

CPSC 320 2022W1: Tutorial Quiz 3, Individual Portion Solutions

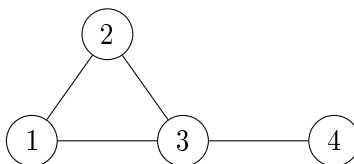
October 2022

Colour Me Puzzled (Grid-Free Version)

We are given a graph $G = (V, E)$. We want to colour each vertex with one of k colours so that two end points of any edge in E receive different colours. This is called a *vertex k -colouring* of the graph.

Recall that the *degree* of a vertex $v \in V$ is the number of edges incident on v . Let d be the *maximum degree* in the graph.

Example:



The maximum degree d in this graph is 3. The graph has a 3-colouring (by using different colours for nodes 1, 2, and 3 and colouring node 4 the same as 1 or 2) and a 4-colouring (by using a different colour for every node), but does not have a 2-colouring.

1. For any value of d , describe a graph with maximum degree d that can be coloured by two colours.

The solution to this question will be included in the solutions to Assignment 3.

2. Design a greedy algorithm that will colour vertices of G with at most $d + 1$ colors. Your algorithm should consider vertices in an arbitrary order and then greedily choose the colour of each vertex.

The solution to this question will be included in the solutions to Assignment 3.

3. Explain why your algorithm works correctly (i.e., why it will use at most $d + 1$ colours).

The solution to this question will be included in the solutions to Assignment 3.

Shelter Skelter









You're the manager of an animal shelter, which is run by a few full-time staff members and a group of n volunteers. Each of the volunteers is scheduled to work one shift during the week. There are different jobs associated with these shifts (such as feeding the animals, interacting with visitors to the shelter, handling administrative tasks, etc.), but each shift is a single contiguous interval of time. There can be multiple shifts going on at once.

You'd like to arrange a weekly meeting with your staff and volunteers, but you have too many volunteers to be able to find a meeting time that works for everyone. Instead, you'd like to identify a suitable subset of volunteers to instead attend the weekly meeting. You'd like for every volunteer to have a shift that overlaps (at least partially) with a volunteer who is attending the meeting. Your thinking is that you can't personally meet with every single volunteer, but you would like to at least meet with people who have been

working with every volunteer (and may be able to let you know if a volunteer is disgruntled or having any difficulties with their performance, etc.). Because your volunteers are busy people, you want to accomplish this with the fewest possible volunteers.

Your volunteer shifts are given as an input list V , and each volunteer shift v_i is defined by a start and finish time (s_i, f_i) . Your goal is to find the smallest subset of volunteer shifts such that every shift in V overlaps with at least one of the chosen shifts. A shift $v_i = (1, 4)$ overlaps with the shift $v_j = (3, 5)$ but not with the shift $v_k = (4, 6)$.

1. In each of the following greedy algorithms, answer "Yes" if the algorithm *always* constructs an optimal schedule, and answer "No" otherwise.

	Yes	No	Algorithm:
1			Choose the shift v that ends last, discard all shifts that overlap with v , and recurse on the remaining shifts.
2			Choose the shift v that starts first, discard all shifts that overlap with v , and recurse on the remaining shifts.
3			Choose the shift v with longest duration ($f - s$), discard all shifts that overlap with v , and recurse on the remaining shifts.
4			Choose the shift v that overlaps with the most other shifts, discard all shifts that overlap with v , and recurse on the remaining shifts.

2. At least one of the greedy strategies referenced in question 1 is not optimal. For **one** of the greedy strategies for which you answered “No” on the previous page, provide and briefly explain a counterexample that shows the algorithm is not optimal. Your counterexample may not rely on tie-breaking behaviour.

- (a) Greedy strategy (enter the number of the non-optimal greedy strategy from the previous page you've chosen):

- (b) Counterexample:

You might have chosen any of the four strategies above for providing your counterexample, as none of them are optimal (we didn't want to give away the answer to the corresponding assignment question!). Below we'll provide counterexamples for all four.

For strategy 1, a counterexample (drawn below in ASCII art, with each line representing a shift with its start on the left and finish on the right) is:

The optimal solution is to choose the one shift in the bottom row, but the greedy solution will choose the two shifts in the top row.

The same counterexample we gave for strategy 1 also serves as a counterexample for strategy 2, for the same reasons (greedy will choose the two shifts on top, while the optimal solution is to choose the one on the bottom).

For strategy 3, a counterexample is:

If each dash in the dashed line represents a unit of time, the two shifts in the top row are both 7 time units and the shift in the bottom row is 3 units. The greedy strategy would select the two shifts on the top row, while the optimal solution is to select the single shift in the bottom row.

A counterexample for strategy 4 is tricky (especially if we want to construct an example that doesn't rely on the greedy strategy breaking ties in a particular way), and we won't draw it with ASCII art just so we can make all the overlaps (and non-overlaps) entirely explicit. Suppose our shifts are:

- $v_1 = (1, 3)$;
- $v_2 = (2, 4)$;
- $v_3 = (3, 4)$;
- $v_4 = (3, 5)$;
- $v_5 = (4, 5)$;
- $v_6 = (4, 6)$;
- $v_7 = (5, 7)$.

The optimal solution is to select v_2 and v_6 . In terms of overlap: v_1 overlaps with 1 shift, v_2 with 3 shifts, v_3 with 2 shifts, v_4 with 4 shifts, v_5 with 2 shifts, v_6 with 3 shifts, and v_7 with 1 shift. Thus, the greedy strategy will select v_1, v_4 and v_7 .

Film Festivities

You and a few friends are travelling in Europe this summer, and one of your stops is at a 10-day international film festival! Tickets are expensive and you and your friends are big movie fans, so you want to maximize your cultural enjoyment by watching as many films as you can.

A *film* is represented by a pair (s, f) where s is its start time and f is its finish time, relative to the start of the festival. There are n performances across 10 days, hosted across many theatres. Your goal is to maximize the number of non-overlapping films in your film festival itinerary.

1. In each of the following greedy algorithms, answer "Yes" if the algorithm *always* constructs an optimal schedule, and answer "No" otherwise.

	Yes	No	Algorithm:
1	<input type="radio"/>	<input checked="" type="radio"/>	Choose the film m that ends last, discard all films that conflict with m , and recurse on the remaining films.
2	<input checked="" type="radio"/>	<input type="radio"/>	Choose the film m that starts last, discard all films that conflict with m , and recurse on the remaining films.
3	<input type="radio"/>	<input checked="" type="radio"/>	Choose a film m that conflicts with the fewest other films, discard all films that conflict with m , and recurse on the remaining films.
4	<input type="radio"/>	<input checked="" type="radio"/>	If no films conflict, choose them all. Otherwise, discard a films that conflicts with the most other films and recurse on the remaining films.

2. At least one of the greedy strategies referenced in question 1 is not optimal. For **one** of the greedy strategies for which you answered "No" on the previous page, provide and briefly explain a counterexample that shows the algorithm is not optimal.

- (a) Greedy strategy (enter the number of the non-optimal greedy strategy from the previous page

you've chosen):

- (b) Counterexample:

Proof of correctness for strategy 2

You may have noticed that the “film festival” problem is essentially a re-worded version of the interval scheduling problem described in section 4.1 of the textbook. In the proof below we have replaced the word “film” with the word “job” to make the links between the proofs (ours and the textbook’s) more explicit.

First, we already know from the textbook that choosing the latest finish time first is a successful greedy algorithm. Now, take any instance, choose a time t such that $t > f_i$ for all finish times f_i of all jobs. Then, produce a new instance where each job (s_i, f_i) is replaced with $(t - f_i, t - s_i)$. Now, our greedy algorithm run on this produces exactly the same set of jobs as the textbook’s greedy algorithm run on the original instance. Since a pair of jobs conflicts in one instance if and only if it conflicts in the other, the same set of jobs must be in the optimal solution(s) to both instances. Thus, our greedy algorithm is also optimal.

Second, proving this directly: Let our greedy algorithm’s solution to an arbitrary instance be \mathcal{G} . Imagine an optimal solution \mathcal{O} . Consider the jobs in order of decreasing start time and find the first job i that is in \mathcal{O} but not in \mathcal{G} or vice versa. There are three cases which together form an inductive argument that \mathcal{G} is optimal:

- (a) No such job exists, the two solutions are the same, and so \mathcal{G} has at least as many jobs as \mathcal{O} and is also optimal.
- (b) The job is in \mathcal{O} but not in \mathcal{G} . But, that means the job doesn’t conflict with any of the later-start-time jobs that \mathcal{G} and \mathcal{O} agreed on. But in that case \mathcal{G} would have included it in its solution (as that’s how the greedy algorithm selects the next job). Thus, this case cannot arise.
- (c) The job is in \mathcal{G} but not in \mathcal{O} . In that case, we can remove the job with the next lower start time from \mathcal{O} and add this job instead. The new job cannot conflict with any other job that starts after this one in \mathcal{O} or it would have conflicted also in \mathcal{G} and not been selected. The new job also cannot conflict with any other job that starts earlier than this one, because then that job would also have conflicted with the job we removed from \mathcal{O} . Thus, the new solution is valid and has the same number of jobs as \mathcal{O} and so is also optimal but more similar to greedy. (We can inductively extend this observation to make an optimal solution identical to \mathcal{G} , as in case 1.)