CSE 431/531 Final Exam – Spring 2007

Time: 7:00pm to 10:00pm
Place: Knox 109

Tuesday May 08, 2007

There are totally 6 problems. You are given 2 points for writing down your name and person number correctly. In total, the maximum score is 100. There are totally 11 pages.

Please put down your pen when you are told to do so. We shall not accept your submission otherwise.

Name	
Person Number	

Problem Number	Score obtained
name and id (2 max)	
Problem 1 (14 max)	
Problem 2 (9 max)	
Problem 3 (15 max)	
Problem 4 (20 max)	
Problem 5 (20 max)	
Problem 6 (20 max)	
Total Score: (100 max)	

Problem 1 (14 points). In the following sub-questions, let A and B be two decision problems such that $A \leq_p B$. Choose TRUE or FALSE? You don't have to justify your answer either way.

1. If $A \in \mathbf{P}$, then $B \in \mathbf{P}$.

   ☐ TRUE

   ☐ FALSE

2. If $A \in \mathbf{NP}$, then $B \in \mathbf{NP}$.

   ☐ TRUE

   ☐ FALSE

3. If $B \in \mathbf{P}$, then $A \in \mathbf{P}$.

   ☐ TRUE

   ☐ FALSE

4. If $B \in \mathbf{NP}$, then $A \in \mathbf{NP}$.

   ☐ TRUE

   ☐ FALSE

5. If A is $\mathbf{NP}$-complete and $A \in \mathbf{NP}$, then B is $\mathbf{NP}$-complete.

   ☐ TRUE

   ☐ FALSE

6. To show that a problem Π is NP-Complete, it is sufficient to find a polynomial-time reduction from Π to an NP-Complete problem Π'.

   ☐ TRUE

   ☐ FALSE

7. To show that a problem Π is NP-Complete, it is sufficient to find a polynomial-time reduction from an NP-Complete problem Π' to Π.

   ☐ TRUE

   ☐ FALSE

Problem 2 (9 points). Consider the following approximation algorithm for the VERTEX COVER problem. (The input is a graph $G = (V, E)$, and the output is a vertex cover S of this graph.)

VC-APPROXIMATION(G)

$S \leftarrow \emptyset$
while $E \neq \emptyset$ **do**
    Choose a vertex v with maximum degree, break ties arbitrarily
    $S \leftarrow S \cup \{v\}$
    Remove from E all edges incident to v
end while
Return S

Give a graph G for which the above algorithm returns a vertex cover of size at least 3 times the size of an optimal vertex cover. (Draw a graph, specify which vertex cover the above algorithm returns, and specify an optimal vertex cover.) You do not have to justify your answer.

Problem 3 (15 points). TRUE or FALSE? If you choose FALSE, give a counter example to briefly justify the choice; otherwise, you don't have to justify your answer.

1. Consider a flow network $G = (V, E)$ with source s, sink t, and positive integral capacity c_e on every edge $e \in E$. Let (A, B) be a minimum s, t-cut. If we add 1 to each edge capacity, then (A, B) remains a minimum s, t-cut with respect to the new capacities.

   ☐ TRUE

   ☐ FALSE

2. Consider a flow network $G = (V, E)$ with source s, sink t, and positive integral capacity c_e on every edge $e \in E$. Let f be a maximum flow, then for each edge $e = (s, u)$ out of s, we must have $f(e) = c_e$.

   ☐ TRUE

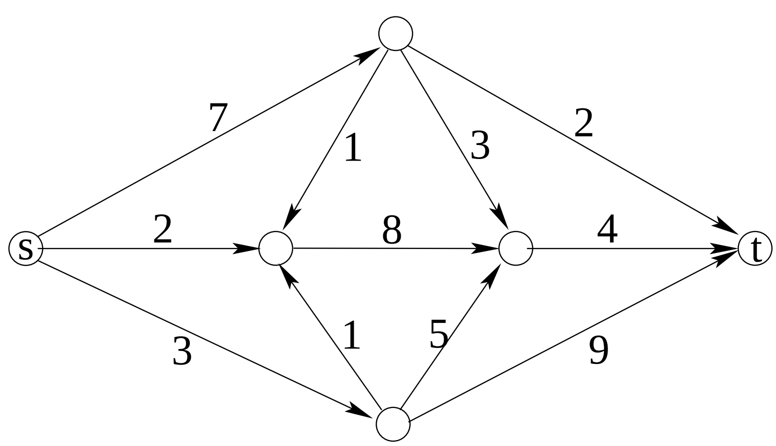
   ☐ FALSE

3. Consider the following flow network. The number next to an edge is its capacity. The source is s and the sink is t.

   (a) Find a maximum flow for this network

   (b) Find a minimum cut for this network

   Note: You don't have to show how you compute the max-flow or the min-cut, just write down the flow value for each edge and draw your cut.



Problem 4 (20 points). A *weird transformation* of a character array A is the task of breaking A into several contiguous sub-arrays, reversing some of the sub-arrays, then reconnecting the sub-arrays in the original order. The cost of a weird transformation is the number of breaks.

For example, the array

$$A = \texttt{thisisveryweird}$$

can be "weird-transformed" into the array

$$B = \texttt{sihtisyrevdriew}$$

by breaking A into `this`, `is`, `very`, `weird`, then reversing the first, third, and fourth sub-arrays, and finally reconnecting them in the original order. The cost of this transformation is 3.

Given two character arrays A and B of length n each, briefly describe an efficient dynamic programming algorithm to find the optimal cost of a weird transformation turning A into B, or report FAIL if A cannot be weird-transformed into B. Justify your answer and derive its running time. (You do not have to write down the pseudo-code.)

(– page intentionally left blank –)

Problem 5 (20 points). Next Fall semester, our CSE department will encounter a problem requiring your expertise in algorithm design. There are totally n students and m courses. Student i has a set S_i of courses that she/he prefers. The difficulty is that course j can only be offered if there are *exactly* s_j students registered for it.

(a) Devise an efficient algorithm to **either** recommend each student i a set T_i of courses to register for, such that $T_i \subseteq S_i$ and that all courses can be offered, **or** report correctly that no such recommendation exists.

(b) The requirement in part (a) might be too restrictive. Given a positive integer k, devise an efficient algorithm to **either** recommend each student i a set T_i of courses to register for, such that the number of non-preferred courses each student i registers for is at most k (i.e. $|T_i \setminus S_i| \leq k$), and that all courses can be offered, **or** report correctly that no such recommendation exists.

(– page intentionally left blank –)

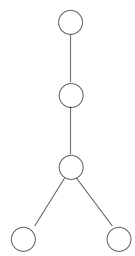
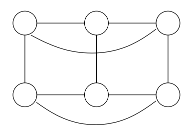
Problem 6 (20 points). A strongly independent set of a graph $G = (V, E)$ is a subset S of vertices such that for any two different vertices u, v in S, there is no path of length ≤ 2 between u and v. The STRONGLY INDEPENDENT SET (SIS) problem is defined as follows: given a graph G and an integer k, decide whether or not G has a strongly independent set of size at least k. In this question, you are to show that 3-SAT $\leq_p$ SIS.

(a) State precisely what you have to do to show that 3-SAT $\leq_p$ SIS.

(b) In this part and part (c), you are asked to describe a polynomial time reduction from 3-SAT to SIS. This part is to describe your reduction with a concrete example. Consider the following instance of the 3-SAT problem:

$$\phi = (x_1 \vee \bar{x}_2 \vee x_3) \wedge (\bar{x}_1 \vee \bar{x}_2 \vee x_3)$$

Draw the graph G and specify the number k that your reduction produces, given the formula ϕ above. Briefly explain how the reduction works using the example. **Hint:** staring at the following two pictures may help.

(c) Formally describe a polynomial time reduction from 3-SAT to SIS.

(d) Show that your reduction works.

(– page intentionally left blank –)

(– page intentionally left blank –)