HA NOI UNIVERSITY OF SCIENCE AND TECHNOLOGY
SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

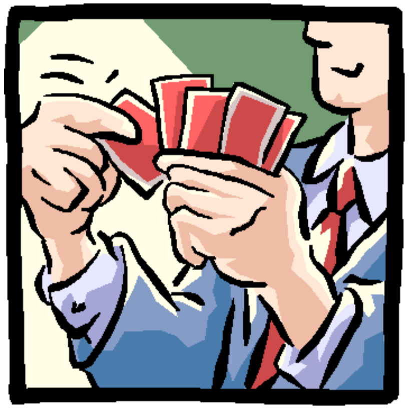
C Programming Basic

Sorting – part I

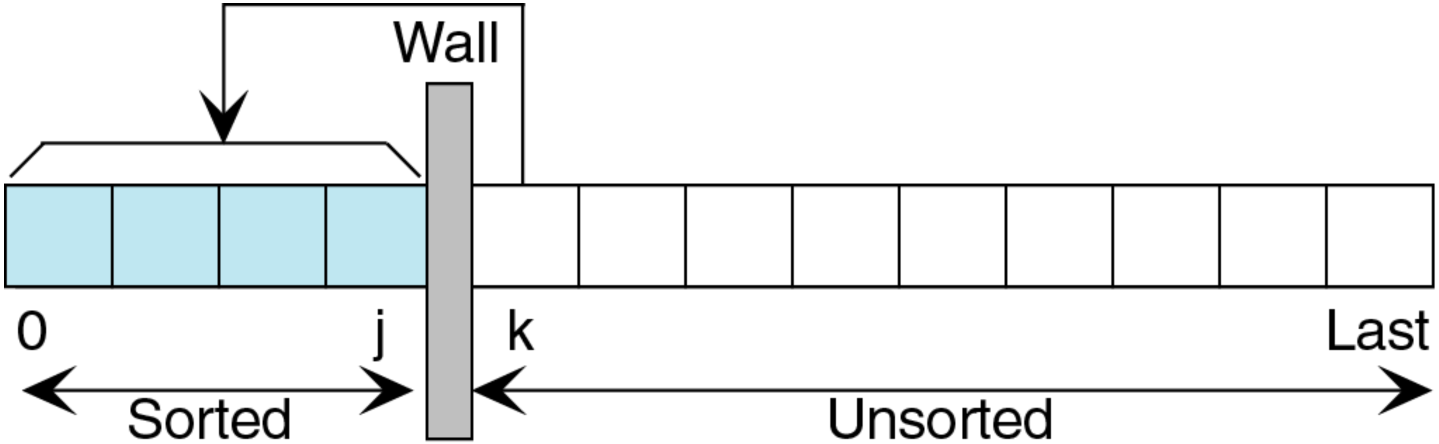
Topics of this week

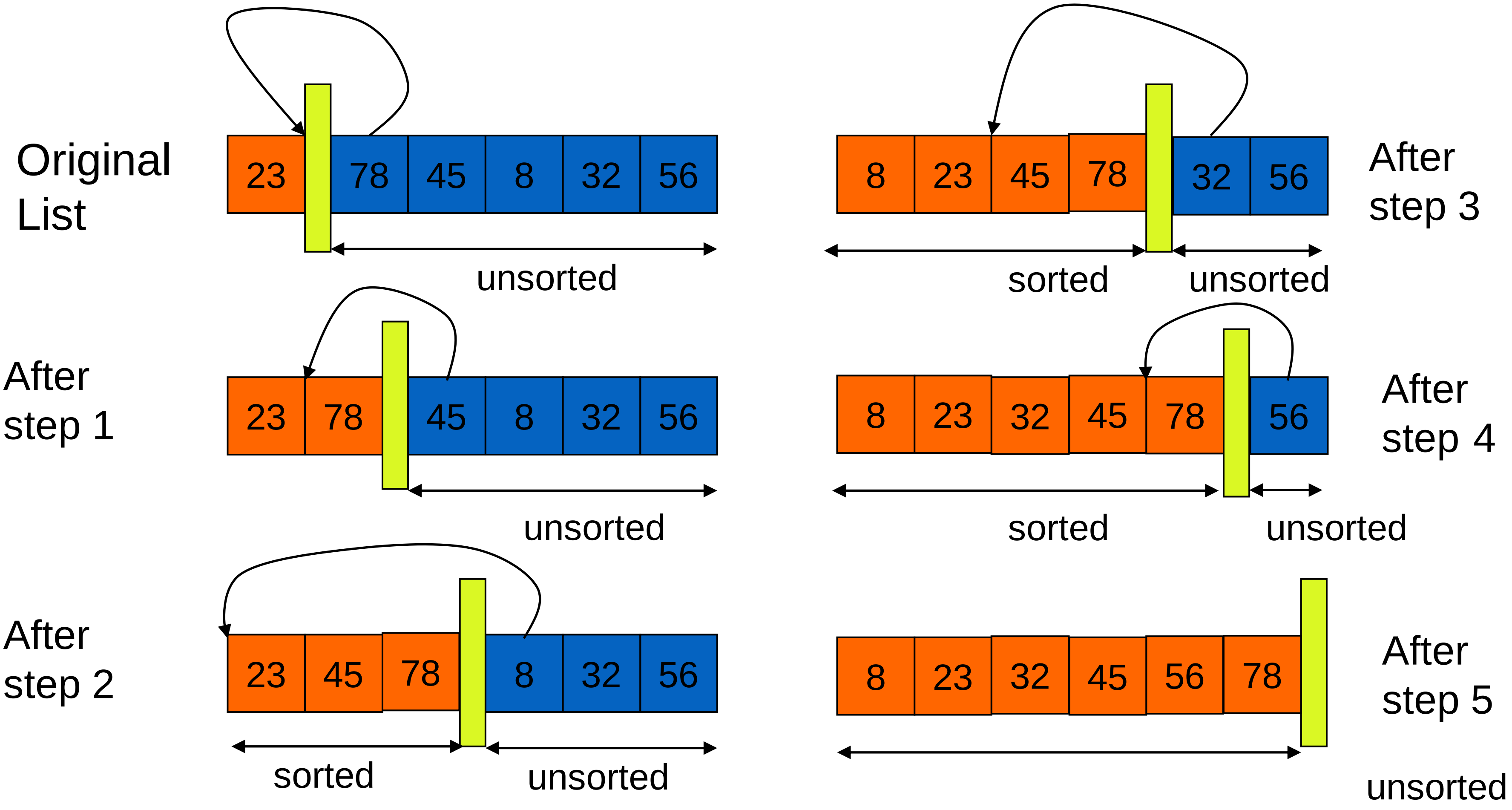
- Elementary Sorting Algorithm
  - Insertion
  - Selection
  - Bubble (exchange)
- Heap sort Algorithm

Insertion sort



- Strategy of Card Players
- Sorts list by
  - Finding first unsorted element in list
  - Moving it to its proper position
  - Efficiency: O(n^2)





Original List

23 | 78 45 8 32 56

unsorted

After step 1

23 78 | 45 8 32 56

unsorted

After step 2

23 45 78 | 8 32 56

sorted unsorted

After step 3

8 23 45 78 | 32 56

sorted unsorted

After step 4

8 23 32 45 78 | 56

sorted unsorted

After step 5

8 23 32 45 56 78 |

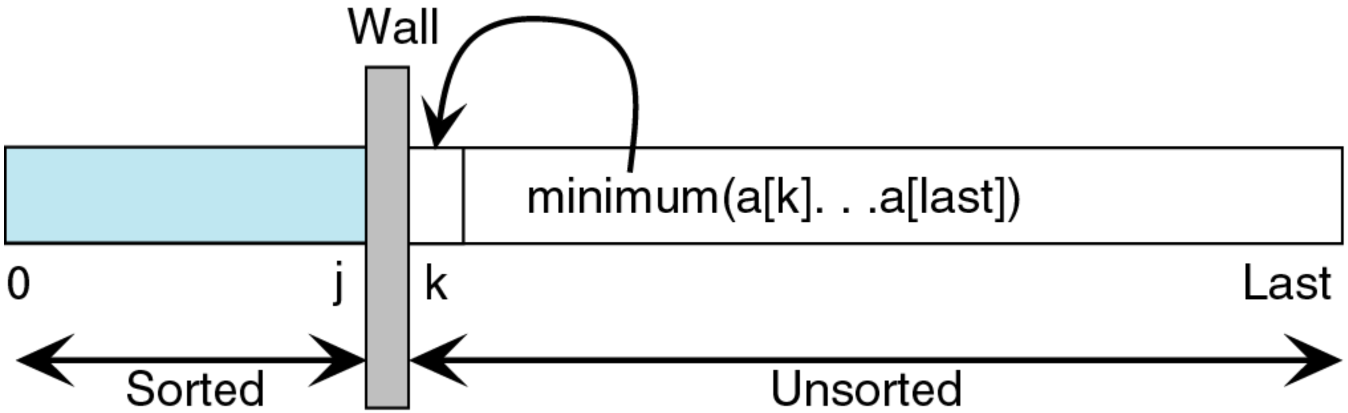
unsorted

Insertion Sort

```
void insertion_sort(element list[], int n)
{
  int i, j;
  element next;
  for (i=1; i<n; i++) {
    next= list[i];
    for (j=i-1;j>=0 && next.key< list[j].key;
         j--)
      list[j+1] = list[j];
    list[j+1] = next;
  }
}
```

Selection sort

- Sorts list by
  - Finding smallest (or equivalently largest) element in the list
  - Moving it to the beginning (or end) of the list by swapping it with element in beginning (or end) position



Selection sort

```
void selection(element a[], int n)
  { int i, j, min, tmp;
    for (i = 0; i < n-1; i++){
          min = i;
        for (j = i+1; j <=n-1 ; j++)
            if ( a[j].key < a[min].key)
                     min = j;
          tmp= a[i];
          a[i]= a[min]);
          a[min] = tmp;
      }
  }
```

Exercise 1

- We assume that you make a mobile phone’s address book.

```
typedef struct Address
{
   char name[30];
   char phone[15];
   char email[30];
};
```

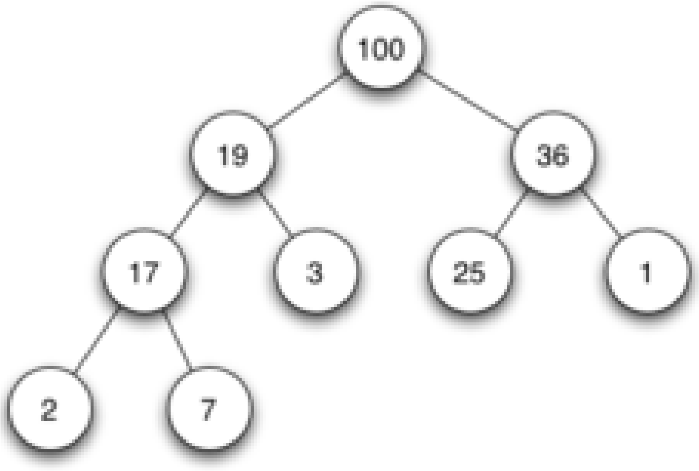
Exercise 1 (cont.)

- We want to write a program that can store about 100 structure data with name and phone number and e-mail address.
- Read about 10 data from an input file to this structure, and write the data that is sorted in ascending order into an output file.
- Use the insertion sort and selection sort

- (1) Write a program that uses array of structure
- (2) Write a program that uses singly-linked list or doubly-linked list.
- In both program, print out the number of comparisons made during the sorting process of each algorithm.

Heap sort

- Heap: a binary tree which
  - The root is guaranteed to hold largest node in tree
  - Smaller values can be on either right or left sub-tree
  - The tree is complete or nearly complete
  - Key value of each node is ≥ to key value in each descendent

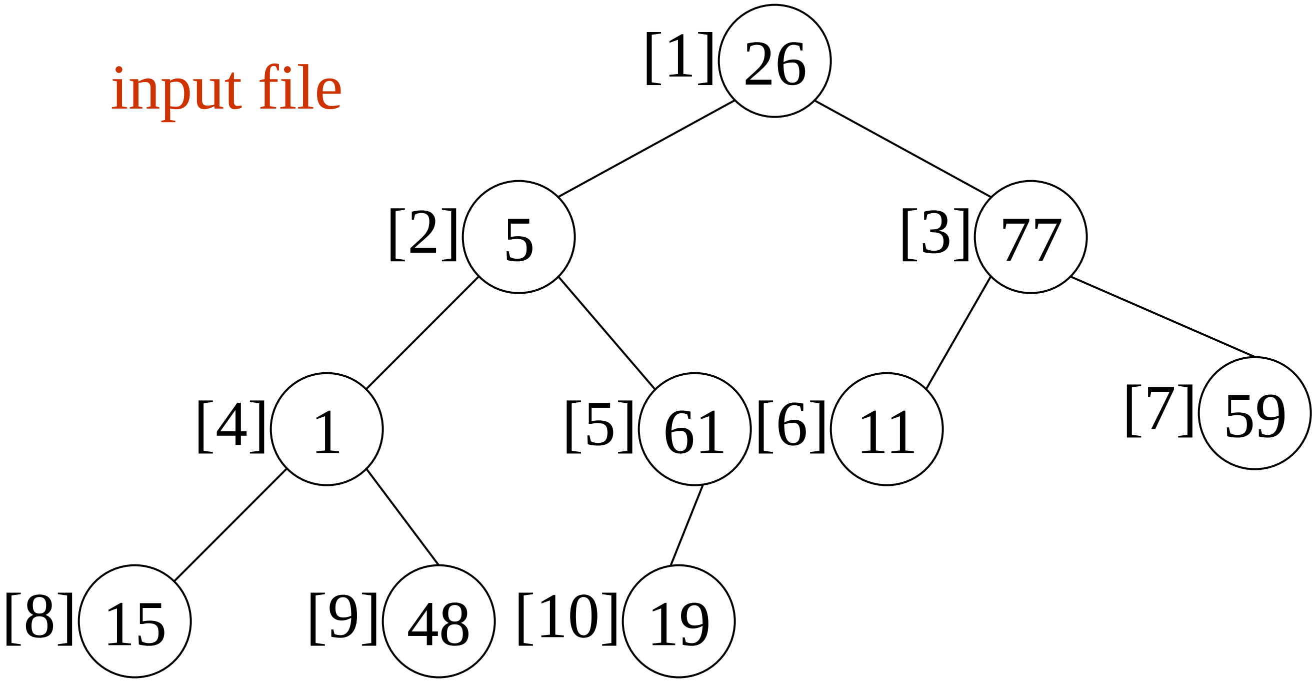


Heap sort

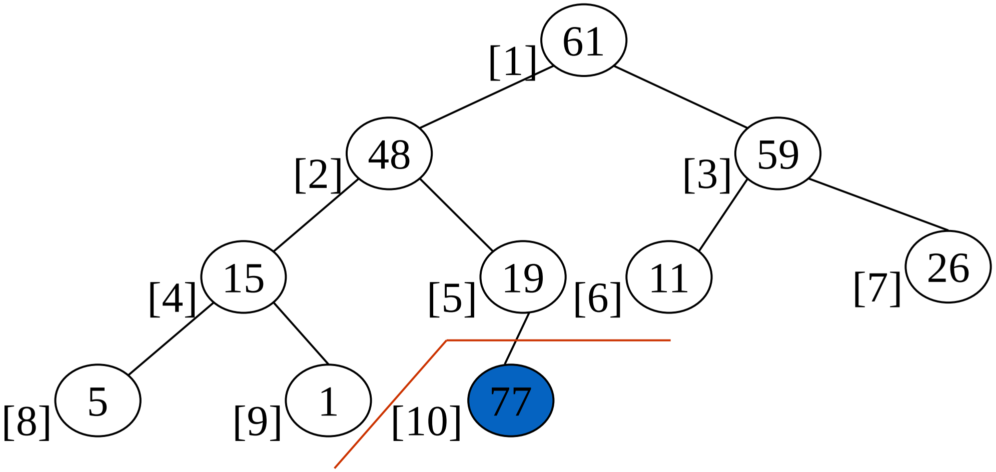
Array interpreted as a binary tree

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

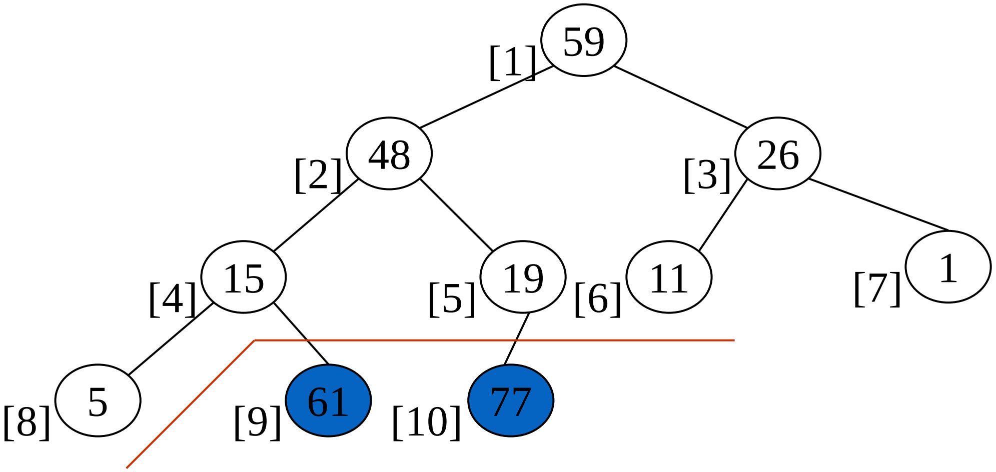
input file



Heap sort illustration

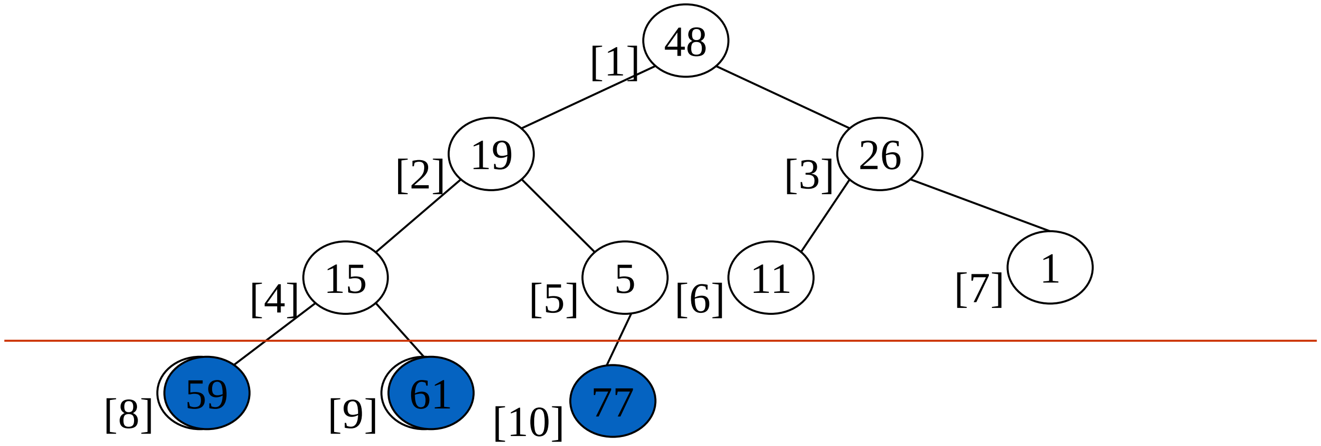


(a)

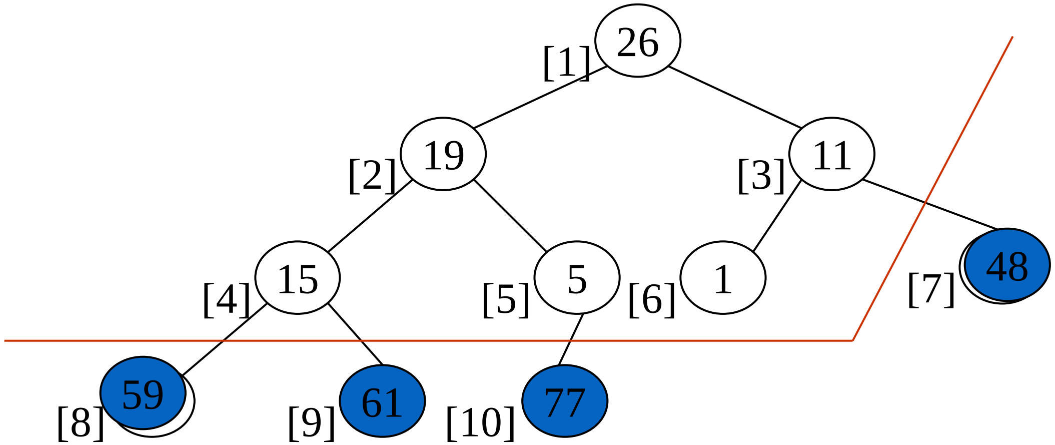


(b)

Heap sort illustration



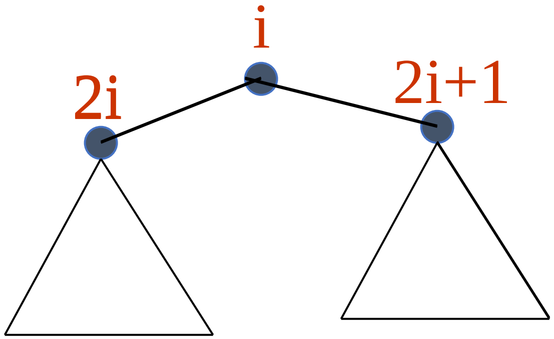
(c)



(d)

Heap sort

```
void adjust(element list[], int root, int n)
{
  int child, rootkey;   element temp;
  temp=list[root];     rootkey=list[root].key;
  child=2*root;
  while (child <= n) {
    if ((child < n) &&
        (list[child].key < list[child+1].key))
           child++;
    if (rootkey > list[child].key) break;
    else {
      swap(list[child/2], list[child]);
      child *= 2;
    }
  }
}
```

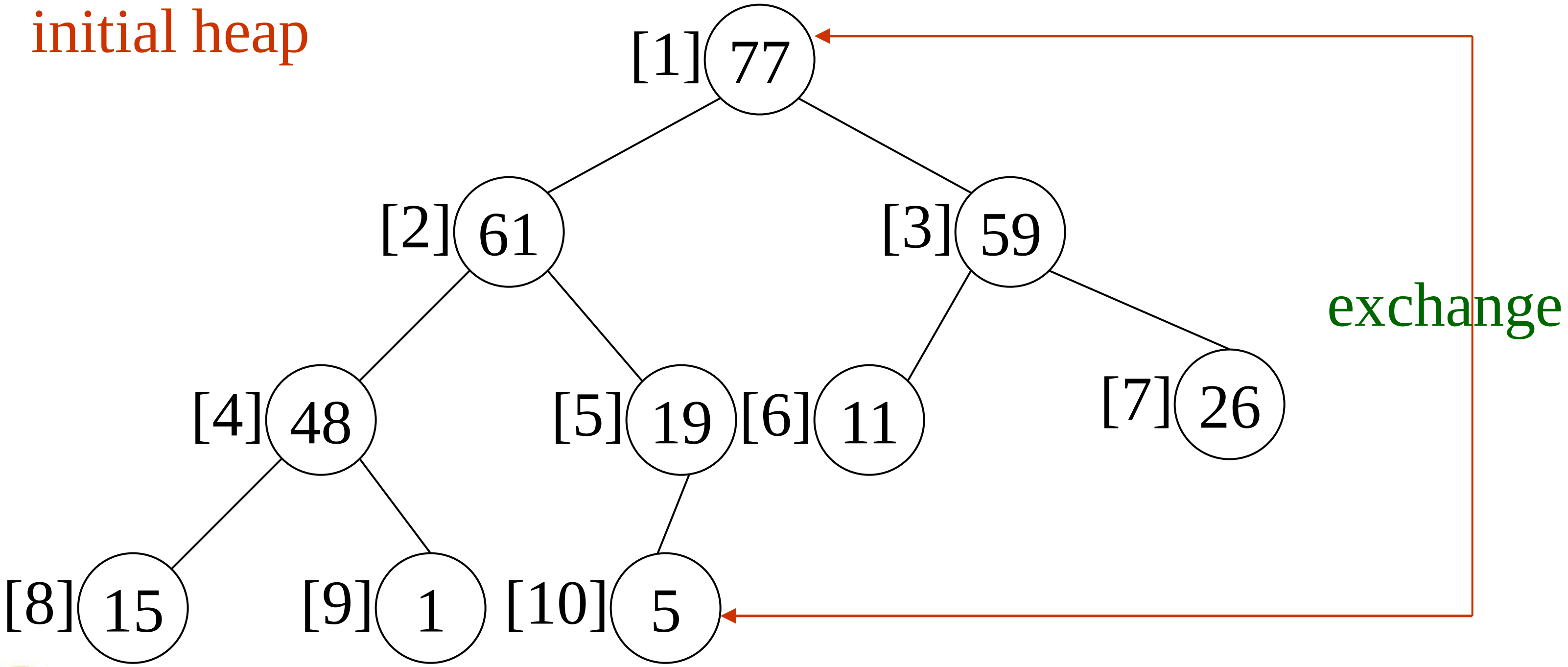


Heap sort

```
void heapsort(element list[], int n)
{  //ascending order (max heap)
    int i, j;
    element temp;
    for (i=n/2; i>0; i--)              bottom-up
             adjust(list, i, n);
    for (i=n-1; i>0; i--) {             n-1 cylces
        SWAP(list[1], list[i+1], temp);
        adjust(list, 1, i);             top-down
    }
}
```

Heap sort

Max heap following first **for** loop of *heapsort*



Exercise 2

- We assume that you make a mobile phone’s address book.
- At least, we want to write a program that can store the declared about 100 structure data with name and phone number and e-mail address.
- Read the about 10 data from an input file to this structure, and write the data that is sorted in ascending order into an output file.
- Use the heap sort. Print out the number of comparisons.

Exercise 3: Comparison of running time

- Create a dynamic memory allocation array to store 1,000,000 integer values.
- Assign random value to array’s elements.
- Build a program with the following menu

Sorting Algorithms Comparison

– 1. Create dataset (Generate integers)

– 2. Insertion Sort

– 3. Selection Sort

– 4. Bubble Sort

– 5. Heap Sort

- For each algorithm, display the running time

Help

- function for generating random numbers:

```
        srand(time(NULL))and rand()
```

- Time functions

```
 #include <time.h>
time_t t1,t2;
time(&t1);
/* Do something */
time(&t2);
durationinseconds = (int) t2 – t1;
```

Compute running time by ticks

```
clock_t tstart,tfinish;
tstart = clock();
/*Thực hiện công việc*/
tfinish = clock();
float tcomp;
tcomp=(float)(tfinish-tstart)/CLOCKS_PER_SEC;
```

Homework

- From the unsorted address book file.
- Sort the data by using Heapsort and display the result to the standard output.

Homework

- Input 10 words from the standard input, and load them to a character type array.
- Sort the array by insertion sort, and output the sorted array into the standard output.

Hints

- You can write a program that processes in the following order.
  - 1. Declare char data[10].
  - 2. Read every 1 word from the standard input by fgetc( ) function and load it on the array “data”.
  - 3. Do the insertion sort to the array “data”
  - 4. Output every 1 word of the value of the sorted array ”sort” by fputc( ) function.