



搜尋演算法

Searching Algorithm

搜尋Searching

- 搜尋(Searching)就是在一堆資料中依照某個「**鍵值**」尋找出所需求的資料。
 - 例如：學生資料表中，利用「學號」進行搜尋，可以利用該鍵值搜尋到對應的學生姓名、班級等。
 - 911632 220 王大明
- 常見的兩種搜尋演算法：
 - 循序搜尋
 - 二元搜尋

循序搜尋演算法

- 循序搜尋Linear Search：
 - 作法：不需任何前製作業，只要在一群資料中，**從頭搜尋到尾**直到找到資料為止。
 - 特色：使用**迴圈**一個一個找資料存放之位置，當搜尋的資料量很大時，**效率很差**。

循序搜尋範例

要找的值

1

value

	<i>i</i>				
data	<u>34</u>	12	5	66	1
	[0]	[1]	[2]	[3]	[4]

data	34	<u>12</u>	5	66	1
	[0]	[1]	[2]	[3]	[4]

data	34	12	<u>5</u>	66	1
	[0]	[1]	[2]	[3]	[4]

data	34	12	5	<u>66</u>	1
	[0]	[1]	[2]	[3]	[4]

data	34	12	5	66	<u>1</u>
	[0]	[1]	[2]	[3]	[4]

二元搜尋法

- Let's play a game !
 - 名畫神偷「Art Thief」
 - 輸入網址：<http://goo.gl/069Vk>

PLAYONLINE close window

QUIT

START

Like this game?

SHARE

Click the SHARE button above to get the FREE code to add this game to your blog or website!

Facebook 上等你來找

Kewlbox

4,377 人覺得 Kewlbox 讚

Facebook 社群外網元件

CHAT LOGIN

You must be logged into your Kewlbox account to take part in chat.

Email:

Password:

LOGIN

Don't have a FREE Kewlbox account? Click here to register.

If you have any problems with kewlbox Chat, please click here to tell us.

Get Adobe FLASH PLAYER The online version of "Art Thief" requires the Adobe Flash player.

二元搜尋法

- 兩人一組，完成名畫神偷學習單。

猜數字 - 名畫神偷(Art Thief)

『你是密碼高手嗎?美術館已經佈下了天羅地網,但是高明如你,能否快速的破解寶庫的密碼,順利地取得寶物?』

班級: _____ 座號: _____

姓名: _____

- 這是款屬於猜數字的遊戲。在密碼破解的機器上輸入一個三位數字，在點一下 Enter，機器會告訴你這個數字太高(Too High)，還是太低(Too Low)，然後你再繼續猜數字，直到猜對。遊戲結果將會顯示你猜的次數和時間，請同學們玩 2 次猜數字遊戲，並將猜的資訊寫在本紀錄單內。
- 紀錄方式：兩人為一組，一人猜數字，一人在旁紀錄。
- 小叮嚀：每人玩兩次，請儘量以有系統的方式來猜，並想想它是否有規律性。

猜數字 - 名畫神偷

範例:	[1] 500 H	← [1] 表示第一次猜, 500 表示猜的數字, H 表示太高			
第一次玩	[1] 500 H	[6] 190 H	[11]	[16]	[21]
第二次玩	[2] 300 H	[7] 185 L	[12]	[17]	[22]
	[3] 150 L	[8] 188	[13]	[18]	[23]
	[4] 200 H	[9]	[14]	[19]	[24]
	[5] 180 L	[10]	[15]	[20]	[25]

結果是：__188__，共猜了__8__次。

第一次猜	[1]	[6]	[11]	[16]	[21]
	[2]	[7]	[12]	[17]	[22]
	[3]	[8]	[13]	[18]	[23]
	[4]	[9]	[14]	[19]	[24]
	[5]	[10]	[15]	[20]	[25]

結果是：_____，共猜了_____次。

(若超過 25 次，表示你陣亡了)

第二次猜	[1]	[6]	[11]	[16]	[21]
	[2]	[7]	[12]	[17]	[22]
	[3]	[8]	[13]	[18]	[23]
	[4]	[9]	[14]	[19]	[24]
	[5]	[10]	[15]	[20]	[25]

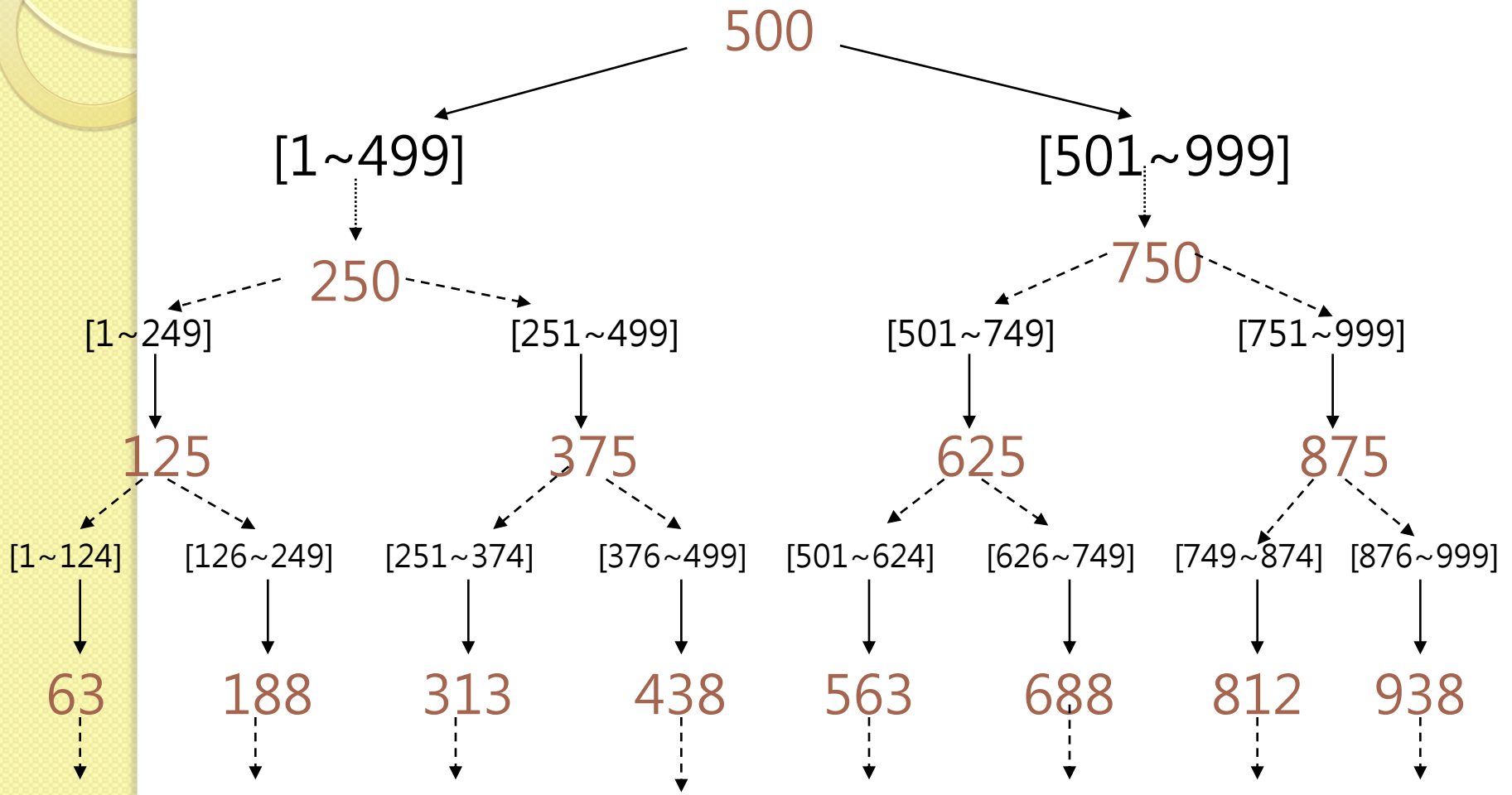
結果是：_____，共猜了_____次。

(若超過 25 次，表示你陣亡了)

二元搜尋法

- 名畫神偷的省思：
 - 若每次猜都可以**消去一半的數字**，是不是可以讓範圍越來越小？
 - 想想看，有沒有什麼方法，可以一次消去一半的數字。想想看我們要用什麼策略？再重新玩玩看吧！

二元搜尋法



以此類推...

二元搜尋演算法

- 二元搜尋Binary Search：
 - 作法：先確保資料**已經排序**完成，接著對一群排序過的資料，使用**二分法**的方式做搜尋。
 - 特色：每一次搜尋可以減少一半的搜尋資料量，當資料量大時可以展現其效率性。
 - 搜尋次數：若有**N筆**資料，搜尋次數為 **$\log_2 N$** 。

二元搜尋法的程式碼：Part 1

```
int main(int argc, char *argv[])
{
    const int num_of_array = 5;
    int a[num_of_array] = {1, 5, 12, 34, 66};
    int value;

    cout << "Which number do you want to search ?";
    cin >> value;
}
```

首先詢問使用者想要搜尋的數字為多少，並且該數字由使用者自行輸入。

二元搜尋法的程式碼：Part 2

```
int low = 0 , high = num_of_array , mid;  
bool is_find = false;
```

```
while( low <= high )
```

```
{
```

```
    mid = ( low + high ) / 2;
```

從陣列最中間數值開始進行搜尋

```
    if( a[mid] > value )
```

```
        high = mid - 1;
```

若最中間數值大於搜尋值，則調整high值為mid值減1

```
    else if( a[mid] < value)
```

```
        low = mid+1;
```

若最中間數值小於搜尋值，則調整low值為mid值加1

```
    else
```

```
    {
```

```
        is_find = true;
```

```
        break;
```

```
    }
```

若搜尋值包含於陣列中，則將bool變數設定為true，並且透過break指令離開while迴圈

```
}
```

```
if( is_find == true )
```

```
    cout << "Find!" << endl;
```

```
else
```

```
    cout << "Not Find!" << endl;
```

利用while迴圈搜尋陣列中是否包含使用者輸入的數字

二元搜尋範例一：找得到的

data	<u>1</u>	<u>5</u>	<u>12</u>	<u>34</u>	<u>66</u>
	[0]	[1]	[2]	[3]	[4]

low

mid

high

要找的值

high = 5
low = 0
mid = 2

34

value

data	1	5	12	<u>34</u>	<u>66</u>
	[0]	[1]	[2]	[3]	[4]

low

mid

high

high = 5
low = 3
mid = 4

data	1	5	12	<u>34</u>	66
	[0]	[1]	[2]	[3]	[4]

low mid high

high = 3
low = 3
mid = 3

因為找到所以break離開
While迴圈

二元搜尋範例二：找不到的

data	^{low} <u>1</u>	^{mid} 5	^{high} <u>12</u>	34	66	^{high} high = 5 low = 0 mid = 2	要找的值 <u>3</u>
	[0]	[1]	[2]	[3]	[4]		value
data	^{low} <u>1</u>	^{mid} 5	^{high} 12	34	66	high = 1 low = 0 mid = 0	
	[0]	[1]	[2]	[3]	[4]		
data	^{low} 1	^{mid} <u>5</u>	^{high} 12	34	66	high = 1 low = 1 mid = 1	
	[0]	[1]	[2]	[3]	[4]		
data	^{high} 1	^{mid} 5	^{low} 12	34	66	high = 0 low = 1 mid = 0	
	[0]	[1]	[2]	[3]	[4]		

跳離while迴圈