CIS507: Design & Analysis of Algorithms
Homework 3, Spring 2014 (With answers)

Q1. (2 points) Exercise 24.4–8 in the text book. Let $Ax \leq b$ be a system of m difference constraints in n unknowns. Show that the Bellman-Ford algorithm, when run on the corresponding constraint graph, maximizes $\sum_{i=1}^{n} x_i$ subject to $Ax \leq b$ and $x_i \leq 0$ for all x_i.

ANSWER:

1. Start with proving $x_i \leq 0$ for all x_i.
   We have $d[v_0] = 0$ and $w(v_0, v_i) = 0$ in the beginning in constraint graph. After the "relaxation" step in Bellman-Ford algorithm, the shortest path $\delta(v_0, v_i) \leq 0$ because "relaxation" step only makes the path shorter (or at least the same). According to Theorem 24.9, $x_i = \delta(v_0, v_i)$ is a feasible solution thus $x_i \leq 0$ for all x_i.

2. Then prove $\sum_{i=1}^{n} x_i$ is maximized.
   The proof is equal to $x_i \leq \delta(v_0, v_i)$ for $i = 1, \ldots, n$ and can be proved by induction.

   (a) Base case: Since $x_i \leq 0$ for all x_i, there must be an x_i satisfying $x_i \leq \delta(v_0, v_1)$.

   (b) Suppose $x_{i-1} \leq \delta(v_0, v_1)$ is true, consider path from v_0 to v_i and the corresponding constraint $x_i - x_{i-1} \leq b_k$. By induction we have $x_{i-1} \leq \delta(v_0, v_1)$, then $x_i - x_{i-1} \leq b_k \Rightarrow x_i \leq x_{i-1} + b_k \leq \delta(v_0, v_1) + b_k = \delta(v_0, v_i)$.

   From the proof above, we have maximized $\sum_{i}^{n} x_i$ when run on the corresponding constraint graph.

Q2. We are given a set of jobs $\mathcal{J}$ to be scheduled on an unlimited number of machines. Each job $j \in \mathcal{J}$ has a processing time $p_j > 0$. There is also a set of *precedence constraints* describing, for each job j, the set of jobs that have to be completed before starting job j. We would like to find, for each job j the *earliest scheduling time* S_j, that is, the first point in time at which the job can be started without violating any precedence constraints.

(I) **(1.0 point).** Model this as a graph problem and give a sufficient and necessary condition for the existence of a feasible schedule.

(II) **(2.0 point).** Give an efficient algorithm for either finding the earliest scheduling times S_j, for $j \in \mathcal{J}$, or declaring that no feasible schedule exists. Analyze the running time of the algorithm.

(III) **(2.0 point).** Suppose now that, instead of processing times, we are given for some pairs of jobs (i, j) the *time delay* $t_{ij} \in \mathbb{R}$ (which can be *negative*), imposing that $S_j \geq S_i + t_{ij}$, that is, job j can only be started after t_{ij} units of time from the start time of job i. Model this as a graph problem and give an efficient algorithm for either finding the earliest scheduling times S_j, for $j \in \mathcal{J}$, or declaring that no feasible schedule exists. Analyze the running time of the algorithm.

(IV) **(3 points).** Implement both algorithms in (II) and (III). For testing purposes, your program in (II) should accept as an input a file "test.in" containing the number n of jobs, followed by a list of n lines; line j contains p_j followed by the list of jobs that have to be completed before starting job j (separated by spaces); your program in (III) should accept as an input a file "test.in" containing the number n of jobs, followed by a list of n lines; line j contains pairs (i, t_{ij}) (separated by spaces). The two programs should output in another file "test.out" either the earliest scheduling times $S_1, \ldots, S_n$ (separated by spaces), or the word "INFEASIBLE", if there is no feasible schedule.

ANSWER: Life is awesome in the United Arab Emirates!

Q3.

1. **(1 point).** Several families go out to dinner together. To increase their social interaction, they would like to sit at tables so that no two members of the same family are at the same table. Show how to formulate finding a seating arrangement that meets this objective as a maximum flow problem. Assume that the dinner contingent has p families and that the ith family has $a(i)$ members. Also assume that q tables are available and that the jth table has a seating capacity of $b(j)$.

2. **(2 points). Exercise 26.3-5 in the text book.** We say that a bipartite graph $G = (V, E)$, where $V = L \cup R$, is d-regular if every vertex $v \in V$ has degree exactly d. Every d-regular bipartite graph has $|L| = |R|$. Prove that every d-regular bipartite graph has a matching of cardinality $|L|$ by arguing that a minimum cut of the corresponding flow network has capacity $|L|$.