CS 577: Introduction to Algorithms — Homework 5

Out: 03/09/21 — **Due: 03/16/21**

Name: Aiden Tepper Wisc ID: 9081242969

Ground Rules

- Answer the questions in the boxes provided on the question sheets. If you run out of room for an answer, add a page to the end of the document. Do **not** feel obligated to fill the entire solution box. The size of the box does **not** correspond to the intended solution length.
- The homework is to be done and submitted individually. You may discuss the homework with others in either section but you must write up the solution *on your own*.
- You are not allowed to consult any material outside of assigned textbooks and material the instructors post on the course websites. In particular, consulting the internet will be considered plagiarism and penalized appropriately.
- The homework is due at 11:59 PM CST on the due date. No extensions to the due date will be given under any circumstances.
- Homework must be submitted electronically on Gradescope.

Problem 1:

1. Recall that in Problem 1 of the midterm, you were given a fully parenthesized numerical expression with n numbers, and your goal was to find the maximum and minimum possible values of the expression by filling in $+$ or $-$ for each of the $n-1$ operators. Denote the sequence of these n numbers as $a_1, \ldots, a_n$, and the sequence of the $n-1$ operators as $o_{1,2}, o_{2,3}, \ldots, o_{i,i+1}, \ldots, o_{n-1,n}$, where $o_{i,i+1}$ is the operator placed between a_i and a_{i+1}.

   Now consider the problem where you are given the number sequence $a_1, \ldots, a_n$, but the choices of these $n-1$ operators and positions of parentheses are both undetermined. Moreover, you can set each of the $n-1$ operators to be either $+$, $-$, or $\times$. Your goal is to find the **maximum** possible value the numerical expression

   $$[a_1]o_{1,2}[a_2]o_{2,3}\ldots[a_{n-1}]o_{n-1,n}[a_n]$$

   can evaluate to when you fill in $+$, $-$, or $\times$ for these $n-1$ operators and insert pairs of parentheses. Note that you should not change the ordering of numbers in $a_1 ... a_n$ but you may give precedence to some operations with appropriate parenthesizing. For example, the following number sequence of

   $$[-3]\ ?\ [-4]\ ?\ [-5]$$

   can achieve a maximum value of 35 by filling in $o_{1,2}$ to $o_{2,3}$ and adding parentheses in the following way:

   $$\big([-3] + [-4]\big) \times [-5]$$

(a) Give a dynamic programming algorithm to solve this problem. That is, describe your algorithm by including a clear statement of your recurrence, any necessary base case(s), and your final output.

Given numbers $a_1 \dots a_n$, we will choose operations $O_{1,2} \dots O_{n-1,n}$ such that $a_1 O_{1,2} \cdots O_{n-1,n} a_n$ is maximized

Base case: for i from 1 to n: $\max(i,i) = \min(i,i) = a_i$

Recurrence relation:

$$\text{maxValue}[i,j] = \max_{i \le k < j} \begin{cases} \text{maxValue}[i,k] + \text{maxValue}[k+1,j] \\ \text{maxValue}[i,k] - \text{minValue}[k+1,j] \\ \text{maxValue}[i,k] \times \text{maxValue}[k+1,j] \\ \text{minValue}[i,k] \times \text{minValue}[k+1,j] \\ \text{maxValue}[i,k] \times \text{minValue}[k+1,j] \\ \text{minValue}[i,k] \times \text{maxValue}[k+1,j] \end{cases}$$

$$\text{minValue}[i,j] = \min_{i \le k < j} \begin{cases} \text{minValue}[i,k] + \text{minValue}[k+1,j] \\ \text{minValue}[i,k] - \text{maxValue}[k+1,j] \\ \text{maxValue}[i,k] \times \text{maxValue}[k+1,j] \\ \text{minValue}[i,k] \times \text{minValue}[k+1,j] \\ \text{maxValue}[i,k] \times \text{minValue}[k+1,j] \\ \text{minValue}[i,k] \times \text{maxValue}[k+1,j] \end{cases}$$

Algorithm would compute base case, then $\max, \min(1,2) \dots \max, \min(n-1,n)$ until eventually reaching $\max, \min(1,n)$ and returning $\max(1,n)$.

(b) Prove the correctness of your algorithm and analyze its runtime.

Runtime: $O(n^2)$ values are computed, with each value taking time $O(n)$. Therefore, total runtime is $O(n^3)$.

Proof of correctness through induction:

Base case: if $j-i=0$ $(i=j)$, then $\max(i,j)$ and $\min(i,j)$ are the only possible value, which is both the max and min. Base case holds.

IH: Assume $j-i=m+1$ and the algorithm holds for all $j-i \leq m$. We will prove that for $j-i=m+1$, the algorithm holds.

Since parentheses must be added somewhere in the formulas for max and min, the string of numbers $a_i \ldots a_j$ must be broken up at some point $a_{k,k+1}$. The operations and the min and max choices in the algorithm consider every possible combination that could lead to a max or min for any position k of this break. We know (by induction) that these max/min are correct. Since the algorithm chooses the max/min over all k, it must choose the same k as a valid solution for max/min.

$\therefore$ $\max(i,j)$ and $\min(i,j)$ are correct and the algorithm is correct. ■

Problem 2:

2. You have a collection of boxes and want to pack them in your attic. Every box i has a width w_i, a height h_i and contains items of total value v_i. Your attic has the shape of a right angled isosceles triangle with two sides of length H. You want to stack the boxes and place them in the attic. You may choose the order in which to stack the boxes but you are not allowed to rotate them and you cannot place two boxes side by side. However, not all boxes can fit in the attic. In order for a box to fit, its width must be at most H minus its height and the heights of all the boxes below it in the stack. Figure 1, illustrates how boxes are packed to fit in the attic. Your goal is to maximize the value of the boxes placed in the attic.

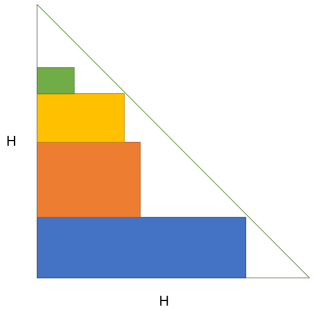

Figure 1: How boxes are stacked and placed in the attic.

Input: A set of n boxes with box i having width w_i, height h_i and value v_i, the height of the attic H, where all heights h_is and H are integers.

Your goal is to maximize the value of the boxes placed in the attic.

(a) Assuming all boxes have the same height $h = 1$, give an efficient algorithm to find the optimal arrangement of boxes in the attic. Prove the correctness of your algorithm and analyze its runtime.

input: set of box objects, attic height H. output: max value of attic boxes

Algorithm BoxStack(S,H):
1. if $w_1 > H-1$:
2. L1:=0
3. else:
4. L1:=value$_1$+BoxStack(S/{1},H-1)
5. L2:=BoxStack(S/{1},H)
6. return max(L1,L2)

Runtime analysis: O(NH) where N is the number of boxes and H is the height of the attic.

Proof of correctness:

Base case: N=1. If the one box fits in the attic (tested on line 1), then

the algorithm will correctly return the value of that one box. Otherwise, it will return 0.

IH: assume the algorithm holds for $n=k$. We will prove that it holds for $n=k+1$.

For each new box added, two options are considered and tested: one with the new box added, and one without (if the new box doesn't fit, it won't be added and the algorithm will remain correct at $n=k$). The algorithm will consider both options and correctly return the maximum sum value of the two, ensuring that the box placement is maximized. Thus, the algorithm holds for $n=k+1$.

∴ The algorithm exhibits program correctness. ■

(b) For the general case where all boxes have arbitrary integer heights, give an $O(n \log n + nH)$ algorithm to find the optimal arrangement of boxes in the attic.

input: set of box objects, attic height H. output: max value of attic boxes

```
Algorithm BoxStack(S,H):
  if w_1 > H - H_1
    L1 := 0
  else:
    L1 := value_1 + BoxStack(S/{1}, H-1)
  L2 := BoxStack(S/{1}, H)
  return max(L1, L2)
```

(c) Prove the correctness of your algorithm from part (b) and analyze its runtime.

Runtime analysis: $O(n \log_4 nt)$