CS570
Analysis of Algorithms
Spring 2017
Exam III

Name: _____

Student ID: _____

Email Address:_____

_____Check if DEN Student

	Maximum	Received
Problem 1	20	
Problem 2	16	
Problem 3	16	
Problem 4	16	
Problem 5	16	
Problem 6	16	
Total		

1) 20 pts
Mark the following statements as **TRUE** or **FALSE**. No need to provide any justification.

[TRUE /]
Given a minimum cut, we could find the maximum flow value in O(E) time.

[/ FALSE]
Any NP-hard problem can be solved in time O(2^poly(n)), where n is the input size and poly(n) is a polynomial.

[TRUE /]
Any NP problem can be solved in time O(2^poly(n)), where n is the input size and poly(n) is a polynomial.

[TRUE /]
If 3-SAT $\leq_p$ 2-SAT, then P = NP.

[/ FALSE]
Assuming P ≠ NP, there can exist a polynomial-time approximation algorithm for the general Traveling Salesman Problem.

[/ FALSE]
Let (S,V−S) be a minimum (s,t)-cut in the network flow graph G. Let (u,v) be an edge that crosses the cut in the forward direction, i.e., u ∈ S and v ∈ V−S. Then increasing the capacity of the edge (u, v) necessarily increases the maximum flow of G.

[/ FALSE]
If problem X can be solved using dynamic programming, then X belongs to P.

[/ FALSE]
All instances of linear programming have exactly one optimal solution.

[/ FALSE]
Let Y $\leq_p$ X and there exists a 2-approximation for X, then there must exist a 2-approximation for Y.

[TRUE /]
There is no known polynomial-time algorithm to solve an integer linear programming.

2) 16 pts

A set of n space stations need your help in building a radar system to track spaceships traveling between them. The i^{th} space station is located in 3D space at coordinates (x_i, y_i, z_i). The space stations never move. Each space station i will have a radar with power r_i, where r_i is to be determined. You want to figure how powerful to make each space station's radar transmitter, so that whenever any spaceship travels in a straight line from one space station to another, it will always be in radar range of either the first space station (its origin) or the second space station (its destination). A radar with power r is capable of tracking space ships anywhere in the sphere with radius r centered at itself. Thus, a space ship is within radar range through its trip from space station i to space station j if every point along the line from (x_i, y_i, z_i) to (x_j, y_j, z_j) falls within either the sphere of radius r_i centered at (x_i, y_i, z_i) or the sphere of radius r_j centered at (x_j, y_j, z_j). The cost of each radar transmitter is proportional to its power, and you want to minimize the total cost of all of the radar transmitters. You are given all of the (x_1, y_1, z_1), ... ,(x_n, y_n, z_n) values, and your job is to choose values for r_1,...,r_n. Express this problem as a linear program.

(a) Describe your variables for the linear program. (3 pts)

Solution: r_i = the power of the i^{th} radar transmitter., i=1,2,….,n (3 pts)

(b) Write out the objective function. (5 pts)

Solution: Minimize $r_1 + r_2 + \cdots + r_n$.
Defining the objective function without mentioning r_i : -3 pts

(c) Describe the set of constraints for LP. You need to specify the number of constraints needed and describe what each constraint represents. (8 pts)

Solution: $r_i + r_j \geq \text{sqrt}((x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2)$. Or, $r_i + r_j \geq d_{i,j}$ for each pair of stations i and j, where $d_{i,j}$ is the distance from station i to station j (6 pts)

we need Sigma_(i=1 to n-1) i = $(n^2-n)/2$ constrains of inequality (2 pts)

3) 16 pts
Given a row of n houses that can each be painted red, green, or blue with a cost P(i, c) for painting house i with color c. Devise a dynamic programming algorithm to find a minimum cost coloring of the entire row of houses such that no two adjacent houses are the same color.

a) Define (in plain English) subproblems to be solved. (4 pts)

Solution: Let C(i, c) be the minimum possible cost of a valid coloring of the first i houses in which the last house gets color c.

b) Write the recurrence relation for subproblems. (6 pts)

Solution: The recurrence is $C(i, c) = \min_{c` \neq c} \{C(i - 1, c`) + P(i, c)\}$.
The base case: C(1, c) = P(1, c), for all colors c.

c) Describe (using pseudocode) how the value of an optimal solution is obtained using iteration. (4 pts)

```
Set r[1] = P(1,r),g[1] = P(1,g), b[1] = P(1,b)
for i=2 to n
  r[i] = P(i,r) + min(g[i-1],b[i-1])
  g[i] = P(i,g) + min(r[i-1],b[i-1])
  b[i] = P(i,b) + min(r[i-1],g[i-1])
endfor
return min{r[n], g[n], b[n]}
```

d) Compute the runtime of the algorithm. (2 pts)

Solution: O(n)
The runtime is O(n) since there are n subproblems each of which takes O(1) time.

Grading rubric:

A. Part (a):
   a. Missing color in subproblem definition -2
   b. Unclear description -2 ~ -3
B. Part (b):
   a. Missing color in recurrence -4

b. Totally wrong recurrence -6
c. Fail to consider no same color in adjacent choice -2
d. Wrong symbols or typos in the recurrence -1 ~ -2

C. Part (c):
a. If wrong answer in part (b) -2
b. If no pseudo code -2
c. If the answer is not according to wrong answer in part (b) -4
d. If wrong order of for loop, color before house id -2

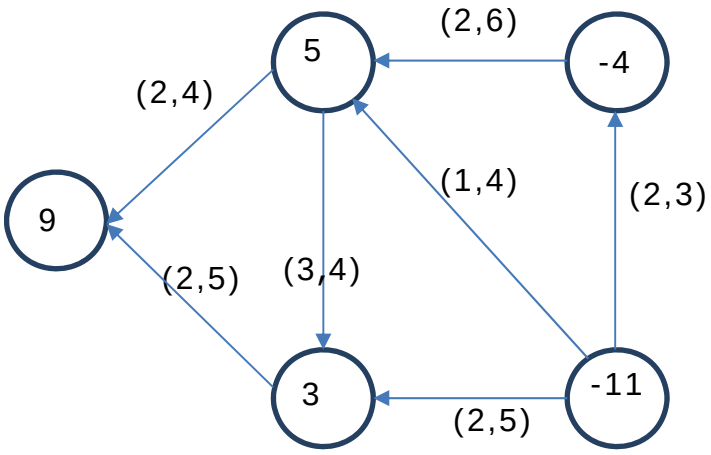
D. Part (d):
a. The question is graded according to recurrence and pseudo on part (b) and (c)
b. If there is additional constant in big O notation -1

Common errors:

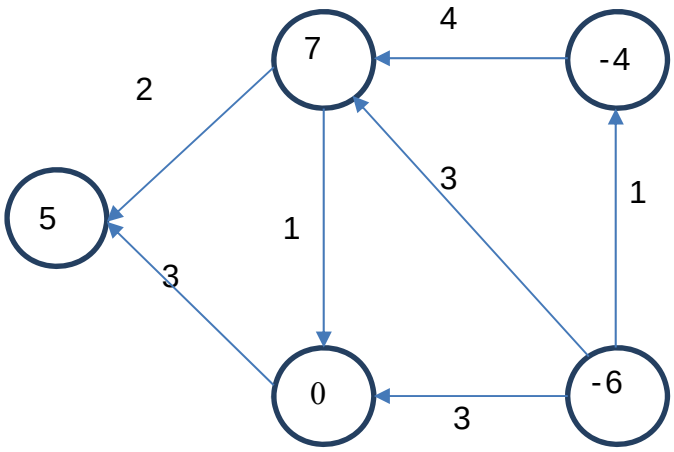
1. Failed to include the color of house in the definition of subproblem.
   a. Score if there are no other mistakes: part (a) -2, part (b) -4, part (c) -2 = 8
2. Each house must be painted into one of the three colors.
   a. Score if there are no other mistakes: part (a) -4, part (b) -4, part (c) -2 = 6
3. Use interval (2d state + 2 colors for the ends)
   a. Score if there are no other mistakes: part (a) -1, part (b) -1, part (c) -1 = 13

4) 16 pts

In the network below, the demand values are shown on vertices (supply value if negative). Lower bounds on flow and edge capacities are shown as (lower bound, capacity) for each edge. Determine if there is a feasible circulation in this graph. You need to show all your steps.



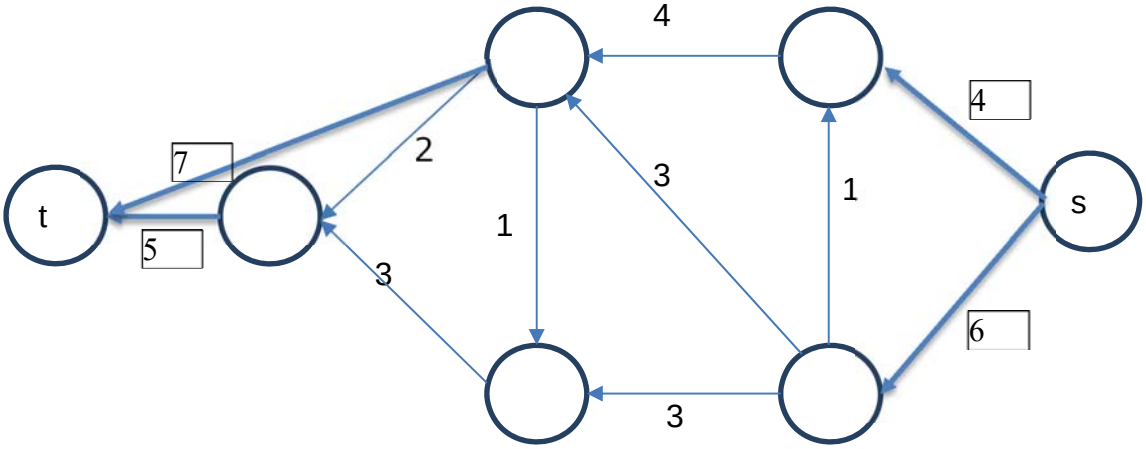
a) Turn the circulation with lower bounds problem into a circulation problem without lower bounds (6 pts)



Grading rubric

1) a) There are total 12 values in the network-flow graph that needs to be written, 5 values (demands) on vertices and 7 values on edges (modified capacities). For each correct value 0.5 marks will be awarded.
b) 2 marks will be deducted for adding a source and sink vertices and edges.

b) Turn the circulation with demands problem into the maximum flow problem (6 pts)



Grading rubric:

a) 2 marks for adding two super nodes, 1 for each node.
b) 2 marks for connecting negative demand vertices to supply nodes and positive demands to sink nodes, 0 if they are connected in opposite way.
c) 2 marks for placing the capacities on the newly added edges between super nodes and the vertices, will lose 0.5 marks for each edge capacity mistake (maximum deduction here is 2 marks).
d) will loose 0.5 marks for any incorrect edge capacity value on the edges in the original network flow.

c) Does a feasible circulation exist? Explain your answer. (4 pts)

No, a feasible circulation doesn't exist. In the given original network, total demand values on the vertices doesn't match with the total supply value on all the vertices.

Grading rubric:

a) 2 marks for answering no, 2 marks for explanation.
If you have answered yes and obtained a max-flow value of 10 and show that edges from source are saturated then 1.5 marks are awarded.

5) 16 pts

We want to become celebrity chefs by creating a new dish. There are n ingredients and we'd like to use as many of them as possible. However, some ingredients don't go so well with others: there is $n \times n$ matrix D giving *discord* between any two ingredients, i.e., $D[i,j]$ is a real value between 0 and 1: 0 means i and j go perfectly well together and there is no discord and 1 means they go very badly together. Any dish prepared with these ingredients incurs a *penalty* which is the sum of the discords between all pairs of ingredients in the dish. We would like the total penalty to be small. Consider the decision problem EXPERIMENTAL CUISINE: can we prepare a dish with at least k ingredients and with the total penalty at most p?

a) Show that Experimental Cuisine is in NP. (4 pts)

Certificate: a dish with at least k ingredients with total penalty of at most p.
Verifier: Go through ingredients and calculate the penalty and the total number of ingredients which takes at most quadratic time.

Grading rubric

-2 points for not mentioning about how to compute the penalty(e.g. add, search, compare, etc.)

b) Show that EXPERIMENTAL CUISINE is NP-complete by giving a reduction from INDEPENDENT SET. (12 pts)

Given an instance of IS, (G,k), where G is a graph, and k is a parameter. Construct the following: Set D to be an incident matrix of G, i.e., D[i,j]=1, iff vertices i and j are connected by an edge, otherwise D[i,j]=0.
Set p=0. (D,k,p) is an instance of EXPERIMENTAL CUISINE.

Claime: (D,p,k) returnes YES for EXPERIMENTAL CUISINE iff (G,k) returns YES for INDEPENDENT SET problem. Proof: Take instances, observe that a collection of vertices form an IS iff corresponding p = 0. k parameter stands for size of solution in both cases.

Grading rubric

Construction(4 points)

Since the problem is stated to reduce FROM IS, it should start with Independent Set to construct a discord matrix for EXPERIMENTAL CUISINE.

Claim (4 points)

Correct claim should be made for case of reduction: (D,k,p) returns YES for EXPERIMENTAL CUISINE iff (G,k) returns YES for INDEPENDENT SET problem.

Proof of reduction case (4 points)

The proof should be made in both directions when p=0.

6) 16 pts

There are *n* positive integers. Your goal is to concatenate them to form a single integer which is as large as possible. For example, for the following three integers 111, 222, and 3, the largest integer is 3222111.

a) Develop a greedy algorithm to solve the problem for arbitrary *n* (10 pts)

1. Sort the n integers using the following rule: If a is larger than b, then put a in front of b.

   (i) The comparison for number a and b is conducted by scanning through each digit. Need to consider several different cases:

   a) if the current digit is the same

   b) if the length of one number is shorter than another.

   (ii) or the comparison can be easily conduct by concatenating two numbers i..e concat(a,b) > concat(b,a)

2. Concatenate all the sorted integers together.

b) Provide proof of correctness. (6 pts)

Exchange argument, induction should be the correct way to do it.

Grading Rubric and common mistakes:

For problem (a),

- Your solution is incorrect, but you explain your own solution in a meaningful way, you will get ~4 points.
- the idea is generally correct, but if doesn't consider complete cases (including how to handle different length of numbers? i.e. 67 vs 676, what about the current comparing digit is the same? and etc), will deduct 2-4 points.
- some students write the solution like this: “comparing integers.. “ this is not correct since the comparison should be conducted digit by digit, thus not a right solution
- Some students answered - “appending zero”, this is not correct. i.e 67 vs 676, after appending zero, 676 > 67, so we get 67667 but the larger one should be 67676

For problem (b),

Common mistakes:

- the idea is in general right, but the proof itself is not a rigorous, get ~2 points deducted
- some students give proof by example, this is not correct. Showing by example is similar to writing a unit-test for programming, which is not a proof at all. deduct ~ 4 points
- some students follow the induction/exchange argument procedure, but does not show any valid evidence among the proof. i.e. use the proof framework and directly get the conclusion without showing any valid evidence. Deduct ~ 2-4 points

Additional Space