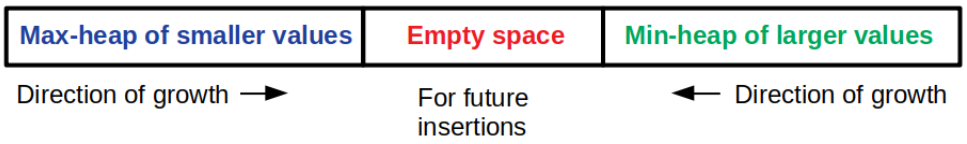
Algorithms Laboratory (CS29203)
Assignment 6: Heap data structure
Department of CSE, IIT Kharagpur

19th October 2023

Question-1 (100 points)

In the theory class, you have learned about max-heap where the maximum element can be found in constant time, and both insertion & deletion can be performed in logarithmic time. Similarly, in a min-heap insertion can also be done in constant time, and both insertion & deletion can be performed in logarithmic time. In this assignment, you will be building a special type of heap where the *median* element can be found in constant time, and both insertion & deletion can be performed in logarithmic time.
Let there are total n number of elements in our heap. Then if $n = 2k+1$ (i.e, odd), then the median is the $(k+1)$-th smallest element. If $n = 2k$ (i.e, even), the k-th smallest element is the median.
Our special type of heap is constructed by maintaining two heaps: a max-heap consisting of the smaller half of the values, and a min-heap consisting of the larger half of the values. The following figure demonstrates how a single array can be used for the contiguous representation of both the heaps back to back:



Let n_1 be the number of elements stored in the max-heap, and n_2 the number of elements stored in the min-heap. So we have $n_1 + n_2 = n$. We maintain the convention $n_1 = \lfloor n/2 \rfloor$ and $n_2 = \lceil n/2 \rceil$. Notice that the max-heap grows from the left, that is, the maximum in this heap can be found at the zeroth index of the array. On the other hand, the min-heap grows from the right end, that is, the minimum in this heap can be found at the $(n-1)$-th index of the array, where n is the total capacity of the array (including the empty space).
Your task is to write the following functions for the heap:

- **(20 points)** Write the *insert* and *deleteMax* functions for the max-heap, and the *insert* and *deleteMin* functions for the min-heap.
- **(10 points)** Write the *findMedian* function for our special heap.
- **(30 points)** Write the *insert* function for the special heap. Suppose that you want to insert x in a non-full heap. Let m be the current median in our special heap. If $x \leq m$, then you should insert x in the max-heap. But then if we already have $n_1 = n_2$, then this insertion will violate the above conventions regarding the new sizes of the two heaps. You therefore shift an element (which one is it?) from the max-heap to the min-heap, and then insert x in the max-heap. On the other hand, if $x > m$, then insert x in the min-heap after a size-adjusting transfer, if necessary.
- **(10 points)** Write the *deleteMedian* function for the special heap (this boils down to a deletion in the max-heap or in the min-heap).
- **(20 points)** Write a *medHeapSort* function that starts with a heap filled to the capacity (that is, with no empty space between the max-heap and the min-heap). The function iteratively deletes the current median. Each deletion creates an empty cell. The deleted median is copied there. After all of the n elements are deleted, the array should store the sorted list.
- **(10 points)** Write a main function which first reads n (the capacity of the array) from the user. It first initializes the array to an empty heap. Subsequently, it inserts n elements in the heap (you can manually enter the numbers or hard

code it). After each insertion, it prints the current median. After all the elements are inserted, the array is filled to the capacity. Now, the *medHeapSort* function is called. When it returns, the (sorted) array is to be printed.

Example:

```
n = 20

Insert elements one by one in MedHeap:

    Insert(3064)  Current median = 3064
    Insert( 545)  Current median =  545
    Insert(2978)  Current median = 2978
    Insert(5176)  Current median = 2978
    Insert(7432)  Current median = 3064
    Insert(2687)  Current median = 2978
    Insert(3003)  Current median = 3003
    Insert(9903)  Current median = 3003
    DeleteMedian()  Current median = 3064
    Insert(7991)  Current median = 3064
    Insert(7963)  Current median = 5176
    Insert(6184)  Current median = 5176
    Insert(5426)  Current median = 5426
    Insert(9981)  Current median = 5426
    Insert(8838)  Current median = 6184
    Insert(8053)  Current median = 6184
    Insert(1069)  Current median = 6184
    Insert(2950)  Current median = 5426
    Insert(3625)  Current median = 5426
    Insert(9130)  Current median = 5426
    Insert(8458)  Current median = 6184
    Insert(9070)  Current median = 6184


Median Heap Sort result:
      545 1069 2687 2950 2978 3064 3625 5176 5426 6184 7432 7963 7991 8053 8458
      8838 9070 9130 9903 9981
```