Instructions

- This problem set is **open book**: you may refer to the lectured material found on Canvas and the recommended books to help you answer the questions.
- This problem set is an **individual effort**. You must arrive at your answers independently and write them up in your own words. Your solutions should reflect your understanding of the content.
- **Posting questions to message boards or tutoring services including, but not limited to, Chegg, StackExchange, etc., is STRICTLY PROHIBITED. Doing so is a violation of the Honor Code.**
- Your solutions must be submitted typed in LaTeX, **handwritten work is not accepted**. If you want to include a diagram then we do accept photos or scans of hand-drawn diagrams included with an appropriate `\includegraphics` command. It is your responsibility to ensure that the photos you obtain are in a format that pdflatex understands, such as JPEG.
- The template tex file has carefully placed comments (% symbols) to help you find where to insert your answers. There is also a **STUDENT DATA** section in which you should input your name and ID, this will remove the warnings in the footer about commands which have not been edited. You may have to add additional packages to the preamble if you use advanced LaTeX constructs.
- You must CITE any outside sources you use, including websites and other people with whom you have collaborated. You do not need to cite a CA, TA, or course instructor.
- Take care with time, we do not usually accept problem sets submitted late.
- Take care to upload the correct pdf with the correct images inserted in the correct places (if applicable).
- Check your pdf before upload.
- **Check your pdf before upload.**
- **CHECK YOUR PDF BEFORE UPLOAD.**

Quicklinks: 1 2 (2a) (2b) (2c)

Problem 1

Magnetic tapes are an old but still-used technology for archiving data. A big problem with tape is that it must be read sequentially, so to access a file stored in the middle of the tape we have to wind on to find that file. Suppose we have stored n files on the tape in order, with lengths $L_1, \ldots, L_n$. Then to access the k-th file we have to wind on a length $L_1 + L_2 + \ldots + L_{k-1}$ to find the beginning of the file, and L_k more to read the entire k-th file. So the cost of accessing file k is $C(k) = \sum_{i=1}^{k} L_i$.

Supposing that each file is equally likely to be accessed, let X be the random variable representing the cost of accessing a random file. Then $E(X)$ is the expected cost of accessing a random file, and

$$E(X) = \frac{1}{n}\sum_{k=1}^{n} C(k) = \frac{1}{n}\sum_{k=1}^{n}\sum_{i=1}^{k} L_i.$$

Consider the greedy algorithm that at each step stores a shortest file on the tape and removes this file from consideration, and iterates this step until no files are left. So the greedy solution will have the files stored in increasing order of length.

A different solution must have an index k such that $L_k > L_{k+1}$. Now suppose that $L_k > L_{k+1}$ and show that exchanging the order of these files, so the list now reads

$$L_1, \ldots, L_{k-1}, L_{k+1}, L_k, L_{k+2}, \ldots, L_n$$

reduces $E(X)$ by the positive number $(L_k - L_{k+1})/n$.

Hint: consider each file and the cost of accessing it before and after the swap.

To finish this proof that the greedy algorithm is optimal by writing one or two sentences describing how this fact shows that putting the files in increasing order of length minimizes $E(X)$.

Let greedy solution G have files stored such that $L_1 < L_2 < L_3 < ... < L_{k-1} < L_k < L_{k+1} < ... < L_n$

Let different solution D have files stored such that $L_1 < L_2 < L_3 < ... < L_{k-1} < L_{k+1} < L_k < L_{k+2} < ... < L_n$

Then we have:

$$E(D(k)) = \frac{C(L_1) + C(L_2) + C(L_3) + ... + C(L_{k-1}) + C(L_{k+1}) + C(L_k)}{n}$$
$$E(G(k)) = \frac{C(L_1) + C(L_2) + C(L_3) + ... + C(L_{k-1}) + C(L_k)}{n}$$
$$E(D(k)) - E(G(k)) = \frac{C(L_{k+1})}{n}$$

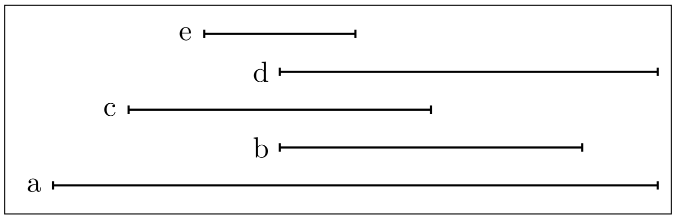
Therefore, we see that $E(D(k)) > E(G(k))$ and thus a greedy algorithm that puts the files in increasing order of length minimizes $E(X)$.

Problem 2

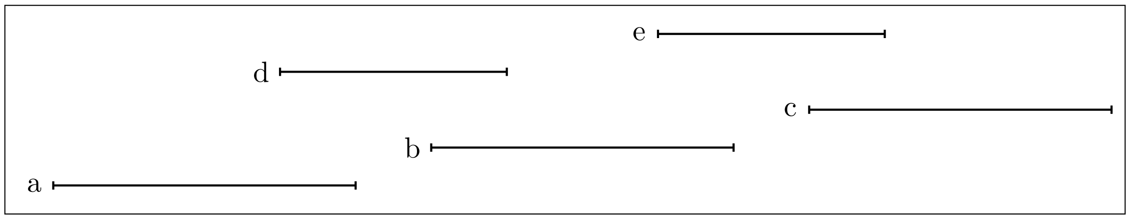
In this question we consider the interval scheduling problem, as covered in class.

A comment on level of justification (applies to all of problems 2 and 3): To help us understand your thinking, it is worth writing a little about the order in which the algorithm selects the intervals. For example "The algorithm takes intervals in the order a,f,c: first the algorithm takes interval a because that is the shortest. Interval a conflicts with intervals b and e, so they are removed. The next shortest is f, which conflicts with interval d, and the only remaining interval is c."

We have provided you with some sample code to draw such a figure natively in LaTeX, you just need to modify the start/end points and possibly add more intervals if you need (as well as answer the rest of the question!). You may also hand-draw the intervals and `\includegraphics` your images, provided they are legible. Your explanations must still be typed using LaTeX. A sample figure is below:



(2a) Consider a greedy algorithm which always selects the shortest interval at each step. Draw an example with at least 5 intervals and at most 10 intervals where this algorithm fails. List the order in which the algorithm selects the intervals, and also write down a larger subset of non-overlapping intervals than the subset output by the greedy algorithm.



For clarity, the intervals are as follows where L_i represents the length of interval i:

a: [0,4] ($L_a = 4$)

b: [5,9] ($L_b = 4$)

c: [10,14] ($L_c = 4$)

d: [3,6] ($L_d = 3$)

e: [8,11] ($L_e = 3$)

Assume that in the event intervals are of the same length, the algorithm will choose alphabetically.

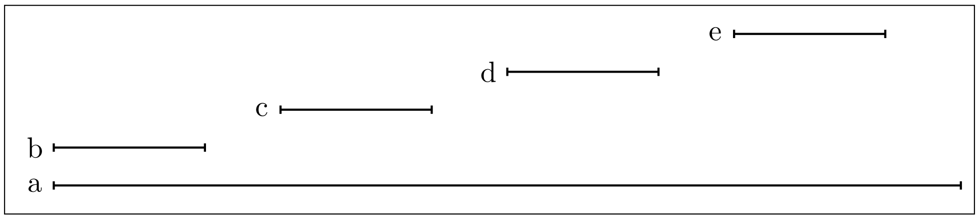
Let set G be the set of intervals chosen by our greedy algorithm, set O be the optimal set, and set A be the subset of non-overlapping intervals available to choose from after each iteration of the algorithm.

Step 1: The shortest intervals are d and e with $L_d = L_e = 3$. We choose d as d is first alphabetically. So we have $G = \{d\}$ and $A = \{e, c\}$. Intervals a and b are removed from subset A because they overlap with our choice d.

Step 2: We now choose the shortest interval in subset $A = \{e, c\}$. We choose e as $L_e < L_c$. So we have $G = \{d, e\}$ and $A = \{\}$. Interval c is removed from subset A because it overlaps with our choice e.

Step 3: Since subset A is empty, there are not any non-overlapping intervals left to choose from and our final set G has two intervals d, e. (That is $|G| = 2$). We note that the optimal set is $O = \{a, b, c\}$ and $|O| = 3$. Since $|O| > |G|$, we can conclude that a greedy algorithm which always selects the shortest interval at each step is not always optimal.

(2b) Consider a greedy algorithm which always selects the longest interval at each step. Draw an example with at at least 5 intervals and at most 10 intervals where this algorithm fails. Show the order in which the algorithm selects the intervals, and also show a larger subset of non-overlapping intervals than the subset output by the greedy algorithm. The same comments apply here as for (2a) in terms of level of explanation.



For clarity, the intervals are as follows where L_i represents the length of interval i:

a: [0,12] ($L_a = 12$)

b: [0,2] ($L_b = 2$)

c: [3,5] ($L_c = 2$)

d: [6,8] ($L_d = 2$)

e: [9,11] ($L_e = 2$)

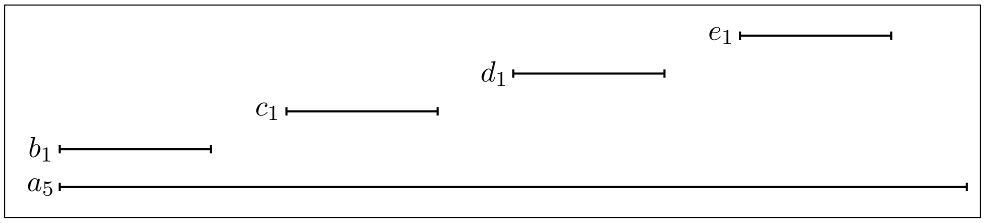
Let set G be the set of intervals chosen by our greedy algorithm, set O be the optimal set, and set A be the subset of non-overlapping intervals available to choose from after each iteration of the algorithm.

Step 1: The longest interval is a with $L_a = 12$. We choose a and so we have $G = \{a\}$ and subset $A = \{\}$. Intervals b, c, d, e are removed because they all overlap with our choice a.

Step 2: Since subset A is empty, there are not any non-overlapping intervals left to choose from and our final set G has one interval a. (That is $|G| = 1$). We note that the optimal set is $O = \{b, c, d, e\}$ and $|O| = 4$. Since $|O| > |G|$, we can conclude that a greedy algorithm which always selects the longest interval at each step is not always optimal.

(2c) Consider now the *weighted interval scheduling problem*, where each interval i is specified by $(\text{start}_i, \text{end}_i, \text{weight}_i)$. Here, you may assume $\text{weight}_i > 0$. We seek a set S of pairwise non-overlapping intervals that maximizes $\sum_{i \in S} \text{weight}_i$. That is, rather than maximizing the number of intervals, we are seeking to maximize the sum of the weights.

Consider a greedy algorithm, which selects the interval which ends earliest at each step. Draw an example with at at least 5 intervals and at most 10 intervals where this algorithm fails. Show the order in which the algorithm selects the intervals, and also show a subset with larger weight of non-overlapping intervals than the subset output by the greedy algorithm.



For clarity, the intervals are as follows where E_i represents the end of interval i and W_i represents the weight of the interval:

a: [0,12] ($E_a = 12$, $W_a = 5$)

b: [0,2] ($E_b = 2$, $W_b = 1$)

c: [3,5] ($E_c = 5$, $W_c = 1$)

d: [6,8] ($E_d = 8$, $W_d = 1$)

e: [9,11] ($E_e = 11$, $W_e = 1$)

Let set G be the set of intervals chosen by our greedy algorithm, set O be the optimal set, and set A be the subset of non-overlapping intervals available to choose from after each iteration of the algorithm.

Step 1: The interval which ends earliest is b, with $E_b = 2$. We choose b and so we have $G = \{b\}$ and subset $A = \{c, d, e\}$. Interval a is removed because it overlaps with our choice b.

Step 2: We now choose the interval that ends the earliest in subset $A = \{c, d, e\}$. We choose interval c because $E_c = 5 < E_d < E_e$. So we have $G = \{b, c\}$ and subset $A = \{d, e\}$. No additional intervals are removed from A because none of them overlap with our choice c.

Step 3: Again we choose the interval that ends the earliest in subset $A = \{d, e\}$. We choose interval d because $E_d = 8 < E_e$. So we have $G = \{b, c, d\}$ and subset $A = \{e\}$. No additional intervals are removed from A because none of them overlap with our choice d.

Step 4: Again we choose the interval that ends the earliest in subset $A = \{e\}$. We choose interval e because it is the only choice available. So we have $G = \{b, c, d, e\}$ and subset $A = \{\}$.

Step 5: Since subset A is empty, there are not any non-overlapping intervals left to choose from and our final set G has four intervals b, c, d, e. The sum of the weights is $W_b + W_c + W_d + W_e = 1 + 1 + 1 + 1 = 4$. We note that the optimal set is $O = \{a\}$ and the sum of the weights in set O is equal to $W_a = 5$. Since the sum of the weights in set O is greater than the sum of the weights in set G, we can conclude a greedy algorithm that always selects the interval which ends earliest is not always optimal when the goal is to maximize $\sum_{i \in S} \text{weight}_i$.