Massachusetts Institute of Technology
6.854J/18.415J: Advanced Algorithms
David Karger

Handout 7
Wednesday, September 26, 2012

Problem Set 4

Due: Wednesday, October 3, 2012.

Collaboration policy: collaboration is *strongly encouraged*. However, remember that

1. You must write up your own solutions, independently.
2. You must record the name of every collaborator.
3. You must actually participate in solving all the problems. This is difficult in very large groups, so you should keep your collaboration groups limited to 3 or 4 people in a given week.
4. **No bibles. This includes solutions posted to problems in previous years.**

Problem 1. In class we saw how to build a perfect hash table for a set of keys given in advance. In this problem we will generalize this to allow insertion and deletion of keys in the table online. Recall that perfect hashing involves a top-level table hashing to multiple second-level tables using 2-universal hash functions. We initially consider only inserts. The basic approach is to *rebuild* any table (top level or second tier) when it violates the invariants on which perfect hashing relies.

(a) Consider a second-level table, that in the static case uses $f(s) = O(s^2)$ space to store s items. We gave a construction of such a table that succeeds with probability 1/2. Suppose that we take the following approach to insertions: if inserting the s^{th} item violates perfection of the table, repeatedly build a table of size $f(2s)$ until you get a perfect one. Show that when s items are inserted (one at a time, with rebuilds as necessary) the expected total cost of all the rebuilds is $O(s)$.

(b) Let s_i be the number of items in bucket i in the second level table. Make a similar modification for the top level: any time the table on n items has bucket sizes s_i violating the requirement that $\sum s_i^2 = O(n)$, we rebuild the top table and all the second level tables. Show the expected total time is $O(n)$ when n items are inserted.

(c) Now consider deletions. When an item is deleted, we can simply mark it as deleted without removing it (and if it reinserted, unmark the deletion). This makes deletion cheap. But if there are many deletions, the table might end up using much more space than it needs, which could mess up the amortized bounds of the previous parts. To fix this, we can rebuild the hash table any time the

number of "holes" (items marked deleted) exceeds half the total items in the table. Show that this cleanup work does not change (can be charged against) the amortized $O(1)$ cost of insertions.

NONCOLLABORATIVE Problem 2. Which of the following statements about flows are true and which are false? Justify your answer with a (short) proof or counterexample.

(a) In any maximum flow, and for all vertices v and w, either the raw flow from v to w or the raw flow from w to v is 0. (The net flow between two vertices v and w is the raw flow from v to w minus the raw flow from w to v, where the raw flow on a directed edge is less than the capacity of that directed edge.)

(b) There always exists a maximum flow such that, for all vertices v and w, either the raw flow from v to w or the raw flow from w to v is 0.

(c) If all directed edges in a network have distinct capacities, then there is a unique maximum flow.

(d) If we replace each directed edge in a network with two directed edges in opposite directions with the same capacity and connecting the same vertices, then the value of the maximum flow remains unchanged.

(e) If we add the same positive number λ to the capacity of every directed edge, then the minimum cut (but not its value) remains unchanged.

(f) Flow is transitive: if in a graph there is a flow of value v from s to t (i.e. with source s and sink t), and there is a flow of value v from t to u, then there is a flow of value v from s to u.

Problem 3. The US Census Bureau produces a variety of tables from its Census data. Suppose that it wishes to produce a p by q table $D = \{d_{ij}\}$ of non-negative integers. Let r_i denote the sum of the matrix elements in the i^{th} row, and let c_j denote the sum of the elements in the j^{th} column. Assume that each sum r_i and c_j is strictly positive. The Census Bureau often wishes to disclose all the row and column sums along with some matrix elements (denoted by the set Y) and yet to suppress the remaining elements to ensure the confidentiality of privileged information. Unless it exercises care, by disclosing the elements in Y the Bureau might permit someone to deduce the exact value of one or more suppressed elements. It is possible to deduce the exact value of an element d_{ij} if only one value of d_{ij} is consistent with the row and column sums and the disclosed elements in Y. We say that any such suppressed element is *unprotected*. We wish to develop an efficient, flow-based algorithm for identifying all the unprotected elements of the matrix and their values.

(a) Given the table as described above, give an efficient, flow-based algorithm for deciding if there is *some* set of cell values that produces the disclosed values.

(b) Given a graph G and a max-flow f that places flow value f_e on edge e, give an efficient algorithm for determining whether there is a max-flow that places a *different* flow value on edge e.

(c) Combine the previous two parts to solve the stated problem.

Problem 4. At lunchtime it is crucial for people to get to the food trucks as quickly as possible. We will show how to use maximum flow to solve this problem. Consider the following model. The building is represented by a graph $G = (V, E)$, where each room, landing, or other location is represented by a vertex and each corridor or stairway is represented by an edge. Each corridor has an associated capacity c, meaning that at most c people can pass through the corridor at once. Traversing a corridor from one end to the other takes one timestep. (Traversing a room takes zero time.)

(a) Suppose all people are initially in a single room s, and that the building has a single exit t. Show how to use maximum flow to find a fastest way to get everyone out of the building. Hint: create an auxiliary graph G' that has vertices to represent each room *at each time step*.

(b) Show that the same technique can be used when people are initially in multiple locations and there are multiple exits.

(c) Generalize the approach to where different corridors have different (integral) transit times.

OPTIONAL (d) Suppose transit times are not integral. Is there still a way to solve the problem?

OPTIONAL (e) The above algorithm is polynomial in the number of people. Can you improve it to be polynomial (in the graph size) regardless of the number of people?

Problem 5. In this problem we develop a scaling algorithm for shortest paths that achieves running time $O(m \log C)$ with no data structure more complicated than an array. Recall that when we discussed shortest paths with positive integer edge lengths, we showed how Dial's algorithm uses a single array of buckets to find shortest paths in $O(m + nC)$ time for maximum edge length C.

Consider a solution to a shortest path problem with edge lengths l_{vw} from v to w in which the (shortest path) distance of vertex w from the source s is d_w. Define the *reduced edge lengths* l^d_{vw} by the rule $l^d_{vw} = l_{vw} + d_v - d_w$.

(a) Argue that the running time of Dial's algorithm is $O(m + D)$, where D is the maximum distance, and that the algorithm still works if some edges have length 0.

(b) Show that the *reduced edge lengths* defined above are all nonnegative integers.

(c) Show that the shortest paths under the reduced length function are the same as those under the original length function. What is the length of shortest paths under l^d?

(d) Devise a scaling algorithm for shortest paths. Use the reduced edge lengths and Dial's bucketing algorithm to keep the task of shifting in a new bit simple.

(e) Is base-2 scaling the best possible, or can you achieve a (slightly) better running time by scaling with a different base?

OPTIONAL Problem 6. Arbitrary augmenting paths are guaranteed to terminate in finite time with a maximum flow only if the edge weights are rational.

(a) Give a graph with non-rational capacities on which some augmenting path algorithm fails to terminate in finite time (even if each augmenting path is fully augmented).

(b) (Harder) Give a graph on which some augmenting path algorithm fails to terminate it finite time, and whose limiting flow is not maximum.

Problem 7. How long did you spend on this problem set? Please answer this question using the Google form that is sent to you via a separate email. This problem is mandatory, and thus counts towards your final grade. It is due by the Monday 2:30pm after the pset due date. You can find the link to the form on the course website.