Design and Analysis of Algorithms April 24, 2014
Massachusetts Institute of Technology 6.046/18.410
Instructors: Manolis Kellis, Ran Libeskind-Hadas, and Michael Kapralov Handout 12

Problem Set 5

This problem set is due on **Friday, May 9 at 11:59 PM**. This is the "ultimate" 6.046 pset (that is, it's the last one of the semester).

- Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered by the exercises and the best way to master that material is to have solved the problems yourself.
- Mark the top of each sheet with (1) your name, (2) the name of your recitation instructor, and the time your recitation section meets, (3) the problem number, (4) the names of any people you worked with on the problem, or "Collaborators: none" if you solved the problem completely alone.
- Each problem must be turned in as a separate PDF file to Stellar.

Exercise 5-1. 34.4-3

Exercise 5-2. 34.4-7

Exercise 5-3. 34.5-4

Exercise 5-4. 35.1-1

Exercise 5-5. 35.2-3

Exercise 5-6. The Partition and Bin Packing Problems

(a) Let $S = \{a_1, \ldots a_n\}$ and let $f : S \to \mathbb{Z}_+$ (that is, f maps elements of S to the positive integers). A *partition* of S is a pair of sets A, B such that $S = A \cup B$ and $A \cap B = \emptyset$. The Partition Problem asks if there exists a partition of S into sets A, B such that

$$\sum_{x \in A} f(x) = \sum_{x \in B} f(x)$$

Prove that the Partition Problem is NP-complete using a reduction from a problem that we have seen in class or on a previous homework assignment. (Notice that while this definition of the problem is seemingly awkward, it permits us to have multiple items with the same value, that is a "multiset" of numbers.)

Conversely, assume that the answer to the constructed Partition instance is "yes". Let A, B be the partition. The sum of A is equal to the sum of B is equal to $(T + 2t - T)/2 = t$. Without loss of generality, our "new" element $2t - T$ is in A. Therefore, the elements in B add up to t and include only elements from the original set S. Thus, there is a subset of S that adds up to t and the answer to the Subset Sum instance is also "yes".

(b) The Bin Packing Problem is the following: We are given a collection of objects $S = \{a_1, \ldots, a_n\}$, a function $s : S \to \mathbb{Z}_+$ indicating the "size" of each object, a bin capacity C such that C is greater than or equal to the size of the largest object, and a target t. The question is whether or not it is possible to pack all of the objects in S into t or fewer bins, each of capacity C. Prove that Bin Packing is NP-complete.

Problem 5-1. The Shmorbodian Fire Station Problem

The Republic of Shmorbodia comprises numerous small towns connected by roads. Since it is expensive to place a fire station at every town, the Shmorbodian parliament has decided that it suffices to ensure that each town either has a fire station or is connected by a road to a neighboring town that has a fire station.

Thus, the Shmorbodian Fire Station Problem (SFSP) is stated as follows: Given an undirected graph $G = (V, E)$ and a positive integer ℓ, does there exist a subset S of ℓ vertices in V such that every vertex in V is either in S or is connected by an edge to some vertex in S?

Take a close look at this problem to make sure that you understand how this problem differs from the seemingly similar VERTEX COVER problem. Prove that SFSP is NP-complete using a reduction from 3SAT. Write your proof carefully and methodically, using the "How to Write a Proof of NP-completeness" document as an example.

Problem 5-2. Network Reliability

Millisoft has decided to purchase the entire Internet and rent links to Internet Service Providers (ISPs). We represent the internet as an undirected graph. Assume that each edge in the graph has a positive integer *rental cost* associated with it. An ISP is confronted with the challenging problem of spending the minimum amount of money on link rental so that it can provide adequately reliable service to its customers.

Here is how we quantify reliability: We say that two paths in the network are *disjoint* if they have no vertices in common, except for the endpoints. For example, there can be two disjoint paths from vertex v_{42} to v_{100}, but v_{42} and v_{100} can be the only vertices that these paths have in common (since these vertices are the endpoints of the paths). In the interest of reliability, it is desirable to have multiple disjoint paths between pairs of nodes in the network. Some ISPs provide more reliability than others and they can charge their clients more accordingly.

The Network Reliability Optimization Problem (NROP) is now defined as follows: Given is an undirected graph with n vertices $v_1, \ldots, v_n$, a positive integer weight on each edge, and a $n \times n$

symmetric matrix R_{ij}. The objective is to find a subset S of the edges such that the total cost of the edges in S is minimized *and* for every pair of vertices v_i and v_j there exist at least R_{ij} disjoint paths from v_i to v_j such that all paths use only edges in S.

Your boss, Gill Bates, has asked you and your team members to develop an efficient algorithm for solving NROP optimally. After working on an algorithm for awhile, your team conjectures that the problem is NP-complete. State the decision version of this problem, which we will call NRDP, and prove that NRDP is NP-complete using a reduction from a problem that we have already shown to be NP-complete.

The last page of this pset contains the list of problems that you may use in your reduction.

Problem 5-3. Task Scheduling at Millisoft

Millisoft has m employees and a "boss" named Gill Bates. Gill's only job is to assign work to the m employees. Each task has an associated positive integer weight (representing something such as the difficulty of the task, how much time the task takes, or some other measure). The *load* on an employee is the sum of the weights of all of the tasks assigned to that employee. The objective of the LOAD BALANCING problem is to distribute the n tasks to the employees so that the maximum load over all of the m employees is minimized.

(a) On your first day at work, Gill asks you to find a polynomial time algorithm for the LOAD BALANCING PROBLEM. After racking your brain on this for several hours, you begin to feel that the problem is probably NP-complete. State the decision version of this problem and explain why it is NP-complete. You need only describe the problem that you would reduce from and how the reduction works. This should take two sentences. The "if and only if" part of the proof will be quite obvious and you don't need to write it here. **The last page of this pset contains the list of problems that you may use in your reduction.**

(b) Here's a simple polynomial time algorithm for this problem: Take the given tasks (in arbitrary order) and assign each task to the employee who currently has the lowest load (breaking ties between employees arbitrarily). Repeat this process until all tasks have been given out. Our claim is that this algorithm is a 2-approximation algorithm for the LOAD BALANCING problem. That is, it finds solutions that are guaranteed to have a maximum load no more than twice the optimal maximum load. Your job is to prove this claim. To do so, you'll need some "proxies" for OPT. (*Note:* One proxy is the duration of the longest task. What's another proxy for OPT? We've seen in class that two proxies are often useful in these kinds of analyses.)

Problem 5-4. Disk Storage

First the gratuitious story. You have been hired by Greenhat to work on their popular Lunatix operating system. When Lunatix does a backup, it typically uses two large backup disks and a tape drive. Since disks have lower access time than tape, it is desirable to store as many files as possible on disk and the remainder will go on tape. Let $F = \{f_1, \ldots, f_n\}$ denote the n files to be backed up and let ℓ_i denote the length of file i. Without loss of generality, let $\ell_1 \leq \ell_2 \leq \cdots \leq \ell_n$.

There are two identical disks, each with storage capacity of L. A file stored on disk cannot be broken up - it must be stored entirely on one of the two disks. The Disk Storage Optimization Problem is to store the maximum number of files from F onto the disks.

Your boss has suggested that you implement a greedy algorithm for the Disk Storage Optimization Problem: Since the files in F appear in non-decreasing order of size, simply go through F from shortest file to longest file, putting as many of the files on the first disk as possible. When the first disk fills up, continue iterating through F putting as many of the remaining files as possible on the second disk.

(a) Give an example that demonstrates that the greedy algorithm does not necessarily find an optimal solution. That is, give a set of specific file sizes, show the solution found by the greedy algorithm, and then the optimal solution. Show that the solution found by the greedy algorithm is not optimal.

(b) State the decision problem corresponding to this optimization problem. Then prove that the decision problem is NP-complete. **The last page of this pset contains the list of problems that you may use in your reduction.**

(c) Prove that the greedy algorithm is an approximation algorithm that finds solutions that are at most 1 smaller than optimal. This is really neat, since all of the approximation algorithms we've seen so far were some *multiplicative* factor worse than optimal (typically 2 times optimal). Here, the algorithm finds solutions which are an *additive* amount worse than optimal! (Recall that we've seen that finding a proxy is a nice way of relating the optimal solution to the solution found by our approximation algorithm.)

Problem 5-5. Robots on a Tree!

You've been hired by `Nile.com`, a huge online bookstore. Nile's warehouses have tracks embedded in the floor and a collection of k robots move along these tracks picking up books, etc. The network of tracks has no cycles and each piece of track is straight (although not necessarily horizontal or vertical). Notice that the tracks are continuous so a robot can be located at any point along the track. In other words, any undirected acyclic graph in which the edges are drawn as straight line segments forms a legitimate network of track except that the robots can be located anywhere along any edge of the tree and not just at the vertices.

We say that a robot R can "see" a request if there is no other robot between R and the request. Recall that in the algorithm for the line, we dispatch all robots that can see the request and they move towards the request at equal speed. On the line, that means that either one or two robots are dispatched. The tree algorithm will work analogously: All robots that can "see" the request move towards the request at equal speed. Now, there can be many robots moving towards the request! At some point, some of the moving robots may no longer see the request. At that point, those robots stop moving. But the robots that can see the requests keep moving. This process continues until at least one robot reaches the request.

To make your analysis as clean and clear as possible, it's helpful to divide your analysis into a number of "phases" where the first phase is the time period when the initial group of robots are

moving. That phase ends and the next one begins the moment one or more robots stop moving. If you can analyze one phase, you'll be just about done!

Show that this algorithm is k-competitive. (The good news is that you can use the same potential function as we used in class. The analysis will be just a bit more involved, but it will all work out!)

Problem 5-6. Ghosbusters re-re*-visited

Due to unusually high paranormal activity in the past few months, Ghosbusters Inc. is now getting so many contracts that they are running out of space in their containment unit (a high-tech device used to store captured ghosts). While another unit is being constructed (which is a very costly and time consuming process!), they need to decide which of the available contracts to accept in order to maximize their profit while not exceeding the storage capacity of their one and only containment unit. This is where you come in!

You are given a set of contracts $C_i = (v_i, p_i), i = 1, \ldots, n$, where v_i is the *metaphysical volume* of the ghost to be caught and stored under contract i, and p_i is the profit received from fulfilling contract C_i. The containment unit only has space to store a set of ghosts of total metaphysical volume at most V. All parameters $v_i, p_i, i = 1, \ldots, n$ and V are non-negative integers. Let $P = \max_{i=1,\ldots,n} p_i$ be the maximum profit.

(a) (Warm-up) Devise an efficient algorithm that given $C_i, i = 1, \ldots, n$ and V finds a feasible set of contracts that maximizes the profit. The runtime of your solution should be polynomial in n and P.

(b) It turns out that the profits that Ghostbusters are now getting are so large that solutions with runtime polynomial in P are no longer feasible! So in order to achieve better efficiency, Ghostbusters are happy to tolerate some approximation. Design an algorithm that given a precision parameter $\epsilon > 0$, the set of contracts $C_i, i = 1, \ldots, n$ and the total available metaphysical volume V, finds a set of contracts that is within $1 - \epsilon$ factor of the most profitable set. The runtime should be polynomial in n and $1/\epsilon$ only.

A List of Some Known NP-Complete Problems

1. 3SAT: Given an expression in conjunctive normal form with 3 literals per clause, is it satisfiable?
2. Vertex Cover: Given a graph G and number k, is there a vertex cover of size k or less?
3. Independent Set: Given a graph G and number k, is there an independent set of size k or more?
4. Clique: Given a graph G and a number k, is there a clique of size k or more?
5. Graph Coloring: Given an undirected graph and positive integer k, is it possible to color the vertices with k or fewer colors so that two adjacent vertices do not have the same color.
6. Directed Hamiltonian Path From s to t: Given a directed graph G and vertices s and t, is there a directed Hamiltonian path from s to t?
7. Directed Hamiltonian Path: Given a directed graph G, is there a directed Hamiltonian path in the graph.
8. Directed Hamiltonian Cycle: Given a directed graph G, is there a directed Hamiltonian cycle in the graph?
9. Undirected Hamiltonian Path From s to t: Given an undirected graph G and vertices s and t, is there an undirected Hamiltonian path with endpoints s and t.
10. Undirected Hamiltonian Path: Given an undirected graph G, is there an undirected Hamiltonian path in the graph.
11. Undirected Hamiltonian Cycle: Given an undirected graph G, is there an undirected Hamiltonian cycle in the graph?
12. Traveling Salesperson Problem: Given a completely connected weighted undirected graph and a number k, does there exist a traveling salesman tour of cost k or less?
13. Subset Sum: Given a set S of positive integers and a target integer k, does there exist a subset of S with sum equal to k?
14. Partition: Given a set S of positive integers, can the integers be partitioned into two sets S_1 and S_2 such that the sum of the elements in S_1 is equal to the sum of the elements in S_2?
15. Bin Packing: Given a multiset S of positive integers "objects", a bin capacity C such that C is greater than or equal to the size of the largest object, and a target t, is it possible to pack all of the objects in S into t or fewer bins, each of capacity C?