$) since the mapping only takes away 1 chip on both 2 ***ends*** respectively. Due to that it also reduce the degree of both 2 ***ends*** by 1, the mapping doesn't effect whether those $i+1$ points can be ***fired*** or not.

Thus, we know the total number of N_n is equal to $W_{n+1,0}$, which is F_{2n}. Furthermore we enumerate it by $N_{n,ab}$ (*a* and *b* denote the number of chips on ***ends***).

Let *n* = 4, we'll have the table below:

$N_{n,00}$	$N_{n,01}$	$N_{n,10}$	$N_{n,11}$	
0210	0111	1110	1101	1011
0120	0211	1120	1021	1201
0220	0121	1210	1211	1121
	0221	1220	1111	1221
	0201	1020		
3	5	5	8	

Through the mapping we found that $N_{n,00}, N_{n,01}, N_{n,10}$ and $N_{n,11}$ are responding respectively to $W_{n+1,101}, W_{n+1,102}, W_{n+1,201}$ and $W_{n+1,202}$. We know:

1. $N_{n,0} = N_{n,0...1} + N_{n,0...0} = W_{n+1,102} + W_{n+1,101} = W_{n+1,10} = F_{2n-2}$
2. $N_{n,1} = N_{n,1...0} + N_{n,1...1} = W_{n+1,201} + W_{n+1,202} = W_{n+1,20} = F_{2n-1}$

Therefore we generalize the table to be:

ends	$N = 1$	2	3	4	5	6	7	8	
00	0	0	1	3	8	21	55	144	F_{2n-4}
01	0	1	2	5	13	34	89	233	F_{2n-3}
10	0	1	2	5	13	34	89	233	F_{2n-3}
11	1	1	3	8	21	55	144	377	F_{2n-2}
Total	1	3	8	21	55	144	377	987	F_{2n}

References

[1] N.L. Biggs. Chip-firing and the critical group of a graph. *J. Algebr. Comb.* **9** (1999), 25-45.

[2] S.P. Eu, T.S. Fu, and C.J. Lai. On the enumeration of parking functions by leading numbers. accepted by *Adv. Appl. Math.*

[3] D. Foata and J. Riordan. Mappings of acyclic and parking functions. *Aequationes Mathematicae* **10** (1974), 10-22.

[4] A.G. Konheim and B. Weiss, An occupancy discipline and applications, *SIAM J. Applied Math.* **14** (1966), 1266-1274.

[5] R. Stanley. Enumerative Combinatorics, volume 2. Cambridge University Press, 1999.

[6] C.H. Yan, Generalized parking functions, tree inversions and multicolored graphs. *Adv. Appl. Math*. **27** (2001) 641–670.