CS120: Intro. to Algorithms and their Limitations Prof. Salil Vadhan

Problem Set 6

Harvard SEAS - Fall 2021 *Due: Wed Oct. 26, 2022 (11:59pm)*

Your name: Aika Aldayarova
Collaborators:
No. of late days used on previous psets: 5
No. of late days used after including this pset:

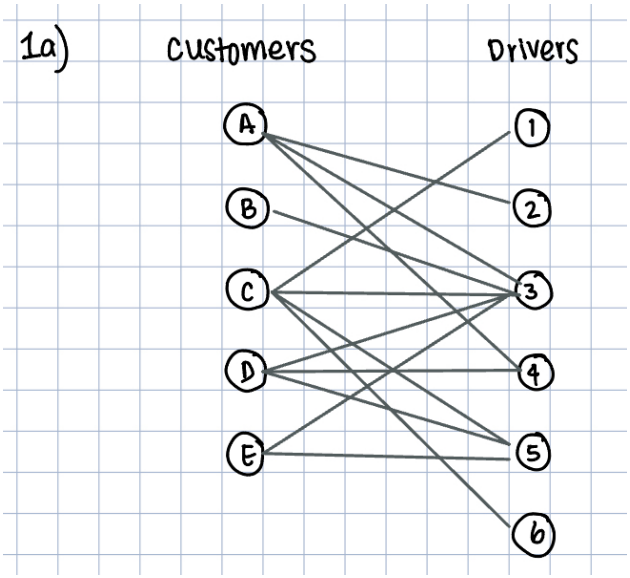
1. (Matching Algorithms)

   One practical application of matching algorithms is planning logistics, like in the following example from (fictional) ridesharing service Lyber in (real) New York City's Times Square. When a customer books a Lyber ride, the ride request is sent to a Lyber server and combined with others to create a schematic

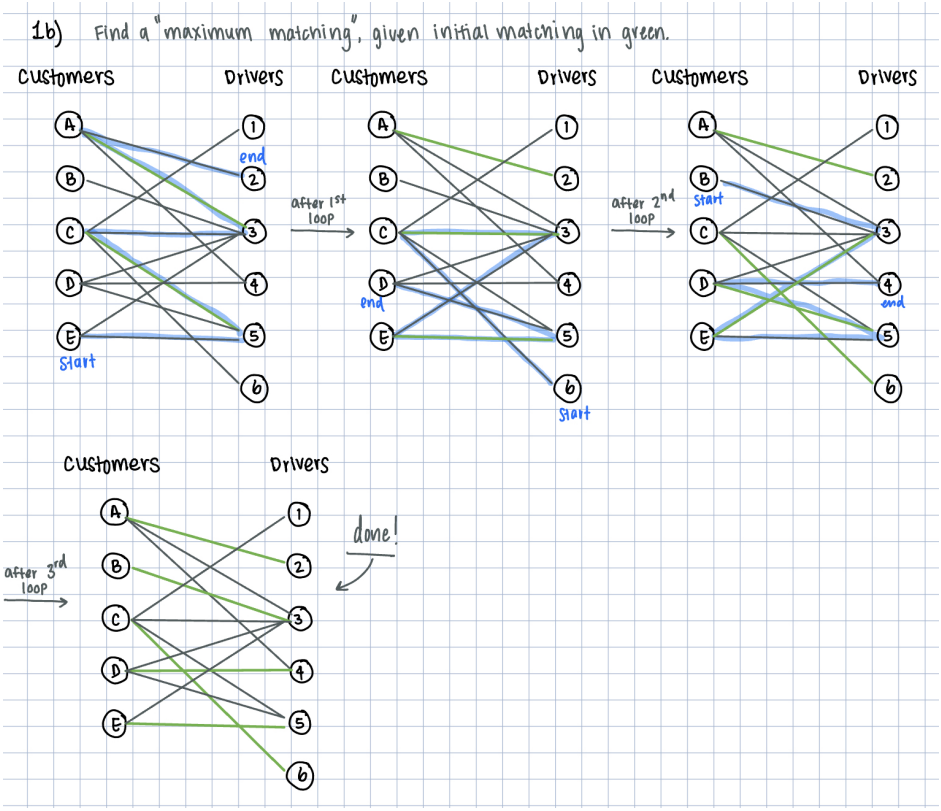
   like the one drawn in the map below:

Given a schematic like this, Lyber's goal is to serve as many customers (labeled A–E in the map) as possible, by assigning each one to a driver (labeled 1–6 in the map). For simplicity, each customer and driver is at an intersection, and assume driving between adjacent streets (vertical segment) takes 30 seconds, and driving between adjacent avenues (horizontal segments) takes 1 minute. However, the one twist is that they want to make sure that *no customer is waiting for longer than 2 minutes*. They also do not want to assign a driver to more than one customer at once, since serving a single customer can take more than 2 minutes.

(a) To perform the assignment, they reduce to Maximum Matching in bipartite graphs. Draw a bipartite graph corresponding to the drivers and customers in the map above.



(b) The Lyber app first prioritizes customers on Broadway, so they initially assign customer A to driver 3 and customer C to driver 5. Using the algorithm from class, find a *maximum matching* in the bipartite matching graph you've drawn, starting from the initial matching of A to 3 and C to 5. Draw pictures showing the sequence of matchings and augmenting paths you find. (No need to break down the steps of the algorithm to find the augmenting paths.)



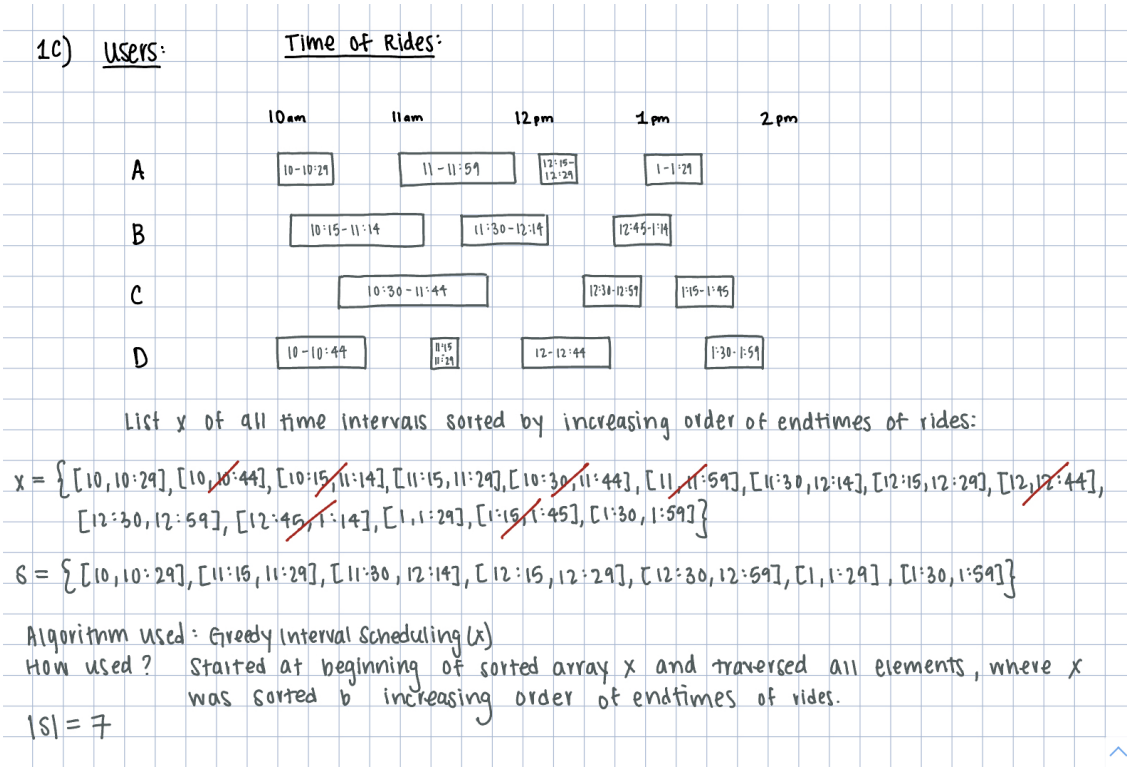
(c) The Lyber app also allows users to schedule trips in advance. One Lyber driver receives the following pre-programmed trips scheduled by 4 different users (all of them need multiple rides on the same day):

- User A: one trip from 10 to 10:29, another from 11 to 11:59, another from 12:15 to 12:29, and another from 13 to 13:29.
- User B: one trip from 10:15 to 11:14, another from 11:30 to 12:14, and another from 12:45 to 13:14.
- User C: one trip from 10:30 to 11:44, another from 12:30 to 12:59, and another from 13:15 to 13:44.
- User D: one trip from 10 to 10:44, another from 11:15 to 11:29, another from 12 to 12:44, and another from 13:30 to 13:59.

The Lyber driver wants to maximize the total number of different rides, because they earn a fixed rate per ride. (The driver is *not* trying to maximize the driving time).

We also assume for simplicity that the driver needs no time to move between different rides (i.e., it is possible that one ride finishes at 10:29 and the next one starts immediately at 10:30).

Use a greedy algorithm that we saw in class to find a maximum-sized set of non-conflicting rides. State which algorithm you are using, how you are applying it to this ride-selection problem, and write down the order in which the greedy algorithm selects the rides in the solution.



Answer: To elaborate on the written justification provided in the image, I am using the "Greedy Interval Scheduling" algorithm here by applying it to an array of time intervals (which are sorted by an increasing order of endtimes). Doing so, per lecture 12 notes' theorem 2.3, the algorithm will yield the optimal solution to the interval scheduling problem. It begins at the first element in the array and through the last element, checks to see if there exist conflicting time intervals. Once the algorithm is done running, the set S in the image above is the output set. It has a cardinality of 7 intervals.

2. (EthiCS Reflection) Suppose there are two patients in need of an immediate kidney transplant, but only one donor is currently available. The donor's kidney is compatible with both patients. Patient A is 30 years old, and is expected to live 6 additional years as a result of the transplant. Patient B is 60 years old, and is expected to live 10 additional years as a result of the transplant. *All else being equal* (e.g., both patients have an urgent need for the transplant; both patients have been on the kidney exchange waiting list for the same length of time), **which patient should the kidney go to, and why?** Your response should take the form of a short paragraph (3-4 sentence) reflection. *In explaining your ethical reasoning about the case, be sure to draw on at least one concept discussed in class.*

Answer: Although I sympathize with people who would prioritize saving the 60 year old patient, I think I lean more towards using the "Need-based Approach" discussed in lecture and saving the 30 year old first here. Generally, I see more benefit with the utilization of the "Outcome-based Approach", however in this specific scenario, I feel the difference between gaining 6 and 10 years of life to be small enough (in comparison to other patients) to start

to weigh other factors – specifically, how much life a given patient has lived. As much as I hate assigning a weight on someone's life, I think that an age difference between 30 and 60 years is sufficient enough to weigh the life of a 30 year old more in importance than that of the 60 year old. This is because even to this day, many people die at or around the age of 60; on the other hand, unless the person experiences a tragic accident, not many die as young as 30 years old. Being able to grant the 30 year old 6 more years of life would allow them to do more (e.g., get married, work, have children, explore passion projects) with their life, generally, than would giving the 60 year old 10 more years of life (when they're at a point in life when most begin to retire and slow their pace anyway).

3. (Vertex-Weighted Matching) For a graph $G = (V, E)$ and a subset $F \subseteq E$, let $V(F)$ denote the set $\bigcup_{f \in F} f$ of vertices that are an endpoint of at least one edge in F.

   (a) Prove that if $G = (V, E)$ is a graph and $M \subseteq E$ is a matching in G, then there is a maximum-size matching M' such that $V(M) \subseteq V(M')$. (Hint: consider constructing a maximum matching via augmenting paths, but starting with $M_0 = M$ rather than $M_0 = \emptyset$. What can you say about the $V(M_i)$'s?)

   Proof. Let us construct a maximum-size matching M' using the given initial matching M, by utilizing the "Maximum Matching Augmenting Paths Algorithm" from lecture 13. Let us have $p_0, \ldots, p_k$ be the augmenting paths produced by our algorithm, and let $M_0, \ldots, M_k$ be the associated matchings created from appending the initial matching M with the alternating augmenting paths; M_0 is our initial matching M. Since we will be using the "Maximum Matching Augmenting Paths Algorithm" which has repetitive steps within, let us prove the validity of the statement $V(M) \subseteq V(M')$ by using induction and inducting on k – the number of possible augmenting paths.

   Base Case: When k = 0, working under the assumption that $M_0, \ldots, M_k$ are matchings created from appending M with the associated alternating augmenting paths, M_k (the next largest matching) would be M_0, which we defined to be M, the initial matching given. Then, if $M = M$ and $M' = M_k$, $V(M) \subseteq V(M')$ will be $V(M) \subseteq V(M_k)$, and we know this to true since any set is a subset of itself.

   Inductive Hypothesis: Let us assume $V(M) \subseteq V(M_i)$ for some $i < k$ is true.

   Inductive Step: We want to show that $V(M) \subseteq V(M_{i+1})$ for some $i < k$ will be true as well. Using our inductive hypothesis, we can claim that $V(M) \subseteq V(M_i) \subseteq V(M_{i+1})$ will be true; the first part of the claim ($V(M) \subseteq V(M_i)$) can be said because that is simply our inductive hypothesis; the second part of the claim ($V(M) \subseteq V(M_{i+1})$) can be said because by the definition of the algorithm we are using in this problem and by the definition of augmenting paths, augmenting paths with respect to M_i will never unmatch currently matched vertices. In other words, as we increase the matching, though the set of matched edges will change, the set of matched vertices will always contain the same if not more vertices and never less. Having justified our created claim, we can then say that $V(M) \subseteq V(M_{i+1})$ by the definition of subsets. We can tie this back to the initial assertion we wanted to prove, which was $V(M) \subseteq V(M')$, by realizing

that $V(M_{i+1}) = V(M')$ by the explanation in our base case step that outlined the subscripted Ms and their meanings. Thus, by the Principle of Mathematical Induction, our assertion has been proved.
Q.E.D.

(b) In the Embedded EthiCS module, we saw how simply maximizing the *size* of a matching may not always be the right objective. Thus, it is natural to consider weighted versions of the matching problem. Suppose we we consider vertex-weighted graphs $G = (V, E, w)$, w is an array specifying a nonnegative edge weight $w(v)$ for every $v \in V$. (For example, the weight assigned to a patient might correspond to the number of extra years of life they would gain from a donation.) The goal of the *vertex-weighted maximum matching problem* is to find a matching M maximizing its *total weight*

$$w(M) = \sum_{\{u,v\} \in M} (w(u) + w(v)).$$

(This corresponds to the utilitarian objective discussed in Embedded EthiCS module.) Using Part 3a, prove that every graph G has a matching M^* that simultaneously maximizes both total weight and size. That is, for every matching M in G, we have both $w(M) \leq w(M^*)$ and $|M| \leq |M^*|$.

Proof. Let us assume that our initial matching M is a matching that maximizes $w(M)$ over all possible matchings. Let us also assume that any further matching M', that has a greater number of vertices than M, will have appended vertices whose weight is 0; thus even if M' exceeds M in size, it will always be of equivalent weight to M.

In part 3a of the problem set, we proved that for a maximum-sized matching M', $V(M) \subseteq V(M')$ is true. We begin with a matching M and assume that it is a matching that yields the maximum possible weight. Using the work proved in 3a of the problem set, we can find a matching M' that maximizes the size of the matching. Since all the vertices of $V(M)$ are included in $V(M')$ and have non-negative weights, if we begin with M and use the work from 3a, we not only build out a matching M' that increases the size of the possible matching (satisfying the statement $|M| \leq |M^*|$), but we also build out a macthing M' that has at least the same weight as the original matching M (satisfying the statement $w(M) \leq w(M^*)$).
Q.E.D.

(c) (optional[1]) Explain why the same holds for the maximin objective discussed in the Embedded EthiCS module. That is, there is always a matching M that simultaneously maximizes the maximin objective and $|M|$.

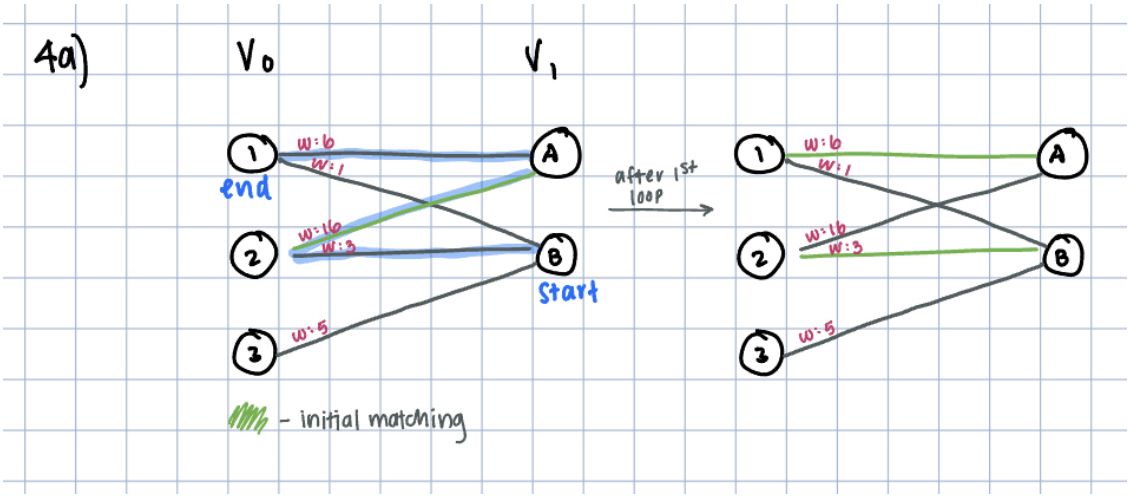
4. (Edge-Weighted Bipartite Matching) Instead of considering vertex weights, we could instead study matching on *edge-weighted* bipartite graphs $G = (V, E, w)$, where w is an array specifying a nonnegative edge weight $w(e)$ for every $e \in E$.

[1]This problem won't make a difference between N, L, R-, and R grades. As this problem is purely extra credit, course staff will deprioritize questions about this problem at office hours and on Ed.

The goal of the *edge-weighted maximum matching problem* is to find a matching M maximizing

$$w(M) = \sum_{e \in M} w(e).$$

(a) Construct an edge-weighted bipartite graph $G = (V, E, w)$ such that there is no matching M in G that simultaneously maximizes the weight $w(M)$ and the size $|M|$. Thus, there can be an inherent tension between these two objectives.



Explanation of the graph: This graph shows that if the matching maximizes size, then the matching cannot simultaneously maximize the weight. The initial matching between vertices 2 and A maximized the matching in weight; once the algorithm "Maximum Matching Augmenting Paths" was applied, the matching between vertices 1 and A, and between 2 and B (together) had a lower weight despite the maximized size.

(b) (optional[1])

One real-life constraint in kidney exchange is that donors d are often associated with a particular patient p (e.g. a close family member) such that d is only willing to donate a kidney if p receives a kidney from someone. (d would donate their kidney directly to p if they could, but they are incompatible.) Suppose we have an (unweighted) bipartite graph $G = (V_0 \cup V_1, E)$ representing n such donor-patient pairs, i.e. $V_0 = \{d_0, d_1, \ldots, d_{n-1}\}$, $V_1 = \{p_0, p_1, \ldots, p_{n-1}\}$, where p_i is the patient associated with donor d_i. We assume $\{d_i, p_i\} \notin E$ for each i. Our goal is to find a matching M of the largest possible size $|M|$, subject to the constraint that d_i is matched in M only if p_i is matched in M.

Show that we can reduce this constrained version of the maximum matching problem to finding a maximum-weight *perfect* matching in an appropriate edge-weighted bipartite graph, where a perfect matching is a matching that matches all the vertices in the graph. (Hint: add edges $\{d_i, p_i\}$ with appropriate edge weights.)