CSL 630, Tutorial Sheet 2

1. Given a binary tree of N leaf nodes, show that the average distance from leaf nodes to the root node is $\Omega(\log N)$ for *any* binary tree.
   Hint: Suppose the minimum tree has x nodes in the left subtree and $N-x$ in the right subtree - minimize average path length wrt x.

2. If we are given a number x, let N_x denote the number of elements in a heap $\leq x$. Show how to output all elements $\leq x$ in $O(N_x)$ steps in a heap.
   In a binary min-heap of n elements, prove that the smallest k elements are present in a subtree rooted at the root (i.e. these elements are not arbitrarily scattered in the heap).

3. For n distinct elements $x_1, x_2 \ldots x_n$ with positive weights $w_1, w_2 \ldots w_n$ such that $\sum_i w_i = 1$, the *weighted median* is the element x_k satisfying

   $$\sum_{i|x_i<x_k} w_i \leq 1/2 \ \sum_{i|x_i \geq x_k, i \neq k} w_i \leq 1/2$$

   Describe an $O(n)$ algorithm to find such an element. Note that if $w_i = 1/n$ then x_k is the (ordinary) median.

4. Given two sorted arrays A and B of sizes m and n respectively, design an algorithm to find the median in $O(polylog(m+n))$.
   (You can do this in exactly $O(\log(m+n))$ steps).
   Can you generalize it to m sorted arrays ?

5. Use a divide and conquer based approach to find the maximum and minimum element among a set of n numbers in $3n/2$ comparisons.
   Can you do any better ?

6. * We want to sort n integers in the range $0..2^{b-1}$ (b bits each) using the following approach. Let us assume that b is a power of 2. We divide each integer into two $b/2$ bit numbers - say x_i has two parts x'_i and $x"_i$ where x'_i is the more significant part. We view the more significant bits as buckets and create lists of $b/2$ bit numbers by associating the lower significant $b/2$ bit numbers with the bucket with the more significant bits. Namely $x"_i$ is put into the list corresponding to x'_i. To merge the list we now add the $b/2$ bit numbers corresponding to the non-empty buckets to the earlier list (to distinguish, we can mark them). We can now sort the list of $b/2$ bit integers recursively and output the merged list by scanning the sorted elements. Note that this list can have more than n numbers since we added the buckets also. Suggest a method to avoid this blow up (since it is not good for recursion) and analyze this algorithm.

7. Consider a job scheduling problem where each job J_i has a start and a finish time (s_i, f_i). Two jobs cannot run simultaneously and once started, a job must run to its completion (i.e. we cannot split a job into parts). Given a set of jobs
   (i) If we schedule greedily in increasing order of finish times can we maximize the number of jobs completed ? Justify.
   (ii) If job J_i is associated with a profit p_i (≥ 0), can you apply a greedy algorithm to maximise the profit (of all completed jobs) ? Justify.

8. Consider a long straight road from left to right with houses scattered along the road (you can think of houses as points on the road). You would like place cell phone towers at some points on the road so that each house is within 4 kilometers of at least one of these towers. Describe an efficient algorithm

which achieves this goal and uses as few cell phone towers as possible.
Hint: Consider a solution where each tower is located as much to its right as possible (without changing the number of towers). How would you construct such a solution ?

9. We are given a set of events (with starting and finishing times) that have to be scheduled in a number of halls without conflicts. Design an algorithm to find the minimum number of halls needed for this purpose. Note that the timings are fixed and no two events can happen at the same time in the same hall.
   You can think about the events as intervals on the real line such that that we have to assign a colour to each interval in a way that no two overlapping intervals are assigned the same colour. What is the minimum number of colours required ?

10. Given a graph with negative weights, can you transform it to a graph that doesn't have any negative weights and the shortest paths between every pair of vertices remain unchanged. The lengths may change but the sequence of edges do not change. Given this transformation, we can then run Dijkstra's algorithm.
    Hint: Define weights on the vertices using a function U such that weight of an edge (x, y) is redefined as $w'(x, y) = w(x, y) + U(x) - U(y)$. Choose a function U such that $w'(x, y) \geq 0$. Then the weight of a path $x_1, x_2, x_3, \ldots x_k = w(x_1, x_2) + U(x_1) - U(x_2) + w(x_2, x_3) + U(x_2) - U(x_3) \ldots$. The intermediate $U(x_i)$ will cancel out.

11. Design an efficient algorithm to compute the second shortest path in a weighted graph without negative weights.
    Hint: Modify Dijkstra's algorithm appropriately.

12. The second minimal spanning tree is one that is distinct from the minimal spanning tree (has to differ by at least one edge) and is an MST if the original tree is ignored (they may even have the same weight). Design an efficient algorithm to determine the second MST.

    Hint: Show that the second MST differ from the MST by exactly one edge.

13. Show that
    (i) By excluding the heaviest edge h_C in any simple cycle C of a weighted graph G, the MST of $G(V, E - h_C)$ is no worse than the MST of $G(V, E)$.
    (i) By including the lightest cut edge of any cut A, the MST is no worse than te MST of the original graph.

14. Describe an efficient algorithm to find the *girth* of a given undirected graph. The *girth is defined as the length of the smallest cycle.*

15. Given a directed acyclic graph, that has maximal path length k, design an efficient algorithm that partitions the vertices into $k + 1$ sets such that there is no path between any pair of vertices in a set.