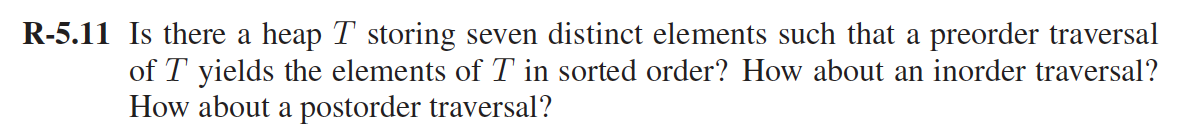
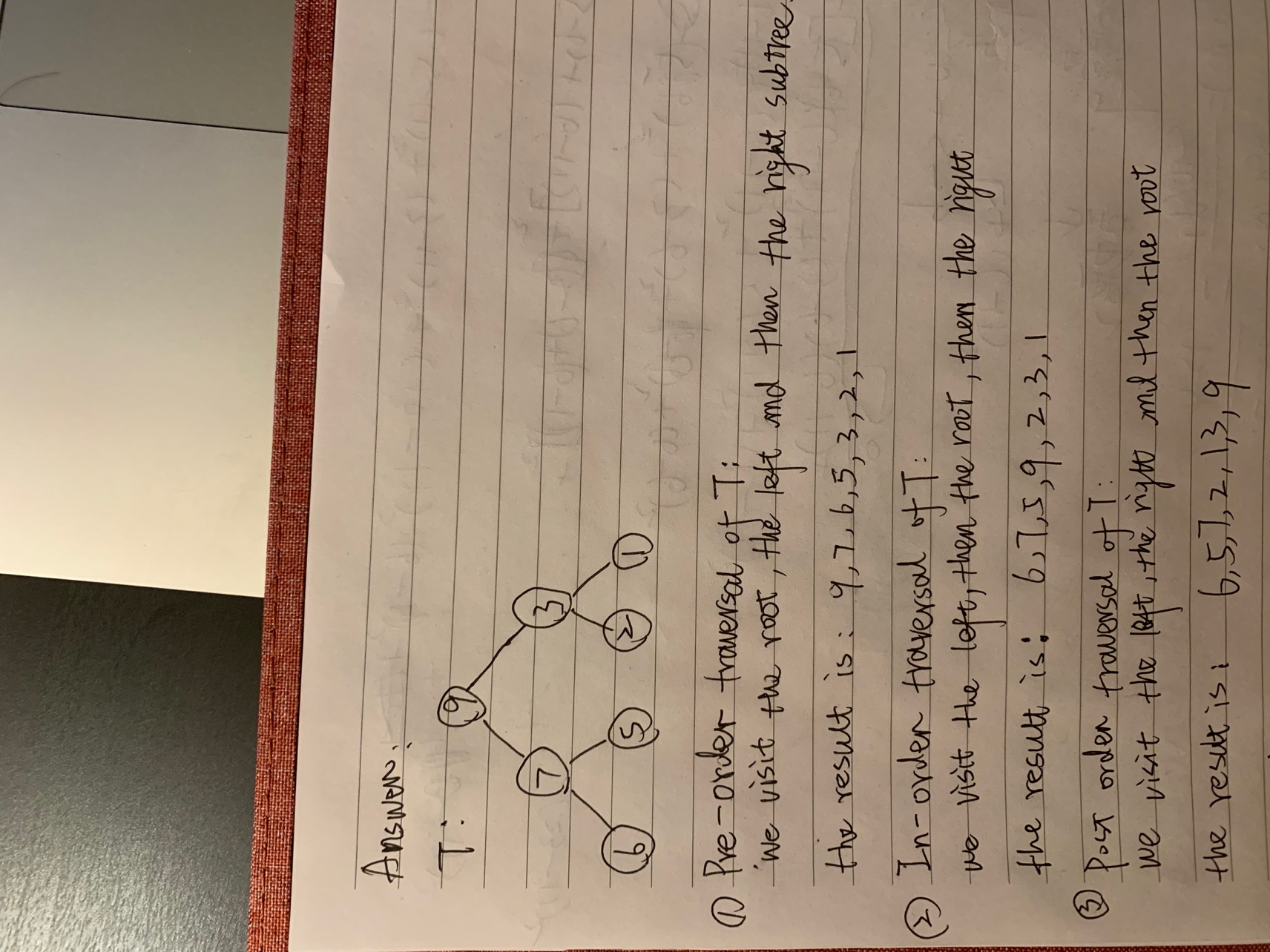
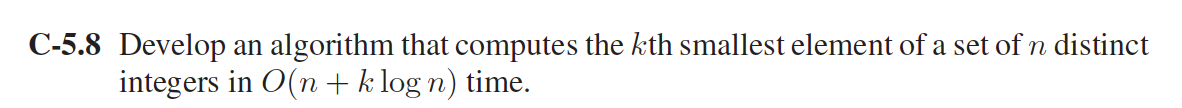
R-5.11,





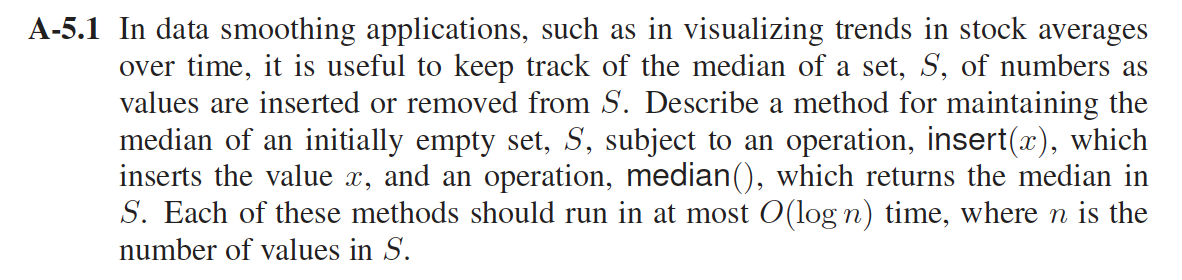
C-5.8,



1. build a min heap (build min heap method , all elements at a time ) ,the complexity will be O(n)
2. remove the root of the min heap for k times，which takes O(klogn)time

the total time is O(n+klogn) time.

A-5.1



Answer：

1. Median data structure using minHeap and maxHeap，the median() function, return average of largest element from maxheap and smallest from minheap.

Median():

     if (maxHeap.length==null)

    return null

if (maxHeap.length>minHeap.length)

          return max(maxHeap)// Return largest element from maxHeap

     return (max(maxHeap)+min(minHeap))/2

1. Method to insert an element into S should compare the median with x,

* if x is less than the return value of median(), Insert x into maxheap, find the difference of maxheap & minheap length, if the difference is 2 , Remove the largest element from maxheap and insert it into minHeap.
* If x is larger than the return value of median(), Insert x into the min heap, find the difference of maxheap & minheap length, if the difference is -1, then remove the smallest element from minheap, and insert it into maxheap.

insert(x):

     if ((Median()==null) || (x<=Median()))

     {

                 maxHeap.insert(x)

          if(maxHeap.length-minHeap.length==2)

              minHeap.insert(maxHeap.removeMax())

     }

     else

     {

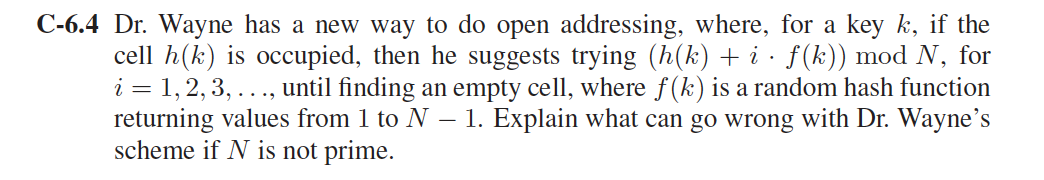
          minHeap.insert(x);

          if(maxHeap.length-minHeap.length==-1)

              maxHeap.insert(minHeap.removeMin())

     }

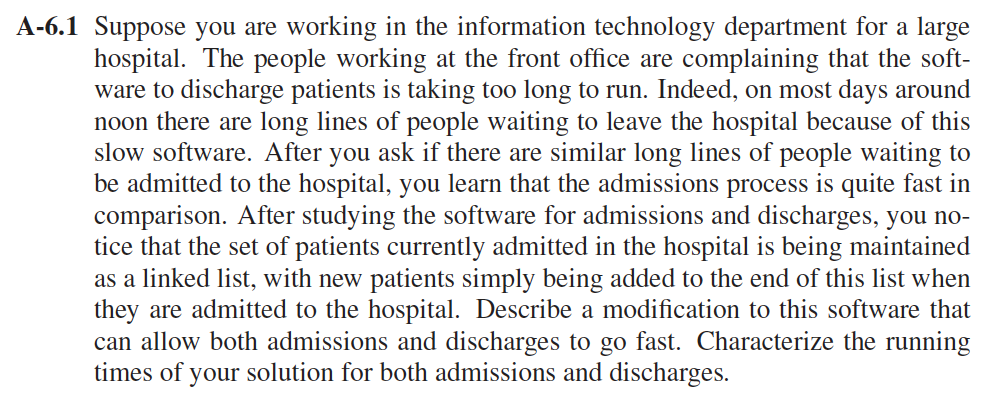
C-6.4,



Answer:

Let N=6 , and say f(k) produces 3 each time ,and( h(k) + i\*f(k)) mod N will map to h(k) + 0 or h(k)+3 , if these two cells are occupied, there will generate confiction.

A-6.1,



Answer：

Use Hash table to solve this problem

1. AdmitPatient(patient)

1.1Check if the patient exist by hash map lookup

1.2 if the patient not exist then add the patient at the last index of the array.

1.3 also add the patient to hash table, the key is the patient,and the value is the index of the array.

1. SearchPatient(patient)

Lookup for the patient in Hash map

1. dischargePatient(patient)

3.1check if the patient exist by using the SearchPatient(patient) method.

3.2 if patient exist, get the value(which is the array index) and remove that index from the hashmap

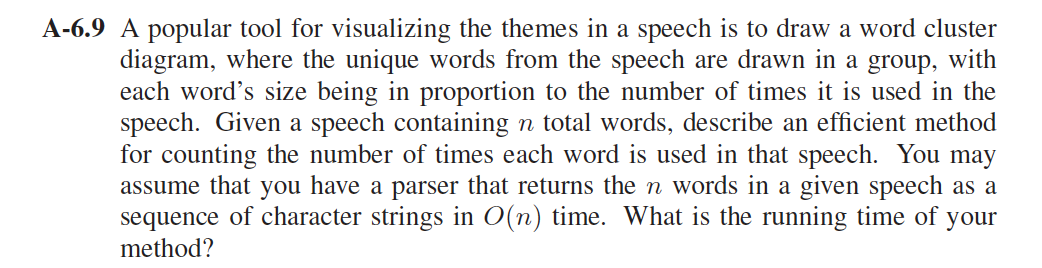
3.3 swap the current patient item with the last patient item in the array and remove the last patient item in the array.

The Time complexity of Hash table is O(1) in average and amortized time complexity. Because it is very rare case that many patients will be hashed to the same key and Rehash operation.

Rehash operation is of O(n) and will happen after n/2 operations and these are all assumption of O(1)

(O(n)+n\*O(1))/n= O(1).

A-6.9



Answer:

Use the hash table to solve this problem, we need key-value pairs,which key is the word, the value is the frequency words appeared in the speech.

CountWordFreq()

For word in TheGivenWords

1. check the word exist in the table by hash map lookup
2. if it exists then get the current value and add one to the value.
3. If it not exists then add the word to the hash table,and initiate the value

The running time of the method is O(n), since every hash map lookup takes O(1), and we traverse each word from the speech once for n times and maintain its count value.