Heap

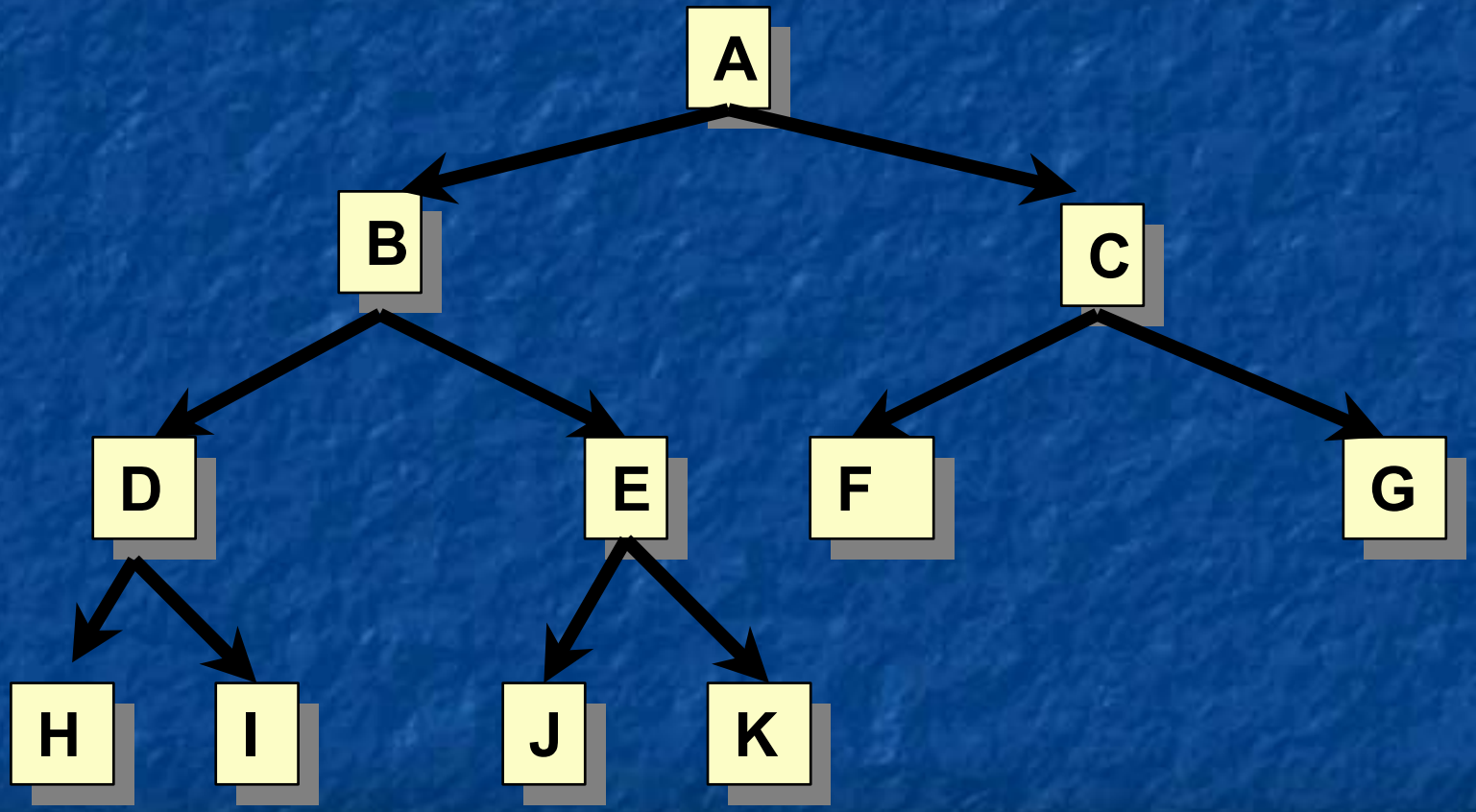
Heap

- Definition in Data Structure
  - Heap: A special form of complete binary tree that key value of each node is no smaller (larger) than the key value of its children (if any).
- Max-Heap: root node has the largest key.
  - A *max tree* is a tree in which the key value in each node is no smaller than the key values in its children. A *max heap* is a complete binary tree that is also a max tree.
- Min-Heap: root node has the smallest key.
  - A *min tree* is a tree in which the key value in each node is no larger than the key values in its children. A *min heap* is a complete binary tree that is also a min tree.

Complete Binary Tree

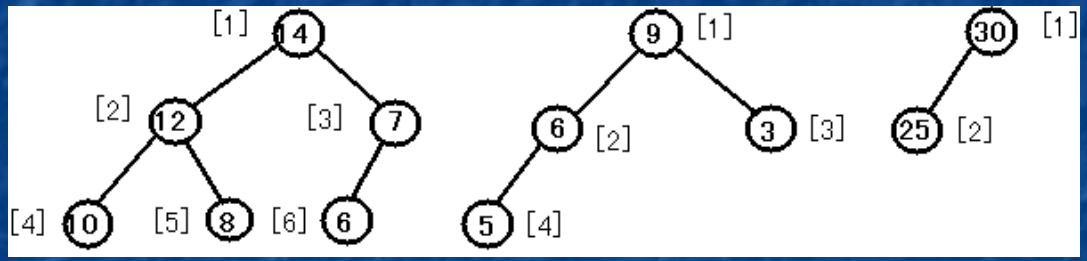
- A **complete binary tree** is a binary tree in which every level, *except possibly the last*, is completely filled, and all nodes are as far left as possible.

Complete Binary Trees - Example

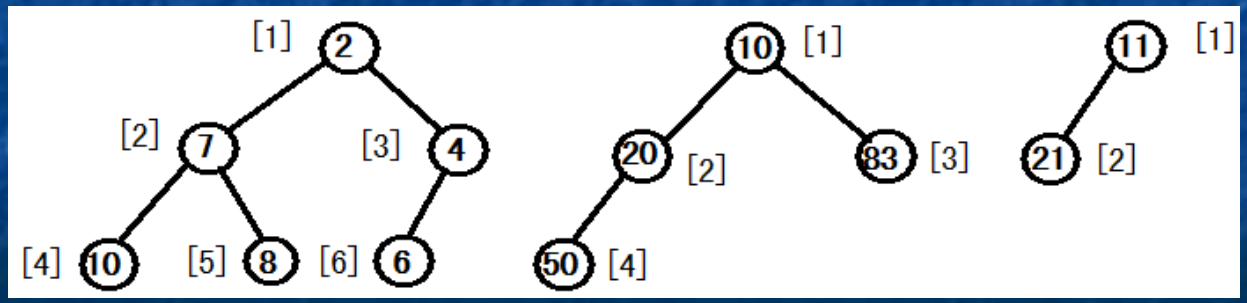


Heap

- Example:
  - Max-Heap

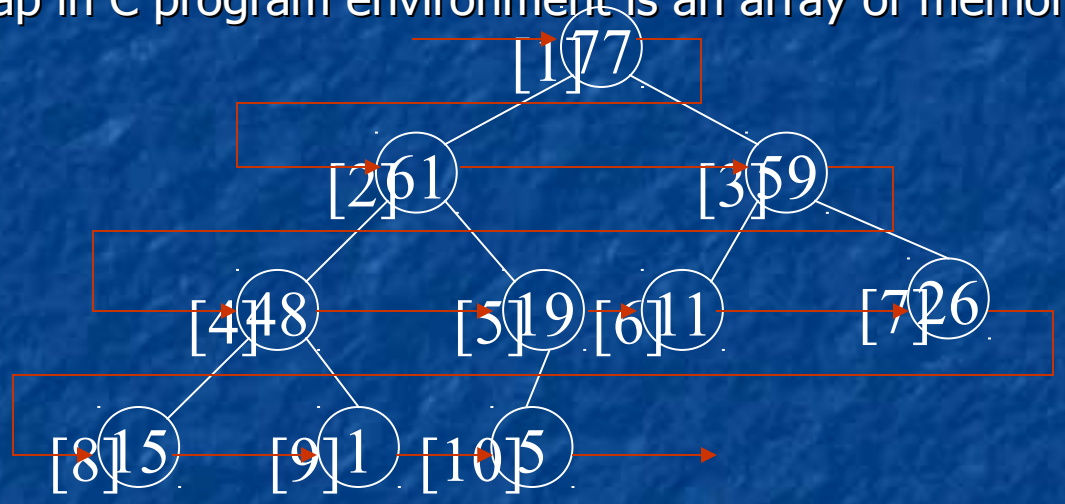


  - Min-Heap



Heap

- Notice:
  - Heap in data structure is a complete binary tree! (**Nice representation in Array**)
  - Heap in C program environment is an array of memory.



  - Stored using array in C

index	1	2	3	4	5	6	7	8	9	10
value	77	61	59	48	19	11	26	15	1	5

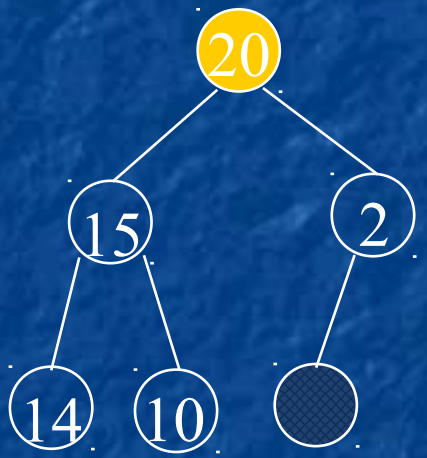
Heap

- Operations
  - Creation of an empty heap
  - Insertion of a new element into the heap
  - Deletion of the largest(smallest) element from the heap
- Heap is complete binary tree, can be represented by array. So the complexity of inserting any node or deleting the root node from Heap is O(height) = O($\log_2 n$).

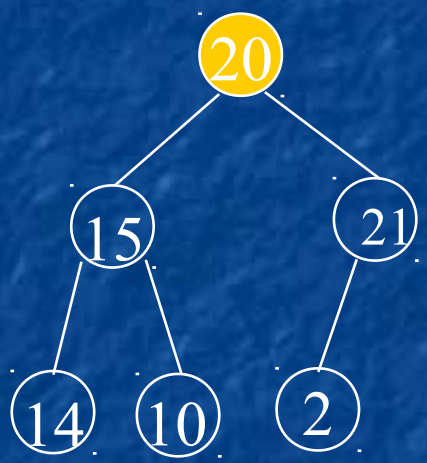
Heap

- Given the index i of a node
- Parent(i)
  - return i/2
- LeftChild(i)
  - return 2i
- RightChild(i)
  - Return 2i+1

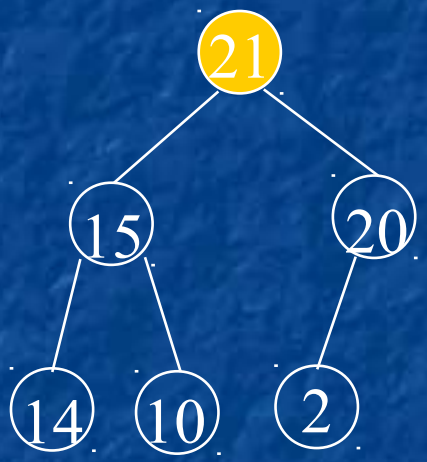
Example of Insertion to Max Heap



initial location of new node



insert 21 into heap

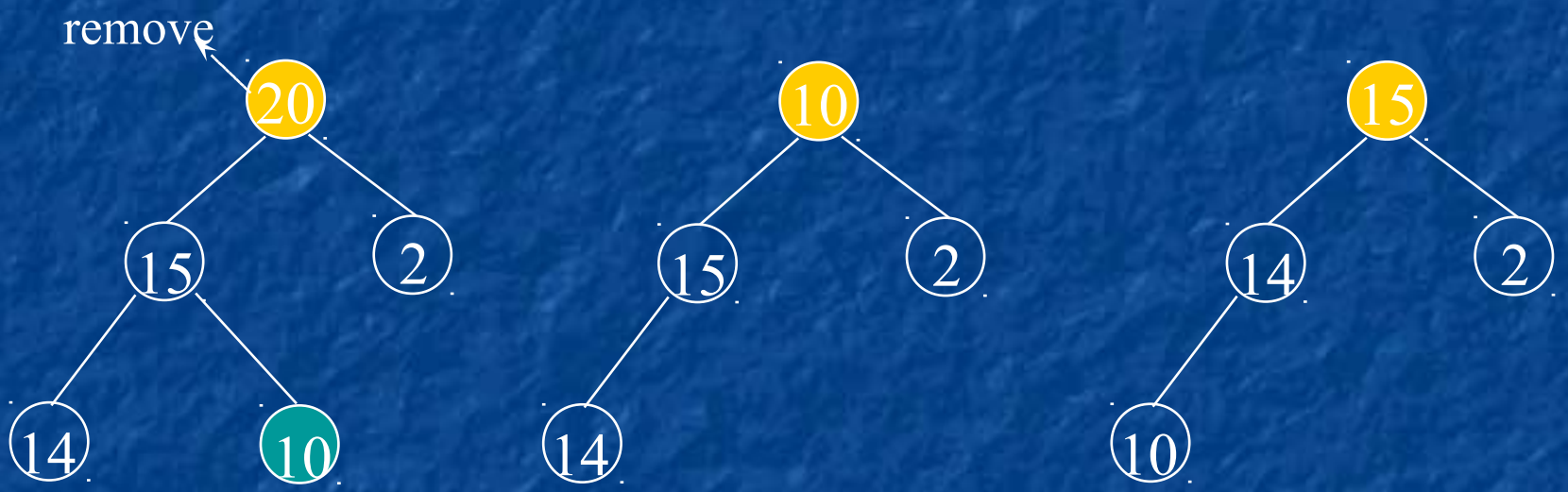


insert 21 into heap

Insertion into a Max Heap

```
void insert_max_heap(element item, int *n)
{
  int i;
  if (HEAP_FULL(*n)) {
    fprintf(stderr, "the heap is full.\n");
    exit(1);
  }
  i = ++(*n);
  while ((i!=1)&&(item.key>heap[i/2].key)) {
    heap[i] = heap[i/2];
    i /= 2;
  }
  heap[i]= item;
}
```

Example of Deletion from Max Heap



Deletion from a Max Heap

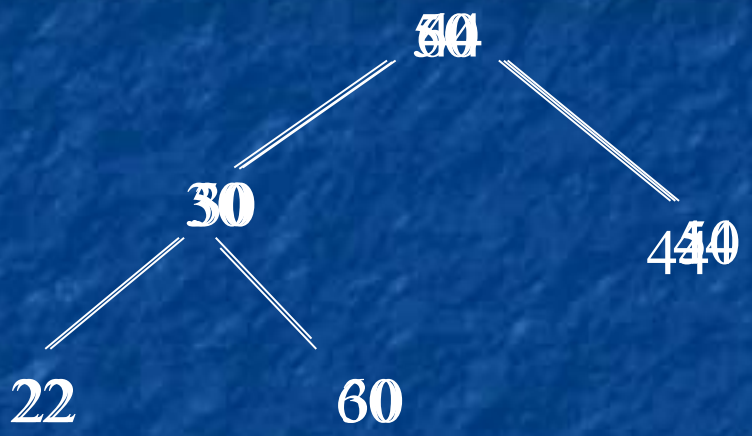
```
element delete_max_heap(int *n)
{
  int parent, child;
  element item, temp;
  if (HEAP_EMPTY(*n)) {
    fprintf(stderr, "The heap is empty\n");
    exit(1);
  }
  /* save value of the element with the
     highest key */
  item = heap[1];
  /* use last element in heap to adjust heap */
  temp = heap[(*n)--];
  parent = 1;
  child = 2;
```

Deletion from a Max Heap

```
    while (child <= *n) {
     /* find the larger child of the current
     parent */
     if ((child < *n)&&
        (heap[child].key<heap[child+1].key))
        child++;
     if (temp.key >= heap[child].key) break;
     /* move to the next lower level */
     heap[parent] = heap[child];
     parent = child
     child *= 2;
   }
   heap[parent] = temp;
   return item;
}
```

Suppose we want to build a heap H from the following list of numbers

44, 30, 50, 22, 60, 55, 77, 55



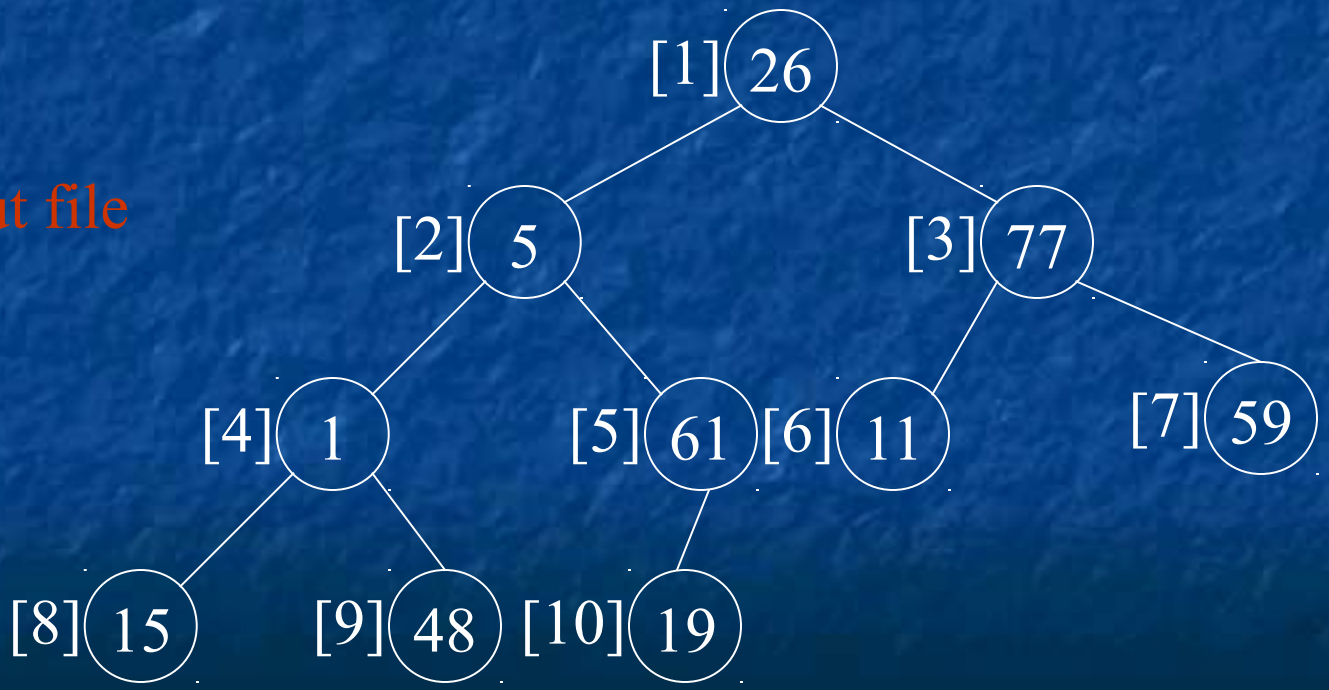

77
55
60
50
30
44
55
22

Application On Sorting: Heap Sort

- See an illustration first
  - Array interpreted as a binary tree

1	2	3	4	5	6	7	8	9	10
26	5	77	1	61	11	59	15	48	19

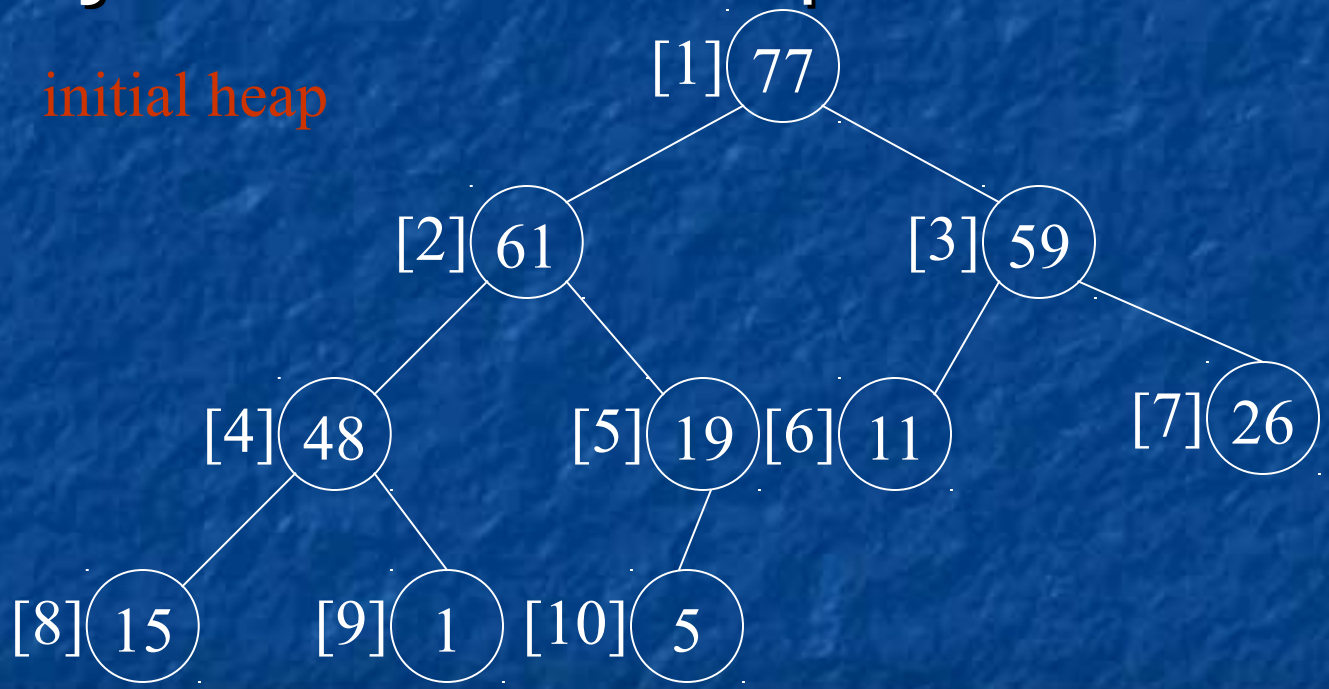
input file



Heap Sort

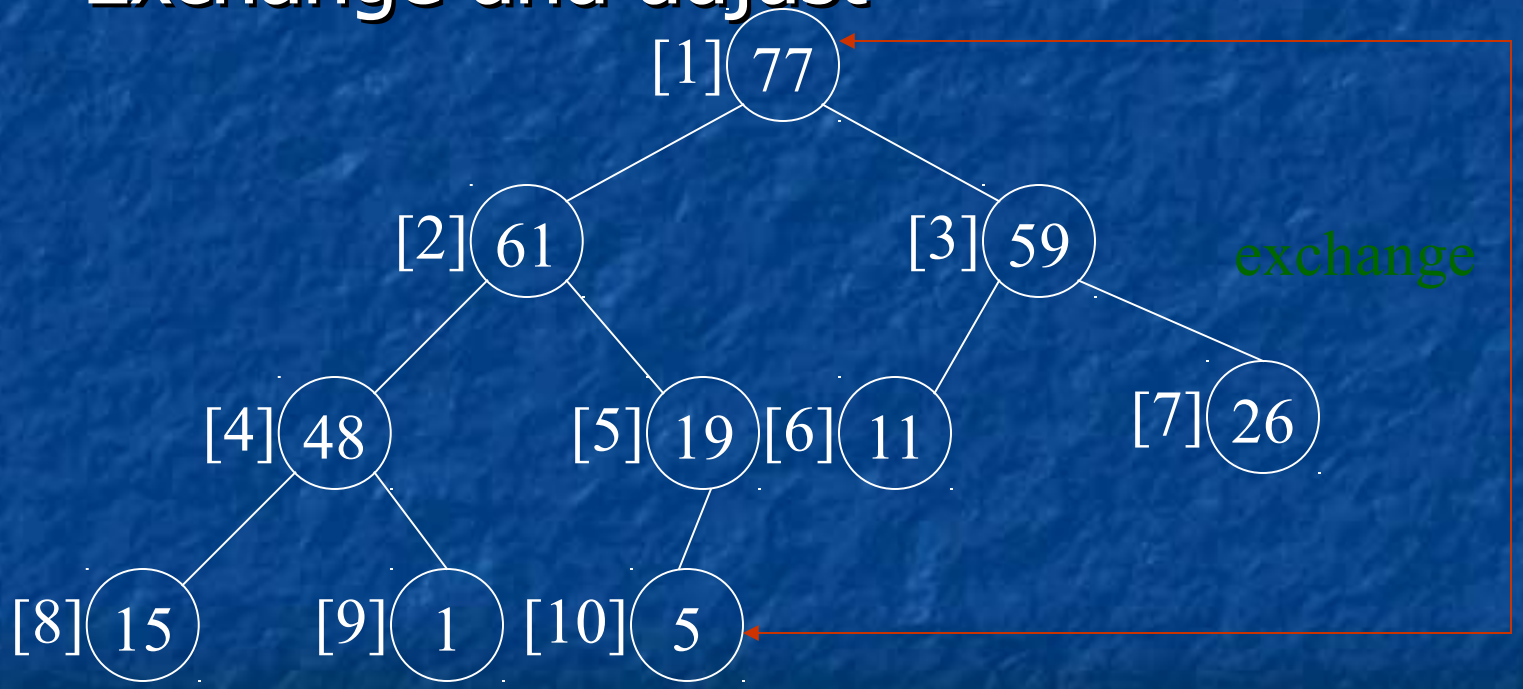
- Adjust it to a MaxHeap

initial heap

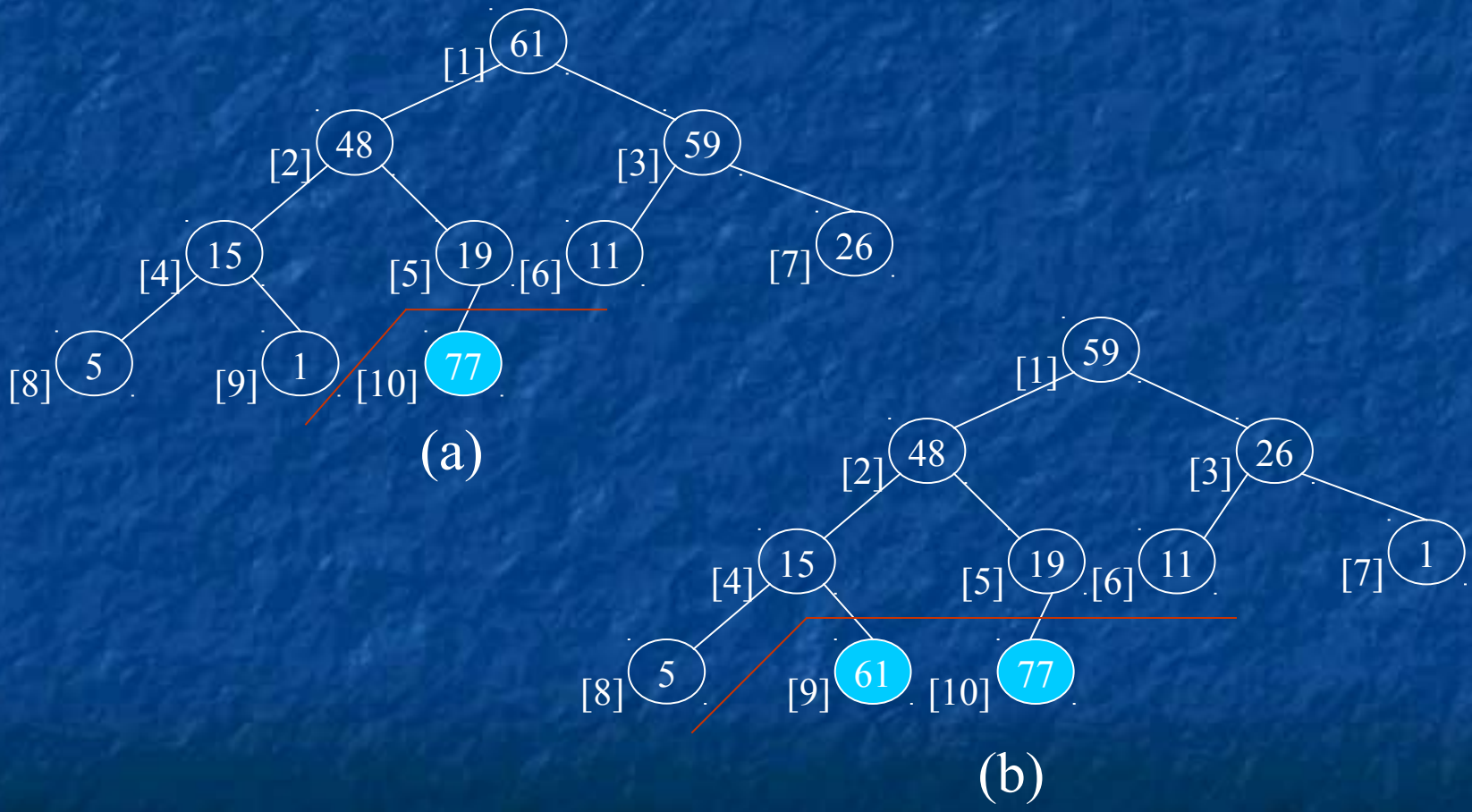


Heap Sort

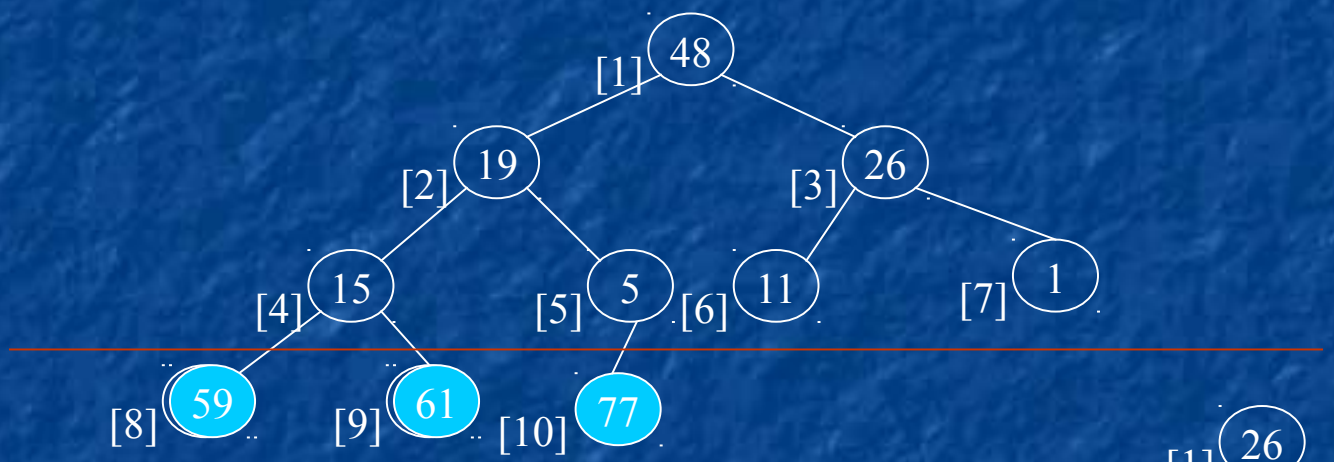
- Exchange and adjust



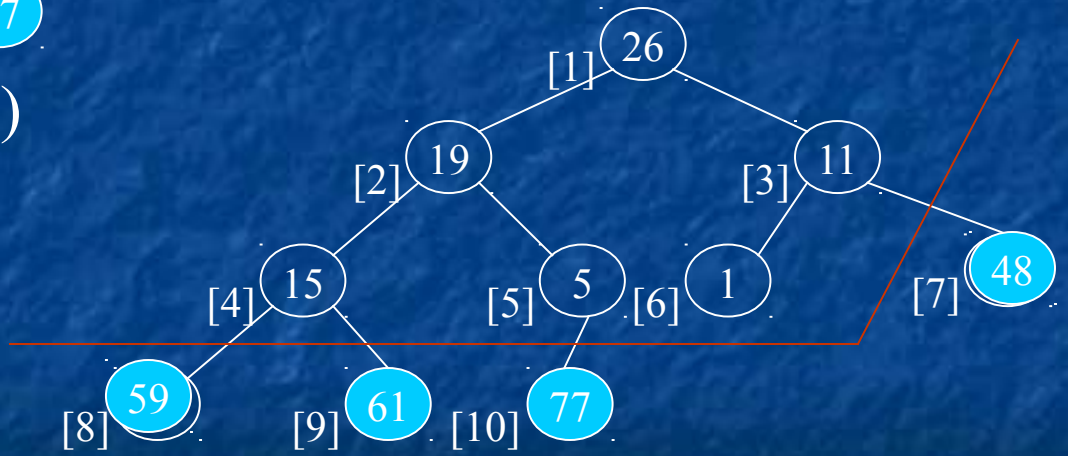
Heap Sort


[1] 61
[2] 48
[3] 59
[4] 15
[5] 19
[6] 11
[7] 26
[8] 5
[9] 1
[10] 77
(a)
[1] 59
[2] 48
[3] 26
[4] 15
[5] 19
[6] 11
[7] 1
[8] 5
[9] 61
[10] 77
(b)

Heap Sort

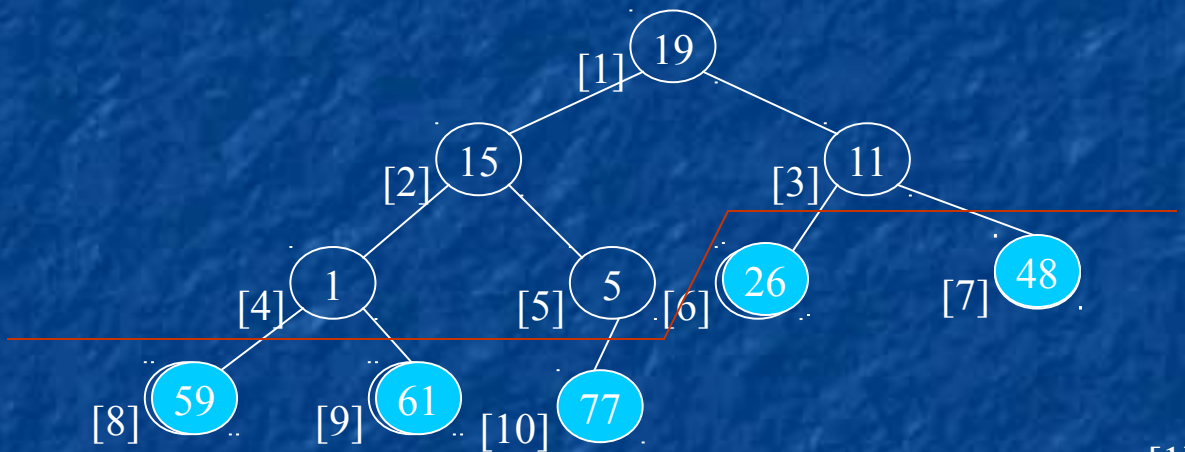

[1] 48
[2] 19
[3] 26
[4] 15
[5] 5
[6] 11
[7] 1
[8] 59
[9] 61
[10] 77

(c)

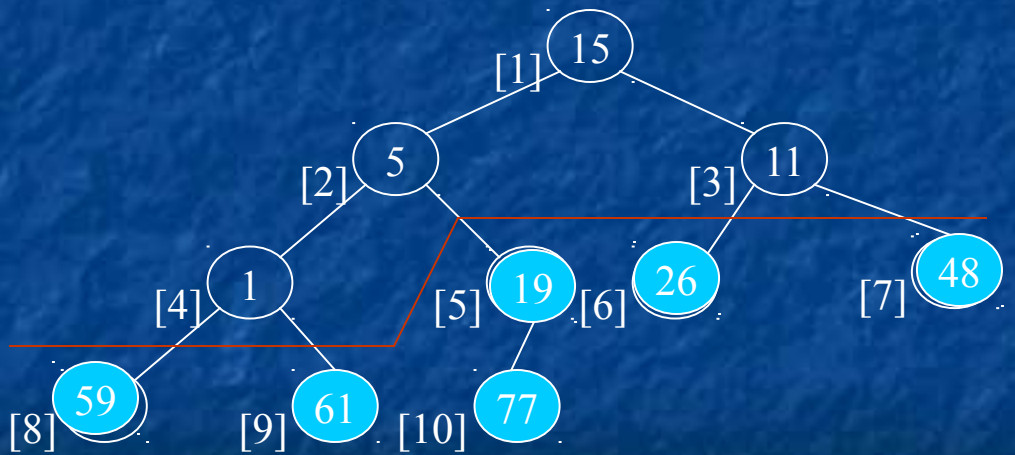

[1] 26
[2] 19
[3] 11
[4] 15
[5] 5
[6] 1
[7] 48
[8] 59
[9] 61
[10] 77

(d)

Heap Sort

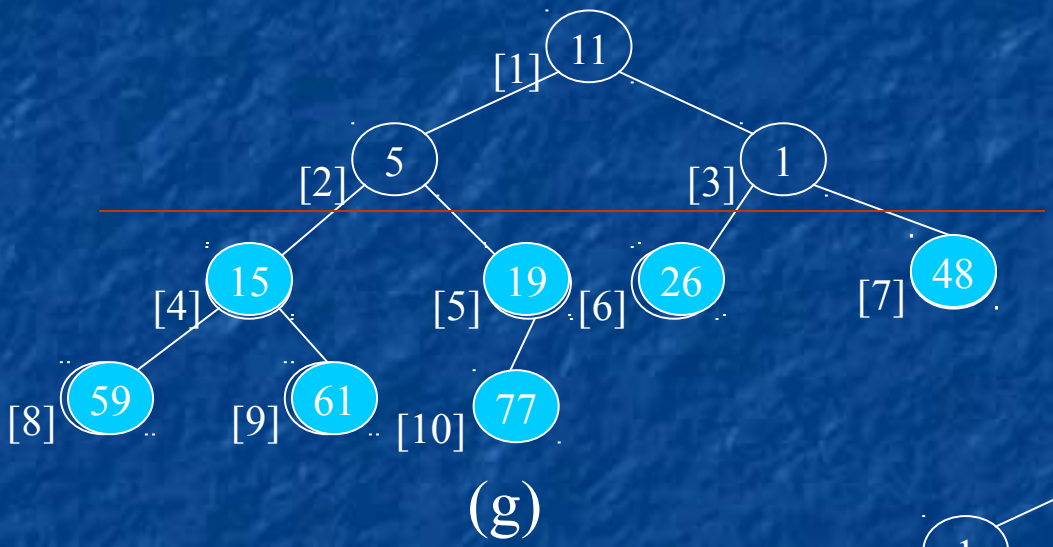

[1]
19
[2]
15
[3]
11
[4]
1
[5]
5
[6]
26
[7]
48
[8]
59
[9]
61
[10]
77

(e)

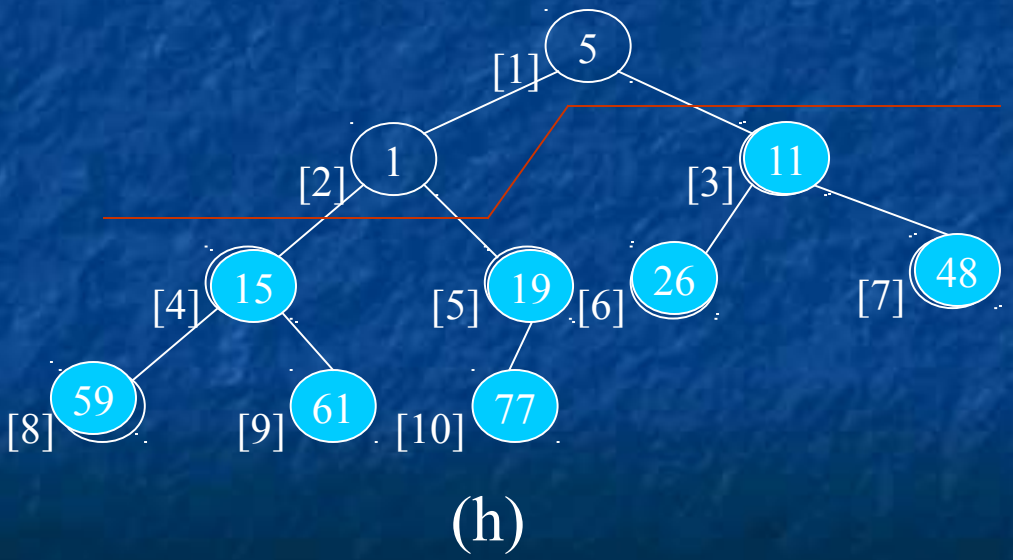

[1]
15
[2]
5
[3]
11
[4]
1
[5]
19
[6]
26
[7]
48
[8]
59
[9]
61
[10]
77

(f)

Heap Sort

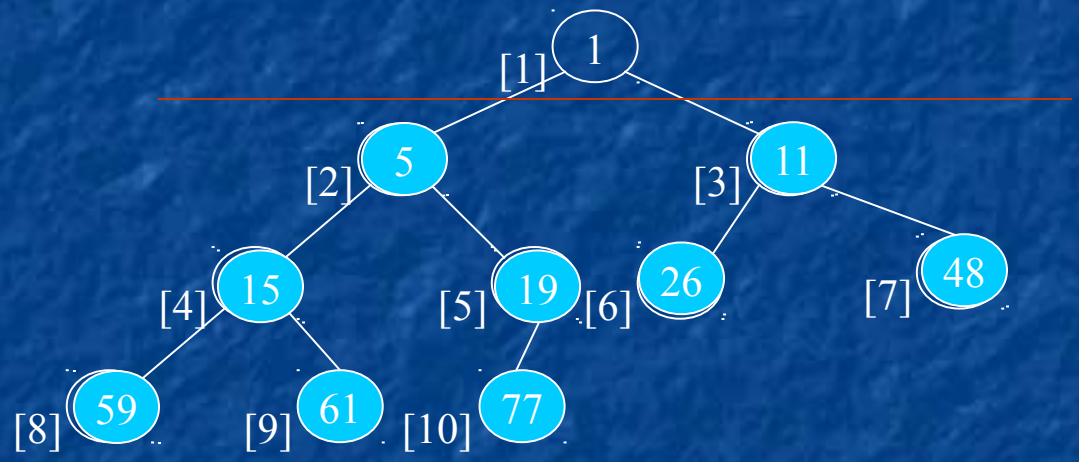

[1]
11
[2]
5
[3]
1
[4]
15
[5]
19
[6]
26
[7]
48
[8]
59
[9]
61
[10]
77

(g)


[1]
5
[2]
1
[3]
11
[4]
15
[5]
19
[6]
26
[7]
48
[8]
59
[9]
61
[10]
77

(h)

Heap Sort



(i)

- So results

77 61 59 48 26 19 15 11 5 1

Questions

?