Final Examination

CS 4104 (Fall 2021)

Assigned: December 7, 2021.
Due: submit PDF file containing your solutions on Canvas by 11:59pm on December 14, 2021.

Instructions

1. The Honor Code applies to this exam. In particular, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, the solutions to homeworks and the midterm examination, and the instructor. Do not use a search engine.

2. Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*

3. For every algorithm you describe, prove its correctness, and state and prove the running time of the algorithm. I am looking for clear descriptions of algorithms and for the most efficient algorithms and analysis that you can come up with. Except for one problem, I am not specifying the desired running time for each algorithm. I will give partial credit to non-optimal algorithms, as long as they are correct.

4. You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince me that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*

5. If you prove that a problem is $\mathcal{NP}$-Complete, remember to state the size of the certificate, how long it takes to check that the certificate is correct, and what the running time of the transformation is. All you need to show is that the transformation can be performed in polynomial time. Your transformation need not be the most efficient possible.

6. Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.

7. You must prepare your solutions digitally, i.e., do not hand-write your solutions. I prefer that you use LaTeX to prepare your solutions. However, I will not penalise you if you use a different system. To use LaTeX, you may find it convenient to download the LaTeX source file for this document from the link on the course web site. At the end of each problem are three commented lines that look like this:

   ```
   % \solution{
   %
   % }
   ```

   You can uncomment these lines and type in your solution within the curly braces.

Good luck!

Problem 1 (30 points) Let us start with some quickies. For each statement below, say whether it is true or false or fill in the blanks. You do not need to provide a rationale for your solution.

1. We can compute the minimum spanning tree in an undirected, unweighted graph in time that is linear in the size of the graph.
2. In a directed graph, let $d(x, y)$ denote the length of the shortest path between nodes x and y. If $e = (u, v)$ is an edge on a shortest path from s to t, then $d(s, t) = d(s, u) + d(v, t) + l(e)$, where $l(e)$ is the length of the edge e. In this and the next question, we are using the standard definition of the length of a path as the sum of the weights of the edges in it.
3. In a directed graph, $G = (V, E)$, each edge has a weight between 0 and 1. To compute the length of the *longest* path that starts at u and ends at v, we can change the weight of each edge to be the reciprocal of its weight and then apply Dijkstra's algorithm.
4. A divide and conquer algorithm partitions an input of size n into two subsets of size $n/2$ and uses $O(n \log n)$ time in the conquer step. The running time of this algorithm is $O(n(\log n)^2)$.
5. Every edge in a flow network has a capacity of two. The value of the maximum flow in this network is the ____________________________ from s to t. (Fill in the blanks with a phrase.)
6. If A and B are problems such that $A \in \mathcal{NP}$ and $A \leq_P B$, then $B \in \mathcal{NP}$.
7. If $\mathcal{P} = \mathcal{NP}$ and problem X is $\mathcal{NP}$-Complete, then there is a polynomial time algorithm to solve X.
8. To prove that the Hamiltonian Cycle problem is in $\mathcal{NP}$, a certificate can be the set of all nodes in the input graph. (I mean a *set* of nodes here, as opposed to the certificate we used in class, which was a sequence of nodes.)
9. Suppose that, by accident, the jobs input to `Greedy Balance` are sorted in non-decreasing order of running time. Then the makespan of the solution computed by the algorithm is at most 1.5 times the optimal makespan.
10. James Bond wrote the ornithology book "Birds of the West Indies".

Problem 2 (10 points) Consider the residual graph G_f when the Ford-Fulkerson algorithm terminates; we have used the notation $\nu(f)$ to denote the value of this flow. Let $rc(e)$ denote the residual capacity of an edge e in G_f. Let A^* be the set of all nodes reachable from s in G_f (by a directed path of one or more edges) and $B^* = V - A^*$. Consider the set of backward edges (u, v) in G_f, where u is a node in B^* and v is a node in A^*. What is the total residual capacity of these edges, i.e., what is the value of

$$\sum_{\substack{(u,v) \text{ is a backward edge in } G_f,\\ u \in B^*, v \in A^*}} rc(e)?$$

Provide a brief explanation.

Problem 3 (15 points) In the field of social networks, scientists often look for groups of highly connected people. Here is one way to define such a group. In a undirected, unweighted graph $G = (V, E)$, we say that a subset S is *complete* if every pair of nodes in S is connected by an edge in E. Given a graph G and an integer k, the `Completeness` problem asks whether G contains a complete subset of size at least k. Prove that Vertex Cover $\leq_P$ Completeness. Do so without using the transitive property of reductions, i.e., without using some intermediate problem X where Vertex Cover $\leq_P$ X and X $\leq_P$ Completeness.

Problem 4 (20 points) Here is the correct version of what I wanted to assign as Problem 3 in Homework 5. There are many ways to formulate the cost of a secondary structure for an RNA molecule. In class, we defined the cost as the number of matching base pairs in the secondary structure and sought to maximise this quantity. Here is another approach. Suppose S is a secondary structure for an RNA molecule $b_1 b_2 \ldots b_{n-1} b_n$ that obeys all the rules we described in class. We define a *gap* as a maximal, consecutive sequences of indices that do not appear in S. More formally, a gap is a subsequence

$b_k b_{k+1} \dots b_{m-1} b_m$ where for every $k \leq l \leq m$ b_l, is not matched with any other base in S but both b_{k-1} and b_{m+1} are matched in S (not necessarily with each other). The *length* of the gap is $m-k+1$. Define the cost of S as the sum of the *squares of the*[1] lengths of the gaps. See Figure 1 for an illustration. Devise a recurrence relation to minimise this quantity. If you can derive the recurrence clearly from first principles (as we did in class), this derivation will serve as the proof of correctness. *I am not asking you to devise an algorithm to evaluate this recurrence. Hint:* One recurrence may not be sufficient.

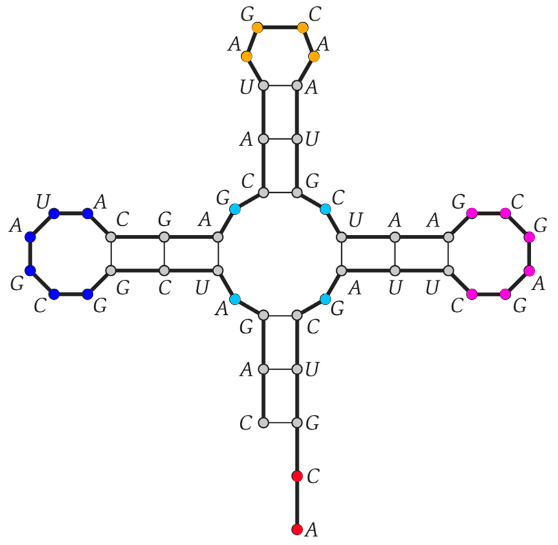

Figure 1: An illustration of gaps in an RNA secondary structure. To understand what we mean by a gap being maximal, consider the orange sequence AGC at the top. It is not a gap since it is a part of the sequence AGCA, which is maximal and matches our definition of a gap, consequently. There are four gaps of length one (light blue, in the centre) , one gap of length two (red, at the bottom), one gap of length four (orange, at the top), and two gaps of length six (dark blue and purple, on the left and right, respectively.) The total cost of this secondary structure is $96 = 4 \times 1^2 + 2^2 + 4^2 + 2 \times 6^2$.

Problem 5 (25 points) Consider the proof that HAMILTONIAN CYCLE $\leq_P$ TRAVELLING SALESMAN that we developed to prove that the second problem is $\mathcal{NP}$-Complete. We started with an undirected, unweighted graph $G = (V, E)$ that was an input to the HAMILTONIAN CYCLE problem. We created an input for TRAVELLING SALESMAN as follows: for every node $i \in V$, we created a city c_i, and for every pair of cities v_i and v_j, we defined $d(v_i, v_j) = 1$ if (i, j) is an edge in E and $d(v_i, v_j) = 2$ if (i, j) is not an edge in E. Consider the following modification: in the second case, i.e., (i, j) is not an edge in E, we set $d(v_i, v_j) = c$, where c is a constant ≥ 2. Let us denote this input to TRAVELLING SALESMAN as $I(G, c)$. Answer the following questions.

(a) (4 points) If G contains a Hamiltonian cycle, what is the length of the shortest travelling salesman tour in $I(G, c)$?

(b) (4 points) If G does not contain a Hamiltonian cycle, what is the length of the shortest travelling salesman tour in $I(G, c)$?

(c) (11 points) Suppose there is an α-approximation algorithm for the TRAVELLING SALESMAN problem that runs in polynomial time, i.e., for every input to this problem, this algorithm computes a travelling salesman tour whose cost is at most α times the cost of the optimal tour. Determine a value of c so that you can use this approximation algorithm to distinguish between the two cases: G has a Hamiltonian cycle and G does not have a Hamiltonian cycle. Explain your approach.

(d) (6 points) What can you conclude from these arguments?

[1] I omitted this italicised phrase in Homework 5.