Homework 4

CS 4104 (Summer 2024)

Assigned on July 22, 2024.
Submit a PDF file containing your solutions on Canvas by 11:59pm on Monday, July 29, 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 (30 Points) In the kingdom of Algoria, there is a sequence of magical stones. Each stone has an enchanting property that allows it to glow if placed in a strictly increasing order based on their inherent magical power. The task of the royal mage is to determine the longest glowing sequence of stones possible. Write an algorithm to help the mage find this sequence.

Examples:
Input 1: {10, 22, 9, 33, 21, 50, 41, 60, 80}
Output 1: 6 (The longest glowing sequence is {10, 22, 33, 50, 60, 80})

Explanation 1: The sequence {10, 22, 33, 50, 60, 80} is strictly increasing and has the maximum length of 6 in the given input.

Input 2: {3, 10, 2, 1, 20}
Output 2: 3 (The longest glowing sequence is {3, 10, 20})

Explanation 2: The sequence {3, 10, 20} is strictly increasing and has the maximum length of 3 in the given input.

Problem 2 (30 Points) In a mystical forest, there is a path guarded by various magical creatures. Each creature requires a specific amount of magical energy to be passed. The kingdom's bravest knight must find the path that requires the minimum energy to reach the end of the forest. The knight can move either right or down at each step. Write a dynamic programming algorithm to assist the knight in his journey.

Examples:

Input 1:

$$\begin{bmatrix} 1 & 3 & 1 \\ 1 & 5 & 1 \\ 4 & 2 & 1 \end{bmatrix}$$

Output 1: 7 (The path with minimum energy is $1 \to 3 \to 1 \to 1 \to 1$)

Explanation 1: The knight starts at (0,0) and follows the path (0,0) $\to$ (0,1) $\to$ (0,2) $\to$ (1,2) $\to$ (2,2), which requires a total energy of 7.

Input 2:

$$\begin{bmatrix} 1 & 2 & 5 \\ 3 & 2 & 1 \\ 4 & 2 & 1 \end{bmatrix}$$

Output 2: 7 (The path with minimum energy is $1 \to 2 \to 2 \to 1 \to 1$)

Explanation 2: The knight starts at (0,0) and follows the path (0,0) $\to$ (0,1) $\to$ (1,1) $\to$ (2,1) $\to$ (2,2), which requires a total energy of 7.

Problem 3 (30 Points) A dragon guards a treasure in a distant kingdom. To reach the treasure, a knight must solve a puzzle where he needs to determine the number of ways to climb to the top of a tower with `n` steps. The knight can climb either 1 step or 2 steps at a time. Write a dynamic programming algorithm to help the knight.

Examples:
Input 1:
Number of steps: 5
Output 1: 8 (There are 8 ways to reach the top of the tower with 5 steps)

Explanation 1: The 8 ways are: (1,1,1,1,1), (1,1,1,2), (1,1,2,1), (1,2,1,1), (2,1,1,1), (2,2,1), (2,1,2), (1,2,2).

Input 2:
Number of steps: 6
Output 2: 13 (There are 13 ways to reach the top of the tower with 6 steps)

Explanation 2: The 13 ways are: (1,1,1,1,1,1), (1,1,1,1,2), (1,1,1,2,1), (1,1,2,1,1), (1,2,1,1,1), (2,1,1,1,1), (2,2,1,1), (2,1,2,1), (1,2,2,1), (1,1,2,2), (2,1,1,2), (1,2,1,2), (2,2,2).

Problem 4 (30 Points) In the realm of Computronia, a powerful spell is protected by an ancient code. To break the code, a sorcerer needs to find the maximum sum of non-adjacent numbers in a sequence of integers. The sorcerer can pick numbers in such a way that no two picked numbers are adjacent. Write a dynamic programming algorithm to help the sorcerer break the code.

Examples:
Input 1:
Sequence: {3, 2, 5, 10, 7}
Output 1: 15 (The maximum sum is obtained by picking 3, 10, and 2)

Explanation 1: The maximum sum is achieved by picking the numbers 3, 10, and 2, which sum up to 15.

Input 2:
Sequence: {3, 2, 7, 10}
Output 2: 13 (The maximum sum is obtained by picking 3 and 10)

Explanation 2: The maximum sum is achieved by picking the numbers 3 and 10, which sum up to 13.