CS 577 # Assignment 8 – Dynamic Programming 2 Spring 2023

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.

Name: Zinnia Nie Wisc id: 908 319 4044

## Dynamic Programming

Do **NOT** write pseudocode when describing your dynamic programs. Rather give the Bellman Equation, describe the matrix, its axis and how to derive the desired solution from it.

1. *Kleinberg, Jon. Algorithm Design (p. 327, q. 16).*

   In a hierarchical organization, each person (except the ranking officer) reports to a unique superior officer. The reporting hierarchy can be described by a tree $T$, rooted at the ranking officer, in which each other node $v$ has a parent node $u$ equal to his or her superior officer. Conversely, we will call $v$ a direct subordinate of $u$.

   Consider the following method of spreading news through the organization.

   - The ranking officer first calls each of her direct subordinates, one at a time.
   - As soon as each subordinate gets the phone call, he or she must notify each of his or her direct subordinates, one at a time.
   - The process continues this way until everyone has been notified.

   Note that each person in this process can only call *direct* subordinates on the phone.

   We can picture this process as being divided into rounds. In one round, each person who has already heard the news can call one of his or her direct subordinates on the phone. The number of rounds it takes for everyone to be notified depends on the sequence in which each person calls their direct subordinates.

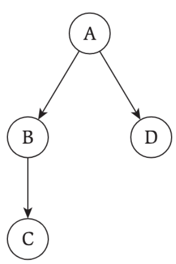


Figure 1: A hierarchy with four people. The fastest broadcast scheme is for A to call B in the first round. In the second round, A calls D and B calls C. If A were to call D first, then C could not learn the news until the third round.

The questions are on the next page.

Give an efficient algorithm that determines the minimum number of rounds needed for everyone to be notified, and outputs a sequence of phone calls that achieves this minimum number of rounds by answering the following:

(a) Give a recursive algorithm. (The algorithm does not need to be efficient)

**Solution:** NumCalls

If is leaf: return 0
Sort children by number of children
max = NumCalls (child)
for all children:
    max = max(max, Num calls (child))
end
return max

(b) Give an efficient dynamic programming algorithm.

**Solution:**

Use a tree that looks the same as the given tree to store the values. (Tree S)

Bellman: $S_p = \min_i [i + S(i)]$ where $i$ is the index of the child in sorted order by $S(i)$

Populate: all leafs are 0, and start from the leaves and move up to the root.

Trace back through subtrees to get the call sequence.

(c) Prove that the algorithm in part (b) is correct.

**Solution:** Base Case: At a leaf, there are no more subordinates to inform, so Ø more rounds are needed.

Inductive Step: all of the children of a node have the correct rounds needed.

For each subtree, if it informs the max number of children child first, then the rest of the children can be informed while the largest child subtree is already being traversed.

Runtime: find min $O(n)$, traceback is $O(n \log n)$

2. Consider the following problem: you are provided with a two dimensional matrix $M$ (dimensions, say, $m \times n$). Each entry of the matrix is either a **1** or a **0**. You are tasked with finding the total number of square sub-matrices of $M$ with all **1**s. Give an $O(mn)$ algorithm to arrive at this total count by answering the following:

(a) Give a recursive algorithm. (The algorithm does not need to be efficient)

**Solution:**
At each index of the matrix, find the maximum size square
and add all squares together

FindSq: If (past bounds of M || M[row, col] == 0) return 0
return min(FindSq(row+1, col, matrix), FindSq(row, col+1, matrix),
FindSq(row+1, col+1, matrix)) + 1

(b) Give an efficient dynamic programming algorithm.

**Solution:**
Matrix: S[i,j], m×n matrix
↳ number squares with M[i,j] as bottom right
Bellman: S[i,j] = min(S[i-1,j], S[i,j-1], S[i-1,j-1]) + 1
Populate: For first row and column, populate with the
values in A[0,j] and A[i,0]
Solution: sum all values in the matrix

(c) Prove that the algorithm in part (b) is correct.

**Solution:** Base case: a 1×1 sq that has 1 is one square of all ones. In the matrix M,
it is in the bottom right of the first row/col, so it counts as one square.
Inductive step: IH: The matrix is filled correctly to M[i, j].
Each index essentially stores how many squares can be formed
with the element M[i,j]. If there is a Ø then it would
block square formation for all area around it. If we look
at the squares in the top left, top right, and bottom left
of the matrix, the minimum possible squares, plus the index
at the very bottom would be the total squares including the bottom right index

(d) Furthermore, how would you count the total number of square sub-matrices of $M$ with all **0**s?

**Solution:** Invert the matrix by switching 1 and 0
values.
Or count Øs instead of 1s. In initialization,
put a 1 when there is a 0 in the first row
and column. Then follow the same procedure

3. *Kleinberg, Jon. Algorithm Design (p. 329, q. 19).*

String $x'$ is a *repetition* of $x$ if it is a prefix of $x^k$ ($k$ copies of $x$ concatenated together) for some integer $k$. So $x' = 10110110110$ is a repetition of $x = 101$. We say that a string $s$ is an *interleaving* of $x$ and $y$ if its symbols can be partitioned into two (not necessarily contiguous) subsequences $x'$ and $y'$, so that $x'$ is a repetition of $x$ and $y'$ is a repetition of $y$. For example, if $x = 101$ and $y = 00$, then $s = 100010010$ is an interleaving of $x$ and $y$, since characters $1, 2, 5, 8, 9$ form 10110a repetition of $x$and the remaining characters $3, 4, 6, 7$ form 0000a repetition of $y$.

Give an efficient algorithm that takes strings $s$, $x$, and $y$ and decides if $s$ is an interleaving of $x$ and $y$ by answering the following:

(a) Give a recursive algorithm. (The algorithm does not need to be efficient)

**Solution:** Input : x, y, s

Interleaving :

repeat x and y till they are the length of s

If s is empty: return True

If s[0] == x[0] : reurn Interleaving(x[1,n], y, s[1,0]

If s[0] = y[0] : return Interleaving(x, y[1,m], s[1,0]

return False

(b) Give an efficient dynamic programming algorithm.

**Solution:** Matrix : M[i,j], nxn where n = len(s)

↳ if s[1, i+j] is an interleaving of x[1: i] and y[1:j]

Bellman: $M[i,j] = \begin{cases} \text{True} & \text{if } M[i-1,j] = \text{yes and } s[i+j] = x'[i] \\ & M[i,j-1] = \text{yes and } s[i+j] = y'[j] \\ \text{False} & \text{otherwise} \end{cases}$

Populate: from begining of x', y', and s to the end

Solution: Yes if there is i+j = n where M[i,j] = yes

(c) Prove that the algorithm in part (b) is correct.

**Solution:** Base Case: if the first char of s is part of x' or y', then it is an interleaving.

Inductive step: If the char s[i+j] is equal to either x'[i] or y'[j], then it is still part of the correct position for interleaving. Then if the previous location of x' or y' is also a True, then the previous string is interleaved to the correct position. However if there is ever an index that is not correctly interleaved with x' or y', then the entire string s is not an interleaving

4. *Kleinberg, Jon. Algorithm Design (p. 330, q. 22).*

   To assess how well-connected two nodes in a directed graph are, one can not only look at the length of the shortest path between them, but can also count the number of shortest paths.

   This turns out to be a problem that can be solved efficiently, subject to some restrictions on the edge costs. Suppose we are given a directed graph $G = (V, E)$, with costs on the edges; the costs may be positive or negative, but every cycle in the graph has strictly positive cost. We are also given two nodes $v, w \in V$.

   Give an efficient algorithm that computes the number of shortest $v - w$ paths in $G$. (The algorithm should not list all the paths; just the number suffices.)

**Solution:**

Matrix: $M[i,v]$, shortest path to $v$ using $i$ edges

Bellman: $M[i,v] = \min_{(w,v) \in E} \{ M[i-1, w] + c_{vw} \}$

This calculates the optimal shortest path cost from v to w

Shortest path to w is $\min_i \{ M[i,w] \}$.

The total number of paths with optimum cost is $N(i,v) = \sum N(i-1, w)$.

Then add up all the paths that achieve the min cost using $\sum N(i,w)$.

Populate: $M[i][v] = 0$

if edge doesn't exist from v to w,

$M[i][v] = \infty$

$N(i,v) = 1$ and $N(i,v) = 0$ if edge doesn't exist.

5. The following is an instance of the Knapsack Problem. Before implementing the algorithm, run through the algorithm by hand on this instance. To answer this question, generate the table, indicate the maximum value, and recreate the subset of items.

| item | weight | value |
|---|---|---|
| 1 | 4 | 5 |
| 2 | 3 | 3 |
| 3 | 1 | 12 |
| 4 | 2 | 4 |

Capacity: 6

**Solution:**

weight

first i elements

| | 1 | | | | | 6 | |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 0 | 5 | 5 | |
| | 0 | 0 | 3 | 5 | 5 | 5 | |
| | 12 | 12 | 12 | 15 | 15 | 15 | |
| 4 | 12 | 12 | 16 | 16 | 16 | 19 | ← solution |

Max value = 19

Take items 2, 3, 4 for a total weight of 6 and total value of 19.

6. Implement the algorithm for the Knapsack Problem in either C, C++, C#, Java, or Python. Be efficient and implement it in $O(nW)$ time, where $n$ is the number of items and $W$ is the capacity.

The input will start with an positive integer, giving the number of instances that follow. For each instance, there will two positive integers, representing the number of items and the capacity, followed by a list describing the items. For each item, there will be two nonnegative integers, representing the weight and value, respectively.

A sample input is the following:

```
2
1 3
4 100
3 4
1 2
3 3
2 4
```

The sample input has two instances. The first instance has one item and a capacity of 3. The item has weight 4 and value 100. The second instance has three items and a capacity of 4.

For each instance, your program should output the maximum possible value. The correct output to the sample input would be:

```
0
6
```