NCKU Programming Contest Training Course
Course 4
2013/01/21

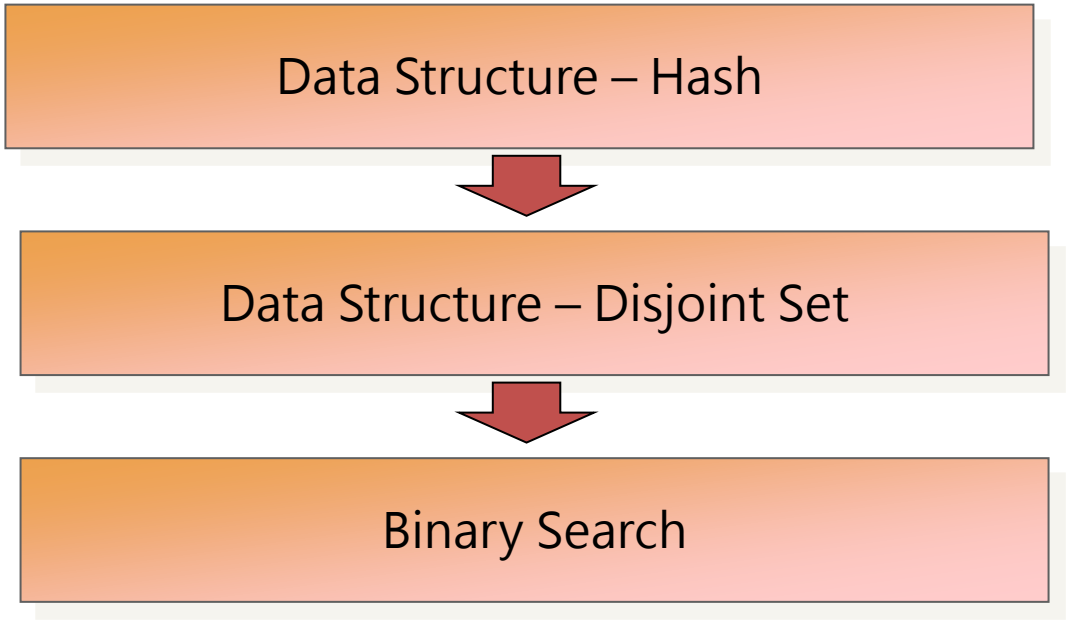
Pin-Chieh Huang (free999)

Pinchieh.huang@gmail.com

Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan

made by electron & free999

Outline



made by electron & free999

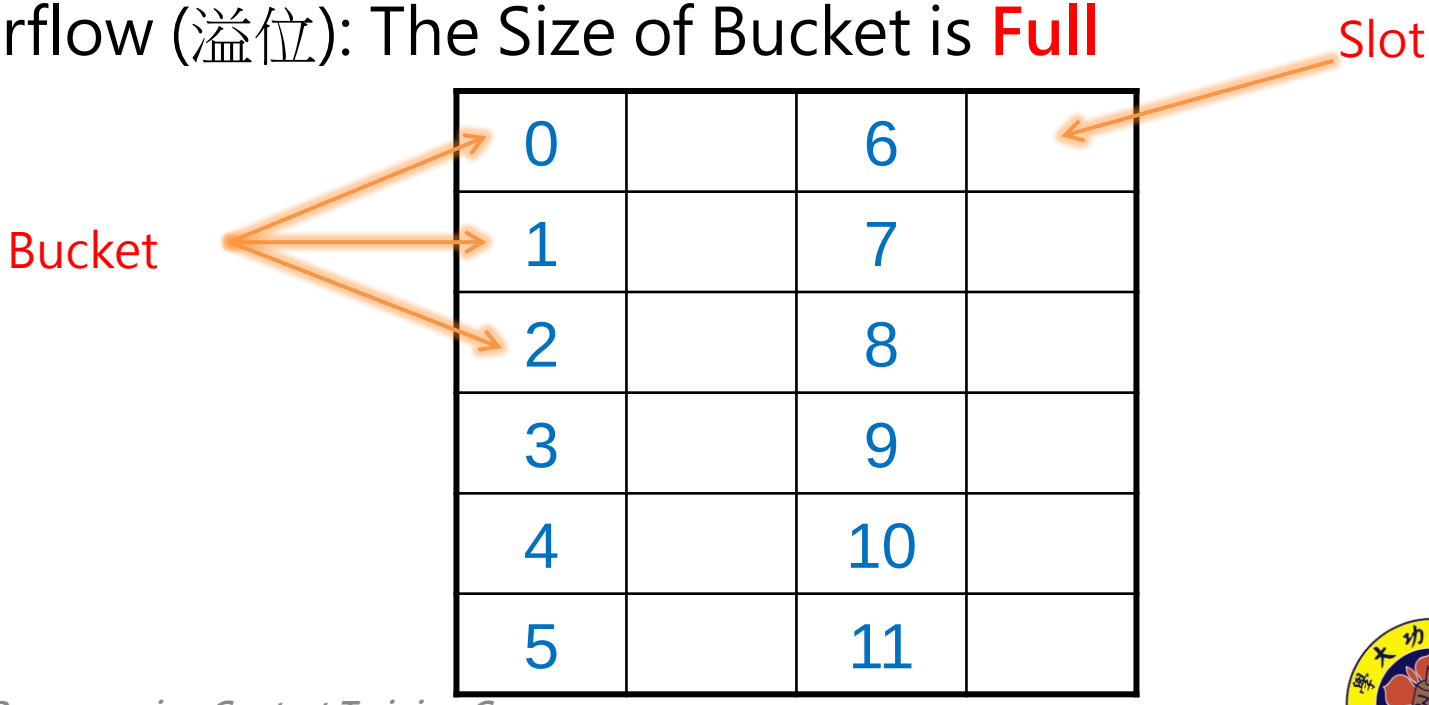
Hash

- Hash
  - A Static Table with Fixed Size
  - hash represent the procedure that using a function f(x) to transform a key word x into a new address

made by electron & free999

Hash

- Bucket (桶):The Size of f(x) for the Hash Table
- Slot (槽): The Size of each Bucket
- Collision (碰撞): x_1 != x_2, while $f(x_1)=f(x_2)$
- Overflow (溢位): The Size of Bucket is **Full**



made by electron & free999

Hash

- Execution Time of Hash Technique
  - Depend on the Defined Hash Function
  - A **hash function** is any **well-defined procedure** or **mathematical function** which converts a large, possibly variable-sized amount of data into a small datum, usually a single integer that may serve as an index to an array. The values returned by a hash function are called **hash values**, **hash codes**, **hash sums**, or simply **hashes**.

Hash Function 1

- Example of Hash Function (Mid-Square)
  - 將識別字轉成一個數值,再求它的平方值,然後再取其中間的幾個位數作為桶的位址
  - 由於平方值的中間幾個位數通常和識別字的所有字元有關,所以會有較高的機率產生不同的位址
    - EX.
    - CFGA→3671(轉成數值)→13476241(取平方值)→762(取中間三位數)

Hash Function 2

- Example of Hash Function (Folding)

- Shift folding
- Ex.

  key word X

  X=167329421598 12

  pick up for each three characters

P1	167
P2	329
P3	421
P4	598
P5	12

summation 1527

Hash Function 3

- Example of Hash Function (Boundary at the Folding)

P1	167
P2	~~329~~ ←reverse 923
P3	421
P4	~~598~~ ←reverse 895
P5	12

summation 2418

Hash Function 4

- Example of Hash Function (Division)
  - Divide the key word by a value M and use the modulo value as the address

$$f(x) = x\%M$$

M是除數,求出的餘數介於0至M-1之間.

Hash Function 4

- push **345, 728, 251, 490, 15** into 12 buckets, M is 12, **f(x)=x%12**

  f(345)=9

  f(728)=8

  f(251)=11

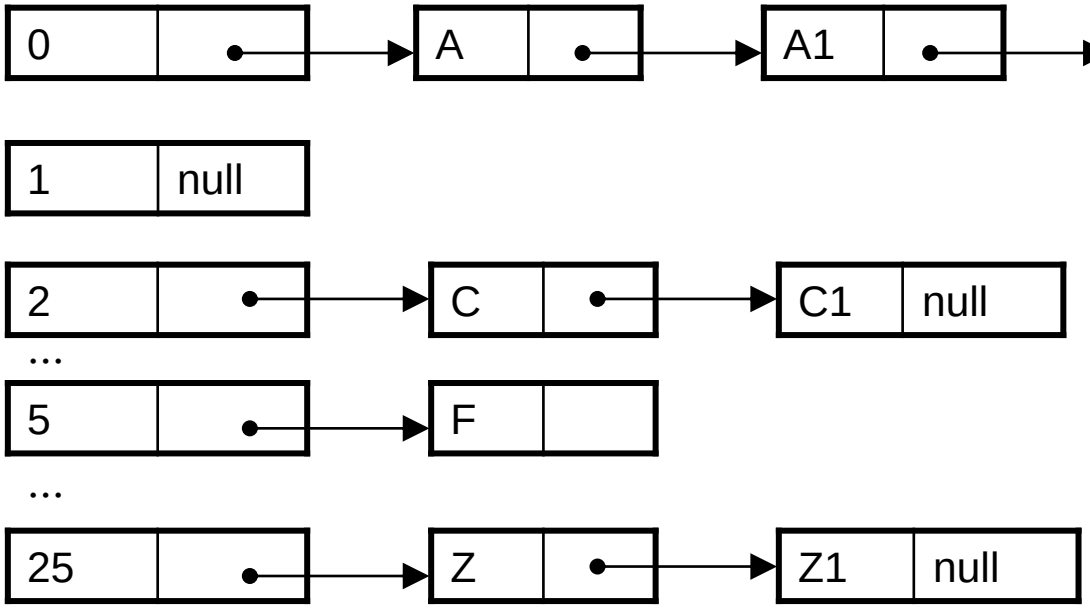
  f(490)=10

  f(15)=3

	X		X
0		6	
1		7	
2		8	728
3	15	9	345
4		10	490
5		11	251

Overflow Handling

- Overflow Handling
  - the size of bucket is overflow due to multiple key words that access the same bucket
  - use the link list to store multiple key words (vector is powerful)



key words:
A, A1, A2, C, C1, F, Z, Z1

made by electron & free999

Example - 1

Snow Snowflakes (POJ 3349)

Problem Description

You may have heard that no two snowflakes are alike. Your task is to write a program to determine whether this is really true. Your program will read information about a collection of snowflakes, and search for a pair that may be identical. Each snowflake has six arms. For each snowflake, your program will be provided with a measurement of the length of each of the six arms. Any pair of snowflakes which have the same lengths of corresponding arms should be flagged by your program as possibly identical.

Example - 1

Snow Snowflakes (POJ 3349)

I/O Description

Input

The first line of input will contain a single integer n, $0 < n \leq 100000$, the number of snowflakes to follow. This will be followed by n lines, each describing a snowflake. Each snowflake will be described by a line containing six integers (each integer is at least 0 and less than 10000000), the lengths of the arms of the snow ake. The lengths of the arms will be given in order around the snowflake (either clockwise or counterclockwise), but they may begin with any of the six arms. For example, the same snowflake could be described as 1 2 3 4 5 6 or 4 3 2 1 6 5.

Output

If all of the snowflakes are distinct, your program should print the message:
No two snowflakes are alike.
If there is a pair of possibly identical snow akes, your program should print the message:
Twin snowflakes found.

Example - 1

Snow Snowflakes (POJ 3349)

Sample I/O

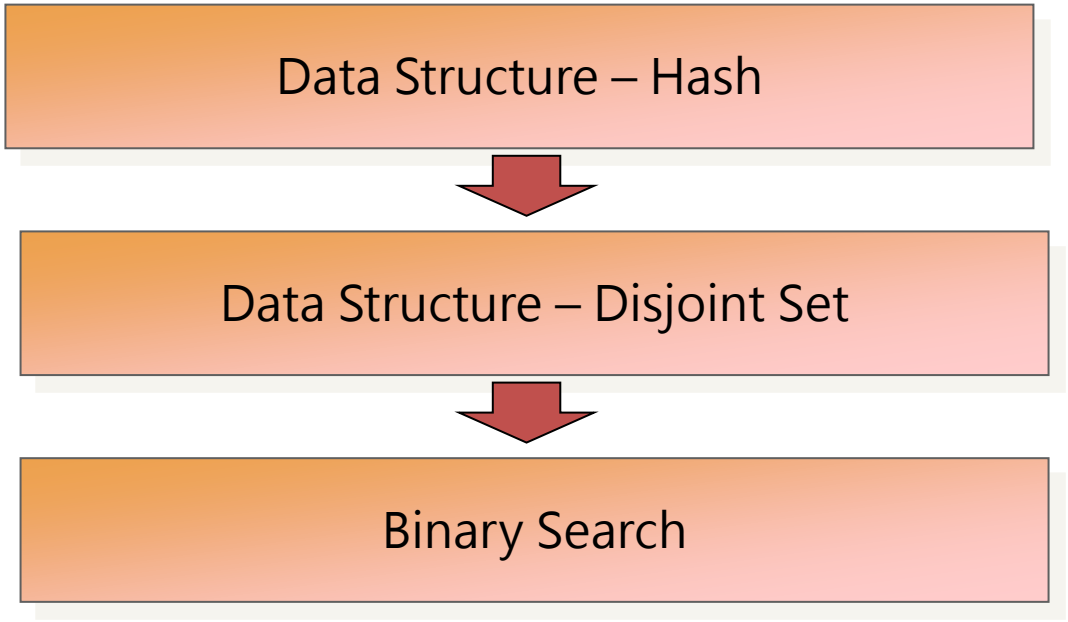
Input

2
1 2 3 4 5 6
4 3 2 1 6 5

Output

Twin snowflakes found.

Outline



made by electron & free999

Disjoint Set

- Disjoint Set
  - We have a collection of disjoint sets of elements. Each set is identified by a representative element. We want to perform union operations, and tell which set something is in. This is useful in a minimum spanning tree algorithm and many other applications. Formally, we have the following operations.
- **Basic Operation**
  - MAKE-SET(x) ：Create new set {x} with representative x.
  - UNION(x,y)：x and y are elements of two sets. Remove these sets and add their union. Choose a representative for it.
  - FIND-SET(x)：return the representative of the set containing x.

Disjoint Set

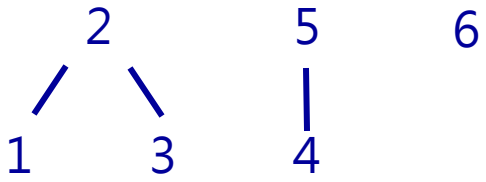
- Example

MAKE-SET(1)		{1}
MAKE-SET(2)		{2}
MAKE-SET(3)		{3}
MAKE-SET(4)		{4}
FIND(3)	(returns 3)	
FIND(2)	(returns 2)	
UNION(1,2)	(representative 1, say)	{1,2}
FIND(2)	(returns 1)	
FIND(1)	(returns 1)	
UNION(3,4)	(representative 4, say)	{3,4}
FIND(4)	(returns 4)	
FIND(3)	(returns 4)	
UNION(1,3)	(representative 4, say)	{1,2,3,4}
FIND(2)	(returns 4)	
FIND(1)	(returns 4)	
FIND(4)	(returns 4)	
FIND(3)	(returns 4)	

Disjoint Set

- Forest Implementation

Here we represent each set as a tree, and the representative is the root . For example, the following forest represents the set {1,2,3}, {4,5}, {6} :



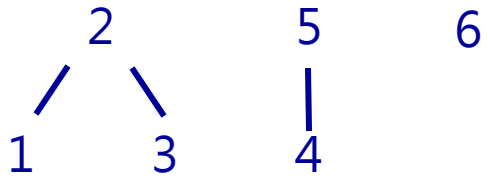
Implementation

MAKE-SET(x)	Create a tree
FIND-SET(x)	Return the root
UNION(x,y)	Combine two trees

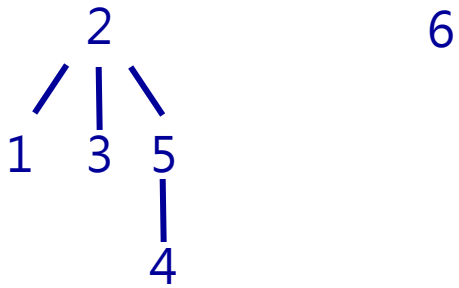
made by electron & free999

Disjoint Set

- Forest Implementation



Thus we would get the following form UNION(1,4)



This representation does not improve the running time in the worst case over the linked list representation.

made by electron & free999

Disjoint Set

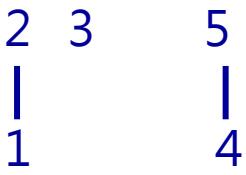
- Path Compaction and Rank
  - These are refinements of the forest representation which make it significantly faster
    - FIND-SET: Do path compression
    - UNION: Use ranks
  - "Path compression" means that when we do FIND-SET(X), we make all nodes encountered point directly to the representative element for x. Initially, all elements have rank 0. The ranks of representative elements are updated so that if two sets with representatives of the same rank are unioned, then the new representative is incremented by one.

Disjoint Set

- Example MAKE-SET(1) … MAKE-SET(6)

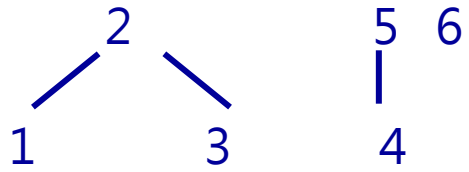
1 2 3 4 5 6 RANKS = 0

UNION(1,2) UNION(4,5)



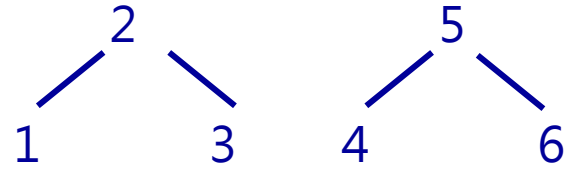
RANK(2)=1
RANK(5)=1

UNION(1,3)



RANK(2)=1
RANK(5)=1

UNION(5,6)



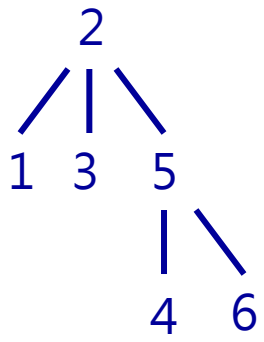
RANK(2)=1
RANK(5)=1

made by electron & free999

Disjoint Set

- Example

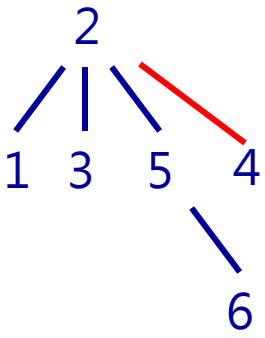
UNION(4,3)



RANK(2)=2

RANK(5)=1

FIND(4) (path compression)



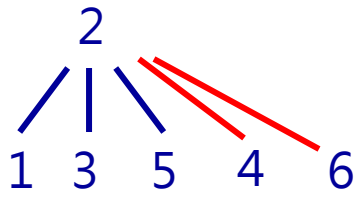
made by electron & free999

Disjoint Set

- Example

FIND(3) no change

FIND(6) (path compression)



made by electron & free999

Disjoint Set

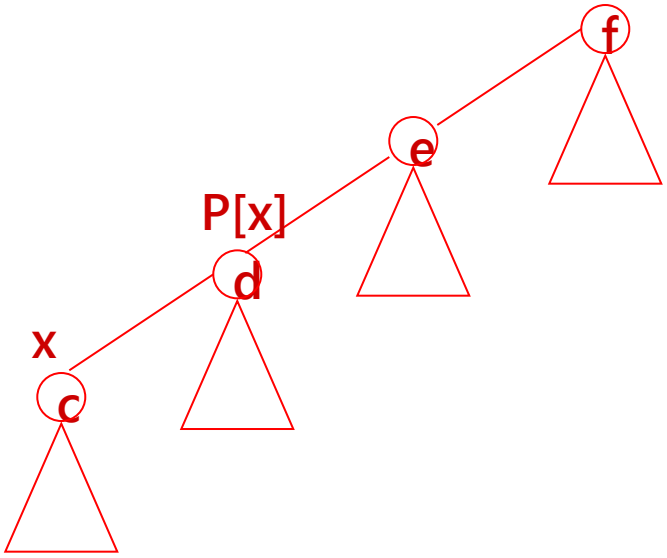
- MakeSet and Union

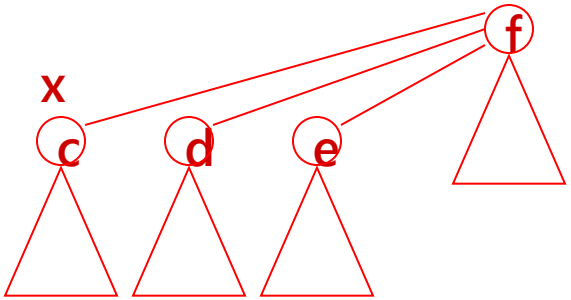
```
void MakeSet(int x)
{
   p[x]    = x;
   rank[x] = 0;
}

void Union(int x,int y)
{
    Link(FindSet(x),FindSet(y));
}
```

Disjoint Set

- FindSet

```
int FindSet(int x)
{
    if(x!=p[x])
        p[x] = FindSet(p[x]);
    return p[x];
}
```



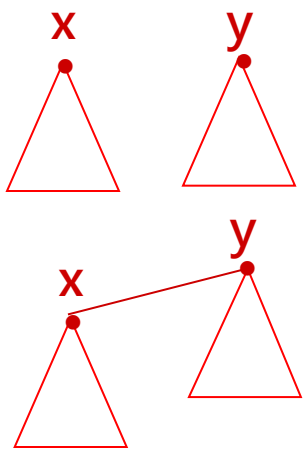


made by electron & free999

Disjoint Set

- Link

```
void Link(int x,int y)
{
    if(rank[x]>rank[y])
        p[y] = x;
    else
    {
        p[x] = y;
        if(rank[x]==rank[y])
            rank[y]++;
    }
}
```



made by electron & free999

Example - 2

ZJ2- d064 NCPC 2005 Problem C – Computer Connectivity

Problem Description

Consider a set of N computers numbered from 1 to N, and a set S of M computer pairs, where each pair (i,j) in S indicates that computers i and j are connected. The connectivity rule says that if computers i and j are connected, and computers j and k are connected, then computers i and k are connected, too, no matter whether (i,k) or (k,i) is in S or not. Based on S and the connectivity rule, the set of N computers can be divided into a number of groups such that for any two computers, they are in the same group if and only if they are connected. Note that if a computer is not connected to any other one, itself forms a group. A group is said to be largest if the number of computers in it is maximum among all groups. The problem asks to count how many computers there are in a largest group.

Example - 2

ZJ2- d064 NCPC 2005 Problem C – Computer Connectivity

I/O Description

The first line of the input file contains the number of test cases. For each test case, the first line consists of N ($1 \le N \le 30000$) and M ($1 \le M \le 100000$), where N is the number of computers and M is the number of computer pairs in S. Each of the following M lines consists of two integers i and j ($1 \le i \le N$, $1 \le j \le N$, $i = j$) indicating that (i,j) is in S. Note that there could be repetitions among the pairs in S.

Example - 2

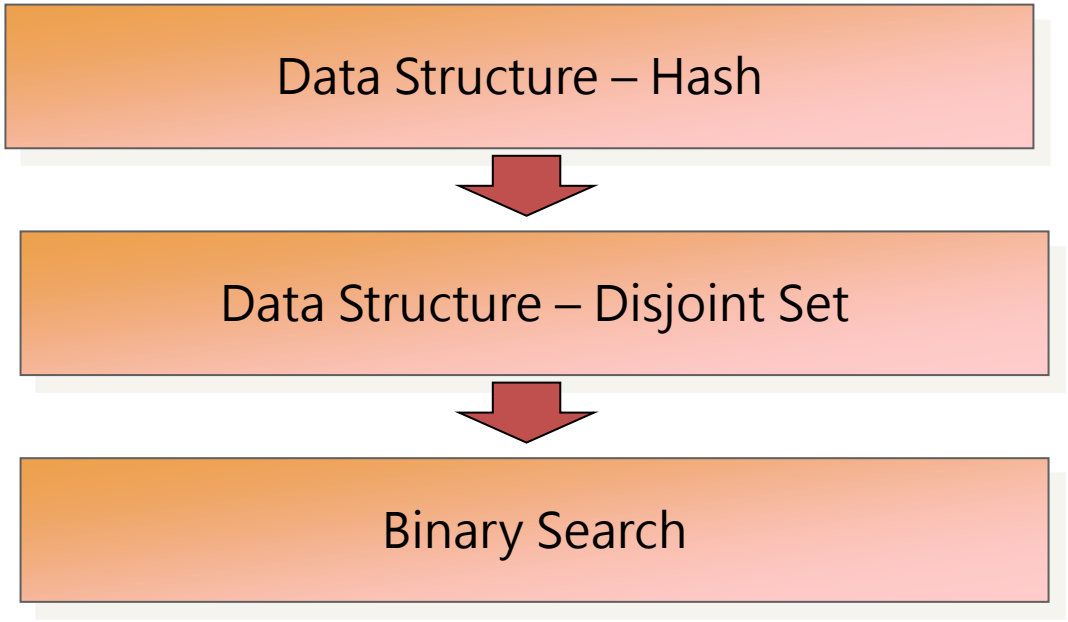
ZJ2- d064 NCPC 2005 Problem C – Computer Connectivity

Sample Input
1
3 4
1 2
3 2
2 3
1 2

Sample Input
3

made by electron & free999

Outline



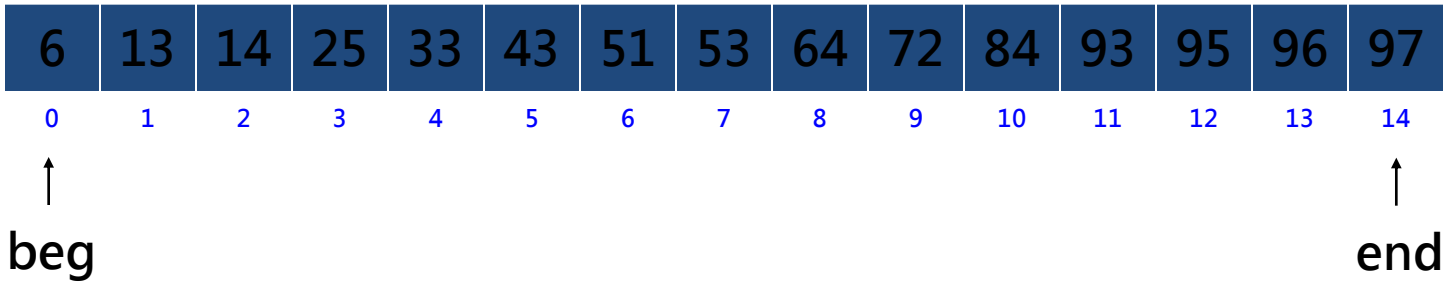
made by electron & free999

Binary Search

- Binary Search
  - Search the Index of an Given Value in a Sorted Array
- Basic Operation
  - Find MAX or Find MIN
  - Set the "canbe" Function
  - Set the while loop (beg, end, mid)
- Types
  - integer
  - double

Binary Search

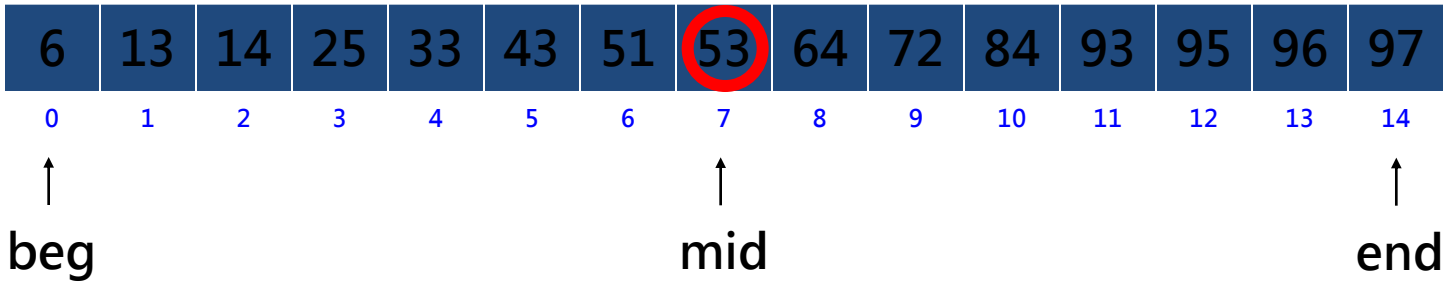
- Example
  - search the value "33"



made by electron & free999

Binary Search

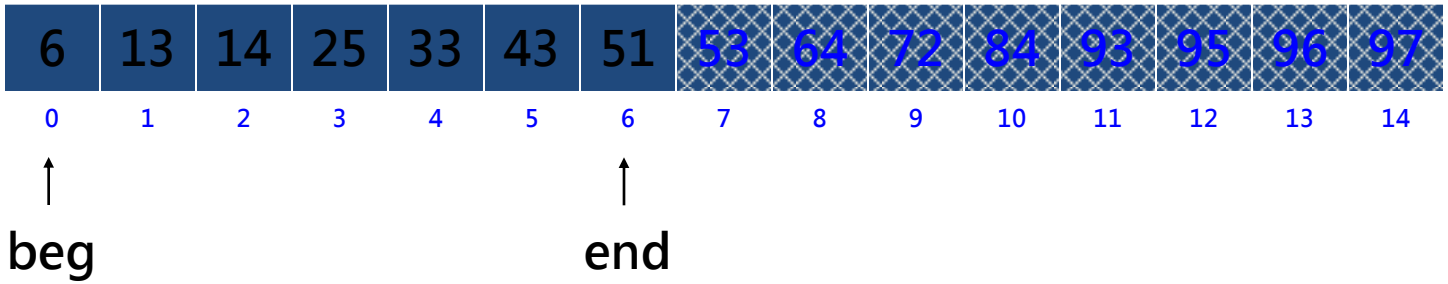
- Example
  - search the value "33"



made by electron & free999

Binary Search

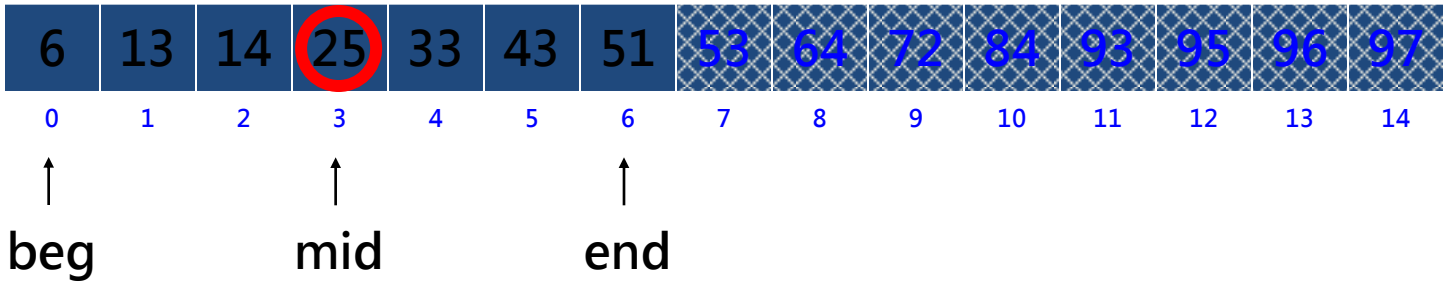
- Example
  - search the value "33"



made by electron & free999

Binary Search

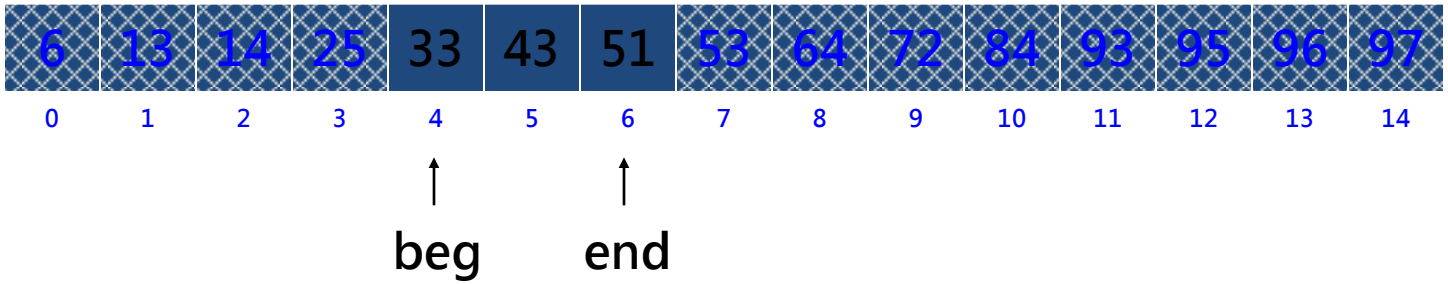
- Example
  - search the value "33"



made by electron & free999

Binary Search

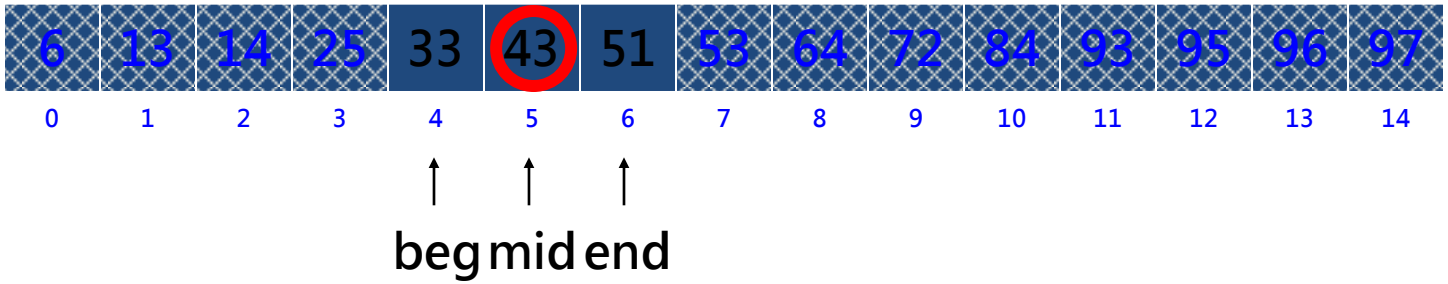
- Example
  - search the value "33"



made by electron & free999

Binary Search

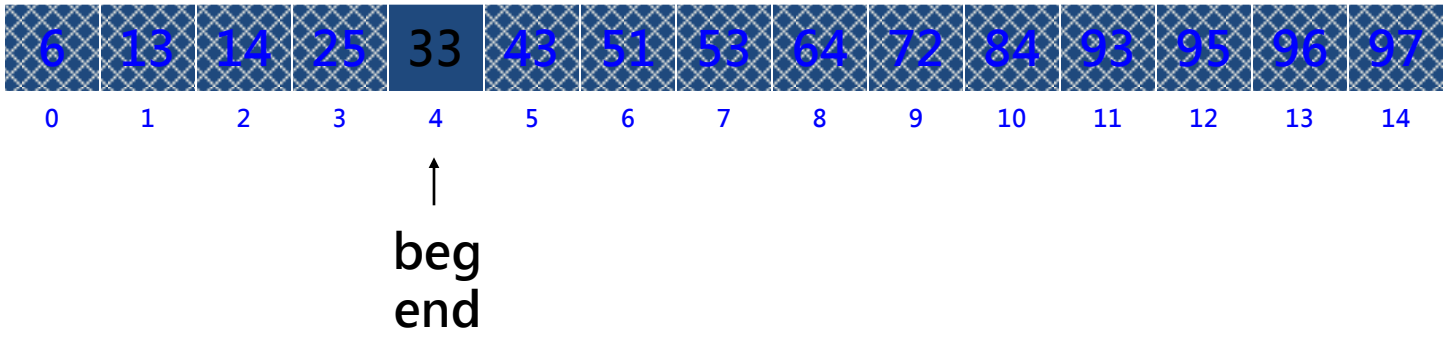
- Example
  - search the value "33"



made by electron & free999

Binary Search

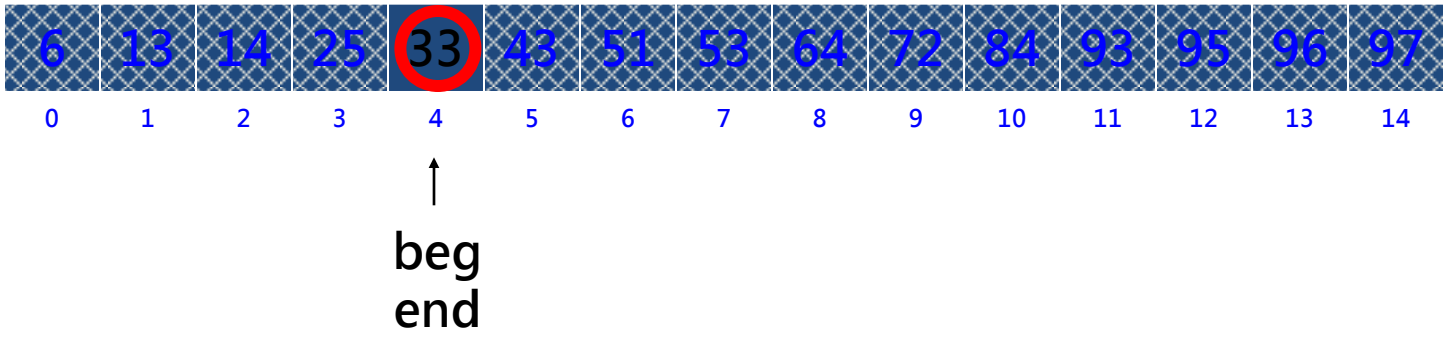
- Example
  - search the value "33"



made by electron & free999

Binary Search

- Example
  - search the value "33"



made by electron & free999

Binary Search

- Find Min or Max (integer)
- Find Min or Max (double)

```
int beg=0, mid, end=N;
while(beg<end)
{
    mid = (beg+end)/2;
    if( canbe(mid) ) beg = mid+1;
    else end = mid;
}
```

Binary Search

- Example
  - a sorted array: 1, 2, 4, 7, 9, 11, 15, 17, 19, 31, 40
  - Find the Minimum Value that > 13

Example - 3

POJ 3273 Expanse

Problem Description

Farmer John is an astounding accounting wizard and has realized he might run out of money to run the farm. He has already calculated and recorded the exact amount of money ($1 \le money_i \le 10{,}000$) that he will need to spend each day over the next N ($1 \le N \le 100{,}000$) days.

FJ wants to create a budget for a sequential set of exactly M ($1 \le M \le N$) fiscal periods called "fajomonths". Each of these fajomonths contains a set of 1 or more consecutive days. Every day is contained in exactly one fajomonth.

FJ's goal is to arrange the fajomonths so as to minimize the expenses of the fajomonth with the highest spending and thus determine his monthly spending limit.

Example - 3

POJ 3273 Expanse

I/O Description

Input

Line 1: Two space-separated integers: *N* and *M*
Lines 2..*N*+1: Line *i*+1 contains the number of dollars Farmer John spends on the *i*th day

Output

Line 1: The smallest possible monthly limit Farmer John can afford to live with.

made by electron & free999

Example - 3

POJ 3273 Expanse

Sample I/O

Sample Input

```
7 5
100
400
300
100
500
101
400
```

Sample Output

```
500
```


made by electron & free999

Example - 4

River

Problem Description

Every year the cows hold an event featuring a peculiar version of hopscotch that involves carefully jumping from rock to rock in a river. The excitement takes place on a long, straight river with a rock at the start and another rock at the end, L units away from the start ($1 \le L \le 1{,}000{,}000{,}000$). Along the river between the starting and ending rocks, N ($0 \le N \le 50{,}000$) more rocks appear, each at an integral distance D_i from the start ($0 < D_i < L$).

To play the game, each cow in turn starts at the starting rock and tries to reach the finish at the ending rock, jumping only from rock to rock. Of course, less agile cows never make it to the final rock, ending up instead in the river.

Farmer John is proud of his cows and watches this event each year. But as time goes by, he tires of watching the timid cows of the other farmers limp across the short distances between rocks placed too closely together. He plans to remove several rocks in order to increase the shortest distance a cow will have to jump to reach the end. He knows he cannot remove the starting and ending rocks, but he calculates that he has enough resources to remove up to M rocks ($0 \le M \le N$). FJ wants to know exactly how much he can increase the shortest distance **before** he starts removing the rocks. Help Farmer John determine the greatest possible shortest distance a cow has to jump after removing the optimal set of M rocks.

Example - 4

POJ 3258 River

IO Description

Input

Line 1: Three space-separated integers: *L*, *N*, and *M*
Lines 2..*N*+1: Each line contains a single integer indicating how far some rock is away from the starting rock. No two rocks share the same position.

Output

Line 1: A single integer that is the maximum of the shortest distance a cow has to jump after removing *M* rocks

made by electron & free999

Example - 4

POJ 3258 River

IO Description

Sample Input

```
25 5 2
2
14
11
21
17
```

Sample Output

```
4
```


made by electron & free999

Homework 4

Total 42 Problems

- uva (12)
  - 311, 834, 846, 10020, 10050, 10098, 10249, 10583, 10608, 10678, 11417, 11489
- zoj2 (10)
  - d003, d004, d006, d008, d009, d011, d017, d030, d062, d077
- pku (20)
  - 1035, 1328, 1703, 1840, 1905, 1936, 2002, 2109, 2151, 2388, 2492, 2503, 2586, 3080, 3122, 3349, 3258, 3273, 3274, 3617

Thank for Your Attention

made by electron & free999