Algorithms Qual Exam 2017

1. This exam is closed book, closed notes, and closed computers.
2. Please write proofs clearly and concisely, stating clearly any assumptions you make. These papers need to be graded fast, so there will be negative points for rambling.
3. Pseudo-code is not required; a clear description of the algorithm in English suffices.

Problem 1. [**20 points**] The following two problems concern removing edges to make a graph acyclic. Such an edge set is termed a "feedback edge set".

1. **(10 points)** Given a weighted connected undirected graph, give an efficient algorithm to find the minimum weight subset of edges whose deletion makes the resulting graph have no cycles in it. You can use any standard graph algorithm as a subroutine. What is the running time of your algorithm?

2. **(10 points)** Given a directed unweighted graph $G(V, E)$ with n vertices and m edges, give a $O(m + n)$ time algorithm to remove at most half the edges from E so that the resulting graph has no directed cycles.

Problem 2. **[20 points]** You are given a binary heap of size n (with the largest element on top), represented as an array. Given a value x and a number k, give an algorithm to decide whether the value of the k^{th} largest element of the heap is at most x. The running time should be $O(k)$. Note that the algorithm *does not* need to explicitly find the value of the k^{th} largest element – all it needs is to decide if the value is at most x.

The running time must depend only on k and *must* be independent of the size of the heap n, else you get no credit. The algorithm can use $O(k)$ extra space.

As an example, if $k = 1$, the algorithm simply compares the first element of the array with x, so the running time is $O(1)$.

Problem 3. **[25 points]** Given a graph $G(V, E)$, the minimum vertex cover problem asks you to find the smallest subset $S^* \subseteq V$ of vertices, such that for each edge $e = (u, v) \in E$, at least one of u, v is present in S^*. The decision version of this problem is NP-Complete.

Consider the following greedy algorithm for the vertex cover problem: Initialize $T \leftarrow \Phi$. Consider the edges of G in arbitrary order, and for each edge $e \in E$ in that order, include e in T iff e does not share *any* end-point with *any* edge previously included in T. The set T of edges so formed is termed a *greedy maximal matching*. Now take the union of all end-points of the edges in T and call this set of vertices S. Output set S as the vertex cover.

1. Prove that the set S is indeed a vertex cover, in the sense that each edge $e \in E$ has at least one end-point in S.

2. Show by an example that the set S constructed by the greedy algorithm need not be the *minimum* vertex cover, in other words, it could be that $|S| > |S^*|$.

3. Prove that $|S| \leq 2|S^*|$. In other words, the greedy algorithm is a *2-approximation* for vertex cover.

Problem 4. [**35 points**] You are given a directed graph with m edges and n nodes, with integer edge lengths which could be positive or negative. Each node in this graph is colored with one of k colors. A *variegated* path is a directed simple path (which does not repeat vertices) such that all vertices on the path have distinct colors. The goal is to find the variegated path of minimum length. Note that the path could start at any node and end at any node. Furthermore, the graph could have negative length cycles.

1. **(20 points)** Give a dynamic programming algorithm which runs in time $O(f(k)g(m,n))$. Here, $f(k)$ is an exponential function of k (*not* depending on m, n); and $g(m,n)$ is a polynomial function of m and n (*not* depending on k). For instance, your running time could look like $O(k^{2k} \cdot m^2 \cdot n^3)$ but *not* like $O(m^k \cdot n^2)$. Precisely state the dynamic programming table, as well as the running time of your algorithm.

2. **(15 points)** Show that if m, n, k are all specified as part of the input, the problem of deciding if there is a variegated path of length at most a given value L is NP-Complete. You can use the NP-Completeness of any standard graph problem to perform the reduction.