

CS270 Problem Set 1 – Spring 2005

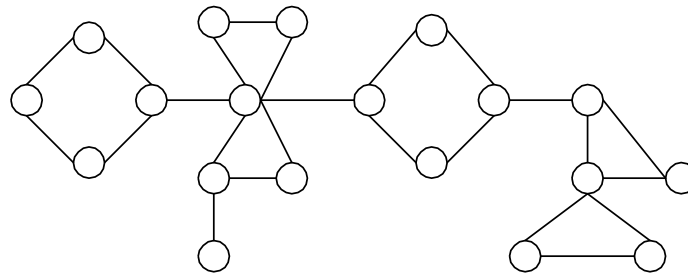
Due: Tuesday, February 15

Rules: You can work in groups and consult papers and books. But you should write your paper by yourself, and you should give credit in it where credit is due, sources and colleagues. Please also write briefly, neatly, and correctly.

1. **Drunken Sailor.** Jack the drunken sailor is walking on a pier of length n , starting at position 0. In each step he either increases his distance from 0 by 1 or decreases it by 1, each with probability $\frac{1}{2}$ — unless he is at 0, from where he always goes to 1.
 - (a) Show that Jack falls of the pier in n^2 steps, with at least some constant probability. (Find a constant that works.)
 - (b) Given your constant from the first part, how long must you wait to ensure that Jack falls of the pier with probability at least $1 - 1/n$?
2. **Biconnected Components.** We discussed strongly connected components (SCCs) as the basis of a fundamental decomposition of directed graphs: *Every directed graph is the DAG of its SCCs.* Undirected graphs also have a decomposition theorem: *Every undirected graph is the tree of its biconnected components.* Let us explain:

In a connected undirected graph (V, E) an *articulation point* is a vertex whose removal disconnects the graph. A *bridge* is an edge whose removal disconnects the graph. A biconnected component is a *maximal set*¹ of edges with the property that any two edges participate in a simple cycle (one without repetitions of vertices). Hence bridges are singleton biconnected components.

- (a) In the graph below, identify the articulation points, bridges, and biconnected components.



- (b) Show that the starting point of a depth first search is an articulation point if and only if there are two or more tree edges out of it.
- (c) Let v be a non-root vertex in the depth-first search tree. Prove that v is an articulation point of G if and only if v has a “bad child” s such that there is no back edge from s or any descendant of s to a proper ancestor of v .

¹That is, if any other edge is added to the set, the property is no longer true.

(d) Let $pre[v]$ be the ordering in which DFS visits nodes (preorder), and let

$$low[v] = \min\{ pre[v], \min\{pre[w] : (u, w) \text{ is a back edge for some descendant } u \text{ of } v\} \}.$$

Note that u is a descendant of itself. Show how to compute $low[v]$ for all vertices $v \in V$ in $O(|E|)$ time.

(e) Show how to compute all articulation points in $O(|E|)$ time.

3. **Fattest augmenting path.** The *fattest path* between s and t is a path that maximizes the value of the smallest edge on the path.

(a) Give an $O((n + m) \log n)$ time algorithm for finding the fattest s - t path in a weighted graph, where (as usual) $n = |V|$ and $m = |E|$.

(b) Show that implementing the Ford-Fulkerson algorithm by augmenting along the fattest path in the residual network finds the optimal solution with $O(m \log |f|)$ augmenting paths, where f is the maximum flow. (Hint: show you route at least $1/m$ of the “residual” flow with each augmenting path. In particular, show how to decompose a flow function into m paths using depth first search on the flow graph.)

4. **Arbitrage.** Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of currency into more than one unit of the same currency. For example, suppose US\$1 buys 0.96 Euros, 1 Euro buys 1.66 Canadian dollars, and 1 Canadian dollar buys US\$0.68. Then, by converting currencies, a trader can start with US\$1 and buy $0.96 \times 1.66 \times 0.68 = 1.083$ US dollars, thus turning an 8.3% profit.

Suppose we are given n currencies c_1, \dots, c_n and an $n \times n$ table R of exchange rates, such that one unit of currency c_i buys $R[i, j]$ units of currency c_j .

(a) Give an efficient algorithm to determine whether or not there exists a sequence of currencies $(c_{i_1}, \dots, c_{i_k})$ such that

$$R[i_1, i_2] \cdot R[i_2, i_3] \cdot \dots \cdot R[i_{k-1}, i_k] \cdot R[i_k, 1] > 1.$$

Analyze the running time of your algorithm.

(b) Give an efficient algorithm to print out such a sequence if one exists. Analyze the running time of your algorithm.

5. **Shortest paths with negative edge weights.** Given a graph G , a potential function $p : V \rightarrow Z$ is a mapping of the vertices to integers. Given integer edge weights $c : E \rightarrow Z$, we define the *reduced cost* under the potential function $p(\cdot)$ of an edge $e = (u, v)$ to be $c_p(u, v) = c(u, v) - p(v) + p(u)$.

(a) Given that the length of the shortest path under $c_p(\cdot, \cdot)$ between u and v has value W , what is the length of the shortest path between u and v under $c(\cdot, \cdot)$? Why?

(b) A potential function is feasible if all the reduced costs are nonnegative. Give an $O(nm)$ time algorithm to find a feasible potential function if one exists. (Hint: one does not exist if there is a negative cycle.)

(c) Give an $O(nm + k(n \log n + m))$ time algorithm for doing k single source shortest path computations in a graph with negative edge weights. (Hint: you may assume that Dijkstra’s algorithm runs in time $O(n \log n + m)$.)

6. **Project.** A typical project will involve (a) choosing a problem, perhaps from your research interests outside Theory, that seems to you interesting and susceptible to algorithmic/theoretical analysis; (b) formulating it, studying it, hopefully solving it; and (c) ideally, publishing the results. You’ll get help along the way. You can work in groups of 1-4.

Please write a paragraph describing possible project topic(s) you are considering.