COL 702, Tutorial Sheet 1

1. Solve the following recurrence equations given $T(1) = O(1)$

   (a) $T(n) = T(n/2) + bn \log n$
   (b) $T(n) = aT(n-1) + bn^c$

2. Show that the solution to the recurrence $X(1) = 1$ and

$$X(n) = \sum_{i=1}^{n} X(i)X(n-i) \text{ for } n > 1$$

   is $X(n+1) = \frac{1}{n+1}\binom{2n}{n}$

3. Instead of the conventional two-way mergesort, show how to implement a k-way ($k \geq 2$) mergesort using appropriate data structure in $O(n \log n)$ comparisons. Note that k is not necessarily fixed (but can be a function of n).

4. **(Multiset sorting)** Given n elements among which there are only h distinct values show that you can sort in $O(n \log h)$ comparisons.
   Further show that if there are n_α elements with value α, where $\sum_\alpha n_\alpha = n$, then we can sort in time

$$O(\sum_\alpha n_\alpha \cdot \log(\frac{n}{n_\alpha} + 1))$$

5. Modify the integer multiplication algorithm to divide each integer into 4 parts and count the number of multiplications and additions required for the recursive approach. Write the recurrence and solve it and compare it with the divide-by-2 approach.

6. In the selection algorithm, if we choose a random element as a splitter, then show that the expected running time is $O(n)$. Prove the correctness and analyse the algorithm rigorously.

   Hint : Write a recurrence and solve for it which is similar to the expected time analysis of quicksort.

7. Given a set S of n numbers, $x_1, x_2, \ldots x_n$, and an integer k, $1 \leq k \leq n$, design an algorithm to find $y_1, y_2 \ldots y_{k-1}$ ($y_i \in S$ and $y_i \leq y_{i+1}$) such that they induce partitions of S of roughly equal size. Namely, let $S_i = \{x_j | y_{i-1} \leq x_j \leq y_i\}$ be the $i-th$ partition and assume $y_0 = -\infty$ and $y_k = \infty$. The number of elements in S_i is $\lfloor n/k \rfloor$ or $\lfloor n/k \rfloor + 1$.
   Note: If $k = 2$ then it suffices to find the median.

8. An element is *common*, if it occurs more than $n/4$ times in in a given set of n elements. Design an $O(n)$ algorithm to find a *common* element if one exists.

9. Construct an example to show that MSB first radix sort can be asymptotically worse than LSB first radix sort.

10. Given two polynomials $P_A(n) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \ldots a_0$ and $P_B(n) = b_{n-1}x^{n-1} + b_{n-2}x^{n-2} + \ldots b_0$, design a subquadratic ($o(n^2)$) time algorithm to multiply the two polynomials. You can assume that the coefficients a_i and b_i are $O(\log n)$ bits and can be multiplied in $O(1)$ steps.
    Note: Don't use Fast Fourier Transform based methods since it has not been discussed in class.

11. Prove that subgraph returned by Dijkstra's algorithm (or Bellman Ford) is a directed tree rooted at source with $n-1$ edges.

12. Given a directed acyclic graph, design a linear time algorithm for computing a SSSP in $O(|V| + |E|)$ time.

13. Let A be an $n \times n$ adjacency matrix of a directed graph $G = (V, E)$ with $A_{i,i} = 0$. We define a operation $B = A \oplus A$ as follows

$$B_{i,j} = \min_{1 \leq k \leq n} \{a_{i,k} + a_{k,k}\}$$

Note the similarity with normal matrix multiplication where we use $\times$ and $+$ instead of $+$ and $\min$. (i) Prove that $B_{i,j}$ equals the shortest path of at most 2 edges between vertex i and vertex j.
(ii) Prove that $B = A \oplus A \oplus \ldots \ell \ \ times \oplus A$ stores the shortest path with at most ℓ edges between i and j in $B_{i,j}$
(iii) Design a fast algorithm to compute A^ℓ under the operation $\oplus$

14. Given a graph G with negative weights (no negative cycles), we want to transform it to another equivalent graph G' that preserves the shortest paths of G but doesn't contain any negative weights.
(i) If we add to all edges a weight greater than the largest negative weight, will shortest paths be preserved ?
(ii) Let $d(v)$ be equal to the shortest path distance to v from source vertex s. Suppose we add to every edge (u, v), the weight $d(u) - d(v)$, i.e. the new weight $w'(u, v) = w(u, v) + d(u) - d(v)$. Then show that

    (a) $w'(u, v) \geq 0$
    (b) Between all pairs of vertices x, y, for two distict paths P_1 and P_2, $w(P_1) \geq w(P_2)$ iff $w'(P_1) \geq w'(P_2)$.