CS 202 Homework 1
Graph Search, Tuning Algorithms with Data Structures
Due Thursday, April 24

General directions:

Algorithms may be described at "high level" without actual code. You may use any lower bound, algorithm or data structure from the text or in class, and their correctness and analysis, but be careful. For example, if you construct a graph with n^2 nodes and n^3 edges and then run Dijkstra's algorithm on the resulting graph, the total run-time is $O(n^3 logn)$,because Dijkstra's algorithm is $O(ElogV)$. If you want to use the correctness of Dijkstra's algorithm on the graph, you must check that edge weights are positive, since otherwise the proof that Dijkstra's algorithm gives shortest paths does not go through. If a problem has multiple size parameters, you should express the run-time as a function of all relevant parameters; e.g., saying maximum bi-partite matching takes time $O(|V||E|)$ is more accurate than to say it takes time $O(V^3)$, although both are correct.

For problems 2-5, you must prove correctness and give a time analysis. For some problems, correctness may be trivial; for others the analysis may be trivial. If it is trivial, you do not need to go into detail, but you should at least give a one or two line explanation. (Conversely, if you only give a one or two line explanation, I will view this as implicitly claiming that it is trivial.)

Each problem is worth 10 points. Grading may be based on any of the following that are relevant for the problem: the efficiency of your algorithm; the correctness and proof thereof; and the time analysis. The number of points depending on each part is given after the problem, as well as a ballpark estimate of the time analysis for my solutions of the problem.

Data structures for all pairs shortest paths You need to solve single source shortest path repeatedly on the same underlying graph G with but with different (non-negative) edge weights. Describe what data structures you would use in Dijkstra's algorithm if :

- **a. 2d grid** G is the $n^{1/2} \times n^{1/2}$ grid, whose nodes are pairs (i,j) with $1 \leq i,j \leq n^{1/2}$ and where (i,j) is connected to $(i+1,j), (i-1,j), (i,j+1)$ and $(i,j-1)$ for $1 < i,j < n^{1/2}$.
- **b. cycle of complete bipartite graphs** G is a cycle of $n^{1/2}$ complete bipartite graphs, each of size $n^{1/2}$. In other words, nodes are as above, but (i,j) is connected to each node $(i',j+1)$ and each node $(i',j-1)$.
- **c. complete bipartite graph** G is the complete bipartite graph with $n/2$ nodes on each side. (All points for efficiency, 4,3,3 pts per problem.)

Flight scheduling You are devising a flight scheduler for a travel agency. The scheduler will get a list of available flights, and the customer's origin and destination. For each flight, it is given the cities and times of departure and arrival. The scheduler should output a list of flights that will take the customer from her origin to her destination that arrives as early as possible, subject to giving her at least 15 minutes for each connection.

a. Give a formal specification for this problem (Instance, Solution Space, Constraints, Objective).

b. Give as efficient as possible an algorithm to solve the problem.

(2 points specification, 4 points correct algorithm, 4 points efficiency)

Palindromic path: You are given a deterministic finite automaton, with alphabet Σ and state space V, specified by its *transition diagram*, a directed graph where each node $v \in V$ has exactly one out-edge labelled with each $\sigma \in \Sigma$, as well as an initial node s_0 and a set of accepting nodes $T \subseteq V$. A word $w \in \Sigma^*$ is accepted by the DFA if the path starting from s labelled by w has its endpoint in T. A palindrome is a word that reads the same backwards as forwards. Give an efficient algorithm to tell whether the DFA accepts any palindrome. Analyze its time in terms of $|V|$ and $|\Sigma|$. (5 pts correct poly-time algorithm and correctness proof, 5 pts efficiency.)

Energy Contracts: UCSD needs long-term contracts for power to keep the lights on. It has a list of n bids $B_I, 1 \leq I \leq n$ from power companies, each with a *rate* (cost per megawatt) r_I and a *capacity* C_I (the maximum number of megawatts the company can guarantee). The regents, to encourage low bids, has guaranteed that they will pay for all accepted bids at the highest rate of any accepted bid. The accepted bids' capacities must sum to at least M, the university's demand for power, to ensure enough power. In addition to the money paid per megawatt, the regents expect each contract to cost a fixed amount F for lawyers, setting up connections to the grid, etc. So the total cost will be $Fk + rM$, where k is the number of accepted bids, and r is the maximum rate of an accepted bid. Give an efficient algorithm, polynomial-time in n, that, given F, M and the list of n bids B_I, computes a subset of bids A that minimizes the cost to the university subject to ensuring enough power. (3 pts correct poly-time algorithm and correctness proof, 7 pts efficiency and time analysis.)

Implementation problem: Popular Websites A web-company wants a data structure that will display webpages by popularity, displaying the most popular. The input will be a sequence of IP addresses. Intermittently, the data structure will need to display the top k most frequently visited sites (where k is given by the user). The data structure should update its list after every new site hit. So the data structure needs to store a list of webpages, ordered by $hit(site)$, the number of hits on the site. It needs to update this list each time a new hit is made, i.e., Update(site) adds *site* to the list with 1 hit, if it isn't already there, or increments $hit(site)$ if it is. Top(k) needs to report the top k most popular sites. Describe and implement at least two data structures for this problem. Compare their performances on test data generated as follows:

Test distribution: A sequence of 1,000,000 random web addresses, each with probability 1/4 of being of the form: random 3 letter word.cs.edu probability 1/4 of the form: free.random 3 letter word.com and probability 1/2 of the form: random 2 letter word.random 2 letter word.com, org, edu. All words are lower case and have only standard letters. After every 1000 sites, perform $Top(k)$ for $k = 2^i$, i uniformly chosen from 0 to 10.

Discuss any conclusions or issues that arose.