CS 577 (Epic) Assignment 8 – Dynamic Programming 2 Spring 2021

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: Zhuoyan Xu Wisc id: 3XU444

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. 329, q. 19).*

   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 s' and s'', so that s' is a repetition of x and s'' is a 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.

   Note: We write x^k to denote k copies of x concatenated together. We say that a string x' is a repetition of x if it is a prefix of x^k for some number k. So $x' = 10110110110$ is a repetition of $x = 101$.

Consider $n = \text{length}(s)$ let x' denote the repetition of x with length n
y' is repetition of y with length n

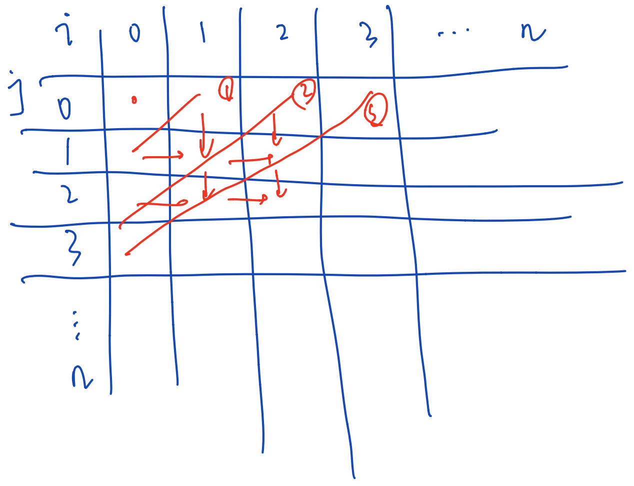
e.g. $x' = \underbrace{\ }_{x}\underbrace{\ }_{x}\underbrace{\ }_{x}\underbrace{\ }_{x}\cdots\underbrace{\ }_{x}\underbrace{\ }$ substring of first x length$(x) = n$

then the question is equivalent to see whether s is interleaving of x', y'

we have 2-D matrix $m(i,j)$ entries are 0 or 1 $i = 1\dots n$, $j = 1\dots n$

let $m(i,j) = 1$ iff s is composed by $x'[1\dots i]$ and $y'[1\dots j]$ (not necessarily contiguous)

then $m(i,j) = 1 \iff \begin{cases} ① \ m(i-1,j) = 1 & s(i+j) = x'(i) \\ ② \ m(i,j-1) = 1 & s(i+j) = y'(j) \end{cases}$ either ① or ② will make $m(i,j)=1$ work



```
for k = 1...n
    for i+j = k.
        check m(i,j) = 1 by ① or ②
    end
end
```

if exist $i+j = k$ s.t. $m(i,j) = 1$. then s is interleaving of x', y'

2. *Kleinberg, Jon. Algorithm Design (p. 328, q. 18).*

   Consider the problem of editing strings over a four-letter alphabet $\{z_1, z_2, z_3, z_4\}$. The edits one can make are (1) you can insert a character, (2) delete a character, and (3) replace a character with another.

   You are given a fixed cost G per insertion, and the same cost G for deletion. You are also given a fixed cost R for replacing a mismatched character. Assume that each of these parameters is a positive integer. The edit distance between the two strings is G times the number of insertions plus G times the number of deletions plus R times the number of replacements.

   Suppose you are given two strings $A = a_1a_2...a_m$ and $B = b_1b_2...b_n$ and a proposed edit to transform A into B. Give an $O(mn)$ algorithm to decide whether this edit is the *unique* minimum-cost edit between A and B.

Consider edit distance for $A[1\cdots i]$, $B(1\cdots j)$

insertion: $E(i,j) = E(i,j-1) + G$

deletion: $E(i,j) = E(i+1,j) + G$

replace: $E(i,j) = E(i-1,j-1) + R \cdot \mathbb{1}(A_i \neq B_j)$

$\begin{cases} E(i,0) = iG \\ E(0,j) = jG \end{cases}$

Bellman equation:

$$E(i,j) = \begin{cases} i, & \text{if } j=0 & ① \\ j. & \text{if } i=0 & ② \\ \min\{E(i,j-1)+G,\ E(i+1,j)+G,\ E(i-1,j-1)+R\cdot\mathbb{1}(A_i \neq B_j)\} & & ③ \end{cases}$$

when we trace back from the finial solution $E(m,n)$,

we see whether we choose tie between ③. if we did.

the edit is not unique.

3. *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.

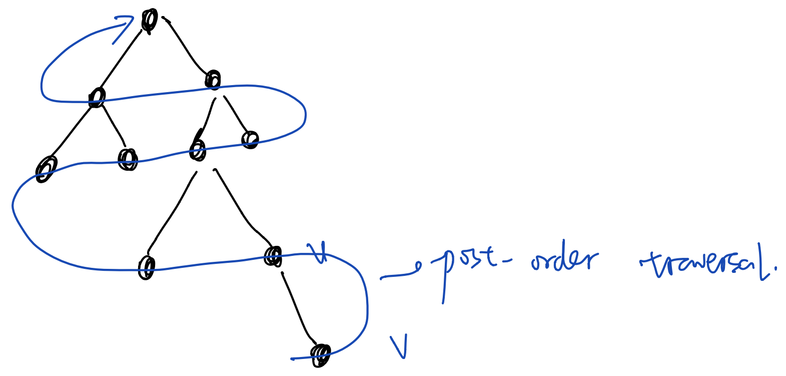
- The ranking officer first calls each of her direct subordinates, one at a time.
- As soon as each subordinate gets the phone call, he or she must notify each of his or her direct subordinates, one at a time.
- The process continues this way until everyone has been notified.

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.

maximum depth of the tree.

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.



Consider $S(i)$ is the set of direct subordinate of i

$d(i)$ is the depth(rank) of node i

if i is leave. $d(i)=0$

Bellman equation: $d(i) = 1 + \max\limits_{v \in S(i)} \{d(v)\}$.

we first implement post-order traversal to our nodes.
then implement our algrithm based on post-order traversal $i=1 \dots |V|$.
and Bellman equation.

complexity: $O(|V|)$

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.)

We first modify our shortest path algorithm. we modify the dichotomy from

- use $\le i-1$ edges
- use $\le i$ edges

to

- use i edges
- use $i-1$ edges

We want to find $v-w$ path. we have 2-D matrix of M: # edges × vertices

- $M(i,t)$ is shortest path from t to w using exact i edges.

  $M(i, w) = 0$

- We have an extra stration $N(i,t)$ is the number of path from t to w

Bellman equation: $M(i,t) = \min_{s \in V} \left(M(i-1, s) + C_{ts} \right)$ ①

$$N(i,t) = \sum_{s \text{ satisfy ①}} N(i-1, s)$$

Once we solve our matrix M, N

we have shortest path $v-w$: $\min_{i \le |E|} M(i, v)$ ②

the number of shortest path:

$$\sum_{i \text{ satisfy ②}} N(i, v)$$

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

$V(2, 2)$

$V(1, x)$

$V(~~1~~, 1) + 3$

$i = 1 \cdots 4 \quad w = 0 \cdots 6 \quad X_{i,w} = \mathbb{1}(w_i \le w)$

$V(i, w) = \max\{V(i-1, w),\ X_{i,w}(V(i-1, w-w_i) + v_i)\}$

i \ w	1	2	3	4	5	6
0	0	0	0	0	0	0
1	0	0	0	5	5	5
2	0	0	3	5	5	5
3	12	12	12	15	17	17
4	12	12	16	16	17	19

maximum value is 19.

trace back, we know

we take 4 (since 19 > 17)

3 (15 > 5)

2 (3 > 0)

the subset is 2, 3, 4

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
n ← 1 3 → W
    4 100
n ← 3 4 → W
    1 2
    3 3
    2 4
```

i \ w	0	1	2	3	4
0	0	0	0	0	0
1	0	2	2	2	2
2	0	2	2	3	5
3	0	2	2	6	4+2

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
```

2-D matrix:

Input n, W

Output $v(n, W)$

i: item $0 \cdots n$

w: max weight $0 \cdots W$

$X_{i,w} = \mathbb{1}(w_i \le w)$

$$v(i,w) = \max\left(v(i-1, w),\ X_{i,w} \cdot \left[v(i-1, w-w_i) + v_i \right] \right)$$

$v(0, W) = 0$ for all w. $v(i, 0) = 0$ for all i

Solution: $V(n, W)$