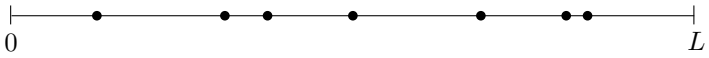
CS 473: Algorithms, Fall 2009
HBS 6

Problem 1. Consider a road with houses scattered along the road. (We can picture the road as a horizontal line segment and the houses as points on this segment, as shown in the figure below.) The residents of these houses are avid cell phone users and therefore we want to place cell phone towers at certain points along the road so that each house gets a very good cell phone reception and the number of towers is as small as possible.



We are given an array $D[1 \cdots n]$ of integers, where $D[i]$ is the distance of the i-th house from the west end of the road, and an integer d. Give an algorithm that finds the minimum number of towers that we need to place on the road and the locations of the towers so that the distance between any house and its closest tower is at most d.

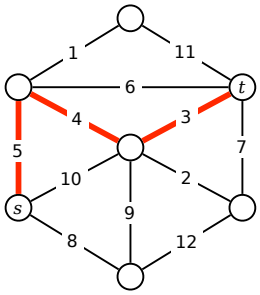
Problem 2. The wildly popular Spanish-language search engine El Goog needs to do a serious amount of computation every time it recompiles its index. Fortunately, the company has at its disposal a single large supercomputer, together with an essentially unlimited supply of high-end PCs.

They have broken the overall computation into n distinct jobs, labeled $J_1, J_2, \ldots J_n$, which can be performed completely independently of one another. Each job consists of two stages: first it needs to be preprocessed on the supercomputer, and then it needs to be finished on one of the PCs. Let us say that job J_i needs p_i seconds of time on the supercomputer, followed by f_i seconds of time on a PC.

Since there are at least n PCs available on the premises, the finishing of the jobs can be performed fully in parallel — all the jobs can be processed at the same time. However, the supercomputer can only work on a single job at a time, so the system managers need to work out an order in which to feed the jobs to the supercomputer. As soon as the first job in order is done on the supercomputer, it can be handed off to a PC for finishing; at that point in time a second job can be fed to the supercomputer; when the second job is done on the supercomputer, it can proceed to a PC regardless of whether or not the first job is done (since the PCs work in parallel); an so on.

A *schedule* is an ordering of the jobs for the supercomputer, and the *completion time* of the schedule is the earliest time at which all jobs will have finished processing on the PCs. Give an algorithm that finds a schedule with minimum completion time.

Problem 3. Consider a path between two vertices s and t in an undirected, connected, and weighted graph G[1]. The bottleneck length of this path is the maximum weight of any edge in the path. The bottleneck distance between s and t is the minimum bottleneck length of any path in G from s to t.



The bottleneck distance between s and t is 5.

(a) Show that there exists a spanning tree T of G such that, for *every* pair of nodes s and t, the bottleneck length of the unique $s-t$ path in T is equal to the bottleneck distance between s and t.

(b) Give an algorithm that finds such a tree T.

[1]There is a weight associated with each edge of G.