

Problem Set 6

Harvard SEAS - Fall 2024

Due: Wed Oct. 30, 2024 (11:59pm)

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

The purpose of this problem set is practice proving optimality and efficiency of a greedy algorithm, practice modelling problems using graphs, reinforce understanding of the matching algorithm covered we learned, and think about ethical issues raised when modelling real-world problems for algorithmic solution.

1. (Greedy Coloring for Interval Scheduling) The IntervalScheduling-Optimization problem we studied in class finds the largest group of nonintersecting intervals. In many applications, it is also natural to consider the *coloring* version of the problem, where we want to partition the input intervals into as few groups as possible so that each group is nonintersecting.

In this problem, you will prove that Greedy Coloring in order of *increasing start time* gives optimal coloring for interval scheduling. (Note the contrast with the *increasing finish time* ordering we used for the version studied in class. It is a common phenomenon that different orderings are better for coloring vs. independent-set problems; for example decreasing vertex degree is a good heuristic for greedy coloring of general graphs, while increasing vertex degree is a good heuristic for independent set.) Let $x = (x_0, \dots, x_{n-1})$ be an instance of IntervalScheduling, where each x_i is an interval $[a_i, b_i]$ with $a_i, b_i \in \mathbb{Q}$. Let k be the maximum number of input intervals that contain any value $t \in \mathbb{Q}$. That is,

$$k = \max_{t \in \mathbb{Q}} |\{i \in [n] : t \in x_i\}|.$$

- (a) Prove that every proper coloring for IntervalScheduling uses at least k colors.
- (b) Show that the Greedy Coloring in order of *increasing start time* uses at most k colors. (To develop your intuition, carry out the algorithm on a few examples.)
- (c) Show that the Greedy Coloring in order of increasing start time can be implemented in time $O(n \log n)$. Hints:
 - i. Keep track of the end times of the most recently scheduled intervals assigned to each color, and use an appropriate data structure to ensure that you spend only $O(\log k)$ rather than $O(k)$ time per iteration, where k is the number of colors used.
 - ii. To make life easier for yourselves, you may instead implement a *variant* of Greedy Coloring in which, at every step, you assign a vertex *any color* not assigned to its neighbours that's also less than the largest color (as opposed to standard Greedy Coloring in which you assign the smallest color).

2. (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.)
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?)
- (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 consider vertex-weighted graphs $G = (V, E, w)$, w is an array specifying a nonnegative vertex 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 $|M| \leq |M^*|$.

This still leaves the question of whether there efficient algorithms to optimize vertex-weighted matching. This problem can be reduced to the maximum-flow problem, which is covered in CS1240.

- (c) (optional¹) Show how to reduce matching with the *maximin* objective to vertex-weighted matching,² and deduce that there is always a matching M that simultaneously maximizes the maximin objective and $|M|$. For simplicity, you may assume that there are no ties in how well off the patients are prior to treatment. (Hint: use weights that are powers of 2.)

¹This 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.

²In lecture, Salil said that he did not know whether there were efficient algorithms to optimize the maximin objective, but afterwards we realized that this reduction allows it to be solved via maximum flow algorithms, as covered in CS1240.

4. (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 starts at 30 QALYs and is expected to live 3 additional QALYs as a result of the transplant. Patient B starts at 45 QALYs, and is expected to live 10 additional QALYs as a result of the transplant. *All else being equal*, **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.

Note: As with the previous psets, you may include your answer in your PDF submission, but the answer should ultimately go into a separate Gradescope submission form.

5. Once you're done with this problem set, please fill out [this survey](#) so that we can gather students' thoughts on the problem set, and the class in general. It's not required, but we really appreciate all responses!