# CS170 Final - Solutions

## Name:

## SID:

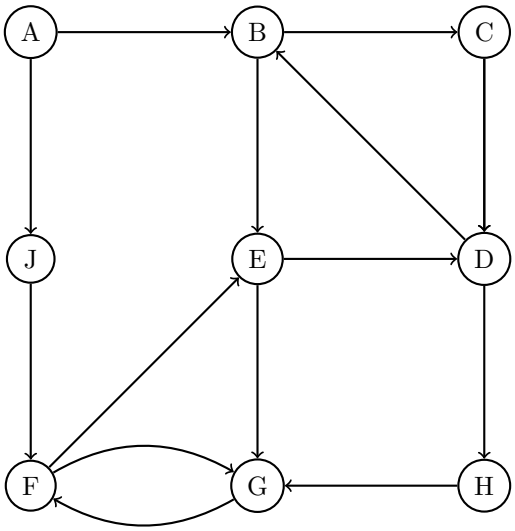
## GSI and section time:

Answer all questions. Read them carefully first. Be precise and concise. The number of points indicate the amount of time (in minutes) each problem is worth spending. Not all parts of a problem are weighted equally. Write in the space provided, and use the back of the page for scratch. By "reduction" in this exam it is always meant "polynomial-time reduction." Also, when you are asked to prove that a problem is **NP**-complete, no need to show that it is in **NP**, unless asked to do so.

Good luck!

# 1 DFS *(10 points)*

For the directed graph below, draw the DAG of the strongly connected components.



## 2 Linear programming (10 points)

Your linear program is this:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ & x_1 \leq 1 \\ & x_2 \leq 1 \\ & x_1, x_2 \geq 0 \end{aligned} \tag{1}$$

(a) Draw the feasible region. What is the optimal solution? How many steps will Simplex take from $(0, 0)$? What are the multipliers that prove optimality?

(b) The above linear program had a unique optimal solution. Can a linear program have exactly two distinct optimal solutions? Justify your answer.

## 3 Assigning students to Sections by Max Flow *(15 points)*

Professor Reynard–Ulysses Nutts is teaching a class with 100 students and five sections A1, A2, ..., A5, with 20 students each, and you are his TA. Now, the professor believes that it is important for students to mingle in a class. For this reason, after the midterm, he wants to redistribute the students in five new sections, B1, B2, ... , B5, so that each of the new sections contains no more than 4 students from each of the old sections.

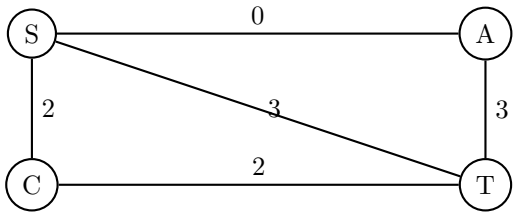
(a) You create a max-flow problem that accomplishes this. The nodes of the network are shown below. Indicate in the sketch the edges with the appropriate capacities (no need to draw all 35 edges). Then describe the max-flow showing that the professor's scheme is doable.

(b) "You were lucky," professes the professor. "But what if the max flow you found was fractional? You cannot transfer 3.4 students from section A2 to Section B1." How do you respond to that?

(c) Five more students register in the class. "No problem," says the professor. "Just increase the section capacities to 21 students." How do you prove to Professor R.-U. Nutts that his scheme is now impossible?

## 4 Bottleneck Dijkstra *(35 points)*

In the *bottleneck path problem* you are given an *undirected* graph $(V, E)$ with $|V| = n$ nodes, $|E| = m$ edges (as always we are assuming that $n < m$), and distances $\ell(i, j)$ on the edges, and two nodes $s$ and $t$, and you are asked to find the *bottleneck* $B$, that is, the smallest number such that there is a path from $s$ to $t$ on which all edges have length $B$ or less. You do not need to find the actual path. For example, in the graph shown, the best bottleneck is 2, because of the path $S - C - T$.




(a) You can solve this problem by a simple modification of Dijkstra. Show how would you change the statement

$$\texttt{...if } dist(v) > dist(u) + \ell(u, v) \texttt{ then } dist(v) = dist(u) + \ell(u, v)\texttt{...}$$

in Dijkstra's algorithm to accomplish this. Argue very briefly that it works.

(b) But suppose that you want instead to use divide-and conquer to solve this problem. You want to divide-and-conquer the *edges* of the graph according to their lengths. If the graph has $m$ edges (assume that $m$ is a power of 2), you want your recurrence to be

$$T(m) = T(\frac{m}{2}) + O(m).$$

Would this algorithm run faster than Dijkstra's algorithm implemented using $d$-ary heap? Justify briefly your answer.

(c) Your divide-and-conquer algorithm, call it `bottleneck(`$V, E$`)`, works in two steps.

In the first step you divide the $m$ edges of the graph into two *disjoint* sets $H$ (for high) and $L$ (for low) such that $|L| = |H| = \frac{m}{2}$, and every edge in $L$ has length $\leq$ to that of every edge in $H$. Can this step be done in $O(m)$ time (as required in the recurrence)? Justify briefly.

(d) The second step is this recursive call:

2. `If there is a path from` $s$ `to` $t$ `in the graph` $(V, L)$ `return bottleneck( ... )`

Fill in the dots and justify your answer briefly.

(e) The last line of the algorithm handles the case where there is no path from $s$ to $t$ in $(V, L)$:

`else return bottleneck(V',E'), where` $V'$ `is ...  and` $E'$ `is ...`

Describe carefully how the set of nodes $V'$ and the set of edges $E'$ is constructed, and briefly justify . Remember that the graph is undirected, $E'$ must contain at most $\frac{|E|}{2}$ edges, and this step must also run in linear time.

(f) Could you solve the same way the *Bottleneck spanning tree* problem (finding the smallest number B such that there exists a spanning tree whose edges all have weight at most B)? Justify very briefly.

## 5 Complete the sentences: *(20 points)*

(a) The solution of the recurrence $T(n) = 4T(\frac{n}{2}) + n^2$ is...

(b) You can linearize a dag by doing depth-first search and then ordering the nodes by...

(c) Dijkstra's algorithm works when the edge lengths are ...

(d) An operation of the union-find data structure with path compression on $n$ elements can never take longer than...

(e) A graph (direct or undirected) has a cycle if and only if depth-first search finds a...

(f) In a Huffman tree with all frequencies different, the item with the second-lowest frequency will always be placed at...

(g) The greedy algorithm for Set Cover approximates the optimum by a factor of...

(h) The FFT on $n$ points can be run with $n$ processors in parallel time...

(i) The FFT on $n$ points can be run with $\sqrt{n}$ processors in parallel time...

(j) In Multiplicative Weights with $n$ experts and $T$ periods you can come O( ... ) close to the optimum total performance.

## 6 True or false? Circle the right answer. No explanation needed (45 points)

*(No points will be subtracted for wrong answers, so guess all you want!)*

**T** **F** The Floyd-Warshall algorithm for all-pairs shortest paths works even when there are negative cycles.

**T** **F** The reduction we did in class is from MAX FLOW to MATCHING, not the other way.

**T** **F** The Bellman-Ford algorithm for shortest paths runs in $O(|V|^2)$ time.

**T** **F** The dynamic programming algorithm for KNAPSACK WITHOUT REPETITION runs in polynomial time.

**T** **F** The algorithm for MST that is best to parallelize is Kruskal's.

**T** **F** Huffman's algorithm runs in linear time.

**T** **F** Euclid's algorithm on $n$-bit numbers can run faster than $O(n^3)$ time.

**T** **F** Doubling the capacities in MAX FLOW just doubles the values of the maximum flow

**T** **F** Adding 1 to the capacities in MAX FLOW just adds 1 to the values of the max flow.

In the remaining questions there are four possible answers: (1) True (T); (2) False (F); (3) True if and only if **P** = **NP** (=); (3) True if and only if **P** ≠ **NP** (≠). Circle one.
**Note:** By "reduction" in this exam it is always meant "polynomial-time reduction."

**T** **F** = ≠ The minmax strategy in a zero-sum game can be found in polynomial time.

**T** **F** = ≠ There is no reduction from RUDRATA PATH to MAX FLOW.

**T** **F** = ≠ There is a polynomial-time algorithm for INDEPENDENT SET.

**T** **F** = ≠ There is a known polynomial-time algorithm for INDEPENDENT SET.

**T** **F** = ≠ There is a reduction from FACTORING to RUDRATA PATH.

a

**T** **F** = ≠ INTEGER LINEAR PROGRAMMING is **NP**-complete.

**T** **F** = ≠ LINEAR PROGRAMMING is **NP**-complete.

**T** **F** = ≠ There are problems in **NP** that cannot be solved in exponential time.

**T** **F** = ≠ There are computational problems that cannot be solved in exponential time.

**T** **F** = ≠ There are problems in **NP** that are neither in **P** nor **NP**-complete.

**T** **F** = ≠ There is a problem in **NP** that is not **NP**-complete.

**T** **F** = ≠ Any two problems in **P** can be reduced to each other in polynomial time.

**T** **F** = ≠ Any two problems in **NP** can be reduced to each other in polynomial time.

**T** **F** = ≠ There is a polynomial-time algorithm that approximates the TSP with triangle inequality with a factor of two.

**T** **F** = ≠ There is a polynomial-time algorithm that approximates the TSP without triangle inequality with a factor of two.

## 7 Dynamic Programming Plus *(25 points)*

(a) You are hired by a startup named Kale Tacos to plan its deployment of food trucks at street corners of Shattuck Ave. There are $n$ street corners where a truck could be deployed, numbered 1 through $n$. You have researched carefully the revenue a truck will get from street corner $i$, but this amount depends on whether or not another truck is deployed at one of the adjacent corners $i-1$ or $i+1$ (street corners 1 and $n$ have only one adjacent street corner). That is, your data consists of $n$ number pairs $(h_i, \ell_i), i = 1, \ldots, n$ where $h_i$ (which is non-negative) is the revenue when there is no truck in any one of the adjacent corners, and $\ell_i$ (which is smaller than $h_i$ and may be negative) is the revenue when there is another a truck in one of the adjacent corners, or if there are trucks in both. You want to find the optimum placement of trucks (there is no limit to the number of trucks you can deploy).

You use, of course, Dynamic Programming! You define $R(i), i = 1, \ldots, n-1$ to be the revenue you can get from corners $1, 2, \ldots, i$ assuming that there is no truck at corner $i+1$, and $R'(i), i = 1, \ldots, n$ to be the revenue you can get from corners $1, 2, \ldots, i$ assuming that *there is* a truck corner $i+1$.

Fill the blanks:

$R(1) =$

$R'(1) =$

For $i = 2, 3, \ldots, n-1$ $\quad R(i) =$

For $i = 2, 3, \ldots, n-1$ $\quad R'(i) =$

The optimum revenue is computed as

The running time is

(b) What if your company only has only a fixed number $k < n$ of trucks? Describe the appropriate subproblem for this situation, and estimate and justify the time complexity of the algorithm (no need to give the precise recurrence).

(c) Suppose next that each location $i$ has an arbitrary list $L_i = \{j_1, j_2, \ldots\}$ of competing locations which can be anywhere (instead of just the two competing locations $i-1$ and $i+1$), and the revenue at location $i$ is $h_i$ if there are no trucks in any competing location in the list, and it is $\ell_i$ otherwise. You have $k$ trucks. Prove that the problem of optimum truck placement is now **NP**-complete. (*Hint:* INDEPENDENT SET..)

## 8 NP-completeness *(25 points)*

**Note:** By "reduction" in this exam it is always meant "polynomial-time reduction." For the reductions in Problem 8 mention the problem you are using, direction and construction of the reduction (*proofs are not necessary*). Also, when you are asked to show that a problem is **NP**-complete, no need to show that it is in **NP**, unless asked to do so.

(a) Recall the 3-Dimensional Matching problem, asking you to match $n$ girls, $n$ boys, and $n$ pets given a list of compatible triples. We know that it is **NP**-complete. In the 4-Dimensional Matching problem you are given compatible *quadruples* of $n$ boys, $n$ girls, $n$ pets, and $n$ *homes*, and again you want to create $n$ harmonious households accommodating them all. Fill the blanks in the following proof that 4-Dimensional Matching is **NP-complete**.

■

(b) You are given a strongly connected directed graph $G = (V, E)$ and a budget $B$, and you are asked to find a subgraph $(V, E')$ of $G$ with $E' \subseteq E$ such that (1) $(V, E')$ is strongly connected, and (2) $|E'| \leq B$. Fill in the blanks in the following proof that this problem is **NP**-complete.

*For every pair of search problems A and B, if problem A is a generalization of problem B then clearly, problem B reduces to problem A.*

*Since we know that the problem Rudrata Cycle is NP-complete, the above problem is NP-complete.* ■

(c) In the $\frac{1}{3}$-Independent Set problem you are given a graph $(V, E)$ and you are asked to find an independent set of the graph of size exactly $\frac{|V|}{3}$. In other words, the target size $g$ is not part of the input, but it is always $\frac{|V|}{3}$. Why is this special case of Independent Set **NP**-complete?

(d) Fill the blanks in the following proof that the $\frac{1}{2}$-Independent Set problem (the variant where the goal is $\frac{|V|}{2}$) is **NP**-complete.

(x) Extra credit, work on this only if you finished all else:

Recall the Non-Partisan Traveling Senator Problem (NPTSP) of the project. Suppose you want to prove that it is **NP**-complete, through a reduction. Which known **NP**-complete problem would you use? And which direction would this reduction go? Fill the spaces below:

Show that the NPTSP is **NP**-complete.