臺灣二 OO 三年國際科學展覽會

科　　別：電腦科學科

作品名稱：關於二元根樹上的鏈、反鏈、獨立集的計數及極值

得獎獎項：電腦科學科第一名

英特爾電腦科學獎

美國第五十四屆國際科技展覽會

學　　校：臺北市立建國高級中學

作　　者：吳宗穎、鄒承孝

作 者 簡 介

我的名字是吳宗穎，目前就讀於台北市立建國高級中學二年級數理資優班。

對電腦科學界的發展來說我只算懂得一點皮毛，到了高中才學了一些簡單的程式語言 Visual Basic，並且能夠開始自己讀一些電腦的語言書籍。由於高中老師給予的資源相當多，因此也使我進步不少，最後對電腦科學領域的興趣也越來越濃，因此開始尋找在這個領域的目標，跟相關領域的研究，並且發現了一些有趣的結果，這就是我做這次科展的主要原因。

我的名字是鄒承孝，目前就讀於台北市立建國高級中學二年級數理資優班。

我喜歡玩電腦，但真正開始學習電腦科學卻是從上高中之前的那個暑假。

在國中老師的建議之下，我才接觸了 Visual C++等的程式語言。剛開始的時候覺得這很無聊，但到了後來一個個程式跑起來的時候，我卻發現了自己對於電腦科學的興趣。後來上了高中的電腦課，發現也滿吸引我的，所以才選擇了資訊做爲我的專題目標，也是我做這次科展的原因。

特別感謝游森棚老師盡心盡力的指導。

摘要

二元根樹是資訊科學上最重要的結構之一，因此其結構的探討就很有價值。在這個研究中，我們探討二元根樹上反鏈(Antichain)，鏈(Chain),獨立集(Independent set)計數。在每一個子題之下，我們求出計算反鏈，鏈，獨立集的方法。之後我們求出極值(包括最大跟最小值)，並且討論該極值之下所有可能的樹形。此外，我們並討論要列出二元根樹上這些資料(反鏈，鏈，獨立集)所需的演算法。

Abstract

In this paper we enumerate the number of antichains, chains, and independent sets of a binary tree of size n. After that we focus on the extreme values (maximal/minimal). we calculate the extreme values and discuss the possible graphs of trees when achieving these extreme values. Moreover, we offer algorithms for listing all the objects in each extreme case.

壹、前言

一、簡介

在電腦科學中，樹(tree)可以說是最基本的結構。而其中平面根樹(plane rooted tree)跟二元根樹(binary rooted tree)更可以說是最基本和重要的。因此分析這些結構有理論和實際上的價值，而且的確在這一方面，經由許多資訊科學家和數學家的努力，也已經有許多重要的成果。

在最近(1997)的一篇論文中，Klazar[1]計算了平面根樹上關於許多統計量的計數，這些統計量包括鏈(chain)，反鏈(antichain)，獨立集(independent set)等等。該篇論文中求出了平面根樹關於這些統計量的計數遞迴式與漸近公式，並且比較了這些統計量的大小。但是，但並沒有實際求出精確的公式，而且雖然從文章中可以隱約看出，但論文中並沒有真正考慮發生極端(極大值或極小值)的情況。

這篇論文是我們研究的動機和出發點。既然平面根樹跟二元根樹都是重要的結構，在二元根樹上應該有相應的結果。我們接著查了許多的資料，發現多半的研究都把焦點放在一般的樹(平面根樹，標號數 labelled tree)上的統計量，卻鮮少有專門討論二元根樹，而且都沒有討論在這些統計量上的最大或最小值。

因此在這個報告中，我們討論二元根樹上關於反鏈、鏈、獨立集的計數。我們的焦點放在計數的遞迴和最大最小值，以及在發生這些極值時二元樹的形狀。最後我們並討論要計算或是要將二元根樹上的這些訊息全部列出來時演算法的複雜度。

在接下來的報告中，我們分成四個部份，分別介紹二元根樹及討論在鏈，反鏈，獨立集三個統計量上的情形。

貳、二元根樹上的反鏈

一、二元根樹上的反鏈

我們將二元根樹視為一個圖形上下顛倒的偏序集，因此任取二元樹上的兩個頂點，不一定能比較大小。因此可定義反鏈：

定義 1：

在二元樹上挑出數點所形成的子集中，若任兩點在二元樹上間都不能比較大小，則該子集稱為反鏈(Antichain)。

例 1：

如圖 1-1 是一個 size 大小為 2 的二元根樹，其中共有 11 個反鏈。分別是

ϕ，{1}，{2}，{3}，{4}，{5}，

{2,3}，{3,4}，{3,5}，{4,5}，{3,4,5}

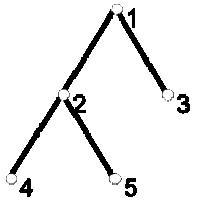


圖 1-1

二、二元根樹上反鏈的計數

很自然的問題是任給一個二元根樹，會有多少個反鏈。

利用遞回的方式可以找到一個算法

```
演算法：ANTI-CHAINCOUNT(A)
1    if    A=NIL
2          then  return 1
3    else
4          return
     ANTI-CHAINCOUNT(left[A])*ANTI-CHAINCOUNT(right[A])+1
```

這個算法的時間複雜度為 O(n)

設二元根樹T的兩個子樹由左至右為T',T''； 又令$a(T)$代表這個二元根樹T的反鏈個數，則有以下引理

引理 1：任一二元根樹反鏈的個數是根所接的兩子樹反鏈個數的積加一。即

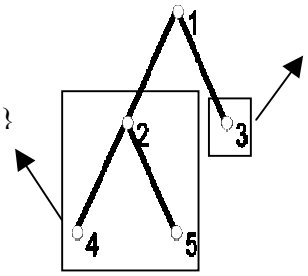
$$a(T)=a(T')a(T'')+1.$$

證明：在該二元根樹中，一子樹的任一個反鏈與另一子樹的任何反鏈的聯集必也為此二元根樹的反鏈，但又因樹根本身自成一個子集，故得證。

$a(T_1)$

ϕ ，{2} ，{4}

，{5} ，{4,5}



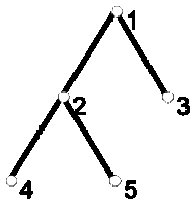
$a(T_2)$

ϕ ，{3}

- ϕ ，{1}，{2}，{3}，{4}，{5}，{2,3}，{3,4}，{3,5}，{4,5}，{3,4,5}

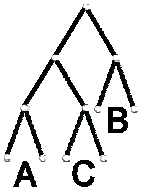
由引理一馬上得出一個遞迴算反鏈個數的方法。

例 2：同圖 1-1，因為以 2 為根的子樹有 5 個反鏈，以 3 為根的子樹有 2 個反鏈，所以全部的有 $5\cdot 2+1=11$ 個反鏈。



引理 2：若將二元樹中，反鏈較少但位置距離樹根較遠的子樹，與反鏈較少位置距離樹根較近的子樹對調，則新二元樹反鏈個數比原二元樹多。

證明：設有一樹 T_1 為  其中 $a(B)>a(C)$ ，

又另一樹 T_2 為  ，即將 T_1 的子樹 B,C 交換。則

$$\begin{aligned} a(T_2)-a(T_1) &= \{(a(A)a(C)+1)\cdot a(B)+1\}-\{(a(A)a(B)+1)\cdot a(C)+1\} \\ &= a(B)-a(C) \\ &> 0 \end{aligned}$$

得證。

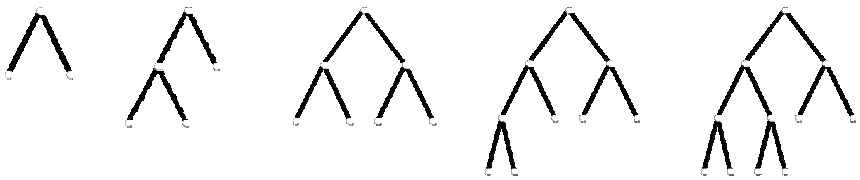
利用這兩個引裡，我們可以求給定二元根樹的大小為 n 時，反鏈的最大值最小值。這是接下來兩個小節的內容。

三、二元根樹反鏈個數的最大值

接下來我們求給定二元根樹的大小為n時，反鏈的最大值。令此值為$\max a(T_n)$。我們先定義"飽和二元根樹"：直觀上就是由左至右，由上到下依次加上分叉。

定義 2：一大小為n的**飽和二元根樹**是指：若此樹有第k層的分叉，則所有位於$\leq k$層的分叉都要填滿。而在第k層中，所有位於該分叉左邊的分叉都要填滿。

例 3：以下為$n=1,2,3,4,5$的飽和二元根樹



我們得到關鍵定理：

定理 1：給定二元根樹的大小為n時，飽和二元根樹的反鏈有最大值。

證明：設該排子樹的反鏈個數分別為$A_1A_2A_3\cdots A_n$，則得到該樹的反鏈個數為

$$\prod_{1}^{k=2^n} A_k + \sum_{1}^{p=2^{n-1}} \frac{\prod_{1}^{k=2^n} A_k}{A_{2p-1}A_{2p}} + \cdots + \sum_{1}^{p=2^n} \frac{\prod_{1}^{k=2^n} A_k}{\prod_{1}^{e=2^n-1} 2^n p - e}$$

第二項達最大值時，為第一，二大的子樹排在一起；第三，四大的子樹排在一起；以此類推。第三項達最大值時，為第一、二、三、四的子樹排在一起；第五、六、七，八的子樹排在一起；以此類推。故使上列所有最大值同時出現的時候，就是飽和二元根樹。得證

有了這個引理：我們可以求出大小為n的二元根樹反鏈有最大值的式子(雖然並不好算！)。底下是我們的主要結果。

定理２：定義數列$\{t_i\}_{i\geq 1}$如下：

$$t_1 = 2 \text{，} t_i = {t_{i-1}}^2 + 1 (i \geq 2).$$

並將n寫為

$$n = 2^k - 1 + a_1 2^0 + a_2 2^1 + ... + a_k 2^{k-1}$$

的形式，其中$a_i \in \{0,1\}$。則給定二元根樹的大小為n時，反鏈的最大值為

$$\max a(T_n) = \prod_{i=0}^{k-1}(t_i + a_{i+1}) + \sum_{p=0}^{k-1}\prod_{i=0}^{k-1}(t_i + a_{i+1}) + 1$$

證明：由定理１和引理１。得證

例４：求$\max a(T_{13})$，即求出大小為13的二元根樹反鏈最大值。

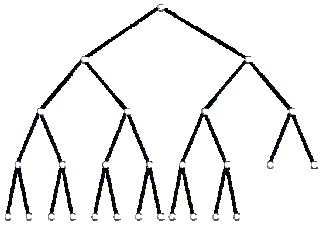
解: 當$n = 13$時，因為

$$n = 2^3 - 1 + 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2 \text{。}$$

因此反鏈最大值為

$$\prod_{e=0}^{2} T_{e+a_{e+1}} + \sum_{p=0}^{2}\prod_{e=0}^{2} T_{e+a_{e+1}} + 1$$
$$= (T_{0+0} * T_{1+1} * T_{2+1}) + (T_{0+0} * T_{1+1} * T_{2+1} + T_{1+1} * T_{2+1} + T_{2+1})$$
$$= 2 * 26 * 677 + 2 * 26 * 677 + 26 * 677 + 677 + 1$$
$$= 88688$$

此時二元樹必為飽和，形如



我們可以把上個定理寫成遞迴的演算法：方便起見考慮 $\max a(T_n)-1$。令 $a_n=\max a(T_n)-1$，底下遞迴求數列 $\{a_i\}_{i\geq 1}$，初始值為 $a_0=1$。

演算法：給定 n，要求 a_n

1. 找 i 使得 $2^{i-1}\leq n\leq 2^i-1$
2. 寫 $n-1=p+q$，若 $2^{i-1}\leq n\leq 3*2^{i-2}-1$，則 $p=2^{i-2}-1$；
   若 $3\cdot 2^{i-1}-1\leq n\leq 2^i-1$，則 $q=2^{i-1}-1$
3. 則 $a_n=(a_p+1)(a_q+1)$

例 4：求 $\max a(T_{10})$，即求出大小為 10 的二元根樹反鏈最大值。

解：由上述演算法將 $10-1=9$ 寫成 $9=3+6$，至少有一個是 2^j-1 的形式。故

$$a_{10}=(a_3+1)(a_6+1)$$

同理，

$$a_6=(a_3+1)(a_2+1)\text{，}$$
$$a_3=(a_1+1)(a_1+1)\text{，}$$
$$a_2=(a_1+1)(a_0+1)\text{，}$$
$$a_1=(a_0+1)(a_0+1)=2\cdot 2=4\text{，}$$

故 $a_2=10$，$a_3=25$，$a_6=286$，$a_{10}=7462$。所求

$$\max a(T_{10})=a_{10}+1=7463\text{，}$$

即大小為 10 的二元根樹反鏈最大有 7463 個。

四、二元根樹反鏈個數的最小值

大小固定為n的二元根樹T_n反鏈最少是多少呢？令此值為$\min a(T_n)$。以下是我們的的結果：

定理 3：大小固定為n的二元根樹反鏈的最小值為$2^{n+2}-2^n-1$.即

$$\min a(T_n)=2^{n+2}-2^n-1$$

證明：根據引理 2，我們馬上得出$a(T_n)$出現最小值時，樹的圖形為每層都只排一個分岔的形狀。以下用歸納法證明：

$n=0$時(只有一個點，取或不取)，$\min a(T_0)=2^2-2^0-1=2$。設$n=k$時反鏈值為$\min a(T_k)=2^{k+2}-2^k-1$. 則當$n=k+1$時，根據引理 1 以及出現最小值時樹的圖形，得到

$$\begin{aligned}&\min a(T_{k+1})\\&=2\left(2^{k+2}-2^k-1\right)+1\\&=2^{k+3}-2^{k+1}-1\end{aligned}$$

由數學歸納法得證$\min a(T_n)=2^{n+2}-2^n-1$。■

由定理我們知道當$a(T_n)$有最小值時二元根樹的結構是很簡單的，即每次長分叉都只能再往下多長一層。

例 6：當$n=5$時，$\min a(T_5)=2^7-2^5-1=95$個。此時樹的形狀必為

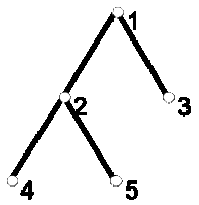
■

參、二元樹上的鏈

一、什麼是二元樹上的鏈

二元樹上的鏈定義為：在二元樹上挑出數點所形成的子集中，若任兩點在二元樹上皆有上下比較關係，則該子集稱為鏈。

例７：如右圖，該二元樹中
{1,2}’{1,3}’{1,4}’{1,5}’{2,4}’{2,5}’{1,2,4}’{1,2,5}
是為此二元樹的鏈。



二、　二元樹上鏈的計數

利用遞回的方式可以找到一個算法

演算法：CHAINCOUNT(A,key)

```
1    if    A=NIL
2          then  return 0
3    else
4          key ← key*2+1
5          return
     key+CHAINCOUNT(left[A],key)+CHAINCOUNT(right[A],key)
```

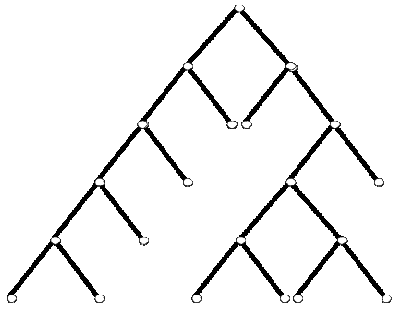
時間複雜度分析：

由於每個節點都會各跑到一次，所以時間複雜度為θ(n)。

定理４：任一樹鏈的計數方法為：計算每一列點的個數
若在第 n 列有 An 個點的話(樹根為第 0 列)
則總點數為 $\sum_{1}^{k=n} A_k\left(2^k-1\right)$

證明：因為任一點都只與他上下點會有關聯，因此每多一個點時，得到的鏈的個數只會跟他在第幾列有關係，所以我們發現當在第 n 列時多出一個點則該點會使樹多出 2^n-1 個，因此得證。

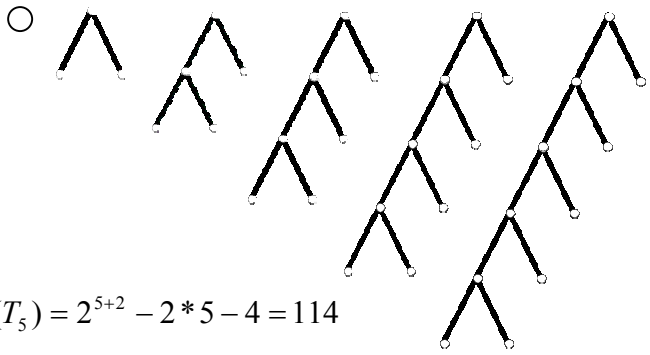
例８：如右圖
其值為
2*1＋4*3＋4*7＋4*15＋6*31
＝288個鏈



三、二元樹上的最大值

根據定理４，我們可以知道ｎ級二元樹鏈的最大值為 $\max c(T_n)2^{n+2}-2n-4$

也就是長成

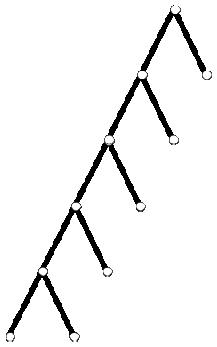


樣子的圖形

例９：

當ｎ＝５時其最大值為 $\max c(T_5)=2^{5+2}-2*5-4=114$

也就是如右圖的情況

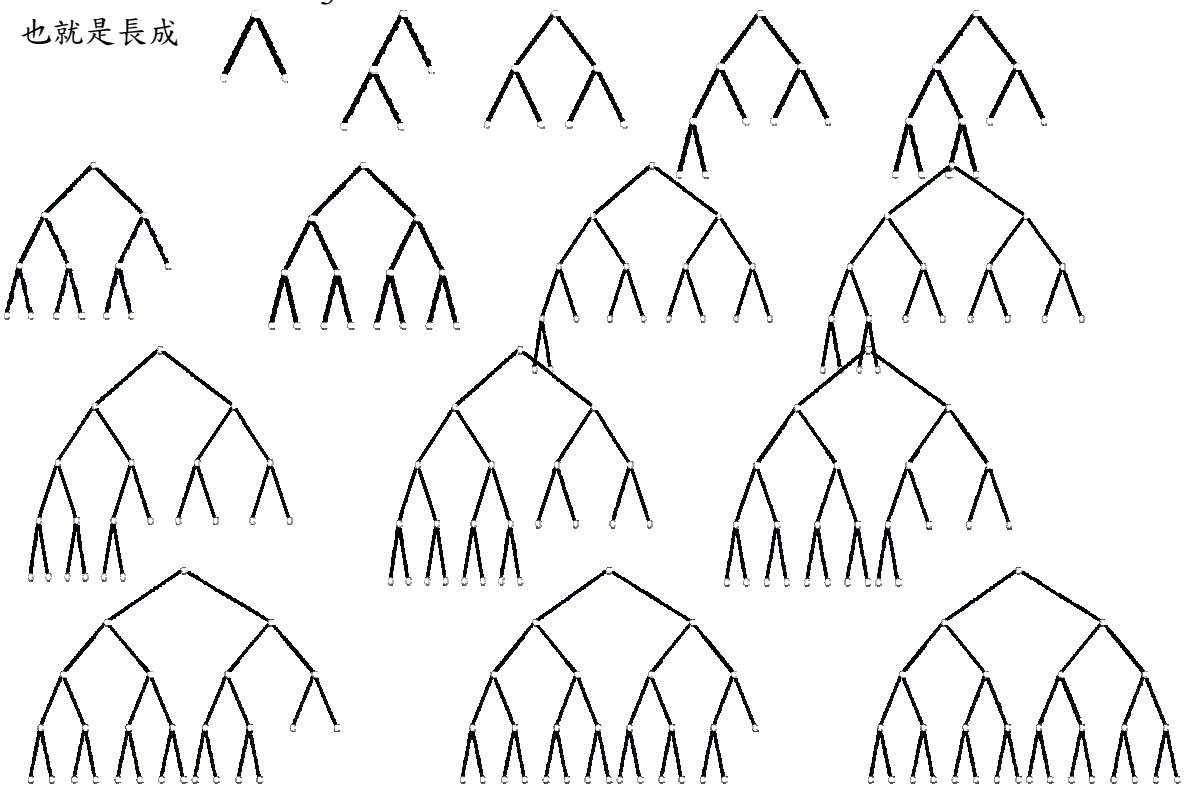


四、二元樹上鏈的最小值

根據定理４，我們可以知道若 $n = 2^k - 1 + m$

則值為 $\min c(T_n) = \frac{4^{k+1}+2}{3} - 2^{k+1} + m \times 2^{k+2} - 2m$

也就是長成

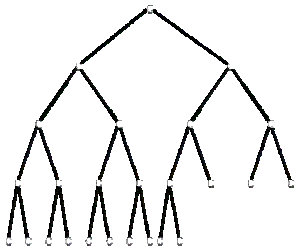


樣子的圖形

例 10：當 $n = 12$ 時 $n = 2^3 - 1 + 5$

$$\min c(T_{12}) = \frac{4^{3+1}+2}{3} - 2^{3+1} + 5 \times 2^{3+2} - 2 \times 5 = 220$$

也就是如



的圖形

肆、二元樹上的獨立集

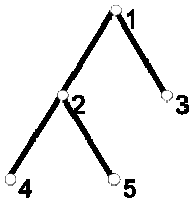
一、什麼是二元樹上的獨立集

二元樹上的獨立集定義為：在二元樹上挑出數點所形成的子集中，若任兩點皆不相連，則該子集稱為獨立集。

例 11：如右圖，該二元樹中

{}’{1}’{2}’{3}’{4}’{5}’{1,4}’{1,5}’{2,3}’{3,4}’{3,5}’{4,5}’
{1,4,5}’{3,4,5}

是為此二元樹的獨立集。

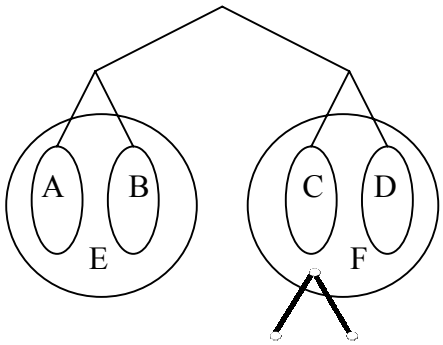


二、二元樹上獨立集的計數

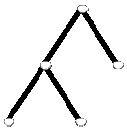
我們可以用遞回的方式找到一個計數算法

```
演算法：INDEPENDENT-SET-SCOUNT(A)
1    if    A=NIL
2          then return 1
3    else
4          return I-SCOUNT(left[A])*I-SCOUNT(right[A])+
5                I-SCOUNT(left[left[A]])*I-SCOUNT(right[left[A]])*
6                I-SCOUNT(left[right[A]])*I-SCOUNT(right[right[A]])
```

引理３：任一二元樹的獨立集若表示為右圖
則其獨立集各數為Ｅ Ｆ ＋Ａ Ｂ Ｃ Ｄ
（定義：空樹定義為 1 個獨立集
一個點的樹的定義為２)

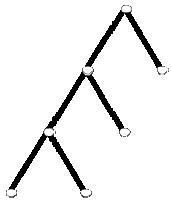


例 12：如右圖
第一圖其值為
2*2+1*1*1*1=5



第二圖其值為
5*2+2*2*1*1=14

第三圖其值為
14*2+5*2*1*1=38



伍、結論與未來展望

一、結論

下面是我們所有結果統整出來的圖表：

	Antichain		Chain		Independent Set	
	Max	Min	Max	Min	Max	Min
極值的樹形	 第參節定理 1	 第參節定理 3	 第肆節定理 4	 第肆節定理 4	 證明中	 證明中
極值計數的方法	ok 第參節定理 2	ok 第參節定理 3	ok 第肆節定理 4	ok 第肆節定理 4	困難，尚無進展	ok 證明中
給定任意樹計數的方法	ok 第參節引理 1		ok 第肆節定理 4		ok 第伍節引理 3	
演算法	Rec. O(n)		Rec. O(n) Non-rec.		Rec. Ω(n)	

*Rec = Recursion 遞迴

二、未來展望

1. 繼續研究 Independent set 的性質並完成證明
2. 完成 Independent set 的極值的計算
3. 推廣研究 maximal independent set 的極值與計數方法
4. 將給定任意樹形中的 Antichain, Chain, Independent set, maximal independent set 列表
5. 計算所有同等級樹形的 Antichain, Chain, Independent set, maximal independent set 值的平均值
6. 推廣研究其他樹(非 binary tree)的 Antichain, Chain, Independent set, maximal independent set

陸、後記

我們探討的題目是在電腦科學中有關於基礎理論的層面。我們從 binary tree 的角度，來分別探討 Antichain, Chain, Independent set 的計數問題。而這些結果都是這個領域中非常重要的新結果。我們最要感謝我們的指導老師，他的用心指導，使這個專題遠遠超出了我們之前的程度。他讓我們真正的了解到什麼才是做科學研究，又什麼才是電腦科學。在研究的過程中，挫折與辛酸是免不了的，但我們一一的克服了它們。而這樣的過程更教給了我們很多深刻的道理，這也是我們除了知識的增長以外額外得到的寶貴經驗。我們真的認爲，這次科展的經驗將會使我們永生難忘。

柒、參考資料

[1]Klazar,*Twelve counting of trees*, European J. Combin (18) , 1997, 195-210.
[2]Herbert S. Wilf, *The Number of Maximal Independent Sets in a Tree*, Siam J. Alg. Disc. (7) 125-130.

評 語

1. 對二元樹的鏈、反鏈與獨立集，推導其計數公式或找出最大與最小計數，是有相當的水準，在獨立集的計數還不完整。

2.理論推導與電腦模擬完整，可用電腦實地視覺化展示模擬，效果更佳。

On enumerations and extreme values of Anti-chains, Chains, and Independent Sets of Binary Trees

Wu, Tsung-Yin Tsou, Cheng-Hsiao

Taipei Municipal Chien-Kuo senior High School, no.56, Nanhai Rd., Taipei, Taiwan 100

Abstract

The Binary Tree is one of the most important data structures in computer science. The concept of Binary Tree can be applied to data sorting and searching. In this project we study the statistics of Chains, Anti-Chains, and Independent Sets in a Binary Tree. The Chain can be used to sort/search a set of correlated data; on the contrary Anti-Chain can be used to sort/search a set of independent data, while the Independent Set can be used to classify the data, which are interfering with each other.

In this project we propose algorithms to enumerate the number of Anti-Chains, Chains, and Independent Sets of any given Binary Tree. Also we calculate the extreme values of these three statistics of Binary Trees of fixed sizes and discuss the extreme graphs when these extreme values are attained. We prove the dual property of extreme graphs between Anti-Chain and Chain of Binary Trees.

Finally we compare the relation among these three statistics of a Binary Tree.

Keyword: anti-chain, chain, independent set, binary tree

A. Problem

The Binary Tree is one of the most important data structures in the computer science. The concept of Binary Tree can be applied to data sorting and searching. On top of that, we think that it is possible to analyze some statistics by ourselves. Therefore, we focus on three statistics, namely, the Anti-chain, Chain, and Independent Set of a Binary Tree.

The notion of Anti-chain, Chain, and Independent Set has been studied outside computer science for a long time. In this project we study interesting combinatorial problems arisen from this notion under the setting of Binary Trees.

B. Engineering Goals

In a recent (1997) paper which Klazar[1] published, he analyzed many statistics of Rooted Trees, included Chain, Anti-chain, and Independent Sets. He obtained the recursive enumeration and the asymptotic formula of these statistics and compared their sizes. But he didn't acquire the exact formulae. In fact, this paper didn't really consider the situation when extreme values were present, but we could appreciate the beauty of extreme values. As a consequence, we try to analyze the three statistics of a Binary Tree. Then we focus on exact extreme values, graphs, and formulae.

C. Method

Firstly we listed by hand all Anti-chains, Chains , and Independent Sets of a given (small) Binary Tree. After careful observation and analysis we then guessed and proved the (recurrence) formulae for enumerating these statistics on a Binary Tree. We analyze our recurrence formula and then designed algorithms to list all the subsets and algorithms to enumerate effectively the exact numbers of these three statistics of a Binary Tree. Next, we focused on the extreme values (maximum/minimum) and the extreme graphs. We first calculated the extreme values from the formulae developed above. We then discussed whether these extreme values could be obtained, and finally found the extreme graphs for each case, respectively.

D. Bibliography

1. Klazar, Twelve counting of trees, European J. Combin (18) , 1997, 195-210.
2. H.S. Wilf, The Number of Maximal Independent Sets in a Tree, Siam J. Alg. Disc. (7) 125-130.

1.Introduction

Binary tree is one of the most important data structures in computer science. The concept of the binary tree can be applied to data sorting and searching. The analysis of the structure of the binary tree is valuable not only for the theory but also for the application in computer science. In what follows all binary trees are assumed **full**: each node has either no child or two children, a left binary subtree, and a right binary subtree. Let U(n) denote the family of all binary trees having n internal nodes.

Klazar[1] investigated the number of anti-chains, chains, and independent sets of a rooted tree. He obtained the recursive relations, the asymptotic formula and the statistics based on these parameters. But little is known about the exact formula. The purpose of our project is to analyze the three basic structures of a binary tree. In this study we focus on the exact enumeration and the derivation of the closed form of the related formulae.

In Section 2, we investigate the number of anti-chains. We calculate the extreme values and characterize the extreme graphs. Section 3 and Section 4 are devoted to the study of the number of chains and independent sets.

2.Anti-chain

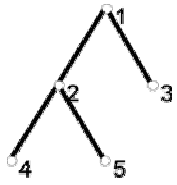
Definition of anti-chain

Recall that a subset of a tree is called an **anti-chain** if no two elements of the subset are comparable in the induced partial ordering.

Example:

All anti-chains in the full binary tree of two internal nodes are

ϕ , {1} , {2} , {3} , {4} , {5} , {2,3} , {3,4} , {3,5} , {4,5} , {3,4,5}



Enumeration of anti-chains

This following algorithm can be applied to enumerate the number of anti-chains of a binary tree recursively.

```
Algorithm: A-CCOUNT(A)
1    if    A=NIL
2          then  return 1
3    Else
4          //let left[A] and right[A] denote the left and right subtrees of
A .
5          return A-CCOUNT(left[A])*A-CCOUNT(right[A])+1
```

The time complexity is O(n).

Lemma 1:

For a binary tree T, we let T' and T'' denote the left and right subtrees of T and a(T) the number of anti-chains contained in T.

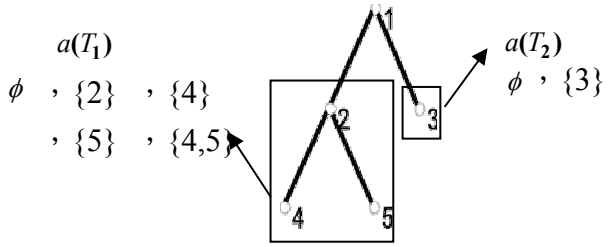
The number of anti-chains of any given binary tree is expressed by the formula:

$$a(T)=a(T')a(T'')+1$$

Proof:

The union of an anti-chain contained in T' and an anti-chain contained in T'' is an anti-chain of T because no node of T' is comparable with any node of T''. On the other hand, other than {root}, every anti-chain A contained in T is a disjoint union $A = (A \cap T') \cup (A \cap T'')$ of anti-chains contained in T' and T'' respectively. Therefore there are a(T')a(T'')+1 anti-chains contained in T.

Example:



ϕ , {1}, {2}, {3}, {4},
{5}, {2,3}, {3,4}, {3,5},
{4,5}, {3,4,5}

The total number of the binary tree of two internal nodes is 11

Lemma 2:

For a node x of a binary tree, let D(x) denote the descendants of x:

D(x): $\{ y \in T \mid y \leq x\}$ together with the edges.

If we interchange the descendants of b and c with depth of b = depth of c+1 assuming a(D(b))>a(D(c)). Then the resulting tree has more anti-chains.

Proof:

This follows from the Lemma 1 and the fact that:

If z<y, then (xy+1)z+1=xyz+z+1<xyz+y+1=(xz+1)y+1.

Maximum value of anti-chains of U(n)

Let **max** a(n) denote the largest number of anti-chains of members of U(n).

Recall that a **binary tree** is said to be complete if the depth of all leaves differs by either 0 or 1, and all leaves are placed toward the left.

Theorem 3:

The maximum value of anti-chains of U(n) will be attained in the case when the tree is complete.

Proof:

Unless a tree is complete, we can always increase the value of a(T) in view of Lemma 2 and the Chebyshev Inequality[3].

Theorem 4:

Recall that a **perfect binary tree** is a complete binary tree with all leaves of the same depth; all internal nodes have two children. Let P(i) be the number of anti-chains in the perfect binary tree of depth i. Then P(1)=2, $P(i)=P(i-1)^2+1$ for i = 2,3,… For a natural number n, let the binary expansion of n+1 be $a_k \ldots a_2 a_1 a_0$ so that $n+\mathbf{1}=\sum_{i=\mathbf{0}}^{k} a_i \mathbf{2}^i$, Then

$$\mathbf{max}\, a(n) = \prod_{i=\mathbf{0}}^{k-\mathbf{1}} (P(i+a_i)) + \sum_{j=\mathbf{0}}^{k-\mathbf{1}} \prod_{i=j}^{k-\mathbf{1}} (P(i+a_i)) + \mathbf{1}$$

If we want to count $\mathbf{max}\, a(T_{\mathbf{13}})$, we have to know that

$$n = \mathbf{13} = \mathbf{2^3 - 1 + 0 \cdot 2^0 + 1 \cdot 2^1 + 1 \cdot 2^2}$$

And then

$$\prod_{i=\mathbf{0}}^{\mathbf{2}} (p(i+a_i) + \sum_{j=\mathbf{0}}^{\mathbf{2}} \prod_{i=\mathbf{0}}^{\mathbf{2}} (p(i+a_i) + \mathbf{1}$$

$$= (P(\mathbf{0}+\mathbf{0}) xP(\mathbf{1}+\mathbf{1}) xP(\mathbf{2}+\mathbf{1})) + (P(\mathbf{0}+\mathbf{0}) xP(\mathbf{1}+\mathbf{1}) xP(\mathbf{2}+\mathbf{1}) + P(\mathbf{1}+\mathbf{1}) xP(\mathbf{2}+\mathbf{1}) + P(\mathbf{2}+\mathbf{1}))$$

$$= \mathbf{2 * 26 * 677 + 2 * 26 * 677 + 26 * 677 + 677 + 1}$$

$$= \mathbf{88688}$$

Proof:

This follows from Lemma 1 and Theorem 3.

There is another recursive algorithm to calculate **max** a(n).

We define a_n=**max** a(n)-1.

```
Algorithm: max_a(n)
1   if n = 0                    //a0=max a(0)-1=2-1=1
2       then return 1
3   i←[log 2n]+1
4   if n≦3*2^(i-2)-1
5       then p←2^(i-2)-1
6   else
7       p←2^(i-1)-1
8   q←n-p-1
9   return (max_a(p)+1)(max_a(q)+1)
```

And then the steps will change into this one:

$$a_{13} = (a_5 + 1)(a_7 + 1)$$

$$a_7 = (a_3 + 1)(a_3 + 1)$$

$$a_5 = (a_3 + 1)(a_1 + 1)$$

$$a_3 = (a_1 + 1)(a_1 + 1)$$

$$a_1 = (a_0 + 1)(a_0 + 1) = 2 * 2 = 4$$

$$a_3 = 25\,,\ a_5 = 130\,,\ a_7 = 676\,,\ a_{13} = 88687$$

$$\max a(13) = a_{13} + 1 = 88688$$

Minimum value of anti-chains of U(n)

Let **min** a(n) denote the smallest number of anti-chains of members of U(n).

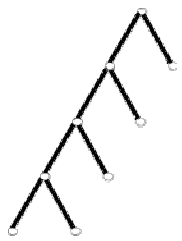
Theorem 5:

The minimum value of anti-chains of U(n) is expressed by the formula:

min a(n)=2^{n+2}-2^n-1

Proof:

It follows from Lemma 2 that the minimum value is attained when the tree takes the form, shown on the right:

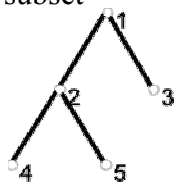


We now prove the formula by mathematical induction. When n=0, there are two anti-chains: empty set and {root}, so that **min** a(0)=2=2^2-2^0-1 and the theorem holds. Now suppose that the formula holds for n=k. When n=k+1, it follows from Lemma 1 that the tree has 2(2^{k+2}-2^k-1)+1=2^{k+3}-2^{k+1}-1 anti-chains. The proof is now complete.

3.Chain

Definition of chain

Recall that a subset of a tree is called a **chain** if any two elements of the subset are comparable in the induced partial ordering.



Example

All of the chains in the binary tree of two internal nodes are
{1,2}'{1,3}'{1,4}'{1,5}'{2,4}'{2,5}'{1,2,4}'{1,2,5}

Enumeration of chains

This algorithm is used to enumerate the number of chains recursively:

```
Algorithm: CHAINCOUNT(A,key)
1    if    A=NIL
2          then  return 0
3    else
4          key ← key*2+1
5          //let left[A] and right[A] denote the left and right subtrees of A .
6          return
key+CHAINCOUNT(left[A],key)+CHAINCOUNT(right[A],key)
```

The time complexity is O(n).

Theorem 6:

For a binary tree T, let c(T) denote the number of chains, h(T) denote the height of T and A_m the number of nodes of depth m ($m \leqq h(T)$). Then c(T) is expressed by the formula:

$$c(T) = \sum_{k=1}^{h(T)} A_k \left(2^{k-1} - 1\right)$$

Proof:

If we add a node to a binary tree T of depth k, the number of chains will be increased by $C_1^{k-1} + C_2^{k-1} + \cdots + C_{k-1}^{k-1} = 2^{k-1} - 1$, since the new chains consist of the new node and some of the nodes which are comparable with the new node in the partial ordering of T.

Maximum value of chains of U(n)

Let **max** c(n) denote the largest number of chains of members of U(n).

According to Theorem 6, we know the maximum value of chains of U(n) is expressed by the formula:

$$\textbf{max } c(n)=2^{n+2}-2n-4$$

The maximum value of chains of Binary Tree of size 5 is calculated as follows:

$$\textbf{max } c(5) = 2^{5+2} - 2 \cdot 5 - 4 = 114$$

Minimum value of chains of U(n)

Let **min** c(n) denote the smallest number of chains of members of U(n).

From Theorem 6, the minimum value of chains of U(n) is expressed by the formula:

min $c(n)=(4^{k+1}+2)/3-2^{k+1}+m*2^{k+2}-2m$, where $n=2^k-1+m$, $k=[\log_2(n+1)]$

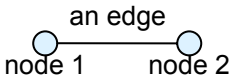
If n is give by 12, we could know that $k = 3$, $m = 5$ by solving the equation:

$12 = 2^k - 1 + m$, $0 \le m < 2^k$. Then the minimum value of chains of Binary Tree of size 12 is calculated as follows:

$$\textbf{min } c(T_{12}) = \frac{4^{3+1}+2}{3} - 2^{3+1} + 5 \times 2^{3+2} - 2 \times 5 = 220$$

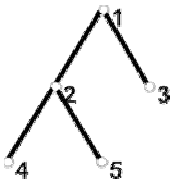
4.Independent set

Definition of independent set



A subset of nodes in a binary tree is said to be **independent** if no two nodes are linked by an edge.

Example:



The number of independent sets of the given Binary Tree (shown at right) equals to fourteen, and

all the objects of independent sets are listed as follows:

ϕ ’{1}’{2}’{3}’{4}’{5}’{1,4}’{1,5}’{2,3}’{3,4}’{3,5}’{4,5}’ {1,4,5}’{3,4,5}

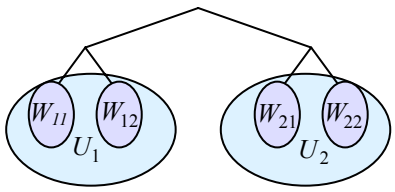
Enumeration of independent sets

The following algorithm enumerates independent sets recursively:

```
Algorithm: I-SCOUNT(A)
1    if    A=NIL
2          then return 1
3    else
4          //let left[A] and right[A] denote the left and right subtrees of
A .
5          return I-SCOUNT(left[A])*I-SCOUNT(right[A])+
6                 I-SCOUNT(left[left[A]])*I-SCOUNT(right[left[A]])*
7                 I-SCOUNT(left[right[A]])*I-SCOUNT(right[right[A]])
```

Lemma 7:

For a binary tree T, let I(T) denote the number of independent sets included in T. Assume that the hierarchy is indicated in the right figure:



Then the number of independent sets of T is expressed by:

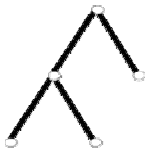
$I(T)=I(U_1)*I(U_2)+$
$I(W_{11})*I(W_{12})*I(W_{21})*I(W_{22})$

Example:

The number of independent sets of the given Binary Tree (shown at right) is calculated as follows:

$$I(T_1) = 2 \cdot 2 + 1 \cdot 1 \cdot 1 \cdot 1 = 5$$

And then the number of independent sets of the second given Binary Tree (shown at right) is calculated as follows:



$$I(T_2) = 5 \cdot 2 + 2 \cdot 2 \cdot 1 \cdot 1 = 14$$

5.Summary

	Anti-chain		Chain		Independent set	
	Max	Min	Max	Min	Max	Min
Exact extreme graph	 Theorem 3	 Theorem 5	 Theorem 6	 Theorem 6	Recursive formulae obtained	Work in progress
Formula of exact extreme value	Theorem 4	Theorem 5	Theorem 6	Theorem 6	Recursive formulae obtained	Work in progress
Formula of enumeration	Lemma 1		Theorem 6		Lemma 7	
Complexity	O(n)		O(n)		O(n)	

6.Conclusions

- The number of anti-chains, chains, and independent sets of the binary tree are formulated.
- Closed form solutions of anti-chains and chains are derived, while the enumeration algorithm for the independent sets is proposed.
- Applications include sorting, searching, and code word generation in cryptography.

7.References

[1] M. Klazar (1997), Twelve countings of rooted plane trees, European J. Combin. (18) , no.2, 195-210.

[2] Herbert S. Wilf (1986), The number of maximal independent sets in a tree, Siam J. Alg. Disc. (7), 125-130.

[3] G.H. Hardy, J.E. Littlewood, and G. Polya, Inequalities, Cambridge U. Press, 1952.