Pset 6

Problem 0 - Birthday Party! (40%)

In the typical algorithmic setup, we have complete information as input and desired descision or optimization as output. For example, given a weight graph, compute its maximum spanning tree. Online algorithms are distinct from this typical setup and are used to model situations of incomplete or streaming data. With incomplete information we cannot guarantee optimality, but, we can reason about an algorithm's effectiveness. This, as we learned in class, is referred to as competitive analysis.
The CSE3500 class is invited to Professor Derek's birthday party in a week's time at 1:00pm. You can't think of a good enough excuse not to go, so you RSVP 'yes'. During lunch time a week later, you realize that you forgot to get Derek a gift. Panicking, you walk over to a fruit bin in the dining hall where you can grab kiwis or pineapples. Pineapples are ten times as heavy as kiwis, so your bag can hold either 40 kiwis, 4 pineapples, or any combination of the two that respect the bag constraints. You do not know if Derek enjoys pineapples or kiwis. But you do know that for each kiwi you bring, Derek will gain K ≥ 0 joy; likewise, for each pineapple that you bring, Derek will gain P ≥ 0 joy.

Task 1: Design an algorithm, inspired by competitive analysis, to choose the number of kiwis and pineapples to bring.
Task 2: What is the competitive ratio of your algorithm with respect to an oracle that brings the optimal presents? Your algorithm does not have to yield the optimal competitive ratio.
Task 3: The optimal offline strategy implemented by the oracle is a function of P and K: Opt(P, K). What is Opt(P, K)? Show that it is optimal. Hint: express joy as a function of n K , K, n P , and P with a constraint indicating how much you can carry in your bag.
Task 4: Find such a case and show that it gives a ratio of 2.
Task 5a: Show that the special case of n K = 0 and n P = 0 yields a joy upper bounded by 2.
Task 5b: Show the uperbound by 2.
Task 5c: Write a statement concluding your results.

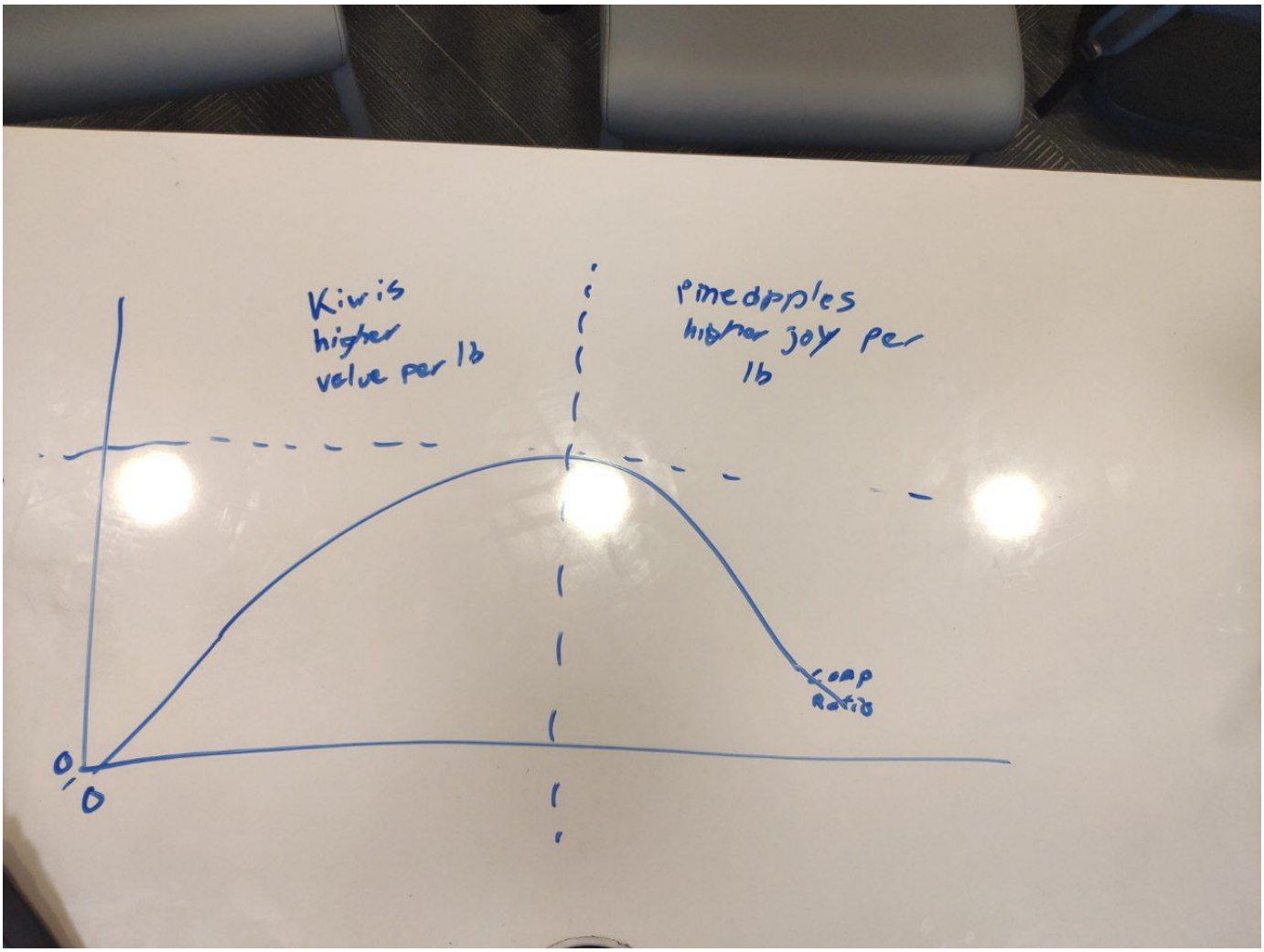
Solutions:

Task 1: The only solution that makes sense to me is to bring equal weights of each of the fruits (2 Pineapples, 20 kiwis) since we don't know what the inputs can be. We know the devil could set either pineapples or kiwis to give 0 joy, in which case favoring either one of these would hurt our competitive analysis as the devil could choose to set that one to be worth 0 joy.
Task 2: An oracle which brings the optimal amount of presents would have the ability to set the joy of anything they bring to an upperbound of infinity. The competitive ratio here could be undefined, as the oracles joy could be infinity and the devils could be nothing.
Task 3: Alg = ($n_K = 20, n_P = 2$) where Joy = J(nk=20), J(np = 2). We choose this algorithm because by prefering either fruit, the devil could lower that fruits weighted value, and increase the other fruits weighted value. Because of this, we must have equal weights of both fruits.

Task 4: Alg = ($n_K = 20, n_P = 2$), lets consider the case Pineapples = 0 Joy, Kiwis = 1 joy. The oracles algorithm brings all kiwis, and our algorithm only brings 20 kiwis. The oracles bring no pineapples and we bring 2 pineapples. In this case, the oracle brought 40 joy, and our algorithm gave 20 joy, bounding us at a competitive ratio of 2. This would happen in the opposite case aswell, if the pineapples were worth one joy, and kiwis were worth no joy. The oracle would choose four pineapples, and we would choose two pineapples, bounding us again at 2. 2 is the maximum competitive ratio this algorithm can produce.

Task 5a: We know that in this case, we do divide by 0, but we are also dividing a real number by 0, meaning we know we should be at minimum upperbounded by 2.

Task 5b:



Task 5c: We know that since our only constraint is weight, the oracle will only take multiple of the items with the highest joy per kg. Because of this, we must take equal weights of each type of fruit, as that hedges our bets against any fruit being worth more by the oracle and end up with the minimum competitive ratio.

Problem 1 - Matryoshka dolls (30%)

Russian nesting dolls are wooden dolls of decreasing size in which the smaller dolls can be placed inside the larger dolls (Figure 1). UConn happens to have the world's largest shadow economy dedicated to the buying and selling of nesting dolls. Professor Derek sees an economic opportunity and decides to plan an elaborate Russian nesting doll heist. Let there be n nesting dolls. The i th nesting

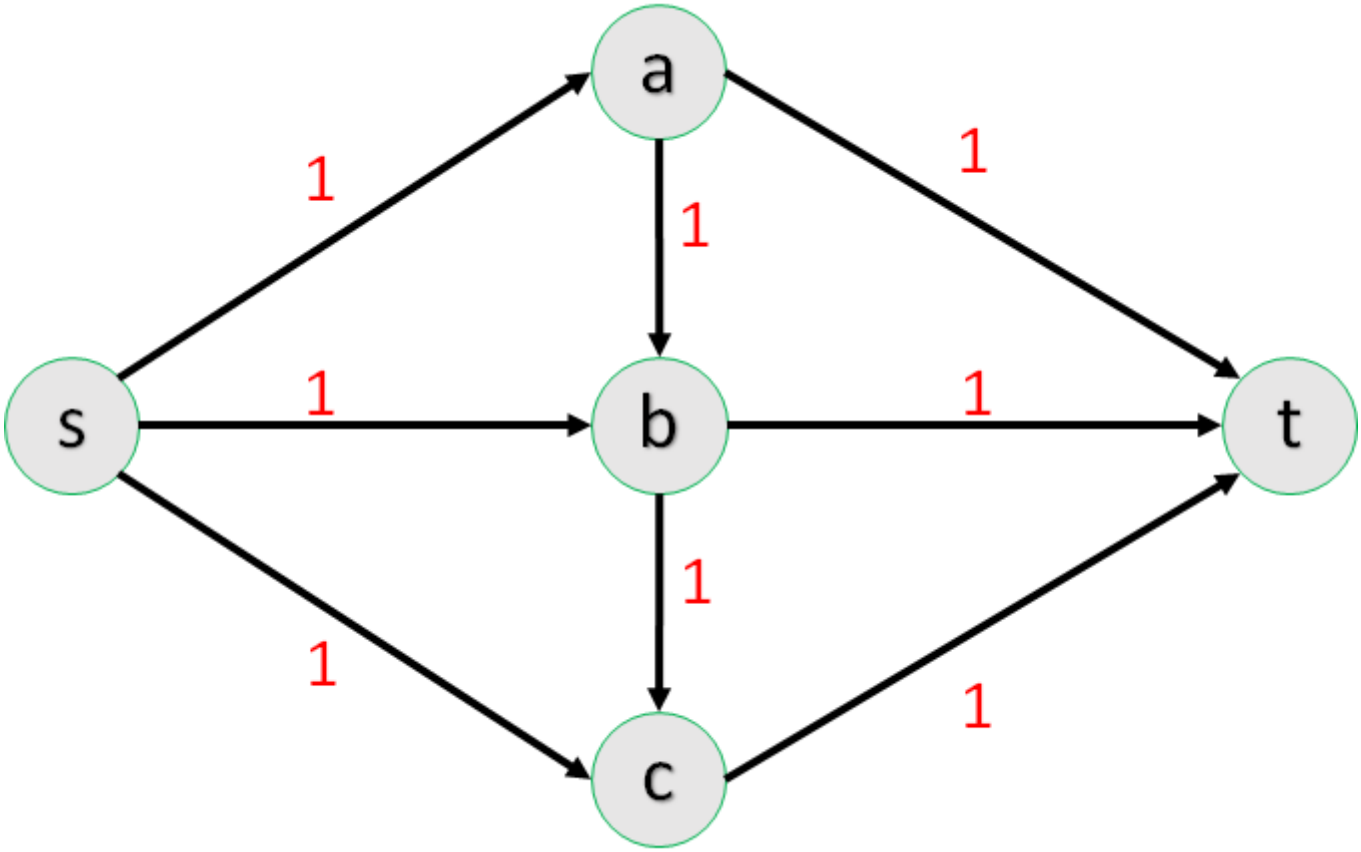
doll has a total weight of w i and total value of v i . However, these particular dolls are unique, i.e., they are infinitely decomposable. In other words, if x i is the normalized amount of the doll that we can take, we can take any fraction of the nesting doll x i ∈ [0, 1]. For example, x i = 1 would mean that we take the whole doll and all infinitesimally small dolls incurring a weight of w i and adding value v i . Whereas, if x i = [0, 1)] we would take the fractional amount xi of wi , incur a weight of xi wi and adding value xi vi. Derek has a backpack with a capacity of Z, meaning that the sum of the items he takes must weigh no more than Z and he wants to select the weights for each item such that his total profits are maximized.

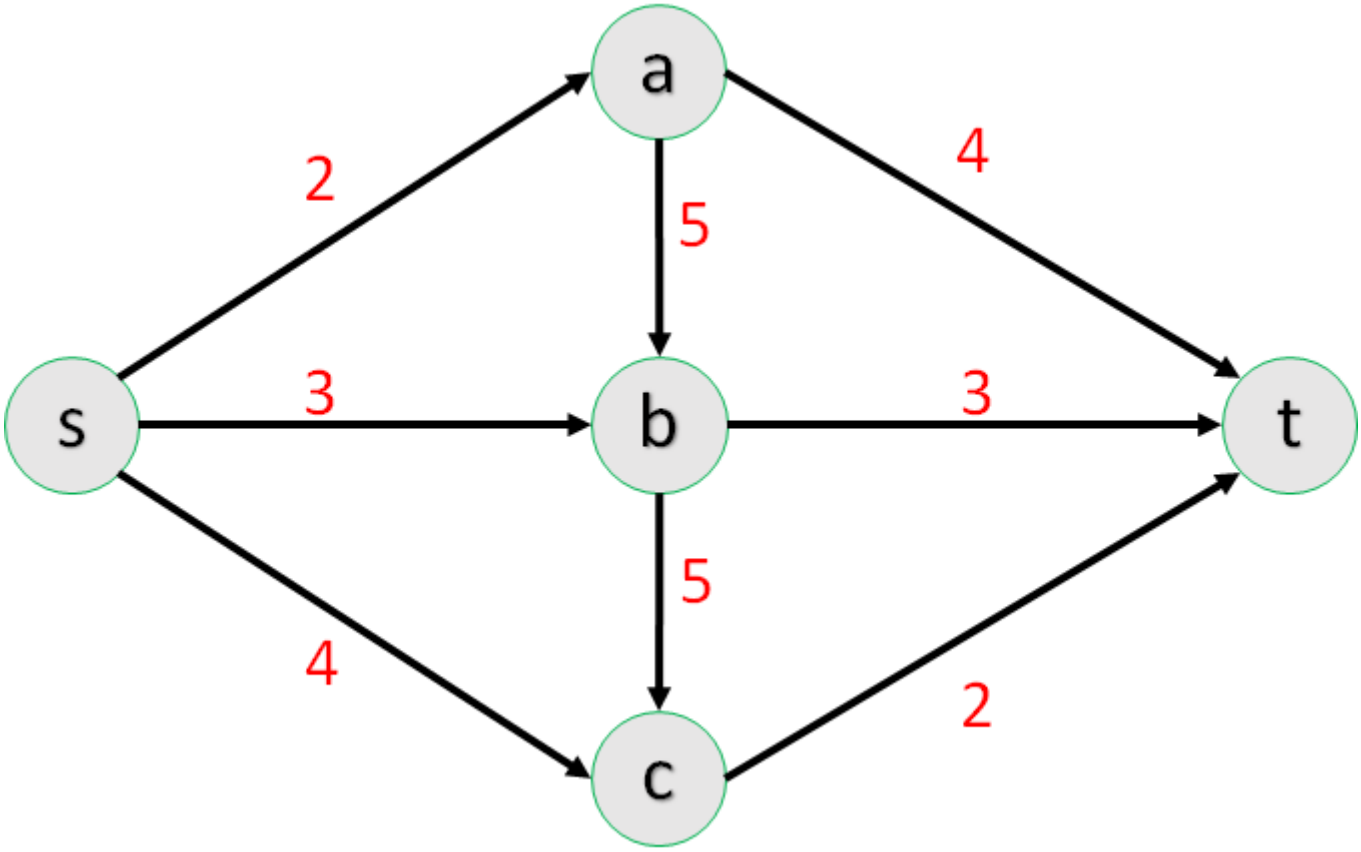
Solutions:

The problem here is I cannot take the entire nesting doll, as it forms a series that gives it infinite weight (an infinite amount of items, each with a weight between 0 and 1). The way I would solve this problem is order the dolls by their value density (value/weight) to get their value/kg. Once these dolls were ordered, I could then begin taking dolls in the order of their value/kg until the kg's of what I've added max out that of my pack. If there is a remainder of space in my pack, I will keep going down the list of dolls until I have a fractional doll that fits in my pack. A way to do this without the potentially costly sorting of an infinite ammount of dolls could be do add these to my pack as I take the dolls apart, raking the dolls I've seen thus far by their value/kg and keeping a running total of which dolls I keep in my pack (comparing all my old dolls value/kg to the next iteration of dolls value/kg). Despite this, I would need to go through the entire list of dolls, as its possible for the most infintessimally small doll to have an extremely high value and low weight.

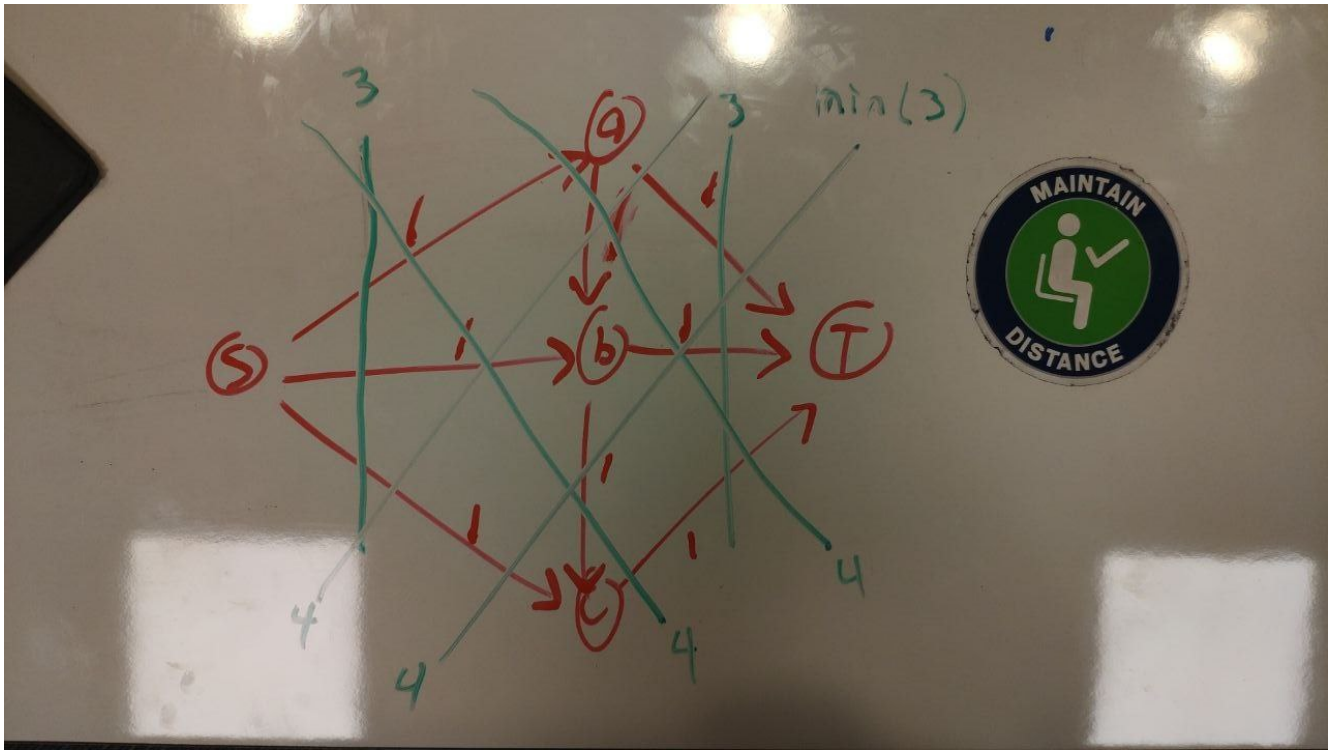
Problem 2 - Min-Cuts (15%)

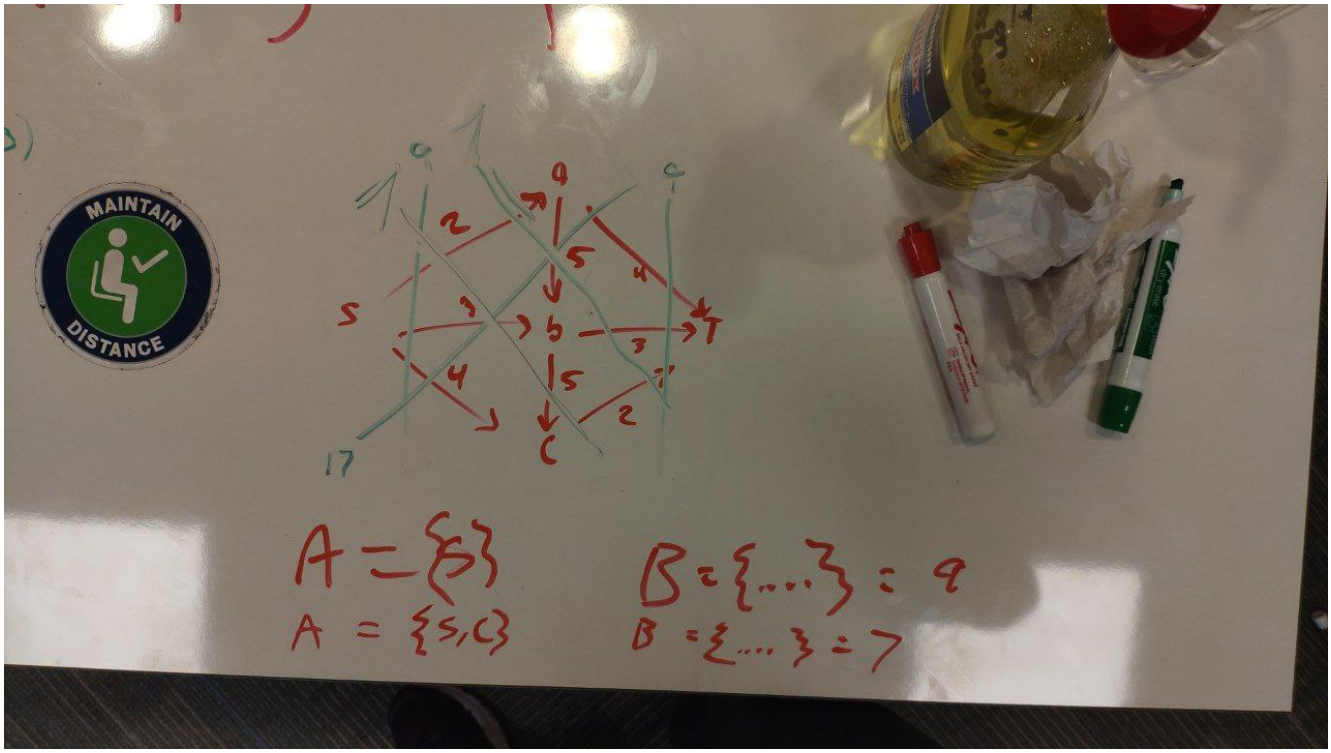
Let A and B be a partition of the nodes of G(V, E) such that s ∈A and t ∈ B. The capacity of a cut denoted c(A, B), is the sum of the capacities out of A. Enumerate all the minimum s − t cuts in the flow networks in Figures 2 and 3; give their value.





Solutions:

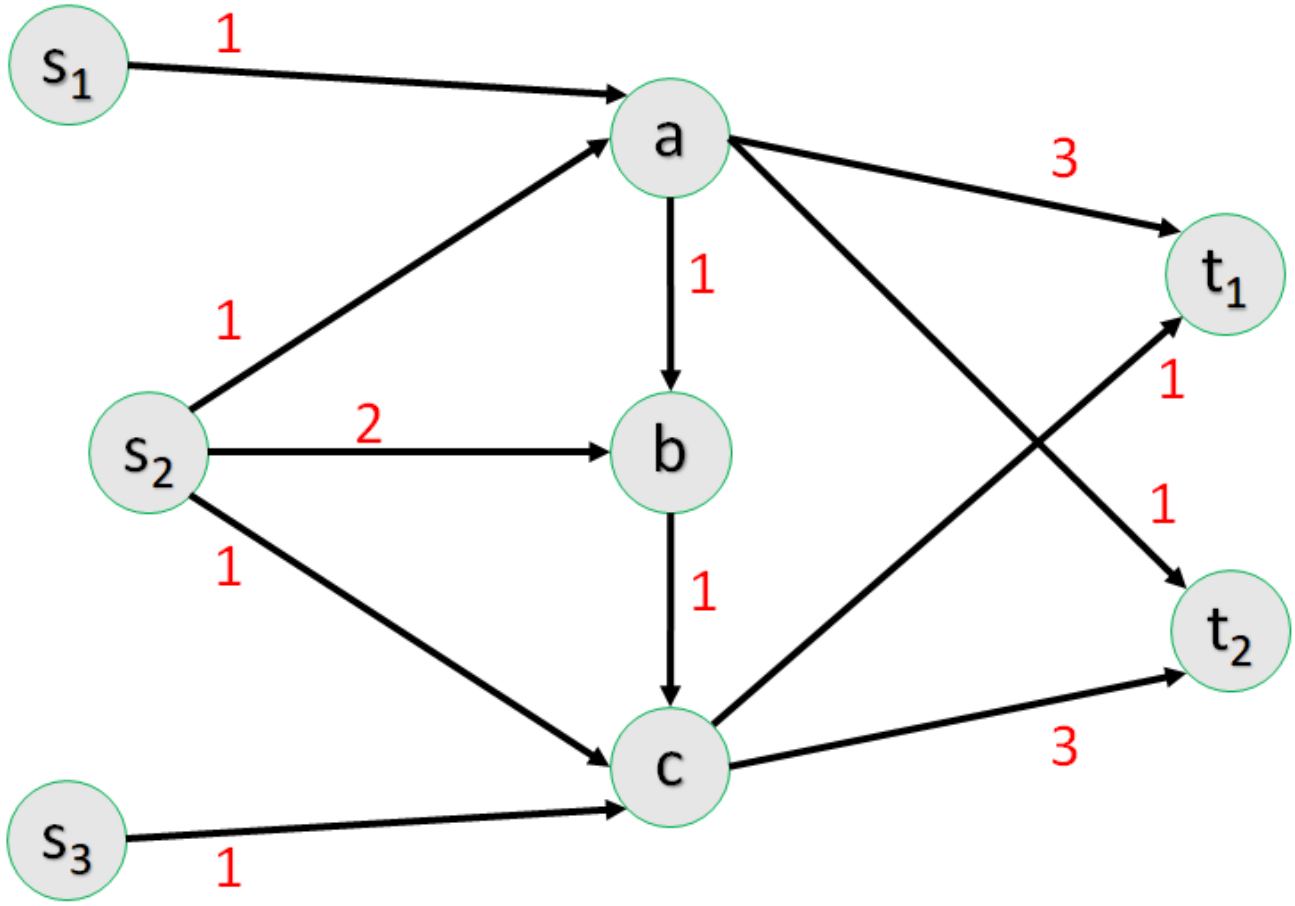
1. 

2. 

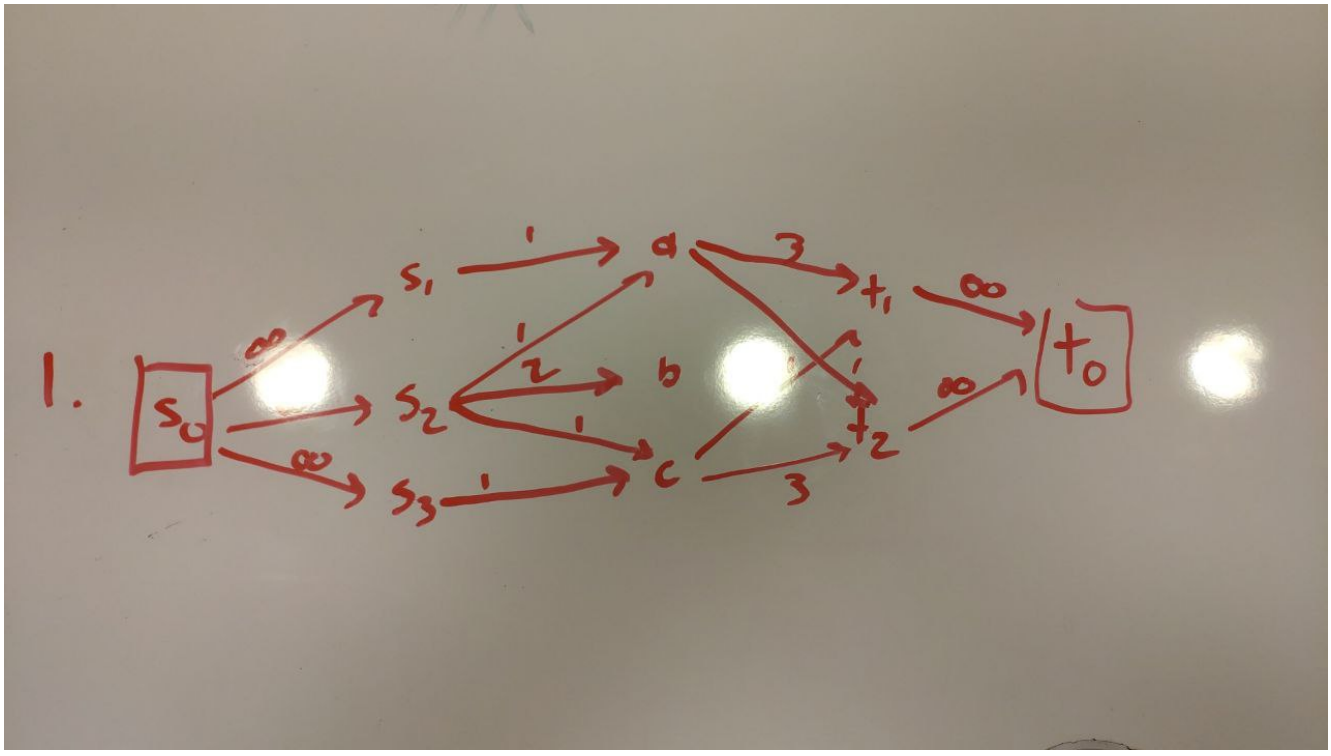
Problem 3 - Reducing Problems to Flow (15%)

1. Let figure 3 represent a company's transportation network from the factory (source) to the warehouse (sink), and the capacity of the roads in between intermediate cities (a, b, and c). Recently, the company has decided to expand and build more factories and warehouses. Their new flow network is shown in Figure 4. Modify Figure 4 to allow the new network to be solved with the single source, single sink Ford-Fulkerson algorithm.
2. Consider a different formulation of the hospital-resident matching problem that is arguably closer to reality. We have n hospitals and m residents. Each resident r ∈ R selects 3 hospitals h ∈ H that are acceptable; we assume that a hospital will accept any resident. A matching M in G is a subset of

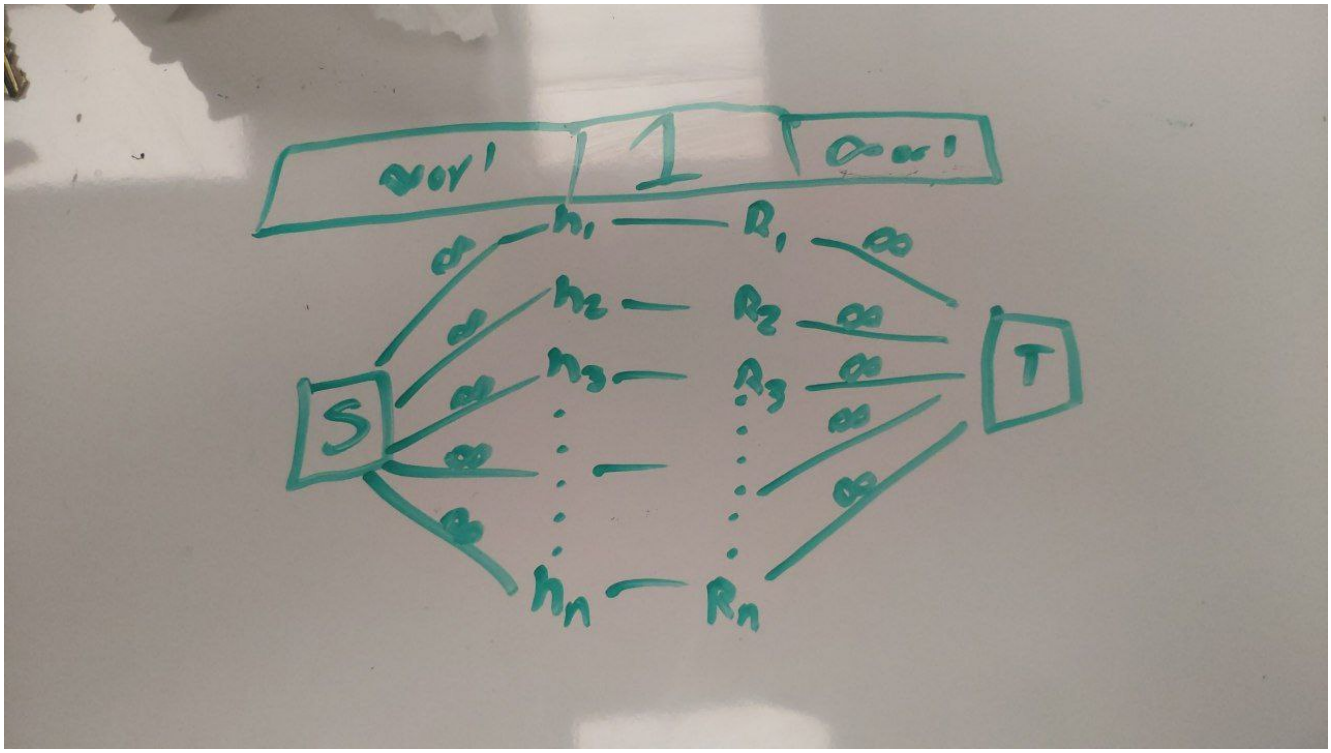
edges such that each node appears in at most one edge in M .
Give a polynomial-time algorithm to find an M with maximum cardinality. For this problem, you need to translate the text into a mathematical formulation, so define the data structures you use to model the problem. If you make assumptions, make sure you state them. You do not need to prove the runtime but justify why it is correct.



Solutions:

1. 

The multiple sources and sinks can be connected to an original source or sink over edges of infinite capacity. This allows for us to mock up the same problem with single source and sink so we can use other tools to evaluate the graph.

2. 

This flow network setup is a good mockup of a visual way to solve the matching problem. Since this graph must have cardinality in matching, each hosptial may potentially have many edges between residents and hospitals, but each hospital may only use one edge between a resident and hosptial. I gave the connections between the source and the hospitals infinite weight, becase they could theoretically have infinite flow (this could also be 1 becasue of later constraints, but for ease of concept I left it as infinity). The connections between the hopsitals and the edges can have an arbitrary edge weight, as long as the edges are equal, for simplicity I used 1. I also connected the residents to the sink over an edge with infinite capacity. Because of this, there are a variety of

possible matchings, but the set of matchings that maximizes flow is the "optimal" matching that maxemizes the number of hospitals matched to residents. Because of this, once we make the flow graph shown above, we can run Ford Folkerson on it to calculate the maximum flow. The value of maximum flow will be equal to the number of residents who are now matched with hospitals.