**CS 577: Introduction to Algorithms** 10/1/2015

# Homework 4

Instructor: Dieter van Melkebeek TAs: Kevin Kowalski, Andrew Morgan, Bryce Sandlund

This assignment covers graph promivites and the "greedy-stays-head" paradigm. Good luck!

## Review problems

1. Suppose that you are given a digraph with nonnegative costs on the edges, and two vertices $s$ and $t$. You want to find a cheapest path from $s$ to $t$. However, the cost for an edge can grow over time. More precisely, the cost of using edge $e$ as the $i$th edge of a path is given by a nondecreasing function $c_e(i)$ of $i$.

   Suppose you are given a subroutine that, on input the graph $G$, edge $e$, and index $i$, returns $c_e(i)$, and that you use the latter quantities in Dijkstra's algorithm. Give an example where this algorithm returns a suboptimal path from $s$ to $t$.

2. The Department of Recreation has decided that it must be more profitable, and it wants to sell advertising space along a popular jogging path at a local park. They have built a number of billboards (special signs for advertisements) along the path and have decided to sell advertising space on these billboards. Billboards are situated evenly along the jogging path, and they are given consecutive integer numbers corresponding to their order along the path. At most one advertisement can be placed on each billboard.

   A particular client wishes to purchase advertising space on these billboards but needs guarantees that every jogger will see its advertisement at least $k$ times while running along the path. However, different joggers run along different parts of the path.

   Interviews with joggers revealed that each of them has chosen a section of the path which he/she likes to run along every day. Since advertisers care only about billboards seen by joggers, each jogger's personal path can be identified by the sequence of billboards viewed during a run. Taking into account that billboards are numbered consecutively, it is sufficient to record the first and the last billboard numbers seen by each jogger.

   Unfortunately, interviews with joggers also showed that some joggers don't run far enough to see $k$ billboards. Some of them are in such bad shape that they get to see only one billboard (here, the first and last billboard numbers for their path will be identical). Since out-of-shape joggers won't get to see $k$ billboards, the client requires that they see an advertisement on every billboard along their section of the path. Although this is not as good as them seeing $k$ advertisements, this is the best that can be done and it's enough to satisfy the client.

   In order to reduce advertising costs, the client hires you to figure out how to minimize the number of billboards they need to pay for and, at the same time, satisfy stated requirements. Design an algorithm that produces an optimal selection of billboards given $k$ and the first and last billboard numbers of each of the $n$ joggers.

   Your algorithm should run in time $O(n \log n)$. You may first want to aim for a solution that runs in time $O(kn + n \log n)$. You can assume that comparisons and standard arithmetic operations take one unit of time.

## Graded written problem

3. [10 points] Your summer job is drive kayakers from the parking lot to the kayak launch platform. The bus you are driving can take up to $k$ kayakers; a round-trip between the parking lot and the launch platform lasts $m$ minutes. You are given an alphabetical list with the names of all $n$ kayakers for a given day, together with the times they will arrive at the parking lot. The kayakers need to be served in the order they arrive but you can decide for each roundtrip when the bus leaves and how many kayakers you take along.

   You would like to organize your schedule in such a way that you are done for the day as early as possible. Design an algorithm that computes the earliest time you can be done. Your algorithm should take $O(n \log n)$ steps.

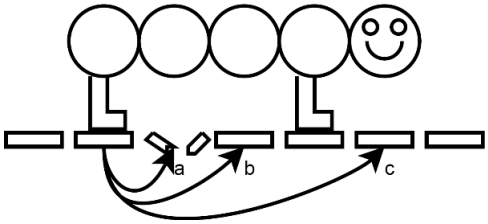
   For example, if $k = 20$, $m = 30$, and there is only a group of 25 people that day, all arriving at noon, then the earliest you can be done is 1:00pm; if in addition there is a group of 5 people arriving at 12:45pm, then the earliest you can be done is 1:15pm.

## Additional written problem

4. For the opening scene of a computer game, you want the main character, Wormly, to cross a bridge. Wormly is a worm made of $k$ equal circular bubbles and $\ell$ legs. At all times each leg has to be under one of the bubbles, and under each bubble there can be at most one leg. The bridge was supposed to be composed of $n$ planks with the width of each plank equal to the diameter of each of Wormly's bubbles. However, some of the planks are missing.

   At every moment, Wormly can do exactly one of the following:

   - Move one of its legs forward over any number of (possibly missing) planks. After the move, the leg should be on a plank and underneath one of Wormly's bubbles. A leg isn't allowed to overtake other legs.
   - Move all of its bubbles forward one plank while its legs remain on the same planks. After the move each leg must still be under one of Wormly's bubbles.

   In the above figure, the only possible move for the last leg is to position $b$. This is because the plank at position $a$ is missing, so the leg cannot move there; to get to position $c$, the last leg would have to overtake the first leg. Also, in this example, moving all the bubbles forward is not allowed because Wormly's last leg would end up without a bubble over it.

   Initially Wormly's bubbles are directly above the leftmost $k$ planks of the bridge and its legs are on the leftmost $\ell$ planks. At the end of the animation Wormly's bubbles have to be directly above the rightmost $k$ planks and its legs have to be on the rightmost $\ell$ planks. The left- and rightmost $\ell$ planks of the bridge are not missing.

Develop an algorithm to determine the smallest number of steps for the animation when given $k$, $\ell$, and a binary string of length $n$ where the $i$th bit indicates whether the $i$th position has a plank. Your algorithm should run in time $O(n)$.

## Optional programming problem

5. [2.5 points] Solve SPOJ problem Mice and Maze (problem code MICEMAZE).

## Challenge problem

6. In some courses you can choose a certain number $k$ of the $n$ assignments that will be dropped in the calculation of your grade. If all the assignments counted equally, the choice would be easy: simply drop the assignments with the lowest scores. However, each assignment may have a different maximum score. Your final homework grade will be the percentage ratio of your total score to the maximum possible score for the retained assignments. This leads to the following problem. You are given a value of $k$ and a list of $n$ assignment results $(s_i, m_i)$, $1 \leq i \leq n$, where $s_i$ denotes your score on the $i$th assignment and $m_i$ denotes the maximum possible score on that assignment. Your goal is to find a set $I \subseteq \{1, 2, \ldots, n\}$ with $|I| = k$ such that $\sum_{i \notin I} s_i / \sum_{i \notin I} m_i$ is as large as possible.

   Give an algorithm that runs in time $O(n^2 \log n)$. For starters, aim for an algorithm that runs in time polynomial in the number of bits in the input.