CS 173, Fall 2015
TAKE HOME Examlet 7, B Lecture
due November 10, 2015 (in class) by 11:05 AM

This is on the honors system. Do not discuss the exam problems with anyone other than CS 173 Lecture B staff. Furthermore, the staff will only answer questions to make sure you understand the questions.

Your answers should be handwritten.

NAME (PRINTED):

Acceptance of Honor code. (Signed):

NETID (e.g. hpotter23, not 314159265):

Circle your discussion:

Fri 11-12 Fri 12-1 Fri 1-2 Fri 2-3 Fri 3-4 Fri 4-5

Problem	1	2	3	4	1-4
Possible	10	10	10	10	40
Score					

Total out of 40 points

Problem 1: (10 pts) Prove (by induction on n, the number of vertices in G) that every simple graph G can be properly vertex-colored using at most $D + 1$ colors, where D is the maximum degree of any vertex in G.

Inductive hypothesis (5 points)
Base case (1 point)
Inductive step (4 points)

Problem 2: (10 pts) Recall that the CLIQUE decision problem takes as input a graph $G = (V, E)$ and a positive integer k, and tells you whether G has a clique of size k. This is an NP-complete problem, by the way.

Recall the INDEPENDENT SET decision problem takes as input a graph $G = (V, E)$ and a positive integer k, and tells you whether G has an independent set of size k. This is also NP-complete.

Suppose you have an algorithm $\mathcal{A}$ that solves the CLIQUE decision problem. Design an algorithm $\mathcal{B}$ that solves the INDEPENDENT SET decision problem, and which satisfies the following constraint:

Given input $G = (V, E)$, graph G on n vertices, and positive integer k, your algorithm $\mathcal{B}$ performs at most a polynomial in n number of steps, where each step is one of the following: (1) applying algorithm $\mathcal{A}$ to some graph with at most n vertices, (2) assigning values to variables or looking at the values of variables, (3) numeric operations, (4) logical operations, and (5) Input/Output operations. In other words, the usual stuff. Analyze the running time of your algorithm and explain why it correctly solves the problem.

Algorithm (5 points)
Explanation for why it's correct (4 points)
Running time analysis (1 point)

Problem 3: (10 pts) Consider the algorithm $\mathcal{A}$ from Problem 2. Design an algorithm $\mathcal{C}$ that takes as input a graph $G = (V, E)$ and returns the subset $V_0 \subseteq V$ that is a clique and has the maximum size of all cliques in G.

Given input $G = (V, E)$, a graph on n vertices, your algorithm $\mathcal{C}$ must perform no more than a polynomial in n number of steps, where each step is one of the following: (1) applying algorithm $\mathcal{A}$ to some graph with at most n vertices, (2) assigning values to variables or looking at the values of variables, (3) numeric operations, (4) logical operations, and (5) Input/Output operations. In other words, the usual stuff.

Algorithm: 7 points
Running time analysis: 2 points
Proof of correctness: 1 point

Note: to prove correctness, you need to show two things: (a) that the algorithm returns a clique, and (b) that the size of the clique is maximum over all cliques for the graph.

Problem 4: 10 points Recall the all-pairs shortest path problem, where the cost of a path is the sum of the edge weights in the path. Consider the following way of solving it. Let $Q[i, j, k]$ denote the cost of the minimum cost path from v_i to v_j that has **at most** k edges; if no such path exists, then set $Q[i, j, k]$ to ∞.

Give a polynomial time dynamic programming algorithm to compute the matrix D of pairwise distances **using this subproblem formulation**. Make sure to clearly explain your solution and why it works. Analyze the running time. (HINT: review the Floyd-Warshall algorithm.)

Answer the following questions:

- Base cases (3 points)
  - What is $Q[i, i, k]$? (1 point)
  - For $i \neq j$, what is $Q[i, j, 1]$ when $(v_i, v_j) \in E$? (1 point)
  - For $i \neq j$, what is $Q[i, j, 1]$ when $(v_i, v_j) \notin E$? (1 point)
- For $i \neq j$, how do you set $Q[i, j, k]$ for $k > 1$? Hint: you need to consider what the minimum cost path using at most k edges looks like when $k > 1$. (3 points)
- How do you compute the matrix of pairwise distances (remember that $D[i, j]$ is the minimum cost of any path between v_i and v_j)? Note, this is something you do after you compute the entire matrix Q, and is what you need to return as your output. (2 points)
- What is the precise order in which you do all your calculations? State this by providing the loop structure for the algorithm. (1 point)
- What is the big-oh running time? Justify your answer. (1 point)

Use this page if you need extra space

Use this page if you need extra space