Midterm Examination

CS 4104 (Spring 2021)

Assigned: March 15, 2021.
PDF solutions due on Canvas by 11:59pm on March 26, 2021, i.e., you have eleven days.

Instructions

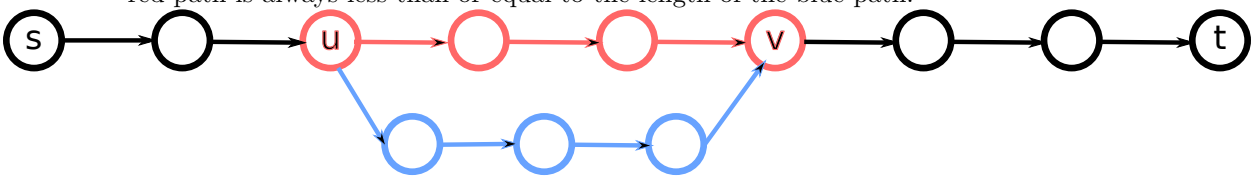
- The Graduate Honor Code applies to this examination. **Unlike in the case of homework, you must work on the examination individually.**
- You are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as* n^2 *and not as "n^2".* You can use the LaTeX version of the homework problems to start entering your solutions.
- 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.
- 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 us 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.*
- If you are proposing an algorithm as the solution to a problem, keep the following in mind (the strategies are based on mistakes made by students over the years):
  - 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.*
  - Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list. If n is the number of nodes and m is the number of edges in the graph, then the input size is $m + n$. Therefore, a linear time graph algorithm will run in $O(m + n)$ time.

Good luck!

Problem 1 (15 points) Let us start with some quickies. For each statement below, say whether it is true or false. *You do not have to provide a proof or counter-example for your answer.*

1. Every bipartite graph is a tree.
2. $\sum_{i=1}^{n} i \log i = \Theta(n^2 \log n)$.
3. Dijkstra's algorithm may not terminate if the input graph contains a negative-weight cycle, i.e., a cycle such that the sum of the weights of the edges in the cycle is less than zero.
4. 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. Here, the *length* of a path is the sum of the weights of the edges in the path.
5. Suppose π is the shortest s-t path in a directed graph. Then, there can be two nodes u and v in π such that length of the portion of π connecting u to v is larger than the length of the shortest u-v path in the graph. Here π is the name I am giving to the shortest path between s and t; it is not the mathematical constant.

   In the figure below, π is the path composed of black and red edges. As an example, I have marked two nodes u and v in π and denoted the portion of π connecting u to v using red edges. The blue edges denote another path from u to v in the graph. The figure does not indicate edge costs. If you answer "yes" to this question, you are saying that there are graphs where the length of the red path can be larger than the length of the blue path (knowing that π is the shortest s-t path). If you say "no", then you are saying that for every pair of nodes u and v in π, the length of the red path is always less than or equal to the length of the blue path.



Problem 2 (15 points) A connected, undirected graph G contains k cycles. Each edge in G has a distinct weight. Develop an algorithm to compute the minimum spanning tree of G in $O(mk)$ time. You may assume that $k < \log n$, so that the running time that I am seeking is faster than what you would obtain using Prim's or Kruskal's algorithm.

Problem 3 (10 points) Consider the problem of minimising lateness that we discussed in class. We are given n jobs. For each job $i, 1 \leq i \leq n$, we are given a time $t(i)$ and a deadline $d(i)$. Let us assume that all the deadlines are distinct. We want to schedule all jobs on one resource. Our goal is to assign a starting time $s(i)$ to each job such that each job is delayed as little as possible. A job i is *delayed* if $f(i) > d(i)$; the *lateness of the job* is $\max(0, f(i) - d(i))$. Define

1. the *lateness of a schedule* as $\max_i \left(\max \left(0, f(i) - d(i)\right)\right)$ and
2. the *delay of a schedule* as $\sum_{i=1}^{n} \left(\max \left(0, f(i) - d(i)\right)\right)$.

Note that although the words "lateness" and "delay" are synonyms, for the purpose of this problem we are defining them to mean different quantities: the lateness of a schedule is the *maximum* of the latenesses of the individual jobs, while the delay of a schedule is the *sum* of the latenesses of the individual jobs.

Consider the algorithm that we discussed in class for computing a schedule with the smallest lateness: we sorted all the jobs in increasing order of deadline and scheduled them in this order. We proved that the earliest-deadline-first algorithm correctly solves the problem of minimising lateness. If we were to use the *same proof* to try to demonstrate that the algorithm correctly solves the problem of minimising delay, what is the first point where the proof breaks down? Explain why the proof is incorrect here.

Problem 4 (20 points) You are given a list of n real numbers (the list can contain both positive and negative numbers). Give an efficient algorithm to determine the contiguous sub-list with the largest sum. More formally, suppose the numbers are $l_1, l_2, \ldots, l_{n-1}, l_n$. Your mission, should you choose to accept it, is to compute two indices $1 \leq i \leq j \leq n$ such that

$$s(i,j) = \sum_{k=i}^{j} l_k$$

is the largest over all possible choices of i and j. Note that to compute $s(i,j)$ we sum up all the numbers between indices i and j inclusive.

Hint: I am looking for an $O(n \log n)$ time algorithm. It is possible that you have seen this problem before and are aware of a solution with an $O(n)$ running time. If you present this algorithm, it is even more essential than ever that you prove its correctness. This proof is quite complex. I will not award any points for an $O(n)$ algorithm without a complete proof of correctness.

Problem 5 (40 points) This summer, you will be working at a synthetic biology company. The company specialises in creating cocktails of microbes (e.g., bacteria and fungi) to synthesize new antibiotics in an effort to combat infectious diseases. Their idea is to start with a compound that is cheap to make and to convert this compound into the desired antibiotic (which is another compound) using a series of chemical reactions that occur inside microbes. If you wonder why microbes are involved, know that several naturally occurring antibiotics are produced by microbes.

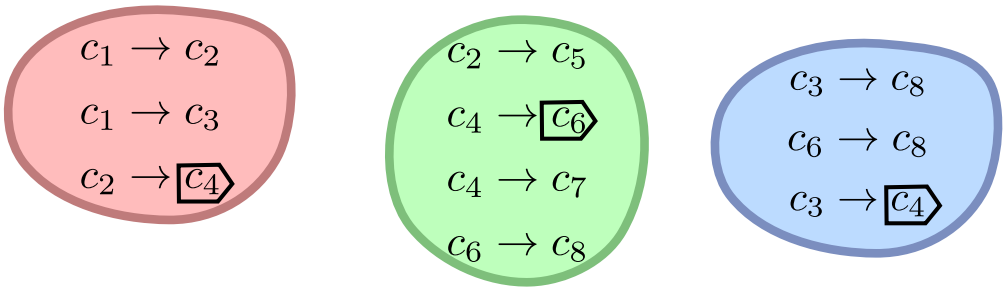


Figure 1: An illustration of microbes and reactions. The red microbe can perform three reactions, the green microbe is capable of four reactions, and the blue microbe has three reactions. A black shape surrounds a compound that can be secreted if it is made. Starting from c_1, the red and green microbes can together make c_8 since c_4 can be secreted by the red microbe and absorbed by the green microbe. Instead, if we started with c_3, then the blue and green microbes could synthesise c_8. (You may notice another way to make c_8 starting from c_1.)

Science background. Here is how the science works. A given species of microbe has the ability to synthesize specific compounds. The microbe does so using reactions that convert one compound into another. For example, in Figure 1, where the arrow sign means conversion, the red microbe can convert the compound c_1 into the compounds c_2 or c_3 (the first two reactions). It can also convert the compound c_2 to c_4. A microbe can also chain together reactions. Therefore, if the red microbe is provided the compound c_1, then it can make c_2, and thereafter use c_2 to make c_4. These chains of reactions can be arbitrarily long, as long as all the reactions can take place in that microbe.

Different microbial species can perform different sets of reactions and hence produce different sets of compounds. For instance, the only reaction that can use c_4 to form another compound exists in the green microbe. If we started with c_1 in the red microbe, produced c_4 in this microbe, and then somehow managed to "ship" c_4 to the green microbe, then we could obtain the compounds c_6, c_7, and c_8 (via c_6). And c_8 may be the antibiotic we desire!

How can we give the green microbe some c_4? Fortunately, biology comes to our rescue once again. Microbes can secrete some compounds to the environment and can absorb some compounds from the

environment. For example, if c_4 is such a compound, then (starting from c_1), the red microbe can make and secrete c_4. Subsequently, the green microbe can absorb c_4 and use it to make c_6, c_7, and c_8.

Only a subset of all compounds can be secreted and absorbed. The reason is that each microbe acts like a closed bag (the solid boundaries in the figure denote the cell wall of the microbe). Therefore, in the figure, if the red microbe has c_1, then it can make c_2 and then c_4. Since the red microbe's bag is closed, c_4 should stay inside the red microbe. However, if c_4 has the right chemical properties and it can be secreted and absorbed, then c_4 will diffuse out of the red microbe and enter the green microbe. Inside the green microbe, c_4 can start new reactions that result in the production of c_8. As another example, consider c_3. Starting with c_1, the red microbe can make c_3. But c_3 cannot be secreted. If it were, the blue microbe could absorb it, giving an alternate path to c_8.

There are only two more rules. Each reaction takes some time to complete; different reactions can have different times. Secretion and ingestion of a compound also take time; these times vary from one compound to another.

Input to the problem. What your company gives you as input are the following:

(i) the names of the microbes $M_1, M_2, \ldots M_k$,

(ii) the set C of compounds,

(iii) for each microbe M_i, where $1 \leq i \leq k$, a set S_i of reactions and their times,

(iv) the subset $S \subseteq C$ of compounds that can be secreted and absorbed, and for each compound $c \in S$, a secretion time s_c and an absorption time a_c. Note that s_c may not be equal to a_c and different compounds may have different absorption/secretion times.

(v) a source compound $\sigma \in C$ and a target compound $\tau \in C$. You can assume that the starting compound σ can be absorbed by any microbe in zero units of time.

To be clear about item (iii), for each microbe M_i, a reaction in S_i is of the form (c, d, t). This triple represents a reaction where compound $c \in C$ is converted to compound $d \in C$ in time t units in the microbe M_i. This triple says that if microbe M_i contains the compound c, then it can convert it into compound d in t units of time. Therefore, a reaction indicates a possible transformation that a microbe can execute. In this case, M_i must somehow obtain c before it can convert it to d. If the microbe has no access to c, then the reaction from c to d cannot take place inside M_i .

The sizes of the sets S_i can vary from microbe to microbe.

Problem statement. Finally, we can state the problems the company asks you to solve:

1. Starting from compound σ, can the microbes produce the target compound τ? If the answer is yes, which set of reactions accomplishes this task in the least amount of time?
2. Solve the first problem in the special case when every reaction in every microbe takes one unit of time and for every compound in S, the total secretion and absorption time is two units.

Note: The problem is not difficult to solve using techniques you have learnt so far in this course. I am looking for clear descriptions of the approach and careful attention to the running time. There are many elements in the input. You have to decide what factors will govern the size of the input and, therefore, the running time of your algorithm. For part 2, I am expecting a faster algorithm than for part 1. And I request you again: please do not write code!