Introduction to Algorithms and Data Structures

Lesson 8: Data Structure (2) Operations on linked lists

Professor Ryuhei Uehara,
School of Information Science, JAIST, Japan.
uehara@jaist.ac.jp
http://www.jaist.ac.jp/~uehara

Example of
Data structures × Algorithms

- Usually, we can choose some data structure, e.g.,
  - array
  - linked list

  for the implementation of the same algorithm.
- Efficiency depends on "data structure" vs "basic operations" you will use on the data.
  - When we "add" and "remove" data, linked list is much better than array, and tree structure is much better than linked list (I'll explain, say, at the last lesson?)
- We will show some simple examples

Sequential search by linked list

- Find x in the linked list from the top of linked list
  - It contains x ➔ address of the record
  - It doesn't contain x ➔ NULL

```
typedef struct{
  double data;
  struct list_t *next;
} list_t;
p = head;
while(p != NULL && p->data != x)
   p = p->next;
return p;
```

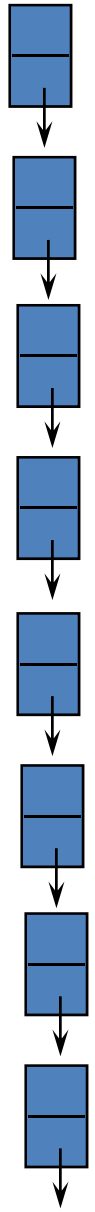
Satisfied?

YES

p==NULL or p->data == x it exits

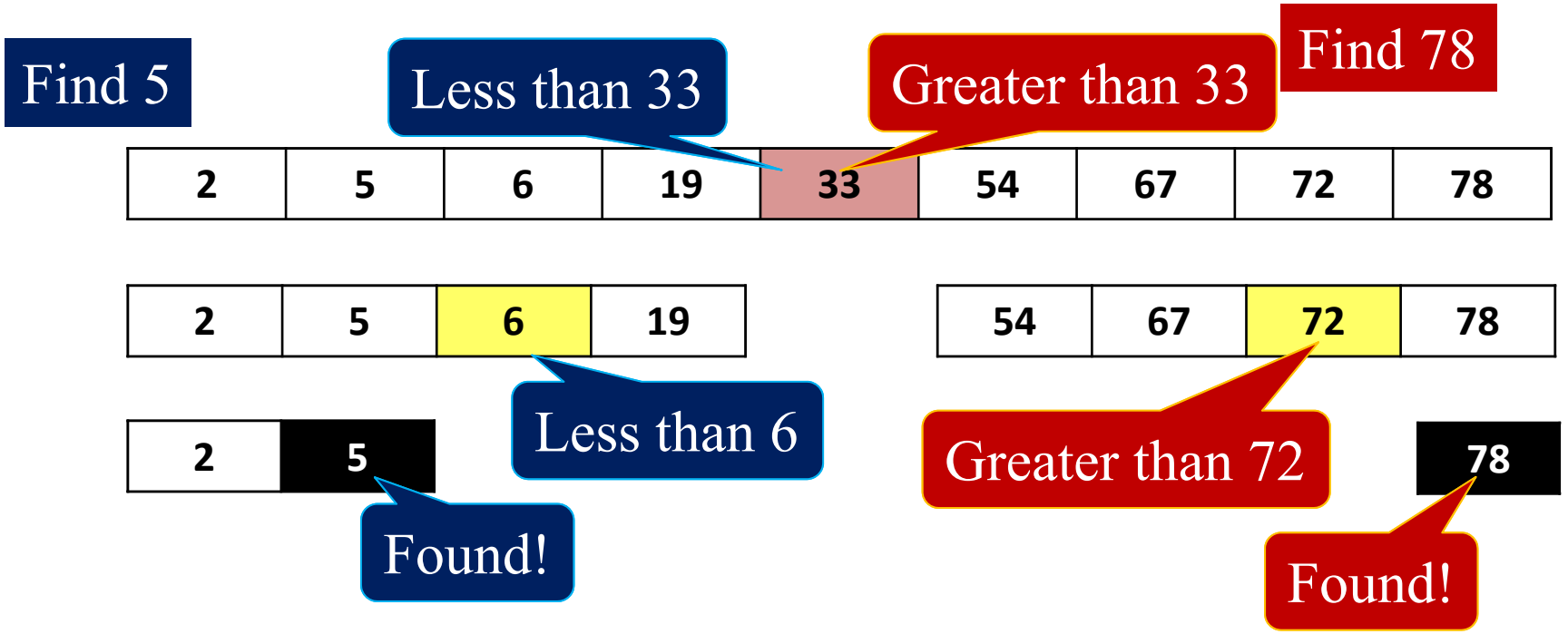
Is this correct?

YES



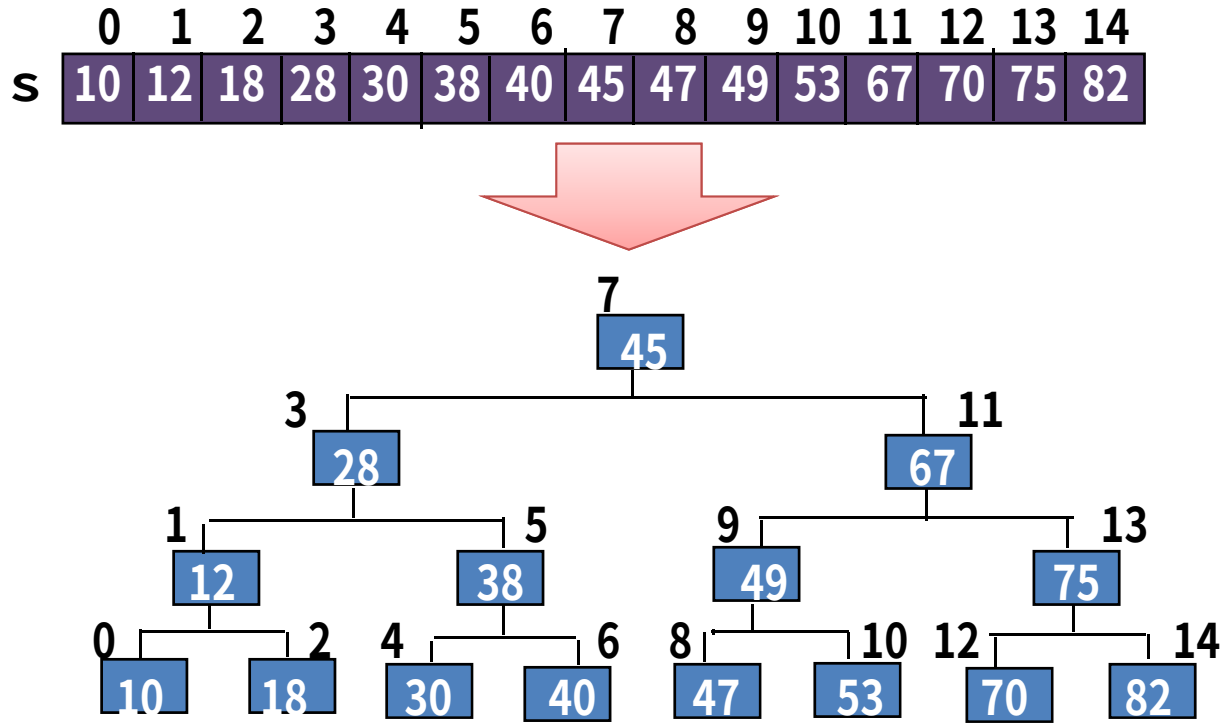
Binary search method

- Search, divide into halves, and repeat to find



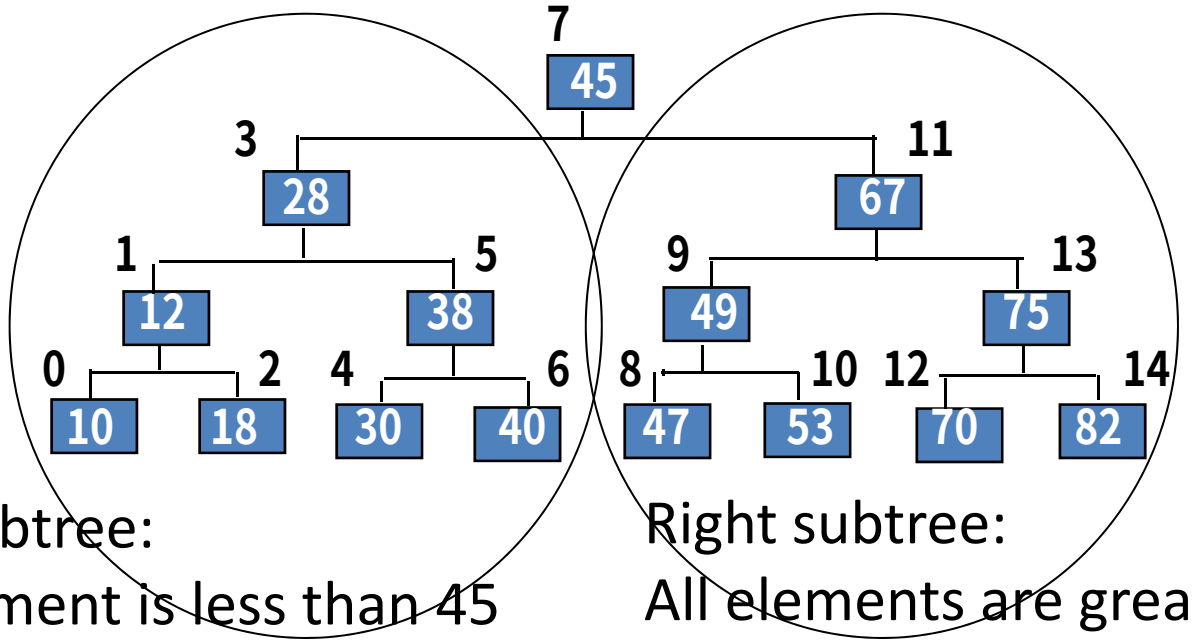
  - Key issue: Divide at the center point.

Binary search tree: data structure of binary search



- When data size is fixed, we can compute the central positions beforehand

Property of binary search tree



Left subtree:

All element is less than 45

Right subtree:

All elements are greater than 45

- In general, for a node n,
  - All elements in right subtree are greater than (or equal to) n
  - All elements in left subtree are less than (or equal to) n

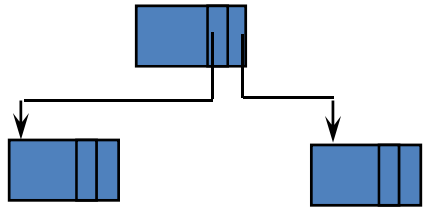
Search in binary search tree

```
BSTnode *root, *v;
x=/*some value*/;
v = root;
while( v ){
  if(v->data == x)
    break;
  if(v->data > x)
    v = v->lson;
  else
    v = v->rson;
}
return v;
```

Left if small

Right if large

```
typedef struct{
  int data;
  struct BSTnode
    *lson, *rson;
}BSTnode;
```



Each record has two pointers to left child and right child.