COL351: Analysis and Design of Algorithms

Tutorial Sheet - 7

October 8, 2022

Question 1 Design a divide-and-conquer algorithm to merge k sorted arrays, each with n elements, into a single sorted array of kn elements. What is the time complexity of this algorithm, in terms of k and n?

Question 2 You are given an n-node complete binary tree T of height h, so $n = 2^h - 1$. The nodes of T are labelled with distinct real numbers. A node in T is a local minimum if its label is smaller than the label of its neighbours. Device an algorithm to find a local minimum of T in $O(\log n)$ time.

Question 3 Given an n sized array A, the *Inversion Count* of A is the number of pairs (i, j) such that $A[i] > A[j]$ and $i < j$. So if A is already sorted, then the inversion count is 0, but if A is sorted in the reverse order, the inversion count is ${}^nC_2$. Design a divide-and-conquer algorithm to compute *Inversion Count* of an array A of size n in $O(n \log n)$ time.

Hint: Use ideas from Merge Sort.

Question 4 Show that the randomized quick sort can be implemented by just using $O(1)$ extra space.

Hint: See next page.

Question 5 Analyze the time complexity to compute Median of a list using Medians-of-Median algorithm (covered in Lecture 24) when the chunk size is (i) 3, and (ii) 7.

Implementation of Randomized Quick Sort

Algorithm 1: Randomized-Quick-Sort(A, L, R)

```
1 if (R ⩽ L) then Return;
2 q ← Random-index-from-interval([L, R])          /* Pivot is A[q] */
3 k ← Partition(A, L, R, q);
4 Randomized-Quick-Sort(A, L, k − 1);
5 Randomized-Quick-Sort(A, k + 1, R);
```

Algorithm 2: Partition(A, L, R, q)

```
1 k ← L + (No. of elements in A[L, R] smaller than A[q]);
2 Swap(A, q, k)                          /* Put pivot at correct index */
3 while (L < k < R) do
4     while (A[L] < A[k]) do L = L + 1;
5     while (A[k] ⩽ A[R]) do R = R − 1;
6     if (L < k < R) then Swap(A, L, R);
7 end
8 Return k;
```