CS 570
Analysis of Algorithms
Summer 2007
Final Exam Solutions

Kenny Daniel (kfdaniel@usc.edu)

Question 1

[FALSE] If A is linear time reducible to B ($A \leq B$), and B is NP-complete, then A must be NP-complete.

[FALSE] If B is linear time reducible to A ($B \leq A$), and B is NP-complete, then A must be NP-complete.

[TRUE] If any integer programming optimization problem can be converted in polynomial time to an equivalent linear programming problem, then $P = NP$.

[FALSE] It has been determined that NP Complete problems cannot be solved in polynomial time.

[FALSE] If $P = NP$, then there are still some NP complete problems that cannot be solved in polynomial time.

[TRUE] When we say that a problem X is NP Complete, then it means that every NP complete problem can be reduced to X.

Question 2

Suppose that there is an ordered list of n words. The length of the i-th word is w_i, that is the i-th word takes up w_i spaces. The objective is to break this ordered list of words into lines, this is called a layout. The length of a line is the sum of the lengths of the words on that line. The ideal line length is L. No line may be longer than L, although it may be shorter. The penalty for having a line of length K is $L - K$. The total penalty is the maximum of the line penalties. The problem is to find a layout that minimizes the total penalty. Prove or disprove that the following greedy algorithm correctly solves this problem.

```
For i = 1 to n
    If the i-th word fits on the current line
       Place the i-th word on the current line
    else
       place the i-th word on a new line
```

Solution: This algorithm does not find the optimal layout. For a counterexample, consider the following sentence:

> *"Greedy is not a good algorithm"*

To print this sentence with width 12, the greedy algorithm gives the following layout:

```
123456789012
Greedyisnota
good
algorithm
```

This layout has total penalty 8, because of Line 2. However, this is not optimal, because the following layout has total penalty 4:

```
123456789012
Greedyis
notagood
algorithm
```

Question 3

You are hired by a consulting company which specializes in hiring/firing employees. When the company consults for an organization, it is given a list of all employees of that organization and a list of all pairs of these employees that have personality conflicts. The company then seeks to find the smallest set of employees that must be fired in order to remove all personality conflicts in the organization. Your boss gives you the assignment of designing an efficient algorithm to solve this problem. In particular, he wants your algorithm to be given as input a list of employees and a list of personality conflicts and to output the minimum number of employees that must be fired to remove all personality conflicts.

(a) Show that you can not be expected to design an efficient algorithm to solve this problem.

Solution: This problem can be shown to be NP-hard by reduction from Independent Set (described in Question 4). Suppose we are given an instance of the Independent Set problem, consisting of a graph G such that we want to find the largest set of vertices such that no two vertices share an edge.

We can express this as a company as follows: Each vertex v is represented by an employee e_v. For each edge (u, v), we say that employees e_u and e_v have personality conflicts. It can then be shown that a solution to this employee problem gives a solution to the Independent Set problem, and so the employee hiring/firing problem must be NP-hard.

(b) Your boss now wants you to design an approximation algorithm for the problem. Can you do this? If yes, provide the solution. If no, explain why.

Solution: As in part (a), Independent Set can be reduced to the employee problem, and Independent Set cannot be approximated if $P \neq NP$.

Question 4

We read about the INDEPENDENT SET problem in class. Here is the statement for your reference. Given a graph $G(V, E)$, find an independent set S as large as possible. (An independent set S is a subset of V such that no two nodes in S share an edge). The goal of this problem is to find an integer programming formulation of INDEPENDENT SET.

(a) What are the decision variables here

Each vertex v is represented by a variable x_v which is either 0 or 1.

(b) What is the objective function of the Integer program

The objective is to maximize the sum: $\max \sum_{v \in S} x_v$.

(c) What are the constraints in the Integer Program

For each edge (u, v), we add one constraint such that $x_u + x_v \leq 1$.

Question 5

In the first midterm, you were given a greedy algorithm for the coin change problem. The greedy algorithm gave the optimal solution for one denomination of coins but did not give the optimal solution for all denominations of coins. Here, you are required to give an algorithm that will optimally solve the coin change problem for any denomination of coins. Formally, If you are given k denominations of coins of values $d_1, d_2, \ldots, d_k$ where each d_i is an integer and given an integer amount X, give an algorithm that gives you the smallest number of coins of the denominations such that the sum of the coins is X.

Solution:

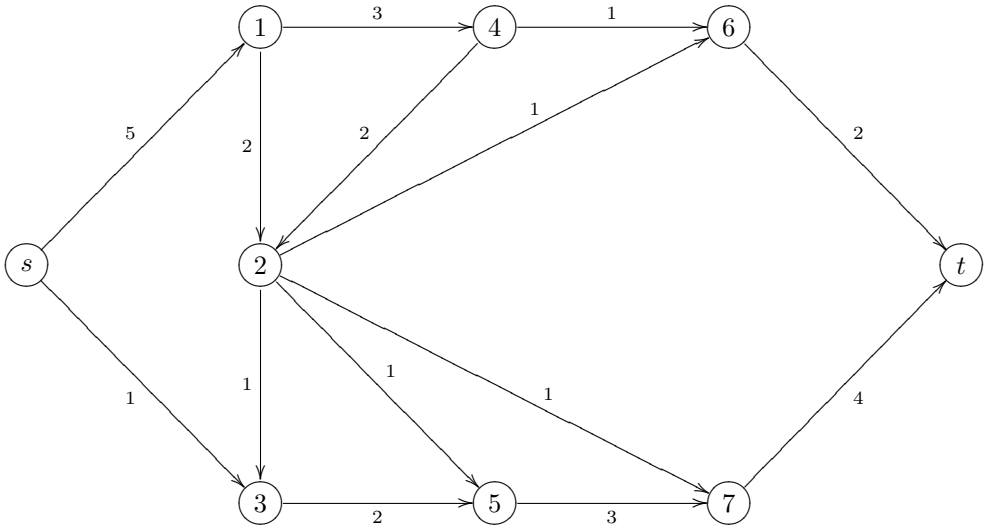
```
int change(X) {
    if (X < 0) return ∞;
    else if (X = 0) return 0;
    else return min_{i=1..k} change(X - di);
}
```

Question 6

(a) Give a maximum s-t flow for the following graph, by writing the flow f_e above each edge e. The printed numbers are the capacities.

(b) Prove that your given flow is indeed a max-flow.

Solution: The figure below shows a flow of 6. This is a max flow, since we the edges going out of s form a cut of weight $5 + 1 = 6$, and so 6 is the min cut / max flow.


1
3
4
1
6
5
2
2
1
2
s
2
t
1
1
1
1
4
3
2
5
3
7