CS3230 : Tutorial - 2

Bakh Khoussainov

28-Aug-2011, 1pm deadline

1. Design an algorithm that given n points $p_1, \ldots, p_n$ in a line, outputs two points with the closest distance. Is your algorithm greedy? Give a very short explanation to your answer.

2. Say you are in country X. The currency of country X uses coins in values of 1, 5, 10, 20, 25, and 50. Assume that you have unlimited number of coins of each type. Given amount M, you would like to find minimum number of coins that sum up to M. This is known as the change problem. Prove or disprove that the greedy algorithm that takes the maximal number of coins of the highest value solves the change problem. For instance, for $M = 197$ the coins produced by this algorithm are $50, 50, 50, 25, 20, 1, 1$.

3. One algorithm that sorts lists is called the *Selection Sort Algorithm*. The input to the algorithm is a list $A = a_1, \ldots, a_n$ of integers. The number n is called the **size** of the input. The *Selection Sort Algorithm* proceeds as follows:

   - Find the smallest number in A.
   - Swap the smallest number with the value in the first position of A.
   - Repeat the steps above for the reminder of the list.

   The **basic operation** in execution of this algorithm is the *comparison operation*. For instance putting the smallest element into the first position requires $n - 1$ comparisons by scanning the whole list from left to right.

   Prove that the *Selection Sort Algorithm* on input A of size n makes at most n^2 comparisons in order to give a sorted output.

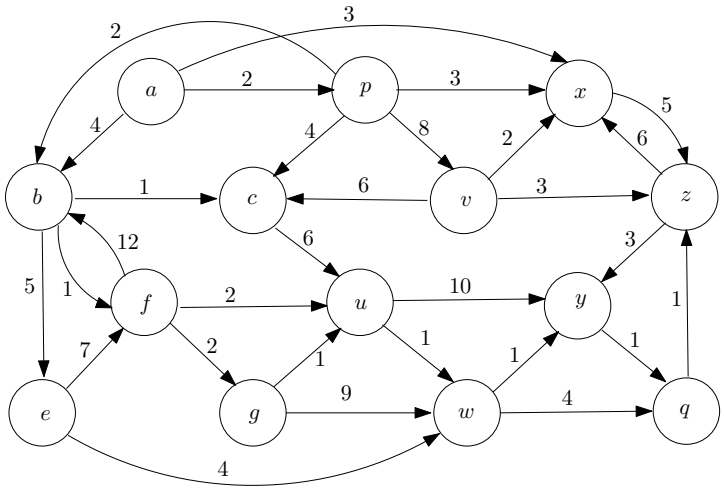
4. Our algorithms from the lectures for solving problems related to intervals use two **basic operations**. The first operation determines if one interval overlaps with the other. The second operation is the comparison operation that is used for sorting the intervals either by their finishing time (Interval scheduling problem) or by starting time (Interval partitioning problem) or by earliest deadlines (lateness minimisation problem). The comparison operation compares the intervals by the finishing or starting times or deadlines depending on the problem. For each of these algorithms

inputs are intervals of $I_1, \ldots, I_n$. The number n is called the **size** of the input. For each of these algorithm, given an input of size n, find the upper bound on the total number of basic operations needed to obtain a solution to the problem. Your bound should be expressed in terms of n. You need to give your answer in the following two cases:

Case 1. The inputs are sorted. Namely, for the interval scheduling problem the intervals are sorted by finishing time, for the interval partitioning problem the intervals are sorted by their starting time, and for the lateness minimisation problem the requests are sorted by their deadlines.

Case 2. The inputs are not sorted.

5. Apply Dijkstra algorithm (proved in the lecture) to the graph depicted below. Your starting vertex is vertex a. Write down each stage of your algorithm, the set S, and the distance values d for the vertices in the graph.



6. Let G be a directed weighted graph. We always assume that the weights are non-negative. Let $v_0, v_1, \ldots, v_n$ be a path P. Recall that **the weight (or the cost) of this path** P, denoted by $w(P)$, is the sum of the weights of its edges, that is:

$$w(P) = \sum_{i=0}^{n-1} w(e_i),$$

where e_i is the edge from v_i to v_{i+1}.

The shortest path-distance from a vertex u to a vertex v is the minimum weight among the weights of all paths from u to v. We denote this shortest path-distance by $\delta(u, v)$. Thus,

$$\delta(u, v) = \min \{w(P) \mid P \text{ is a path from } u \text{ to } v \text{ in } G\}.$$

A **shortest path from** u **to** v is then any path P whose weight is equal to the shortest path-distance $\delta(u, v)$ from u to v.

(a) Show by example that if negative weights are allowed, then the shortest path-distance from u to v might not exist even if there is a path from u to v.

(b) Prove that a subpath of a shortest path is again a shortest path.

(c) For all vertices $x, y, z \in V$ of the directed weighted graph G, the following triangle inequality is true:

$$\delta(x, z) \leq \delta(x, y) + \delta(y, z).$$

7. Consider the lateness minimisation problem.

(a) Suppose we use the following rule to schedule the intervals. Always select the intervals that require the most amount of time (that is start by selecting the interval whose length is the largest among all intervals). Prove or disprove that the greedy algorithm based on this rule produces an optimal solution.

(b) Suppose we use the following rule to schedule the intervals. Always select the intervals that require the least amount of time. Prove or disprove that the greedy algorithm based on this rule produces an optimal solution.