CPSC 365 / ECON 365: Algorithms — Yale University

Problem Set 4

Out: February 22, 2022 — Due: March 8, 2022 at 2:30 pm EST

Instructions

Write your solution to the following problems carefully. Submit the PDF of your solution via Gradescope. Please start your solution to every question on a new page. Make sure to assign the correct page in your document corresponding to each problem. We recommend writing your solution in Latex. Handwritten solutions are accepted if they are clearly legible. Write your name and SID in your answer to Problem 0; do not write them anywhere else in your solution.

Collaboration policy: You may collaborate and discuss these problems with other students. However, you must first make an honest effort to solve these problems by yourself. You may discuss hints and try to solve problems together. But you may not explicitly provide the solution to a problem to other students; nor receive the solution from them. You must write your solution independently, and make sure you understand your solution. You must list all your collaborators (anyone with whom you discussed any part of these problems). You may consult the course textbook and other references related to the class. However, you are forbidden from searching for these problems on the Internet. You must list any resources you consult (beyond the course textbook). You must also follow the Yale academic integrity policy.

0 Your Information

On the first page of your submission, include the following information (but do not include these anywhere else in your solution). Also certify that you have followed the collaboration policy. Your solution to the remainder of the problems should start on a new page.

(a) Your name.

(b) Your SID.

(c) A list your collaborators and any outside resources you consulted for this problem set. If none, write "None".

(d) Certify that you have followed the academic integrity and collaboration policy as written above.

(e) How many hours did you spend in this problem set?

1 Finding Element

You are given an array of integers A of length n, with the property that $|A[i] - A[i+1]| \leq 1$ for all $1 \leq i < n$. That is, adjacent elements of A differ by at most 1 (but A does not need to be sorted). Let $A[1] = p$ and $A[n] = q$, and assume that $p \leq q$. Design an algorithm to find an index j such that $A[j] = t$ for a given value of t in the range $p \leq t \leq q$. Your algorithm should run in $O(\log n)$ time.

Example:
Input: $A = (1, 2, 3, 2, 3, 4, 5)$, $\quad t = 3$
Output: $j = 3$ or $j = 5$ (you only need to output one answer)

Problems:

(a) Describe your algorithm in words. (You don't have to give a formal proof, but explain why your algorithm computes the correct answer. Make sure to note where you use the assumptions on the array.)

(b) Write a pseudocode for your algorithm to solve this problem.

(c) Let $T(n)$ be the running time of your algorithm on input of length n. State the recurrence that $T(n)$ satisfies.

(d) Solve the recurrence. What is the running time of your algorithm?

2 Maximum Sum

You are given an array of integers A of length n. A *subsequence* of A is a contiguous subarray $A[i : i + k - 1]$ for some $k \geq 1$, and the *sum* of that subsequence is $\sum_{j=0}^{k-1} A[i + j]$. Design a divide-and-conquer algorithm for finding the value of the maximum sum of any subsequence of A. Your algorithm should run in $O(n \log n)$ time.

Example:
Input: $A = (2, 1, -1, 3, -1, -2, 2)$
Output: 5

Problems:

(a) Describe your algorithm in words. (You don't have to give a formal proof, but explain why your algorithm gives the correct answer.)

(b) Write a pseudocode for your algorithm to solve this problem.

(c) Let $T(n)$ be the running time of your algorithm on input of length n. State the recurrence that $T(n)$ satisfies.

(d) Solve the recurrence. What is the running time of your algorithm?

3 Longest Common Substring

You are given two string arrays x and y of lengths n and m respectively. Design a dynamic programming algorithm to find the length of the longest common substring of x and y. That is, find the largest value k for which there exist indices i and j such that $x[i : i+k-1] = y[j : j+k-1]$. (Note that the substring must be contiguous.) Your algorithm should run in $O(nm)$ time.

Example:
Input: $x =$ "tacocat", $y =$ "mycatlikestacos"
Output: $k = 4$ (the longest common substring is "taco")

Problems:

(a) In dynamic programming, you are going to keep track of a table $T(i)$ or $T(i, j)$ of subproblems. Define what the entries of your table are, in words. (E.g. $T(i)$ or $T(i, j)$ is ...)

What is the final answer of your algorithm in terms of the table entries?

(b) State the recurrence for entries of the table in terms of smaller subproblems, and state your base case(s) as well. (You don't have to give a formal proof, but explain why the recurrence is correct.)

(c) Write a pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.

4 Infinitely Many Rods

You have a collection of infinitely many metal rods of integer lengths $\ell_1, \dots, \ell_n$. You wish to create a rod of integer length S by welding some of the rods together. Design a dynamic programming algorithm that returns `True` if it is possible to create a rod of length exactly S using rods of lengths $\ell_1, \dots, \ell_n$; and returns `False` otherwise. Note that you can use the rod of length ℓ_i as many times as you like. Your algorithm should run in $O(nS)$ time.

Examples:

- Input: $\ell_1 = 3$, $\ell_2 = 5$, $\ell_3 = 4$, $S = 13$
  Output: `True`
  Explanation: $13 = 1 \times 3 + 2 \times 5 + 0 \times 4$, so we can use 1 rod of length ℓ_1, two rods of length ℓ_2, zero rods of length ℓ_3.

- Input: $\ell_1 = 5$, $\ell_2 = 6$, $S = 13$
  Output: `False`
  Explanation: We cannot make a rod of length 13 with rods of lengths 5 and 6.

Problems:

(a) Define what the entries of your table are, in words. (E.g. $T(i)$ or $T(i, j)$ is ...)

What is the final answer of your algorithm in terms of the table entries?

(b) State the recurrence for entries of the table in terms of smaller subproblems, and state the base case(s). (You don't have to give a formal proof, but explain why the recurrence is correct.)

(c) Write a pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.

5 At Most One Rod

Due to a heatwave, your collection of infinitely many metal rods has melted. Your collection is now left with exactly one rod each of integer lengths $\ell_1, \ldots, \ell_n$. You wish to create a rod of integer length S by welding some of the rods from your collection together. Design a dynamic programming algorithm that returns `True` if it is possible to create a rod of length exactly S using rods of lengths $\ell_1, \ldots, \ell_n$; and returns `False` otherwise. Note that you can use the rod of length ℓ_i at most once. Your algorithm should run in $O(nS)$ time.

Examples:

- Input: $\ell_1 = 2$, $\ell_2 = 3$, $\ell_3 = 5$, $S = 6$
  Output: `False`
  Explanation: There is no way to make a rod of length 6 using only rods of lengths $2, 3, 5$ each at most once.
- Input: $\ell_1 = 2$, $\ell_2 = 3$, $\ell_3 = 5$, $S = 8$
  Output: `True`
  Explanation: $3 + 5 = 8$

Problems:

(a) Define what the entries of your table are, in words. (E.g. $T(i)$ or $T(i, j)$ is ...)

What is the final answer of your algorithm in terms of the table entries?

(b) State the recurrence for entries of the table in terms of smaller subproblems, and state the base case(s). (You don't have to give a formal proof, but explain why the recurrence is correct.)

(c) Write a pseudocode for your algorithm to solve this problem.

(d) Analyze the running time of your algorithm.