

HW 3 Practice Problems

October 12, 2004

Here are LOTS of practice problems. For preparing for HW 3, problems 1-5 are good and answers are included for those. You'll find the rest useful both for HW 4 and also in reviewing for the exams. So focus on 1-5 for now.

1. An investor has decided to invest a total of \$50,000 among three investment opportunities: savings certificate, municipal bonds, and stocks. The annual return on each investment is estimated to be 7%, 9%, and 14%, respectively. The investor does not intend to invest her annual interest returns. She would like to maximize her yearly return (based on the estimates for the returns) while investing a minimum of \$10,000 in bonds. Also, the investment in stocks should not exceed the combined total investment in bonds and savings certificates. And finally, she should invest between \$5,000 and \$15,000 in savings certificates. Formulate a linear program to optimally solve this problem.
2. In SUBSET-SUM the input is a set $S = \{x_1, x_2, \dots, x_n\}$ and the integer t . In the COMPOSITE problem you are given as input an integer y and the question to answer is whether or not y has a factor besides 1 and itself. SUBSET-SUM is NP-complete and COMPOSITE is in NP. Clearly explain whether or not each of the following statements follow from these two facts.
 - (a) SUBSET-SUM \leq_p COMPOSITE.
 - (b) If there is an $O(n\sqrt{t})$ algorithm for SUBSET-SUM, then P = NP.
 - (c) If there is an $O(n^3 \log t)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
 - (d) If there is an $O(\log y)$ algorithm for COMPOSITE, then P = NP.
 - (e) if P \neq NP, then no problem in NP can be solved in polynomial time.
3. Answer each of the following questions and briefly explain your answer. You are given as facts that 3-CNF-SAT and TSP (traveling salesman problem) are NP-complete and that there is a polynomial time algorithm for 2-CNF-SAT.
 - (a) True or false: 3-CNF-SAT \leq_p TSP.
 - (b) True or false: If P \neq NP, then no NP-complete problem can be solved in polynomial time.
 - (c) True or false: 3-CNF-SAT \leq_p 2-CNF-SAT (assume that P \neq NP).
4. Prove that the following problem, the Non-bored Jogger's Problem (NBJP), is NP-complete. You are given as input a weighted undirected graph G (loops and multiple edges are allowed and the weights are positive integers), a specified node v , that is called home, and an integer $\ell \geq 0$. You must determine if there exists a route for the jogger (i.e. a path in G) that starts at vertex v , never repeats an edge, and returns to v after travelling a distance of exactly ℓ . *Hint: Reduce from SUBSET-SUM.*
5. The **subgraph-isomorphism problem** takes two undirected graphs G_1 and G_2 and asks whether G_1 is a subgraph of G_2 . In other words, the problem asks whether there is a one-to-one function f to map the vertices of G_1 to the vertices of G_2 such that there is an edge $\{u, v\}$ in G_1 exactly when $\{f(u), f(v)\}$ is in G_2 . Show that the subgraph-isomorphism problem is **NP-Complete** using a reduction from one of: CIRCUIT-SAT, SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER.

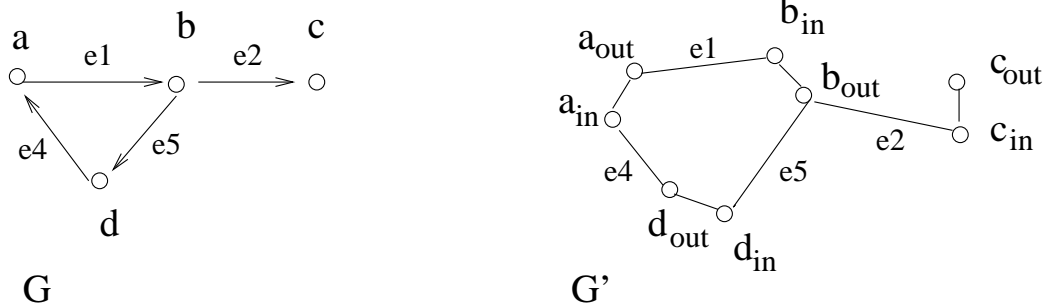
6. Consider the following problem. A gold digger is given a map (undirected graph) of a territory that consists of a set of towns (vertices) and trails (edges) between them. Each trail is labeled with a dollar value of the gold you would find along the trail the first time you travel it. Traveling a given trail a subsequent time yields no gold. Each town is labeled with the dollar cost of visiting it (hotels, meals, etc.), charged every time you enter the town. An “expedition” (cycle in the graph) is said to have profit k if k is the total amount of gold collected on the trails, less the total cost of visiting the towns along the way. The goal is to find an expedition that maximizes the profit. From this optimization problem we create the following decision problem that we will call the GOLD-DIGGER problem: Given an integer p and an undirected weighted graph G where each edge and vertex have a non-negative weight, is it possible to find an expedition with profit at least p .

Either prove that there is a polynomial time algorithm for the GOLD-DIGGER problem or prove that GOLD-DIGGER problem is NP-complete. If you try to prove it is NP-complete, you can use any of the following for your reduction: SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, PARTITION, HAM-CYCLE, TSP.

7. A *feedback vertex set* in a directed graph $G = (V, E)$ is a subset V' of V such that V' contains at least one vertex from each directed cycle in G . The *feedback vertex set problem* (FVS) is: Given a directed graph G and an integer ℓ , does G have a feedback vertex set with at most ℓ vertices? Prove that $\text{VERTEX-COVER} \leq_p \text{FVS}$. Then answer the below question.

Does it follow from the fact that $\text{VERTEX-COVER} \leq_p \text{FVS}$ that FVS is NP-complete? Why?

8. The Hamilton cycle problem (HAM-CYCLE) that we have studied is defined for undirected graphs. We now consider the directed Hamilton cycle problem (DIRECTED-HAM-CYCLE) in which you are given a directed graph G and asked if there is a directed cycle that visits every vertex exactly once. In this problem we consider a proposed reduction to show $\text{DIRECTED-HAM-CYCLE} \leq_p \text{HAM-CYCLE}$. Here is the proposed input transformation function f . Let $G = (V, E)$ be the directed graph that is input for DIRECTED-HAM-CYCLE. We define the undirected graph $G' = (V', E')$ for HAM-CYCLE as follows. For each vertex $v \in V$ there are two vertices v_{in} and v_{out} placed in V' . E' contains $|V| + |E|$ edges. First, for each $v \in V$ an edge is placed between v_{in} and v_{out} . Then for a directed edge from u to v in E , we place an undirected edge from u_{out} to v_{in} in E' . Here is an example to make sure that the definition for G' is clear.



Prove or Disprove: If G has a directed Hamilton cycle then G' has an undirected Hamilton cycle.

Prove or Disprove: If G' has an undirected Hamilton cycle then G has a directed Hamilton cycle.

9. Prove that DIRECTED-HAM-CYCLE is NP-Complete. You may reduce from any of 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, HAM-CYCLE, TSP.

10. The *set intersection* problem (SIP) is defined as follows: Given finite sets A_1, A_2, \dots, A_r and B_1, B_2, \dots, B_s , is there a set T such that

$$|T \cap A_i| \geq 1 \text{ for } i = 1, 2, \dots, r$$

and

$$|T \cap B_j| \leq 1 \text{ for } j = 1, 2, \dots, s?$$

Show that the set intersection problem is NP-complete. (Hint: Reduce from 3-CNF-SAT).

11. You are to prove that the 2-MIN-CLUSTER problem (defined below) is NP-Complete. You must use one of the following NP-complete problems for your reduction: CIRCUIT-SAT, SAT, 3-CNF-SAT, CLIQUE, VERTEX-COVER, SUBSET-SUM, SET-PARTITION, HAM-CYCLE, TSP.

You are given as input an undirected weighted graph $G = (V, E)$ and an integer k . For a partition of the vertices V_1, V_2 (so $V_1 \cap V_2 = \emptyset$ and $V_1 \cup V_2 = V$), let $E' \subset E$ be the edges in which either both endpoints are in V_1 or both endpoints are in V_2 . Then the weight of the partition, $w(V_1, V_2) = \sum_{e \in E'} \text{weight of } e$. The question is whether or not there is a partition of the vertices V_1, V_2 such that $w(V_1, V_2) = k$.

12. For the following problem either prove the decision version is NP-complete or give a polynomial time algorithm. If it is in P give the most efficient algorithm you can for it.

You are given a tree T and must compute a minimum sized vertex cover for T .

Solutions for 1-5 (solve the problems before reading these)

1. Let x_1 be the number of dollars invested in savings certificates, x_2 be the number of dollars invested in bonds, and x_3 be the number of dollars invested in stocks. Then the given problem is optimally solved using the following linear program:

$$\begin{array}{rllll} \text{Maximize} & .07x_1 & +.09x_2 & +.14x_3 & \\ \text{Subject to} & & & & \\ & & & & x_2 & \geq 10000 \\ & -x_1 & -x_2 & +x_3 & & \leq 0 \\ & x_1 & & & & \geq 5000 \\ & x_1 & & & & \leq 15000 \\ & x_1 & +x_2 & +x_3 & & = 50000 \\ & x_1, & x_2, & x_3 & & \geq 0 \end{array}$$

2. (a) SUBSET-SUM \leq_p COMPOSITE does not follow from the facts given above. An NP-complete problem does not necessarily reduce to a problem in NP.
- (b) If there is an $O(n\sqrt{t})$ algorithm for SUBSET-SUM, you cannot conclude that P = NP. $O(n\sqrt{t})$ is not a polynomial time algorithm for SUBSET-SUM since it has an exponential dependence on $\log t$.
- (c) If there is an $O(n^3 \log t)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE. $O(n^3 \log t)$ is a polynomial time algorithm for SUBSET-SUM. Since SUBSET-SUM is NP-complete, all problems in NP reduce to it. Hence, the existence of a polynomial time algorithm for SUBSET-SUM would imply that there is a polynomial time algorithm for all problems in NP (i.e. P=NP). Since we are given that COMPOSITE \in NP, the stated claim follows.

- (d) If there is an $O(\log y)$ algorithm for COMPOSITE, you cannot conclude that $P = NP$. $O(\log y)$ is a polynomial time algorithm for COMPOSITE. However, we only have that COMPOSITE $\in NP$ (not that it is NP-Complete). Hence the conclusion that $P = NP$ would not follow.
 - (e) if $P \neq NP$, one cannot conclude no problem in NP can be solved in polynomial time. $P \subseteq NP$ and there is a polynomial time algorithm for each problem in P. The only conclusion you can draw if $P \neq NP$ is that no NP-complete problem can be solved in polynomial time.
3. (a) True, 3-CNF-SAT is NP-complete and thus in NP. Since TSP is NP-complete it follows that all problems in NP (including 3-CNF-SAT) reduce to it.
 - (b) True. Since all problems in NP can be reduced (via a polynomial time reduction) to every NP-complete problem, if any NP-complete problems can be solved in polynomial time then it would imply that $P = NP$. The contrapositive of this says that if $P \neq NP$ then no NP complete problem can be solved in polynomial time.
 - (c) False. There is a polynomial-time algorithm for 2-CNF-SAT. So if there was a reduction from 3-CNF-SAT to 2-CNF-SAT then it would follow that $P = NP$. And since we were told to assume $P \neq NP$ it is false.
4. Clearly NBJP is in NP since we can use the route as a certificate, and easily verify its correctness in polynomial time. Now we prove SUBSET-SUM \leq_p NBJP.

Let $\langle S = \{a_1, \dots, a_n\}, t \rangle$ be the SUBSET-SUM input. We construct the following NBJP input. The graph G will contain the single home vertex v with n self-loops, one associated with each element of S . The weight on the edge associated with a_i will be a_i . Finally we let $\ell = t$. Clearly this can be constructed in polynomial time.

We now argue that G contains a route that starts at v , never repeats an edge, and returns to v with distance exactly ℓ if and only if there is a solution to the subset sum problem. First note that if $S' \subseteq S$ is a solution to the subset sum problem then the jogger can obtain a solution to the NBJP by just taking those edges that correspond to the elements of S' . Likewise, if there is a solution to NBJP then the subset of S that correspond to the edges used in the NBJP solution forms a solution to the subset sum problem.

5. As the certificate use the one-to-one function f from the vertices of G_1 to the vertices of G_2 . Thus the length of the certificate is $O(n)$. Finally, given the certificate the verification algorithm can confirm that f is a one-to-one function and then take each edge $(u, v) \in G_1$ and verify that $(f(u), f(v)) \in G_2$. Clearly this can be done in $O(n + m)$ time. We now prove that CLIQUE \leq_p subgraph-isomorphism.

Let $\langle G, k \rangle$ be the input for clique. For the subgraph-isomorphism input we let G_1 be a complete graph on k vertices and we let $G_2 = G$. Clearly this can be done in polynomial time. To see this notice that we can assume that $k \leq n$ (or otherwise, clearly G does not have a clique of size k) and thus the time to create G_1 is simply $O(k^2) = O(n^2)$ which is polynomial in the number of bits to represent G .

By definition of a clique being a complete graph on k vertices, there is a clique of size k in G if and only if G_1 is a subgraph of G_2 . That is if there is a clique of k vertices in G then mapping the vertices of G_1 to those vertices in G_2 gives a solution for the subgraph isomorphism problem. Similarly, if there is a solution of the subgraph isomorphism problem, then the vertices in G_2 mapped to by the vertices in G_1 form a clique of k vertices.