Select algorithm (Median-finding algorithm) and Randomized Select Algorithm

1. Why do we use randomization algorithm? (2)
2. Why do we compute expected time complexity for randomized algorithm and not it's worst case running time? (2) ***
3. Give a pseudo-code of Select algorithm. Derive it's time complexity for group of 5 elements. / if the group size is 7?*
4. Prove or Disprove: Median select algorithm does not run in linear time when group size is 3. (4)**
5. Write the recurrence equation and solve it if select algorithm chooses a group of size 11. (4) *
6. How to use randomization to find the k-th smallest element from an array with linear expected time complexity? (2) ***
7. Let 'g' denote the size of group in divide and conquer select algorithm that find k-th smallest element. Derive the running time complexity in term of g. What happen if g is too large compared to 5. (5)***
8. Give a pseudo-code for randomized select algorithm. Suppose we use randomized selection algorithm to select the minimum element of the array A ={3, 2, 9, 0, 7, 5, 4, 8, 6, 1}. Describe a sequence of partition that results in a worst case performance. (5+5) ***
9. In randomized select algorithm for n elements, what is the probability that a worst case pivot is chosen for every call for select? (2) ***
10. Design an efficient randomized algorithm that generates a random permutation of the integers 1,2,….n. Assume that you have access to a fair-coin. Analyze the time complexity of your algorithm. (4)*
11. Given an unsorted array of 50 elements, design an algorithm with O(n) time complexity to print the least 25 elements. (4) **

Quick Sort & Randomized Quick Sort Algorithm

1. Pseudo code for Quick Sort. Derive the time complexity.
2. Can quick sort be made to run in O(nlogn) time in the worst case, assuming that all the elements are distinct. (2) *
3. Pseudocode for randomized quicksort. When randomized quicksort algorithm runs, how many calls are made to random number generator in the (i) worst case (ii) best case? Give your answer in terms of $\boldsymbol{\theta}$ notation. **

Closest pair & Randomized Closest pair

1. Define the closest pair problem. (2)*
2. Describe closest pair algorithm. Derive time complexity.
3. Define universal hash function. (2) **
4. Discuss how randomization is used in closest pair problem? (4) **
5. How hash table used in randomized closest pair. Prove that expected number of insert operation in the hash table in the randomized closest pair algorithm is O(n) (4)**
6. Describe the randomized closest pair algorithm. (6)**
7. Time complexity of randomized closest pair algorithm. *
8. Modify the closest pair algorithm such that Y array is not pre-sorted yet the running time complexity remain O(nlogn). (4) **

String Matching

1. Define String matching problem. (2)
2. Write a pseudocode for the naïve string matching algorithm to find the first occurrence of a pattern P of length m in a text T of length n. What is the worst case time complexity of this algorithm. (6)***
3. Compute the prefix function for the pattern '*ABBBACABB*' in a string matching algorithm. (2)**
4. Compute the prefix function Л for the pattern '*ababbab*' when the alphabet set is ababbab. (2)***
5. Draw a state transition diagram for a string matching automaton for the pattern '*ababbabbababbababbabb*' over the alphabet {a,b}. Compute the prefix function for the same pattern. (5+5)*
6. Suppose all characters in pattern p are different. Show how to accelerate the naïve-string matching algorithm to run in O(n) on a n-character text T. (4) ***
7. Suppose we allow a pattern P to contain occurrences of a gap character '<>' that can match an arbiratary string of characters (one of zero length). For example, the pattern ab<>ba<>c occurs in the text '*cabccbacbacab*' as '*cab<>ba<>cab*'
   Give a polynomial time algorithm to determine wheather such pattern P occurs in a given text and analyze the running time complexity of your algorithm. (10)*
8. Find the KMP prefix function for the string '*aaabbbabacab*'. (2)*
9. KMP algorithm and time complexity.*
10. How is string matching automaton computation difference from the computation of prefix function in KMP algorithm?*
11. Prove that, for any x and character a, we have $\boldsymbol{\sigma}(xa) <= \boldsymbol{\sigma}(x)+1$
12. Prove that, for any string x and character a, if $q=\boldsymbol{\sigma}(x)$, then $(xa)= \boldsymbol{\sigma}(P_q a)$.
13. If $\boldsymbol{\phi}$ is the final-state function of a string matching automaton for a given pattern p and T[1….n] is an input text for automaton, then $\boldsymbol{\phi}(T_i)= \boldsymbol{\sigma}(Ti)$.

Skip List

1. Node structure of a linked list.
2. Derive the expected case time complexity of the search in Skip list. *
3. State and prove the expected case time complexity of the search [a:b] in a randomized skip list. **
4. Prove that the number of tosses required to insert in a skip list is bounded by O(logn) with high probability. **
5. Give example of ideal skip list of 16 key values such that worst case search time complexity is O(logn). How many pointers are required to represent the skip list. *
6. Write an algorithm to merge two skip list.
7. Write a function to delete a given key from the skip list. Assume there is a pointer for pointing the node to be deleted after performing a search operation in the skip list. What is the time complexity? *