University of Houston

Homework 2

COSC 3320
Algorithms and Data Structures

Gopal Pandurangan

NAME

Due: Sunday, March 14, 2021
11:59 PM

Read the University of Houston Academic Honesty policy.

Academic Honesty Policy

All submitted work should be your own. Copying or using other people's work (including from the Web) will result in $-$`MAX` points, where `MAX` is the maximum possible number of points for that assignment. Repeat offenses will result in a failing grade for the course and will be reported to the Chair. If you have any questions, please reach out to the professor and the TAs. The best way to ask is on Piazza.

By submitting this assignment, you affirm that you have followed the Academic Honesty Policy.

Your submission **must be typed**. We prefer you use LaTeX to type your solutions — LaTeX is the standard way to type works in mathematical sciences, like computer science, and is highly recommended; for more information on using LaTeX, please see this post on Piazza — but any method of typing your solutions (e.g., MS Word, Google Docs, Markdown) is acceptable. **Your submission must be in pdf format.** The assignment can be submitted **up to two days late for a penalty of 10% per day.** A submission more than **two days late** will receive a **zero**.

Reading

Chapters 5 and 6. In particular, several worked exercises with solutions are provided at the end of each chapter. Attempting to solve the worked exercises **before** seeing their solutions is a good learning technique.

The exercises below are from the book. The book is updated periodically, so be sure to use the latest version.

Exercises

5.18, 5.23, 6.11, 6.20

Justify your answers. Show appropriate work.

This page intentionally left blank.

1 Class Questions

1.1 February 09

Question 1

Show that the depth of the tree presented in class is $\mathcal{O}\left(\log_{3/2} n\right)$.

Solution. TYPE SOLUTION HERE. □

Question 2

Prove the correctness of the `Select` algorithm by mathematical induction.

Solution. TYPE SOLUTION HERE. □

1.2 February 23

Question 1

Show that the algorithm `mult` given in class takes $\mathcal{O}(n^2)$ operations. You can assume that we are multiplying two n-bit numbers (i.e., binary numbers each of length n bits). Adding or multiplying two bits is counted as one operation.

Solution. TYPE SOLUTION HERE. □

2 Textbook Exercises

Exercise 5.18

Consider the *max-sum* problem: Given an array A of size n containing positive and negative integers, determine indices i and j, $1 \leq i \leq j \leq n$, such that $A[i] + A[i+1] + ... + A[j]$ is a maximum.

(a) Consider the array $A = [5, 10, -15, 20, -4, 6, 4, 8, -10, 20]$. Find the indices i and j that give the maximum sum and state the maximum sum.

(b) Give a *divide and conquer* algorithm that runs in $\mathcal{O}(n \log n)$ time. Assume, that adding or comparing two numbers takes constant time.

(Hint: Split the array into two (almost) equal parts and recursively solve the problem on the two parts. The non-trivial part is combining the solutions — note that it is possible that the indices i and j that give the optimal sum might be on opposite parts.)

Solution. TYPE SOLUTION HERE. □

Exercise 5.23

You are given an unsorted array of $n \geq 1$ integers. Give an efficient divide and conquer algorithm to output an element in the array that occurs more than 3 times, if such an element exists. Show the correctness of your algorithm and analyze its run time.

Solution. TYPE SOLUTION HERE. □

Exercise 6.11

Consider the following *variant* of the max-sum problem.
Given an array A of size n containing positive and negative integers, the goal is to determine indices i and j, $1 \leq i \leq j \leq n$, such that $A[i] + A[i+1] + ... + A[j] - A[k]$ is a **maximum**, where k can be any index between i and j, i.e., $i \leq k \leq j$. In other words, we are allowed to exclude **up to** one element (any one) in the contiguous subarray $A[i], \ldots, A[j]$, including $A[i]$ or $A[j]$. Note that a solution subarray can be contiguous as well — this is captured b allowing k to be $A[i]$ or $A[j]$, which means that the subarray is contiguous.
Give a dynamic programming (DP) algorithm that computes the optimum value of the modified max-sum problem, i.e., we are interested in just computing the value of the optimum solution.

1. Specify the subproblems, clearly explaining your notation.
2. Give a recursive formulation that relates how you can solve larger-sized subproblems from smaller-sized subproblems. Also mention the base cases.
3. Implement your formulation by a DP algorithm and a memoized recursive algorithm.
4. Analyze the runtime of your algorithms.

Solution. TYPE SOLUTION HERE. □

Exercise 6.20

You are given a stick of length n units. Your goal is to cut the stick into different pieces (each piece should be of integer length only) so that the total value of all the pieces is *maximized*. A piece of length i units has value v_i, for $i = 1, 2, \ldots, n$. You should design a dynamic programming algorithm that determines the maximum total value that is possible.

(i) Consider the sequence of 7 numbers: $2, 3, 4, 5, 6, 7, 9$. Let the i^{th} number in the sequence represents the value of piece of length i, i.e., the value of piece of length 1 is 2, of length 2 is 3, length 3 is 4 and so on. What is the best way to cut the pieces for a stick of length 7? What is the maximum value ?

(ii) Define the subproblems for your DP solution on a general instance of the problem where the stick is of length n and a piece of length i has value v_i.

(iii) Give a recursive formulation to solve the subproblems. Don't forget the base cases.

(iv) What is the running time of your solution?

(v) Write a DP algorithm (give pseudocode) that outputs the maximum value.

(vi) Describe an algorithm to output the sizes of the pieces of the cut that corresponds to the maximum value.

Solution. TYPE SOLUTION HERE. □