Review Assignment #3

You should try to solve these problems by yourself. Consult the answer sheet only after you have given it a real attempt. If you do check the answer sheet along the way, set it aside and try to write up a solution without it.

This is not intended to be a comprehensive review.

Topics for exam: optimal substructure, dynamic programming, network flow and intractable problems.

You should be prepared to: explain and reason over whether algorithms have an optimal substructure, how to write and implement recursive formulas for DP problems, how to set up and formulate a network flow problem, and how to prove a problem is NP-Complete.

The year is 2318. Times have changed. The NCAT is now The Republic of NCAT. And the most popular student org on campus is now TRN Spacehoppers, a vibrant mix of undergraduate and graduate students that maintains a garage of spaceships they use for excursions across the galaxy ...

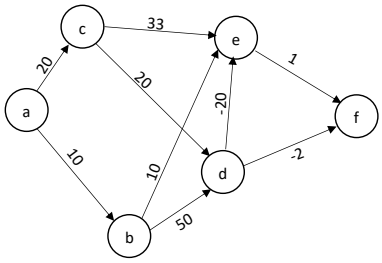
Problem 1: Spacehoppers

First, a warmup...

(a) [2 points] Illustrate, using an example, why Djikstra's shortest path algorithm does not always work when a graph has negative edges. A picture will be useful.

(b) [2 points] Give an example of a small graph (with fewer than five vertices) that has negative edges, but where Djikstra's algorithm still yields the (correct) shortest path.

TRN Spacehoppers is engaged in some trip planning. As part of their planning, they want to devise an algorithm that will allow them to travel between pairs of known star ports. To travel between star ports, ships can use wormholes, which may result in elapsed time going forwards or backwards. The figure below shows an example with six such star ports; traveling from port d to port e goes 20 units backwards in time; traveling from port c to e, however, takes 33 units forward in time.



Note that some super smart physicists (e.g., Einstein, Hawking...) have shown that it's quite unlikely we'll be able to arrive at a previously visited port at the same or time earlier than when we left it, however convoluted the round trip might be. Nothing has happened in the past 300 years to change this belief, so in this problem we assume the use of wormholes for reverse time travel is impossible.

Your goal, as TRN Spacehoppers' Chief Route Planner, is to find an algorithm to help the Spacehoppers get from any space port to a designated port, f, in the shortest possible time using a map of available wormholes. You quickly realize that the dynamic programming algorithm design technique will be particularly useful for this problem. But you have to convince all of your non-algorithmically minded friends that this is the right approach. The remainder of the parts of this question will walk you through doing that.

(c) **[3 points]** If n is the number of star ports and m is the number of wormholes, what is the maximum number of wormholes that an optimal solution will have (assuming that a maximum number exists). Give a proof of your answer.

(d) **[2 points]** What is the impact on your approach if the smart physicists are wrong and we can arrive at a star port earlier in time than we previously visited it?

(e) **[8 points]** Write a recursive definition of your solution. Explain your variables and notations. (Hint: let OPT$[v, k]$ denote the least time take to get from port v to the designated port f using at most k wormholes.)

(f) [**5 points**] For the specific example pictured above, illustrate the execution of your algorithm by filling out a table associated with your recursive definition.

(g) [**2 points**] Circle the value of the minimum elapsed time in the table from a to f and write the value here.

(h) [**3 points**] Provide a tight upper bound on the runtime for your algorithm. Explain your answer.

(i) [**3 points**] What would be the upper bound on the runtime for the brute force approach? An educated guess with an intuitive justification is sufficient.

Problem 2: Spacehoppers Part 2

In the **Vertex Cover** decision problem, we are given as input a graph G and a number k and must test whether G has a vertex cover with k vertices; that is, a set S of k or fewer vertices such that every edge has at least one end in S. This problem is NP-complete.

(a) **[4 points]** If this **Vertex Cover** decision problem has no polynomial time solution (that is, if $P \neq NP$), can there exist a polynomial time algorithm that takes as input a graph G and finds the vertex cover of G with the *largest* number of vertices? Why or why not?

To prepare for the grand adventure, TRN Spacehoppers has decided to send each of its members out on some practice flights. Every practice flight must have at least two members, and every member may be assigned to any number of flights. In addition, for safety, the group will to train a select number of members to be a captain on every flight to which they are assigned. A flight may have more than one captain, but every flight should have at least one.

The **Safe Trips** decision problem asks whether it is possible to train at most k Spacehoppers members to be captains so that every trip has at least one captain?

(b) **[3 points]** Explain in a few sentences what is meant by the expression **Vertex Cover** $\leq_P$ **Safe Trips**; in particular, we are looking for the explanation of the symbol $\leq_P$.

(c) **[3 points]** Is it true to say: "If A and B are NP-Complete problems, both $A \leq_P B$ and $B \leq_P A$ if and only if $P = NP$."? Explain briefly, in a sentence or two.

Finally, complete the proof that the Spacehoppers' problem of **Safe Trips** is NP-Complete:

(d) **[5 points]** Prove that the ST is in NP.

(e) **[10 points]** Complete the proof that the **Safe Trips** decision problem is NP-Complete by relying on the fact that **Vertex Cover** decision problem is NP-Complete. Illustrate your proof with a non-trivial example, along with a diagram.

Problem 3: Spacehoppers Part 3

Now that you've got your path planned, your team is ready to depart. However, interest in joining the trip has grown astronomically (see what I did there?), and there are lots of students that need to be accommodated on this adventure. The TRN Spacehoppers officers have gotten together to inventory the spaceships in the group's garage. Each ship can only accommodate a given number of passengers. Further, Spacehoppers because of limited flight requirements, the ships are all going to different ports.

The Spacehoppers officers have catalogued the n ships in a list SHIPS, indexed by i, where SHIPS$[i]$ contains the pair (p_i, d_i), where p_i is the maximum number of passengers that ship i can accommodate, and d_i is ship i's destination. Each candidate traveler is interested in only a subset of the destinations; each of the m travelers therefore provides a list of acceptable destinations.

(a) **[15 points]** Give a polynomial time algorithm (polynomial in n and m) to determine whether the team's available spaceships can accommodate all of the candidate travelers, and, if it is possible, assigns the travelers to ships. You should write a description in English or in high-level pseudocode (no code!) that should result in a clear passenger manifest for each ship.

Illustrate your solution with a small example, including a diagram.

(b) [**5 points**] Give an asymptotically tight upper bound on the runtime of your algorithm. Briefly explain.

(c) [**5 points**] Explain what is meant by the term pseudo-polynomial, using your algorithm as an example. (Note: if you cannot figure out how to use your algorithm as an example, you may give a brief discussion of the meaning of pseudo-polynomial using an example from class.)