**CS566 Assignment 3**

Yiduo Feng

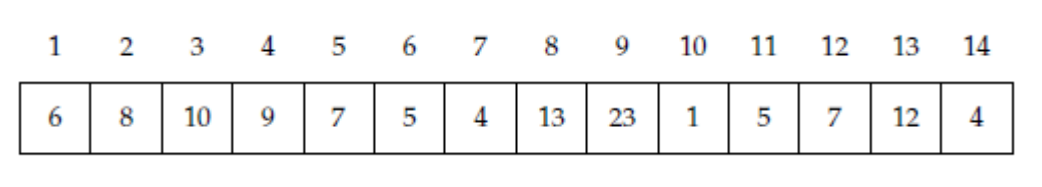
10/17/2022

**Tasks**

1. Max-Heapify (3 points): Illustrate the operation of MAX-HEAPIFY(A, 3) on the array

𝐴 = (6, 8, 10, 9, 7, 5, 4, 13, 23, 1, 5, 7, 12, 4)

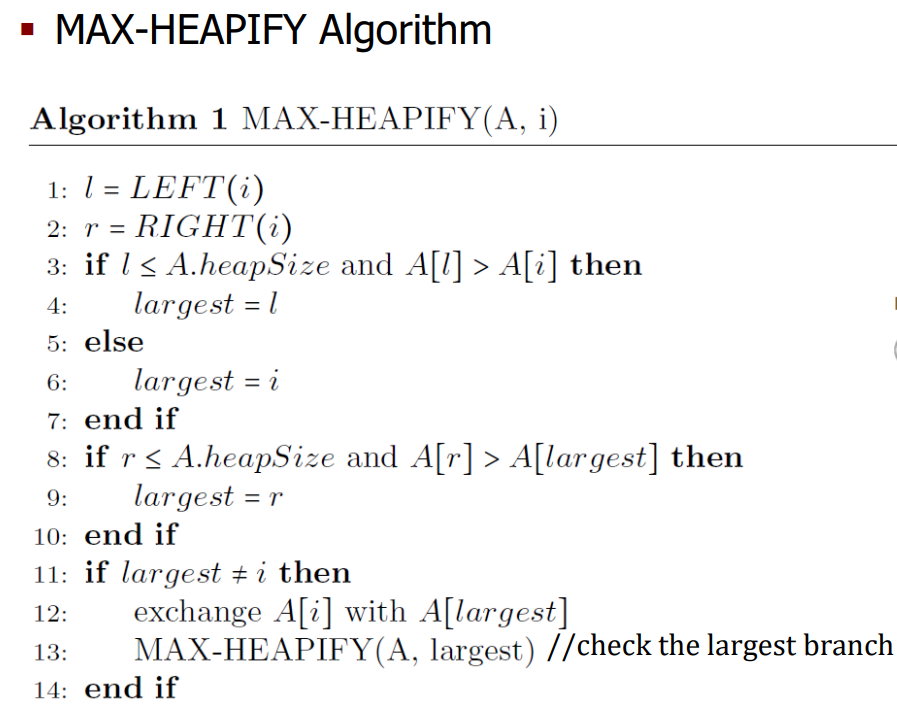
Write your steps, and count the number of total swap operations (exchange of keys)



• The first row is the index

• The second row is the array value (In a heap, the key)

According to the max heapify algorithm from course slide:



**Step1** : cur = 10(i = 3)

Left\_index =2i = 6; right\_index = 2i+1 = 7

Left = 5; right = 4

Because 10 > 5 > 4

There is no swap operation happened.

2. Heap-Sort (3 points): Run the HEAP-Sort Algorithm on the above array and count up the total

number of exchanges (You can assume the array has already been built as a heap)

• Write down the initial Max-heap array

• Write down the array after each key exchange in the HEAP-Sort.

• Write down the total number of key exchanges in the HEAP-Sort algorithm.

• No need to illustrate all your steps.

Note: Number of exchanges is the number of times that you swap the position of two keys (Values

of the array) until it is a sorted array

By algorithm BUILD-MAX-HEAP(A) do max heapify for every nodes in the tree except the leaf nodes. The initial Max-heap array is :

(23 13 12 9 7 10 4 8 6 1 5 7 5 4)

After each key exchange in the HEAP-Sort:

(1 4 4 5 5 6 7 7 8 9 10 12 13 23)

The total number of key exchanges is 43.

3. Heap-Median (4 points): Write pseudocode to extract the Median of an array using a heap structure.

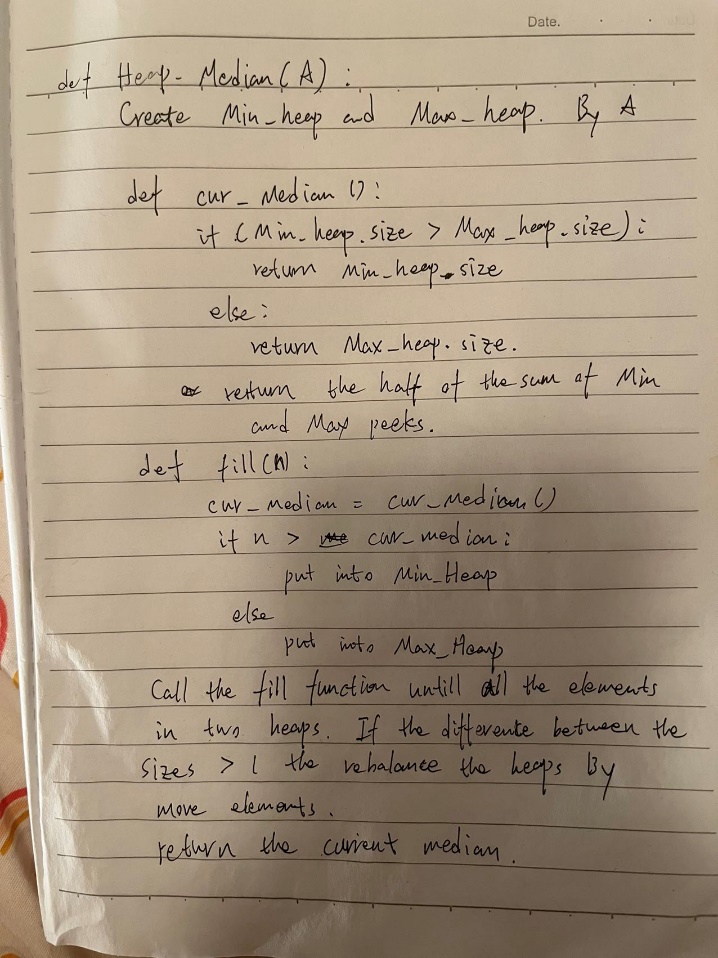
Name it Extract−Median(A) where A is an array. Develop an algorithm with the best running

time.

• You can use similar pseudocode syntax similar to the CLRS book or lecture slides.

• Discuss what is the running time of your best algorithm

Hint: The efficient algorithm to extract median using heap needs to build two heaps: Max-heap to hold smaller half part of the values; Min-heap to save bigger half part of the values. The size difference between two heaps should be within 1. The key algorithm in Extract-Median is Median-Heapify (you may refer Max-Heapify algorithm in the lecture).



Because heap is a tree with 2 children for each node, so for each elements the time complexity is log(n). There is n elements.

Therefore, the time complexity is O(n\*log(n))

4. Hashing by Chaining (3 points): Demonstrate what happens when we insert the keys

(6, 8, 10, 9, 7, 5, 4, 13, 23, 1, 5, 7, 12, 4) into a hash table with collisions resolved by chaining.

Let the table have 9 slots, and let the hash function be ℎ(𝑘) = 𝑘 𝑚𝑜𝑑 9

• Count the number of collisions

• How many collisions would the hash table have if you use ℎ(𝑘) = 𝑘 𝑚𝑜𝑑 7?

Note: Hash table indexes start from zero.

* ℎ(𝑘) = 𝑘 𝑚𝑜𝑑 9:

0:9

1:10-1

2:

3:12

4:4-13-4

5:5-23-5

6:6

7:7-7

8:8

the number of collisions: 6

* ℎ(𝑘) = 𝑘 𝑚𝑜𝑑 7:

0:7-7

1:8-1

2:9-23

3:10

4:4-4

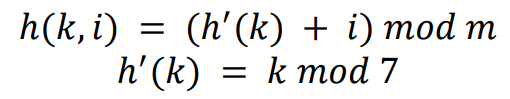
5:5-5-12

6:6-13

the number of collisions: 7

5. Open Addressing - Linear Probing (3 points): Consider inserting the keys {31, 11, 5, 17, 25} into a hash table of length m = 7 using open

addressing with the hash function



• Illustrate the result of inserting these keys using linear probing and provide insertion table index

results

• Count the number of collisions

31: 31%7 = 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 |  |  |  |

11:11%7 = 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 |  |  |

5: 5%7 = 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 | 5 |  |

17:17%7=3 (collision)

(17+1)%7(collision)

(17+1)%7(collision)

(17+3)%7 = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 | 5 | 17 |

25:25%7 = 4(collision)

(25+1)%7(collision)

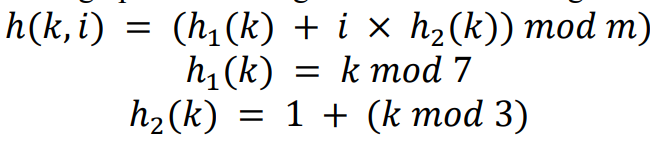
(25+1)%7(collision)

(25+3)%7 = 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 25 |  |  | 31 | 11 | 5 | 17 |

the number of collisions:6

6. Open Addressing - Double Hashing (4 points): Consider inserting the keys {31, 11, 5, 17, 25} into a hash table of length 𝑚 = 7 using open addressing with double Hashing



• Illustrate the result of inserting these keys using Double Hashing, and provide your

resulting hash table

31: 31%7 = 3

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 |  |  |  |

11:11%7 = 4

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 |  |  |

5: 5%7 = 5

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 | 5 |  |

17:h1 = 17%7=3 (collision)

H2 = 1+17%3 = 3(collision)

(3+0)%7 = 3(collision)

(3+3)%7 = 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  | 31 | 11 | 5 | 17 |

25:25%7 = 4(collision)

1+25%3 = 2(collision)

(4+0)%7 = 4(collision)

(4+2)%7 = 6(collision)

(4+4)%7 = 1

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  | 25 |  | 31 | 11 | 5 | 17 |