CSc445 Homework #3 Due: 10/20/2015

Instructions.

1. Solution could **not** be submitted by students in pairs.
2. You could submit a pdf of the homework, either printed or hand-written and scanned, as long as it is **easily** readable.
3. If your solution is not clearly written, it might not be graded.
4. Prove the correctness of your answer. A correct answer without a proof might not be accepted.
5. If you have discussed the solution with other students, mentioned their names clearly on the homework. These discussions are not forbidden and are actually **encouraged**. However, you must write your whole solution yourself.
6. All questions have same weight.
7. You could refer to a data structure studied in class, and just mention briefly their guaranties. For example "*It is known that a Red-Black tree could support the insert/delete/find operations on a set of n elements in time $O(\log n)$.*
8. If your answer uses one of the data structures or algorithms that were studied in class, you could refer to it without having to repeat details studied in class. If you answer requires only modifications of one of the algorithm, it is enough to mention the required modifications, and what's the effect (if any) on the running time and on other operations that the algorithm performs.
9. In general, a complete answer should contain the following parts:
   (a) High level description of the data structures (if needed). *E.g. We use a binary balanced search tree. Each node contains, a key and pointers to its children. We augment the tree so each node also contains a field...*
   (b) High level description of the algorithms
   (c) Proof of correctness (why your algorithm provides what is required).
   (d) A claim about the running time, and a proof showing this claim.

In all discussions about graphs, assume n denote the number of vertices and m is the number of edges. In questions 1,2 and 3 we refer to Rabin-Karp algorithm for string matching. The notation is as used in the slides.

1. This algorithm is used for finding all values s such that $T[s+1 \ldots s+m] = P$. Assume that we actually only looking for a matchings where $s = 16k$ where $k \geq 0$ is an integer. (that is, $s = 0, 16, 32, 48, \ldots$). Explain how to modify the algorithm so t'_s is computed only for these values of s. For example, the case t'_3 need not be computed.

2. Assume the value of the prime q is known to you. You are also given integers m, n. Explain how to generate a text $T[1..n]$ and a pattern $P[1..m]$ for which the number of false positive matchings ($t_s \neq p$ but $t'_s = p'$) is $\Omega(n/m)$.

3. Would you prefer q to be a smaller or a larger prime ? Discuss for example the case $q = 2$ and $q = 3$ vs. the case $q > 2^{20}$. How does this influence the way you construct Π ?

4. Let $G(V, E)$ be a **directed** graph with positive weights on its edges, and let $s \in V$ be a node. Suggest an $O((m + n) \log n)$ time algorithm that finds, for every vertex $v \in V$, the length of the shortest path from v to s.

5. Assume you run Dijkstra algorithm on a directed graph that contain edges with arbitrary weights (some edges with positive wights, some with negative weights). However, the graph does not contain a negative cycle. Show an example in which the resulting output of the algorithm is incorrect.

6. Let $G(V, E)$ be an undirected graph, with positive weights given for its edges, and assume $S_0, S_1, \ldots S_k \subseteq V$ be subsets of V. (that is, each S_i might contain several vertices). Define
$$\delta(S_i, S_j) = \min_{x \in S_i,\ y \in S_j} \delta(x, y),$$
Suggest an algorithm that computes $\delta(S_0, S_i)$ for every i, in time $O((m + n) \log n)$. Assume that $S_i \cap S_j = \emptyset$ for every $1 \leq i < j < k$.

7. Let $G(V, E)$ be an undirected graph with positive and negative weights on its edges. Suggest an algorithm that in time $O(m \log m)$ finds the smallest value d such that between every pair of vertices $u, v \in V$ there is a path where the weights of each edge on this path is $\leq d$.

   Note that we are not interested in the sum of weights.

8. Let $G(V, E)$ be a *directed* graph, and assume that each **vertex** $v_i \in V$ is assigned with a **positive** weight w_i. Edges do not have weights. We define the *vertex-cost* of a path in the graph to be the sum of weights of the *vertices* along this path. Given vertices $s, t \in V$, suggest an algorithm with running time $O((m+n) \log(m+n))$, that computes a path from s to t with minimum vertex-cost.

   Hint: Generate a new graph $G'(V', E')$ where $|V'| = 2n$ and $|E'| \leq m + n$, and run Dijkstra on this graph.

9. Given a graph $G(V, E)$, where each edge (u, v) is associated with a weight $w(u, v)$, which is an integer between 1 to 17. Assume s and t are vertices in V. Suggest an $O(m + n)$-time algorithm for computing the shortest path from s to t.

10. Let $G(V, E)$ be a directed graph, $s \in V$ is a source. You are running Dijkstra's algorithm on this graph. Define

$$U = \{v \in V | \text{ there is no path from } s \text{ to } v \text{ in } G(V, E)\}$$

Explain how you could identify the vertices in U, by reading the array $\pi[1 \ldots n]$ given as output of Dijkstra algorithm.