Algorithms and Data Structures 2

Samy Haffoudhi

September 24, 2022

Dynamic Programming 1

Manhattan Tourists

5.1

We fill the table for the given example:

$W_{i,j}$	1	2	3	4	5
1	0	2	9	16	24
2	3	4	18	26	32
3	12	10	24	35	41
4	13	23	29	39	46
5	18	30	34	47	52

Thus, the maximum weight a shortest path can have is $W[5,5] = 52$.

5.2

$$W[i,j] = \begin{cases} 0 & \text{for (i, j) = (1, 1)} \\ W[i,j-1] + R[i,j-1] & \text{for } i = 1 \\ W[i-1,j] + D[i-1,j] & \text{for } j = 1 \\ \max(W[i-1,j] + D[i-1,j], W[i,j-1] + R[i,j-1]) & \text{otherwise} \end{cases}$$

Indeed,

- $W[1,1] = 0$ since the path is empty when going from s to s
- When $i = 1$ it is only possible to walk right
- When $j = 1$ it is only possible to walk down
- In the last case, there were two available choices for the last move leading to the node $\nu_{i,j}$: walking down from $\nu_{i-1,j}$ or walking right from $\nu_{i,j-1}$. We choose the one which maximizes the sum of the weights along the path

5.3

We write algorithm 1 (page 2) based on dynamic Programming and the recurrence from Question **5.2**. Since we store the results to the subproblems in table W of size nxn we use $O(n^2)$ space. The first two loops make $n-1$ iterations each. Regarding the nested loops, the outer one makes $n-1$ iterations while the inner one makes $n-1$ iterations per iteration of the outer loop. All other operations are done in constant time and thus the total time of the algorithm is $O(n^2)$.

Algorithm 1 Compute the maximum weigh a shortest path can have

Require: n, D, R

Ensure: W[n,n] is the maximum weight a shortest path $s - t$ can have

```
W[1,1] ← 0
for i = 2 to n do
    W[i,1] ← W[i−1,1] + D[i−1,1]
end for
for j = 2 to n do
    W[1,j] ← W[1,j−1] + R[1,j−1]
end for
for i = 2 to n do
    for j = 2 to n do
        W[1,j] ← max(W[i−1,j] + D[i−1,j], W[i,j−1] + R[i,j−1])
    end for
end for
```