BLG 335E – Analysis of Algorithms I
Fall 2015, Recitation 4
25.11.2015

R.A.Doğan Altan
daltan@itu.edu.tr – Research Lab 3
R.A. Hakan Gündüz
hakangunduz@itu.edu.tr –
Research Lab 3

Outline

- Elementary Data Structures
- Medians and Order Statistics
- Hash Tables

Preliminary Information

- Although **MAX-HEAPIFY** costs *O(lg n)* time and there are *O(n)* calls to it, still we can find a tighter bound than *O(n lg n)* for:

```
BUILD-MAX-HEAP(A)
heap-size[A] ← length[A]
for i ← ⌊length[A]/2⌋ downto 1
    do MAX-HEAPIFY(A, i)
```

- Because, cost of **MAX-HEAPIFY** depends on the height of the node in the tree, and and the heights of most nodes are small.

BUILD-MAX-HEAP(A, n)

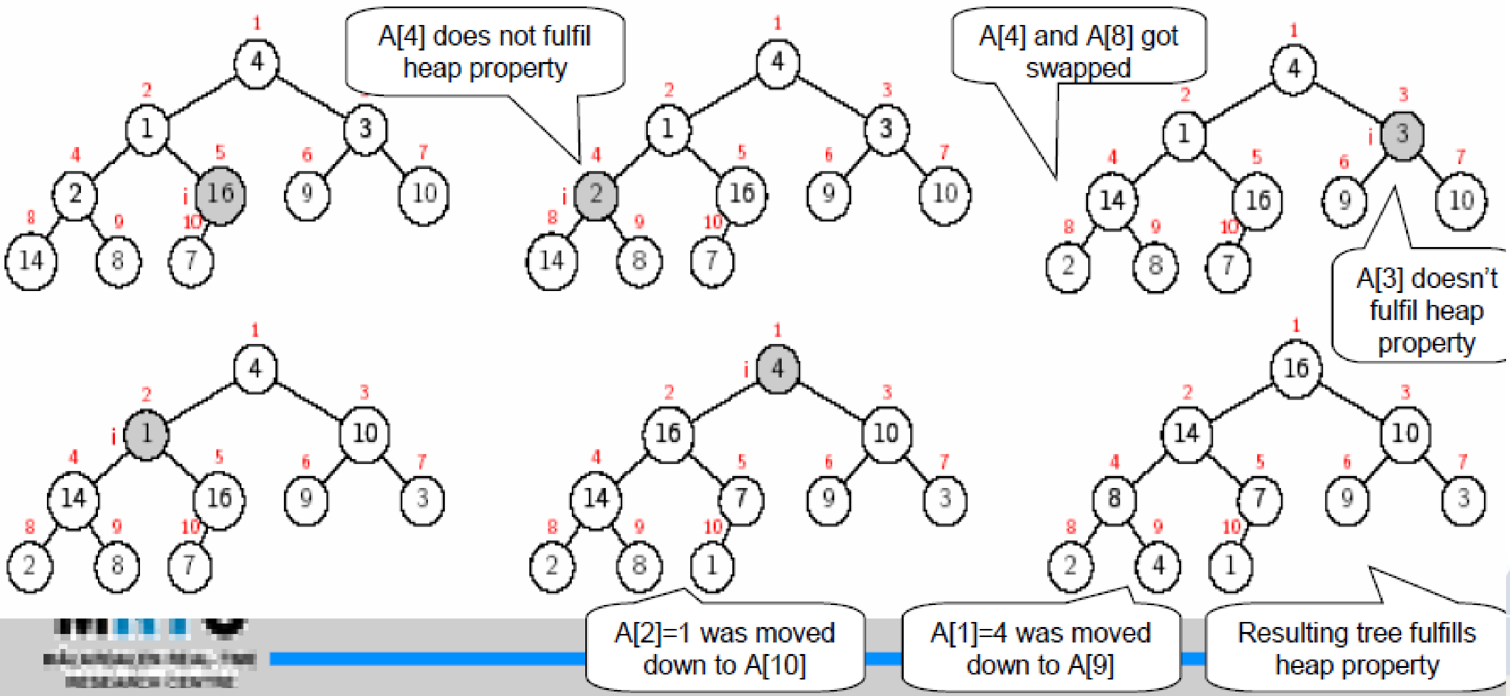
for $i \leftarrow \lfloor n/2 \rfloor$ **downto** 1

    do MAX-HEAPIFY(A, i, n)

Input array A

1	2	3	4	5	6	7	8	9	10
4	1	3	2	16	9	10	14	8	7

A[10], A[9], ..., A[6] have already been inserted, all fulfilling heap property



Exercise 1

- Using a similar reasoning, can we find a tighter bound than *O(n lg n)* for **Heapsort**?

```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i ← length[A] downto 2
    do exchange A[1] ↔ A[i]
       heap-size[A] ← heap-size[A] − 1
       MAX-HEAPIFY(A, 1)
```

Exercise 1-Solution

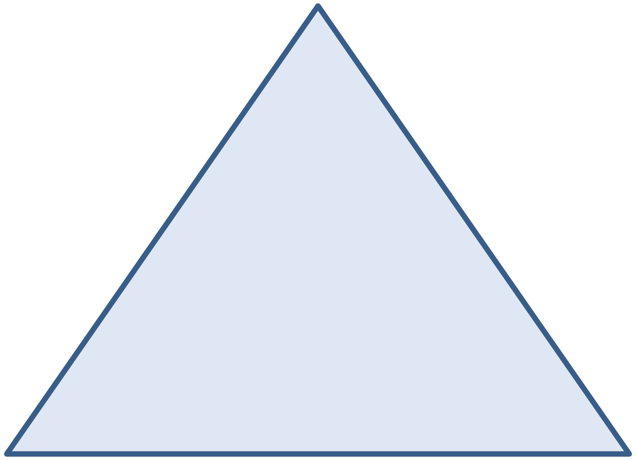
```
HEAPSORT(A)
    BUILD-MAX-HEAP(A)
    for i ← length[A] downto 2
        do exchange A[1] ↔ A[i]
            heap-size[A] ← heap-size[A] − 1
            MAX-HEAPIFY(A, 1)
```

- Call to BUILD-MAX-HEAP takes time *O(n)*
- *n-1* calls to MAX-HEAPIFY
- Each call MAX-HEAPIFY takes *O(lg n)*

Result: *O(n)* + *(n-1)* ∗ *O(lg n)* = *O(n lg n)*

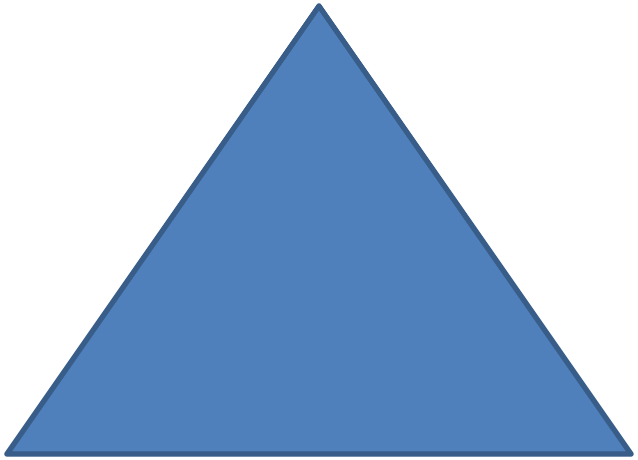
Exercise 1 – Solution

```
BUILD-MAX-HEAP(A)
heap-size[A] ← length[A]
for i ← ⌊length[A]/2⌋ downto 1
        do MAX-HEAPIFY(A, i)
```



$O(n)$

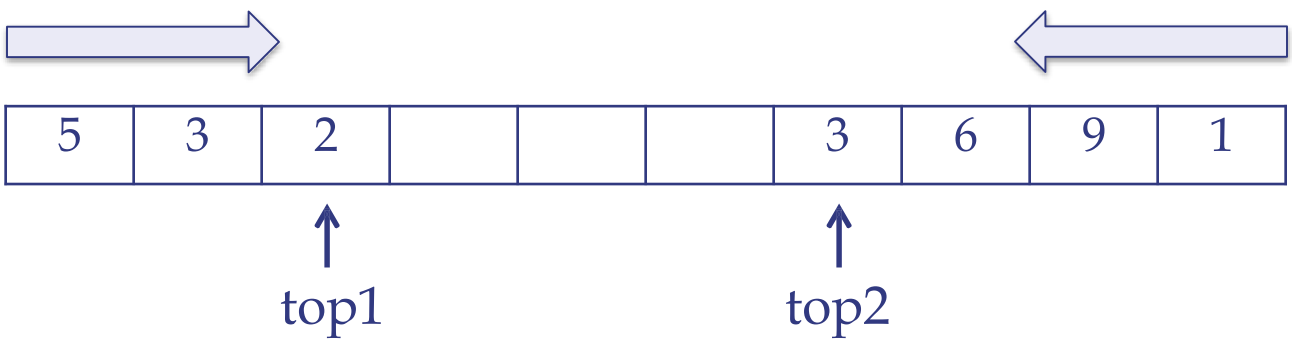
```
HEAPSORT(A)
BUILD-MAX-HEAP(A)
for i ← length[A] downto 2
        do exchange A[1] ↔ A[i]
            heap-size[A] ← heap-size[A] − 1
            MAX-HEAPIFY(A, 1)
```



$O(n \log n)$

Exercise 2a

- Explain how to implement **two stacks** in **one array** A[1…n] in such a way that
  - neither stack overflows unless the total number of elements in both stacks is n.
  - the PUSH and POP operations should run in O(1) time.



Exercise 2b

- Implement a **queue** by a singly linked list **L**. The operations **Enqueue** and **Dequeue** should still take **O(1)** time.

Exercise 2b-Solution

- Implement a **queue** by a singly linked list **L**. The operations **Enqueue** and **Dequeue** should still take **O(1)** time.

Exercise 2b-Solution

- Implement a **queue** by a singly linked list **L**. The operations **Enqueue** and **Dequeue** should still take **O(1)** time.
- Using a tail pointer beside head pointer !

Exercise 2c

- Write an **O(n)-time** procedure that, given an n-node **binary tree**, prints out the key of each node in the tree.

  a) Recursively

  b) Non-recursively using a **stack**

Tree Traversal (Recursively)

- Inorder
- Preorder
- Postorder

Tree Traversal (Recursively)

- Inorder

```
void Inorder(node *nptr){
  if(nptr){
    Inorder(nptr->left);
    cout << nptr->number << endl;
    Inorder(nptr->right);
  }
}
```

Tree Traversal (Recursively)

- Preorder

```
void Preorder(node *nptr){
    if(nptr){
        cout << nptr->number << endl;
        Preorder(nptr->left);
        Preorder(nptr->right);
    }
}
```

Tree Traversal (Recursively)

- Postorder

```
void Postorder(node *nptr){
   if(nptr){
      Postorder(nptr->left);
      Postorder(nptr->right);
      cout << nptr->number << endl;
   }
}
```

Exercise 3

- Show that the second smallest of n elements can be found with

  $n + \lceil \lg n \rceil - 2$

comparisons in the worst case.

(*Hint:* Also find the smallest element.)

Exercise 3

- What about the minimum element?
- Conduct a tournament by using the pairs all the time.
- Consider a tree structure. Leaves are numbers, each inner node corresponds to comparions.
- By doing so, the minimum element can be found with n – 1 comparisons.

Exercise 3

- What about the second minimum?
- In the search for the smallest number, the second smallest number must have come out smallest in every comparison made with it until it was eventually compared with the smallest.
- So the second smallest is one of them !
- What to do now?

Exercise 3

- Second tournament is applied to this subset again.
- At most $\lceil \lg n \rceil$ elements (Depth of the tree).
- What about the number of comparisons?
- $\lceil \lg n \rceil - 1$

Exercise 3

- What about total number of comparisons?
- $n - 1 + \lceil \lg n \rceil - 1$
- $n + \lceil \lg n \rceil - 2$

Exercise 4

- Consider inserting the keys 10, 22, 31, 4, 15, 28, 17, 88, 59 into a hash table of length $m = 11$ using open addressing with the primary hash function $h'(k) = k \bmod m$.
- Illustrate the result of inserting these keys using **linear probing**, using **quadratic probing** with $c_1 = 1$ and $c_2 = 3$, and using **double hashing** with $h_2(k) = 1 + (k \bmod (m - 1))$.

Answer 4: Using Linear Probing

İTÜ

- **Linear probing:** $h(k,i) = (h'(k) + i) mod\, m$
- $h'(k) = k\, mod\, m$
- $m = 11,\ i = \{0, 1, 2, \ldots, m-1\}$
- Set of keys: $\{10, 22, 31, 4, 15, 28, 17, 88, 59\}$

$h(10,0) = 10$
$h(22,0) = 0$
$h(31,0) = 9$
$h(4,0) = 4$
$h(15,0) = 4$
$\boldsymbol{h(15,1) = 5}$

$h(28,0) = 6$
$h(17,0) = 6$
$\boldsymbol{h(17,1) = 7}$
$h(88,0) = 0$
$\boldsymbol{h(88,1) = 1}$

$h(59,0) = 4$
$\boldsymbol{h(59,1) = 5}$
$\boldsymbol{h(59,2) = 6}$
$\boldsymbol{h(59,3) = 7}$
$\boldsymbol{h(59,4) = 8}$

- The resulting hash table:
  - $H = \{22, 88, nil, nil, 4, 15, 28, 17, 59, 31, 10\}$

'ANBUL TEKNİK ÜNİVERSİTESİ

Answer 4: Using Quadratic Probing

- **Quadratic probing:** $h(k,i) = (h'(k) + c_1 i + c_2 i^2) mod\ m$
- $h'(k) = k\ mod\ m, c_1 = 1, c_2 = 3$
- $m = 11,\ i = \{0, 1, 2, \dots, m-1\}$
- Set of keys: $\{10, 22, 31, 4, 15, 28, 17, 88, 59\}$

$h(10,0) = 10$	$h(17,0) = 6$	$\boldsymbol{h(88,3) = 8}$
$h(22,0) = 0$	$\boldsymbol{h(17,1) = 10}$	$\boldsymbol{h(88,4) = 8}$
$h(31,0) = 9$	$\boldsymbol{h(17,2) = 9}$	$\boldsymbol{h(88,5) = 3}$
$h(4,0) = 4$	$\boldsymbol{h(17,3) = 3}$	$\boldsymbol{h(88,6) = 4}$
$h(15,0) = 4$	$h(88,0) = 0$	$\boldsymbol{h(88,7) = 0}$
$\boldsymbol{h(15,1) = 8}$	$\boldsymbol{h(88,1) = 4}$	$\boldsymbol{h(88,8) = 2}$
$h(28,0) = 6$	$\boldsymbol{h(88,2) = 3}$	$h(59,0) = 4$
		$\boldsymbol{h(59,1) = 8}$
		$\boldsymbol{h(59,2) = 7}$

- The resulting hash table:
  - $H = \{22, nil, 88, 17, 4, nil, 28, 59, 15, 31, 10\}$

Answer 4: Using Double Hashing

- **Double Hashing:** $h(k,i) = \big(h_1(k) + ih_2(k)\big) mod\ m$
- $h_1(k) = k\ mod\ m$ and $h_2(k) = 1 + k\ mod\ (m-1)$
- $m = 11,\ i = \{0,1,2,\dots,m-1\}$
- Set of keys: $\{10,22,31,4,15,28,17,88,59\}$

$h(10{,}0) = 10$	$\boldsymbol{h(15,2) = 5}$	$\boldsymbol{h(88,2) = 7}$
$h(22{,}0) = 0$	$h(28{,}0) = 6$	$h(59{,}0) = 4$
$h(31{,}0) = 9$	$h(17{,}0) = 6$	$\boldsymbol{h(59,1) = 3}$
$h(4{,}0) = 4$	$\boldsymbol{h(17,1) = 3}$	$\boldsymbol{h(59,2) = 2}$
$h(15{,}0) = 4$	$h(88{,}0) = 0$	
$\boldsymbol{h(15,1) = 10}$	$\boldsymbol{h(88,1) = 9}$	

- The resulting hash table:
  - $H = \{22, nil, 59, 17, 4, 15, 28, 88, nil, 31, 10\}$

İSTANBUL TEKNİK ÜNİVERSİTESİ
Asırlardır Çağdaş