Homework 2

CS 4104 (Summer 2024)

Assigned on June 20, 2024.
Submit a PDF file containing your solutions on Canvas by 11:59pm on Friday June 28, 2024.

Instructions:

- The Honor Code applies to this homework with the following exception:
  - You are allowed to discuss possible algorithms and bounce ideas with your classmates. Do not discuss proofs of correctness, final answer or running time in detail with your classmate. You must write down your solution individually and independently. Do not send a written solution to your classmates for any reason whatsoever.
  - You are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TA, and the instructor. **In particular, do not use a search engine or large language models such as ChatGPT**.
- Do not forget to typeset your solutions. Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as `n2`. You can use the LaTeX version of the homework problems to start entering your solutions. At the end of each problem are three commented lines that look like the example below. You can uncomment these lines and type in your solution within the curly braces.

```
% \solution{
%
% }
```

- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.
- If you are proposing an algorithm as the solution to a problem, keep the following in mind:
  - Describe your algorithms as clearly as possible. The style used in the text book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed pseudo-code without an explanation, we will not grade your solutions.*Do not describe your algorithms only for a specific example you may have worked out.
  - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
  - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.
- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is m + n, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$time.

Acknowledgement: Some of the questions in this Homework are inspired by/adopted from previous CS 4104 offerings at VT including by T. M. Murali and others.

Problem 1 Imagine you are a freelancer with multiple job offers, each with its own deadline and payment. Your goal is to maximize your earnings by choosing the right jobs to complete within their deadlines. You can only work on one job at a time, so you must be strategic in selecting which jobs to take. Example of the job listening is provided in Table 1.

Job	Deadline	Pay
J_1	2	50
J_2	1	30
J_3	2	40
J_4	1	20
J_5	3	60
J_6	2	10
J_7	3	25
J_8	1	15

Table 1: Job deadlines and payments

1. (5 Points) Design a greedy algorithm that helps you pick the jobs in a way that maximizes your profit. Write the pseudocode for this algorithm.
2. (5 Points) Provide a proof that your greedy algorithm always yields the correct answer, ensuring that the total earnings are maximized. You can explain your proof in plain English without following any strict structure of loop invariant or mathematical induction.
3. (5 Points) You learned from a friend that the some jobs actually require purchasing specific software licenses. Each job requires a different software, and you need to pay for the licenses out of your own pocket. Fortunately, your friend knows the price of each license, and you have received an updated job list as presented in Table 2. She also informed you that the price of each license is guaranteed to be lower than the pay for the corresponding job. How would you modify your algorithm to achieve the same objective as before with this new information?

Job	Deadline	Pay	Licence Cost
J_1	2	50	23
J_2	1	30	17
J_3	2	40	19
J_4	1	20	11
J_5	3	60	49
J_6	2	10	6
J_7	3	25	10
J_8	1	15	0

Table 2: Job deadlines, payments, and license costs

Problem 2 (8 Points) Given a connected, undirected graph $G = (V, E)$ with the following weighted edges:

$$E = \{(A, B, 4), (A, C, 2), (B, C, 1), (B, D, 5), (C, D, 8), (C, E, 10), (D, E, 2)\}$$

Use Kruskal's algorithm to find the Minimum Spanning Tree (MST). List the edges in the MST and the total weight. Show each step of the algorithm by listing the edges/vertices being considered/explored.

Problem 3 (7 Points) Given the directed graph with edge weights presented in Table 3, use Dijkstra's algorithm to find the shortest path from vertex A to vertex C. Show each step of the algorithm and the final shortest path along with its total weight.

Edge	Start	End	Weight
e_1	A	B	6
e_2	A	D	1
e_3	D	B	2
e_4	B	E	2
e_5	D	E	1
e_6	E	C	5
e_7	B	C	5
e_8	A	F	7
e_9	F	G	3
e_{10}	G	C	4
e_{11}	F	C	8

Table 3: Edges of the graph with their weights

Problem 4 (30 points) Your friend designed and built a robot that can solve mathematical puzzles. However, the robot has a difficult time when it comes to fractions. It only understands fractions when they are expressed in the form $\frac{1}{n}$. These are known as *unit fractions*. Interestingly, to understand any other fraction, you need to convert it to a sum of unit fractions. For example, the fractions presented in the first column of Table 4 had to be converted to the representations in the second column. This friend of yours knows that you took CS 4104 and heard that there is a greedy algorithm that can convert any fraction into the sum of unit fractions with an equivalent value. Write the pseudocode for an algorithm that can convert any positive fraction into its equivalent sum of unit fractions.

Number	Equivalent sum of unit fractions
$\frac{2}{3}$	$\frac{1}{2} + \frac{1}{6}$
$\frac{6}{14}$	$\frac{1}{3} + \frac{1}{11} + \frac{1}{231}$
$\frac{12}{13}$	$\frac{1}{2} + \frac{1}{3} + \frac{1}{12} + \frac{1}{156}$

Table 4: Example fractions and their equivalent sum of unit fractions