SetA - Solutions

August 2017

Very Short Answer Questions

1. $\mathcal{O}(n+k)$. [2 points]
   For other answers like $\mathcal{O}(n)$ or $\mathcal{O}(k)$ reward [1 point].

2. Options A, B, and D. [2 points]
   For any other proper subset of the above, reward [1/2 point].

3. Option D.
   Binary Marking.

Short Answer Questions

4. We can use a heap/priority-queue to solve the problem. For any point a_i, the distance from the origin $d_i = \sqrt{(x_i^2 + y_i^2)}$. We store only 20 values in a MAX-HEAP of points sorted in order or their distance from origin. For all points with index greater than 20, check with top of heap and push the smaller point. Complexity: $\mathcal{O}(n \log 20)$

Algorithm 1 Find 20 nearest points to origin

```
PQ ← MAX_HEAP
for i in 1...20 do
  PQ.push(a_i)
end for
for i in 21..n do
  u ← PQ.extract_max()
  v ← a_i PQ.push(min(u, v))
end for
return PQ
```

   Comments: I can't think of any better solution, add any if it is exists. However, if there is no better solution. Reward 3 marks for any correct solution with the same time complexity. Reward 1 mark for any correct sub-optimal solution. E.g. Sort all points and choose first 20.

5. First sort the intervals in order of increasing ending time. Now for every a_i, let the ending time by r_i. The mutually overlapping subset containing a_i has the elements a_j such that $j >= i$ and $l_j <= r_i$. We call this subset S_i. $\mid S_i \mid = j_{last} - i + 1$. We can keep a track of the maximum value and output it in the end. Complexity : $\mathcal{O}(n \log n)$.

   Comments: Check correctness of solution and provide 3 points for correct solution with same time complexity. Any other sub-optimal solution should be rewarded 1 point.

6. This problem can be solved using binary search. Let the given point be k. First, we search for the first occurrence of k, let that be i. This can be done in $\mathcal{O}(\log n)$ Then, we search for the last occurrence of k, let that be j. So the answer becomes $j - i + 1$. Time complexity : $\mathcal{O}(\log n)$.

Algorithm 2 Find number of occurrences of k in sorted array A

```
l ← 0
h ← A.size
while l < h do
  m = (l+h)/2
  if A[m] < k then
    l ← m + 1
  else
    h ← m
  end if
end while
i ← l
l ← 0
h ← A.size
while l < h do
  m = (l+h)/2
  if A[m] <= k then
    l ← m + 1
  else
    h ← m
  end if
end while
j ← l − 1
return j − i + 1
```

Comments: 3 points for $\mathcal{O}(\log n)$. 1 point for $\mathcal{O}(n)$ solution.

Medium Answer Questions

7. Notice that there are two ways in which you can place your tiles - horizontally or vertically. However, when we put a tile horizontally, it is accompanied by another tile to complete the whole array. So that means, if we are at position i, we can put one tile vertically and reach position $i+1$. Also, we can put two tiles vertically and reach position $i+2$. So that means, we can see that to reach a position j, one way is to come from $j-2$ and the second way is to come from $j-1$. Now, let us see the base cases. If $n = 1$, there is only one way to fill it with tiles, also if $n = 2$, there are 2 ways to fill it with tiles. That means our recurrence is :

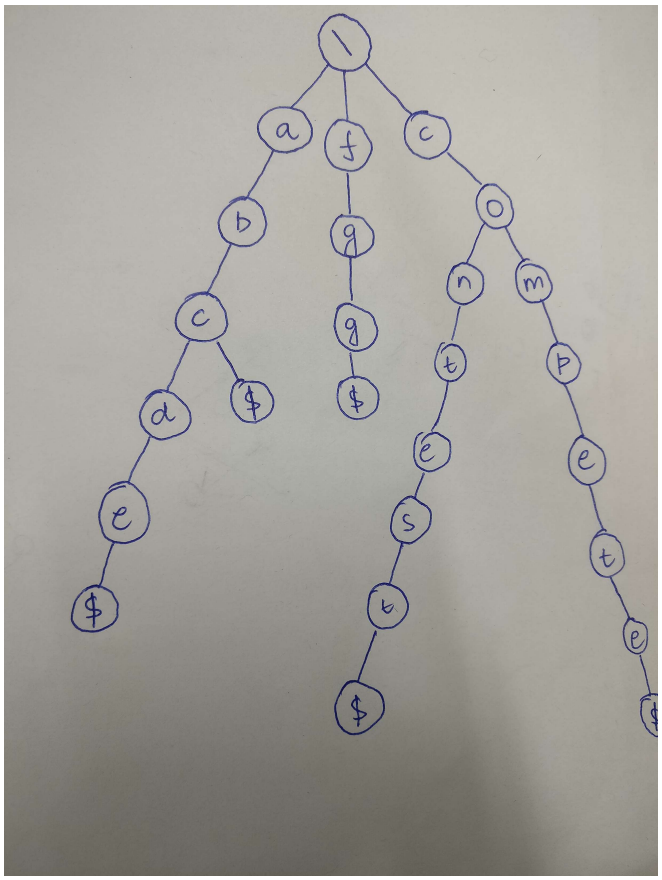
$$f(n) = \{ \begin{array}{l} f(n-1) + f(n-2); n > 2 \\ 2; n = 2 \\ 1; n = 1 \end{array} \}$$

8. Make an undirected graph with the given edge list. Choose a node s and perform a BFS from s marking all parents. Firstly, let us prove that the given graph is a tree. Then we will prove that it is a binary

tree. We know that a graph is a tree if there are exactly $n-1$ edges and the graph is connected. An equivalent proposition is that there is no cycle and the graph is connected. The proof is left as an exercise. We can easily if the second condition is met using the result of BFS from s. Now, to see that the tree is a binary tree, we can see if any node is a parent of at most 2 nodes. If any node is a parent of more than 2 nodes, it is not a binary tree. So the pseudo-code is as follows:

Algorithm 3 Checking whether the given graph is a binary tree or not.

```
G ← makeGraph()
s ← 1
D ← BFS(s).distance
P ← BFS(s).parent
if number of edges ! = n − 1 then
  return false
end if
for i in 1..n do
  if D[i] == INF then
    return false
  end if
end for
freqArray C ← P
for i in 1..n do
  if C[i] > 2 then
    return false
  end if
end for
return true
```

9. 

Comments: No time for making tries in latex. Keep binary marking in this problem. 5 for correct answer and 0 for wrong.

Long Answer Questions

10. We can be modelled as a graph problem. We can see that each word defines a relative ordering of the characters. We see that for any word w_i any letter at position j has a priority greater than any letter at a position $k > j$. Let us define a graph where we say that a directed edge from a to b means that the priority of a is greater than that of b. So, we can read the words in the input and for every letter j, we create an edge from j to $j + 1$. Moreover, for the word w_i, let us create an edge from the first letter of w_i to the first letter of w_{i+1}. After we have the graph, we can do a topological sort and get the ordering of the letters.

    Comments: Keep the uniqueness part as bonus.

11. We can model this as a graph problem. Let us make consider our vertices the number in our set. That means we have m vertices labelled $0..m - 1$.