Introduction to Algorithms April 23, 2004
Massachusetts Institute of Technology 6.046J/18.410J
Professors Erik Demaine and Shafi Goldwasser Handout 22

Problem Set 7

This problem set is due **in recitation** on **Friday, May 7**.

Reading: Chapter 22, 24, 25.1-25.2, Chapters 34, 35

There are **five** problems. Each problem is to be done on a **separate sheet** (or sheets) of paper. Mark the top of each sheet with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated. As on previous assignments, "give an algorithm" entails providing a description, proof, and runtime analysis.

Problem 7-1. Arbitrage

Arbitrage is the use of discrepancies in currency exchange rates to transform one unit of a currency into more than one unit of the same currency. Suppose, 1 U.S. dollar bought 0.82 Euro, 1 Euro bought 129.7 Japanese Yen, 1 Japanese Yen bought 12 Turkish Lira, and one Turkish Lira bought 0.0008 U.S. dollars.

Then, by converting currencies, a trader can start with 1 U.S. dollar and buy $.82 \times 129.7 \times 12 \times 0.0008 \approx 1.02$ U.S. dollars, thus turning a 2% profit. Suppose that we are given n currencies $c_1, c_2, \ldots, c_n$ 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 $\langle c_{i_1}, c_{i_2}, \ldots, c_{i_k} \rangle$ such that:

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

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

Problem 7-2. Bicycle Tour Planning

You are in charge of planning cycling vacations for a travel agent. You have a map of n cities connected by direct bike routes. A bike route connecting cities v and u has distance $d(v, u)$. Additionally, it costs $c(v)$ to stay in city v for a single night.

A customer will provide you with the following data:

- A starting city s.
- A destination city t.
- A trip length m.
- A daily maximum biking distance $u(k)$, where $k \in [1, m]$.

Your job is to plan a tour that takes exactly m days, such that the customer does not stay in the same city two consecutive nights and does not bike more than $u(k)$ on day k. Your customer can bike through several cities on a given day, i.e. there doesn't have to be a direct route between the cities you assign on day k and day $k+1$. You also want to minimize the total cost of staying at the cities on your tour.

Give an $O(n^2(n+m))$ time algorithm to produce an m-city tour $(s = v_0, v_1 \ldots, t = v_m)$ with minimum cost $\sum c(v_i)$, such that $d(v_{i-1}, v_i) \leq u(i)$.

Problem 7-3. P vs. NP

Suppose that $L_1, L_2 \in NP$ and that $L_1 <_p L_2$. For each of the following statements, determine whether it is true, false, or an open problem. Prove your answers.

- **(a)** If $L_1 \in P$, then $L_2 \in P$.
- **(b)** If $L_2 \in P$, then $L_1 \in P$.
- **(c)** L_2 is either NP-complete or is in P.
- **(d)** If $L_2 <_p L_1$, then both L_1 and L_2 are NP-complete.
- **(e)** If L_1 and L_2 are both NP-complete, then $L_2 <_p L_1$.
- **(f)** Suppose there is a linear time algorithm that recognizes L_2. Then there exists a linear time algorithm to recognize L_1.

Problem 7-4. NP-Completeness

- **(a)** Suppose you are given an algorithm A to solve the CLIQUE decision problem. That is, $A(G, k)$ will decide whether graph G has a clique of size k. Give an algorithm to find the vertices of a k-clique in a graph G using only calls to A, if any such k-clique exists.
- **(b)** Prove that the following decision problem is NP-complete.

  LARGEST-COMMON-SUBGRAPH: Given two graphs G_1 and G_2 and an integer k, determine whether there is a graph G with $\geq k$ edges which is a subgraph of both G_1 and G_2. (Hint: reduce from CLIQUE.)
- **(c)** Suppose you are given an algorithm B to solve the LARGEST-COMMON-SUBGRAPH decision problem. Give an algorithm to find a subgraph of size k that appears in both graphs G_1 and G_2, using only calls to B, if any such subgraph exists.

Problem 7-5. Maximum Coverage Approximation

Suppose you are given a set S of size $|S| = n$, a collection $\mathcal{F}$ of m distinct subsets $\{T_1, T_2, \ldots, T_m\}$ where $T_i \subseteq S$, and a number k as input. We would like to pick k subsets from the collection that cover the maximum number of elements in S. Give a greedy, polynomial-time approximiation algorithm for this *maximum coverage problem* with ratio bound of $\min\{k, f\}$, where $f = \max_i\{|T_i|\}$. Analyze the approximation ratio achieved by your algorithm.