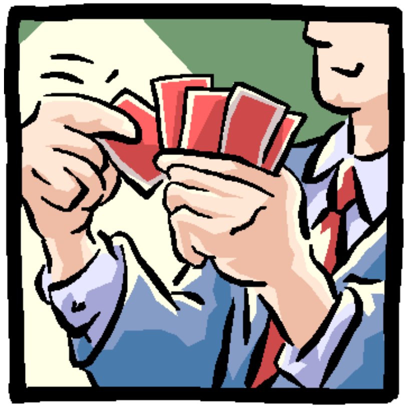
Data Structure & Algorithm Basic Lab – week 10

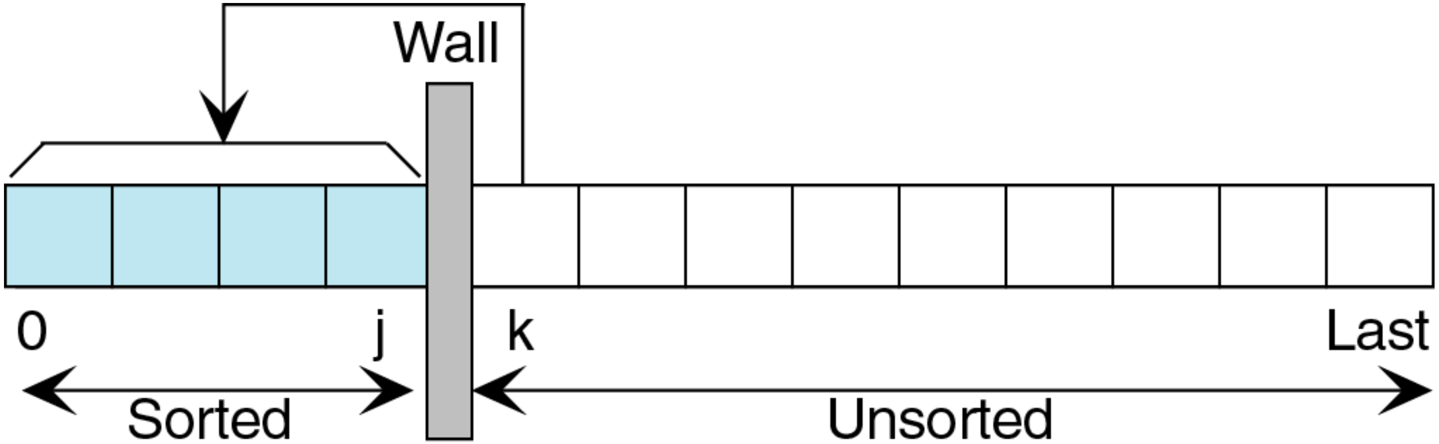
Topics of this week

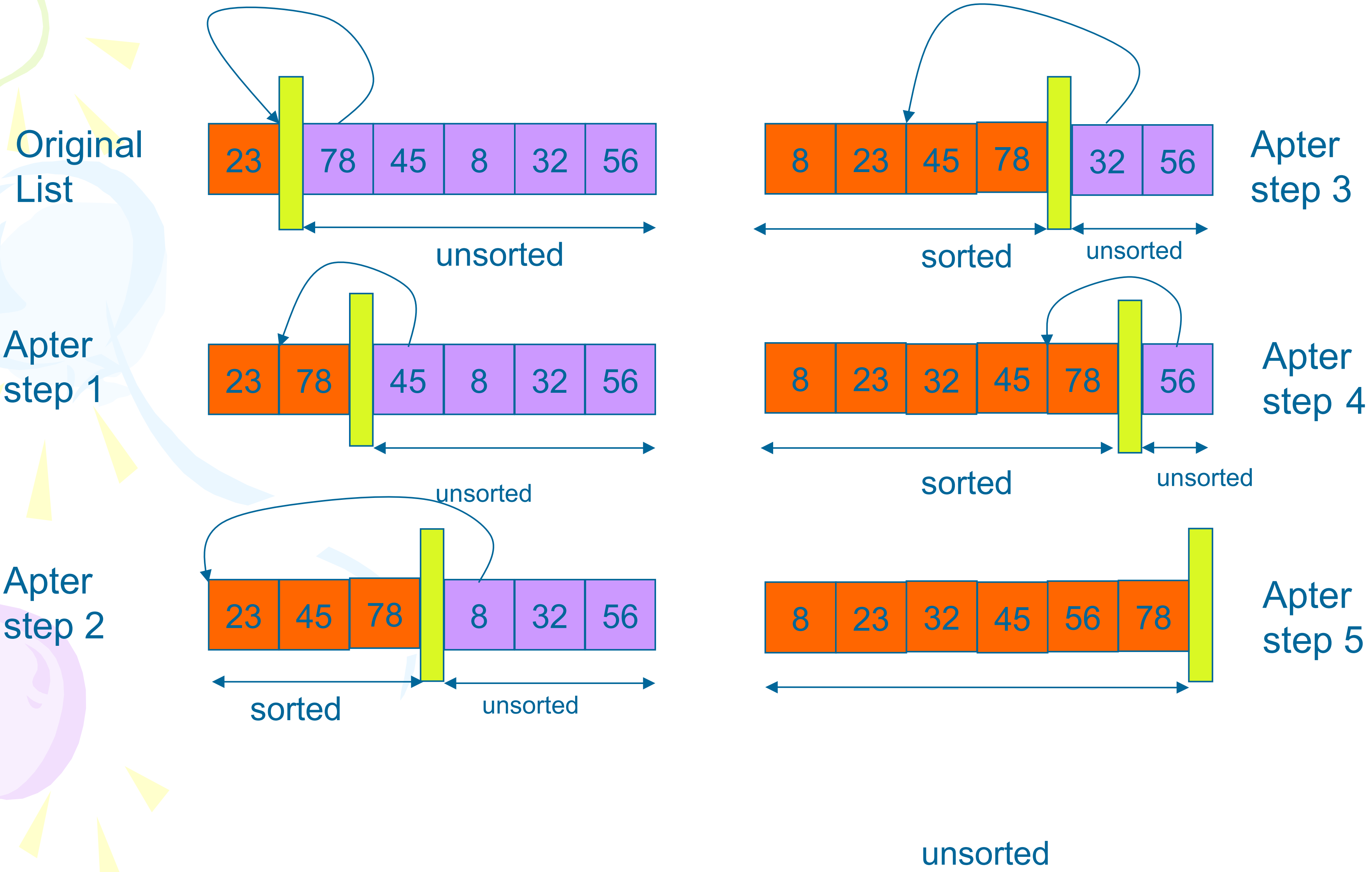
- Elementary Sorting Algorithm
  - Insertion
  - Selection
  - Bubble (exchange)
- Heap sort Algorithm

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

Apter step 1

23 78 | 45 8 32 56

unsorted

Apter step 2

23 45 78 | 8 32 56

sorted

unsorted

8 23 45 78 | 32 56

Apter step 3

sorted

unsorted

8 23 32 45 78 | 56

Apter step 4

sorted

unsorted

8 23 32 45 56 78 |

Apter step 5

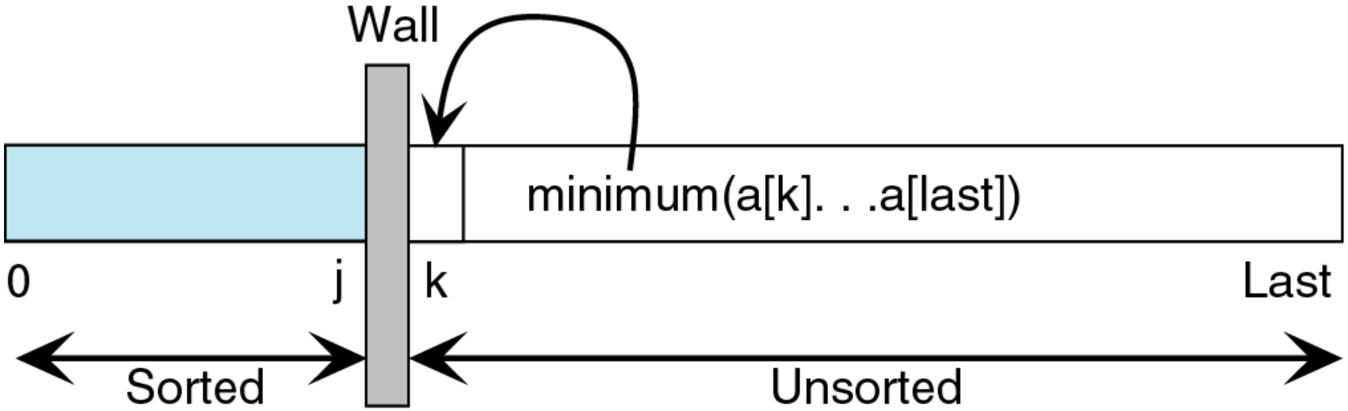
unsorted

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;
      }
  }
```

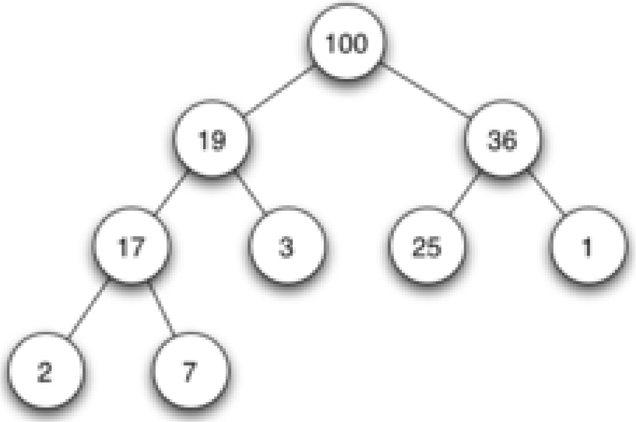
Exercise

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

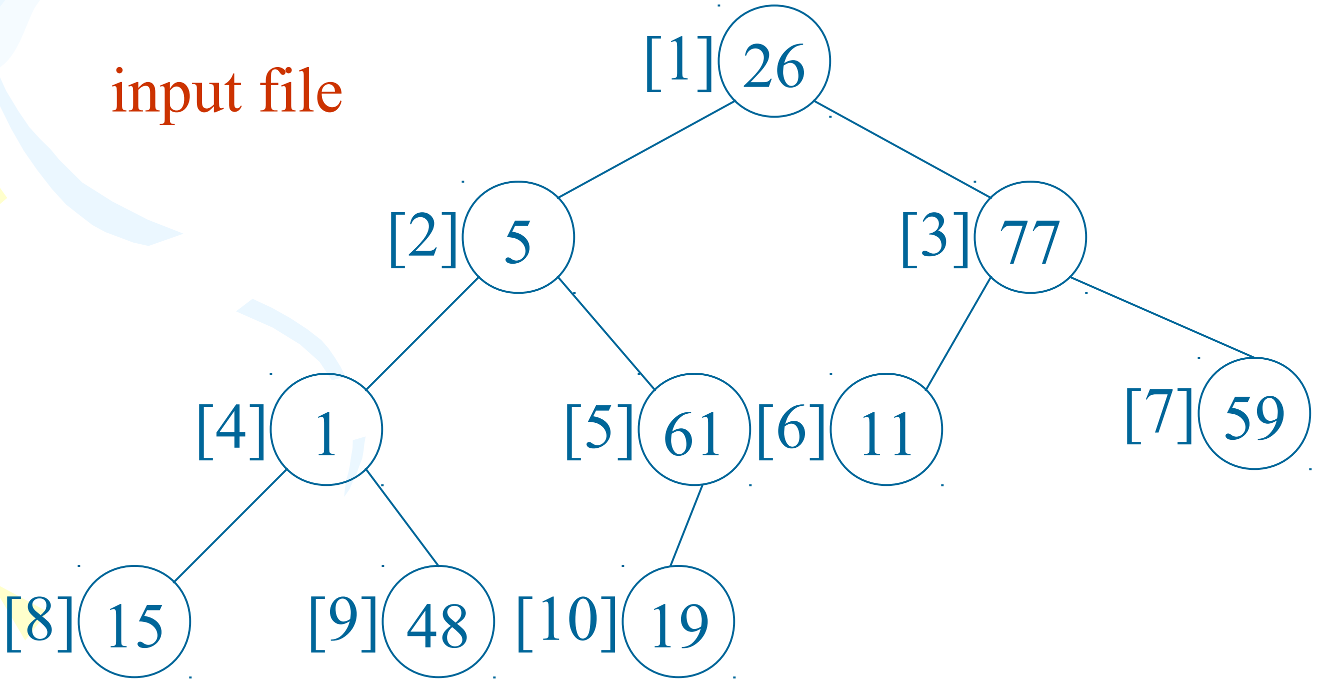
- Heap: a binary tree which
  - The root is guaranteed to hold largest node in tree
  - Smaller values can be on either right or left sub-tree
  - The tree is complete or nearly complete
  - Key value of each node is >= to key value in each descendent



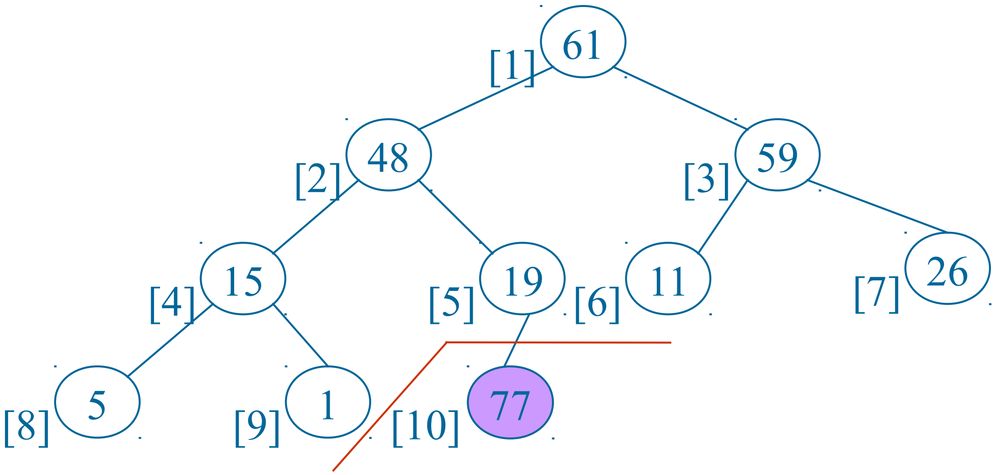
Heap sort

Array interpreted as a binary tree

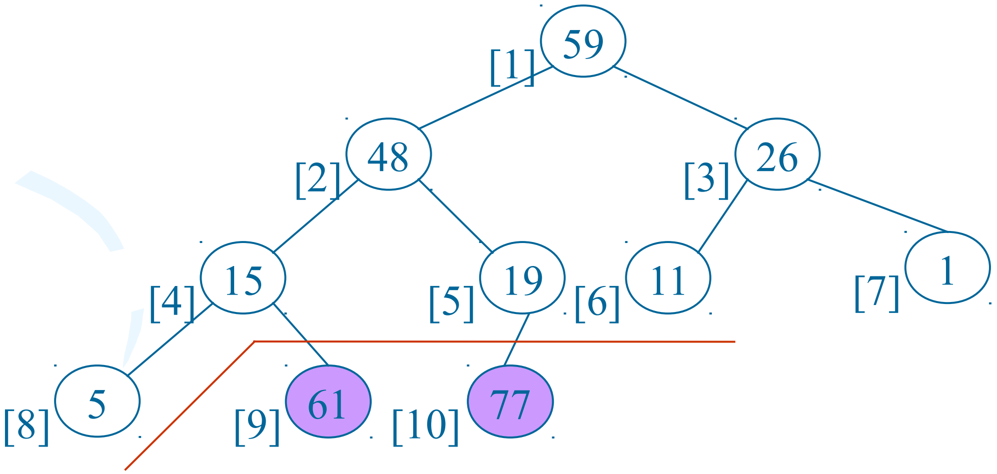
1	2	3	4	5	6	7	8	9	10
26	5	77	1	61	11	59	15	48	19


input file

Heap sort illustration

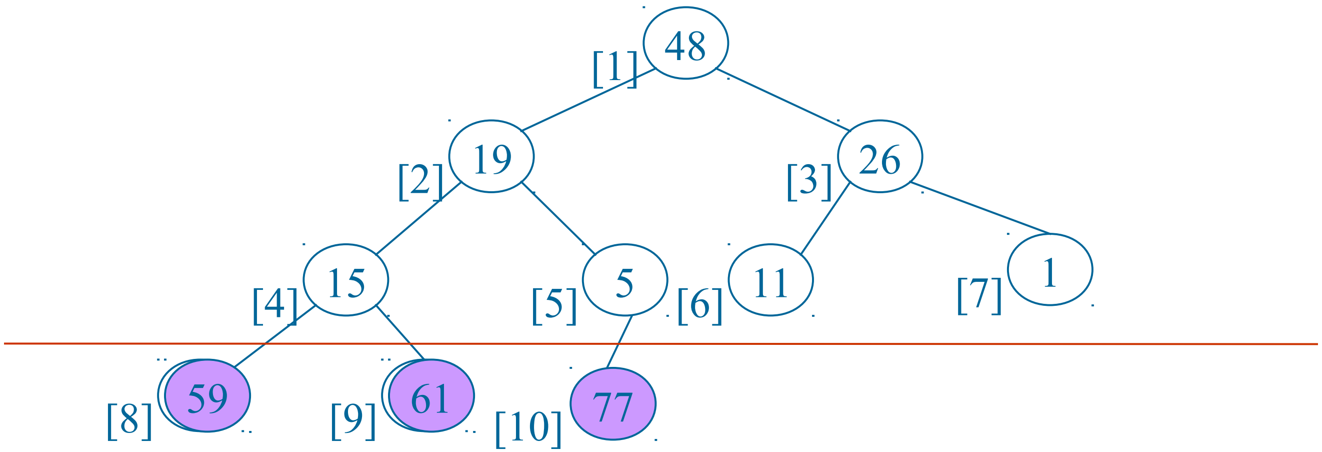


(a)

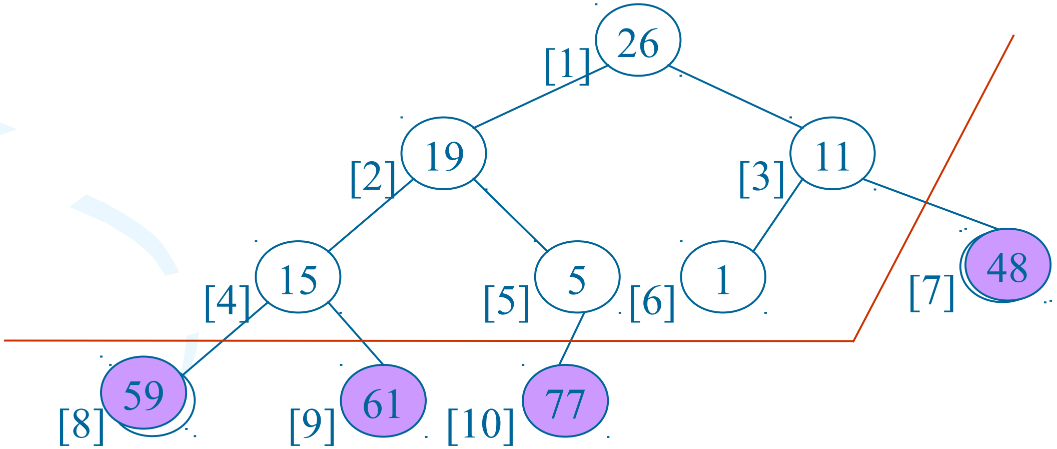


(b)

Heap sort illustration



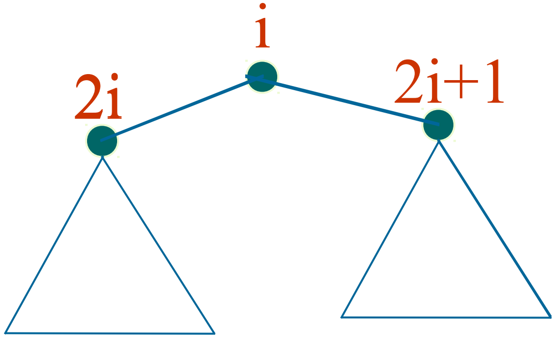
(c)



(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 {
       list[child/2] = list[child];
       child *= 2;
     }
   }
   list[child/2] = temp;
}
```

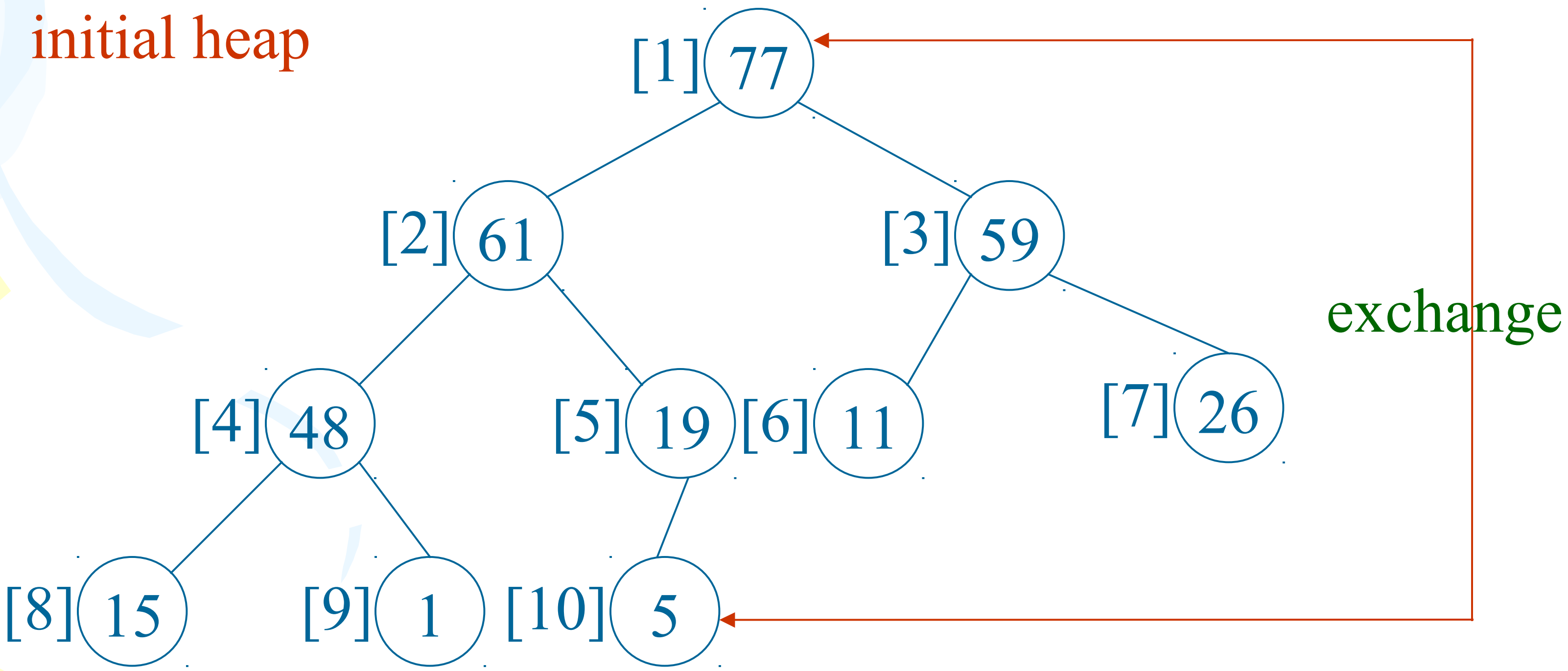


Heap sort

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

Heap sort

<u>Max heap following first **for** loop of *heapsort*</u>



Exercise

- 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: Comparison of running time

- Write a program to initiate an array of 500 integers by using random function.
- Sort this array using insertion sort and heap sort. Calculate the running time of program in each case and print out the results.

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;
```

Exercise

- 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.

BTVN 4EF

- Tạo mảng động chứa 2 triệu số nguyên.
- Chạy vòng lặp sinh giá trị ngẫu nhiên cho các phần tử (rand).
- Cài đặt menu
  - 1. Tạo lại dữ liệu
  - 2. Sắp xếp thêm dần
  - 3. Lựa chọn
  - 4. Nổi bọt
  - 5. Heap
- Với mỗi chức năng in ra thời gian sắp xếp.

BTVN 4EF 3

- Từ file NokiaDB.dat (không sắp xếp – ví dụ chọn lưu file từ chức năng BST tuần trước – nhưng lưu theo thứ tự trước).
- Đọc các bản ghi và sắp xếp theo tên Model dùng Heapsort.