# CS 170 HW 7

Due **2021-10-18, at 10:00 pm**

## 1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, write "none".

**DP solution writing guidelines:**

Try to follow the following 4-part template when writing your solutions.

- Define a function $f(\cdot)$ in words, including how many parameters are and what they mean, and tell us what inputs you feed into $f$ to get the answer to your problem.
- Write the "base cases" along with a recurrence relation for $f$.
- Prove that the recurrence correctly solves the problem.
- Analyze the runtime and space complexity of your final DP algorithm. Can the bottom-up approach to DP improve the space complexity?

## 2 Saving the Spaceship

The famous spaceship captain Ha Nsolo has successfuly blown up the enemy's space station. Now, only one thing is left – to escape back to his base. He realizes that this will not be so easy because the surrounding space is swarmed with hostile fleet. Whenever Captain Nsolo encounters enemy spaceships on his way home, they will damage the hull of his ship, the Millenium Eagle.

However, the Millenium Eagle is not just an ordinary spaceship. In addition to being able to ordinarily fly through space, it can also perform jumps through the hyperspace. When jumping through the hyperspace, the ship does not encounter any enemies and it can go freely without receiving any damage. Captain Nsolo wants to rely on this mechanism to make sure that he can get home safely. But there is one tiny problem. The distance that the Millenium Eagle can travel through the hyperspace is a dynamic quantity that changes each time it performs a jump through hyperspace. Moreover, there is only a limited number of hyperspace jumps it can perform. So, Ha cannot go through hyperspace the whole way back. However, he has been a captain at the Millenium Eagle for a long time so he knows in advance how far each subsequent jump through the hyperspace will travel. He wants to plan his jumps ahead so as to get as little damage as possible on his way home.

The space is represented as a grid. Initially, Captain Nsolo starts at the initial position $(0, 0)$. We assume that he receives the damage at this square, so $W_{0,0}$ should be included in

the damage that the Millenium Eagle receives. He wants to get to the square $(X-1, Y-1)$ for some numbers $X, Y \geq 1$, the width and the height of the grid. Each square $(i, j)$ on the grid $(0 \leq i \leq X-1, 0 \leq j \leq Y-1)$ has a positive number $W_{i,j}$ such that $0 \leq W_{i,j} \leq 9$. $W_{i,j}$ represents the damage dealt to the Millenium Eagle by the enemy spaceship on the square $(i, j)$. If $W_{i,j} = 0$, there is no enemy spaceship on the square $(i, j)$, so Ha can travel freely to that square. Captain Nsolo can only take steps to the right (in the positive X direction) or to the top (in the positive Y direction), one square at a time. However, he also knows a sequence $d_1, ..., d_m$ of distances that the Millenium Eagle can jump through hyperspace. Each $d_i$ is such that $1 \leq d_i \leq 9$. One jump through hyperspace is performed only in one direction (positive X or positive Y), so it cannot go diagonally. The jumps have to be used in the same order as given: the first conducted jump should be of length exactly $d_1$, the second jump should be of length exactly $d_2$ and so on. The Millenium Eagle cannot perform a jump of length $d_i$ before it performed the jumps of lengths $d_1, ..., d_{i-1}$. After it did the jumps of lengths $d_1, ..., d_m$, it cannot make any more jumps.

**Input format**:
The first line contains two integers, $X, Y (2 \leq X, Y \leq 100)$, separated by a space. They give the width and the height of the grid respectively.
The following $Y$ lines contain the description of the grid. Each line $j$ $(0 \leq j \leq Y-1)$ contains $X$ integers separated by a space: $W_{0,j}, ..., W_{X-1,j}$. (So the first line describes the bottom row of the grid, the second line describes the next-to-bottom row of the grid, and so on until the last line that describes the top row of the grid).
The next line contains integers $d_1, ..., d_m (1 \leq m \leq 10^5)$, every two adjacent integers are separated by a single space. $1 \leq d_i \leq 9$, $d_i$ is the length of the $i$'th jump of the Millenium Eagle.

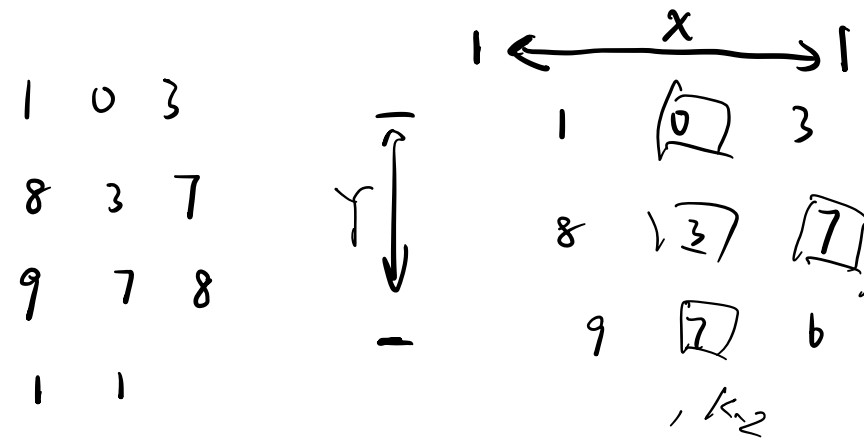
**Output format**:
Output a single integer: the minimal damage that the Millenium Eagle can get as it travels from $(0, 0)$ to $(X-1, Y-1)$

**Note** that $W_{0,0}$ will **always** be included in this sum.

Go to https://hellfire.ocf.berkeley.edu/, access the contest for the current homework and provide a coding solution for the problem.

**Tips**

- When calculating the time your solution will take to run, you can roughly use the following guidelines: $10^8$ basic operations (such as an addition, for example) might run in a second, but might not. $10^7$ operations will most likely run in a second. Anything less than $10^7$, for example, $10^6$ will definitely run in a second.
- To read a line of integers separated by spaces, you can use something like
  d = list(map(int, input().split()))
- You can represent "infinity" in python by float('inf')




(1,1,0)=1

(1,2,0)=1

(1,3,1)=4

jump

(1,2,0)=0

(i,j,k) = min

(i,j+1), k

(i+1,j), k

(i+j, j), k+1

(k-1, k-1, j)

0 1

k+1 (1,1,0)=1 ... (3,3,0)=

(1,2,0)=3 (1,3,1)=4

(2,1,0)=2 (2,3,1)=3

(3,3,2)

for k in jumps

for i in row

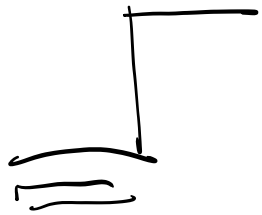
for j in column

dp[(i+1, j, ☐)] = dp[(i, j, k)] +

dp[(i, j+1, k)]

dp[(i+jumps[k], j, k+1)]

dp[(i, j+jumps[k], k+1)]




width = 6

height = 4

(0, 4, 0) = 13

i=2 j=4

i=2 j=3

j=2

(2, 4, 1) = 14

→ (3, 5, 2)

i=2 j=5

(2, 4, 2) = 7

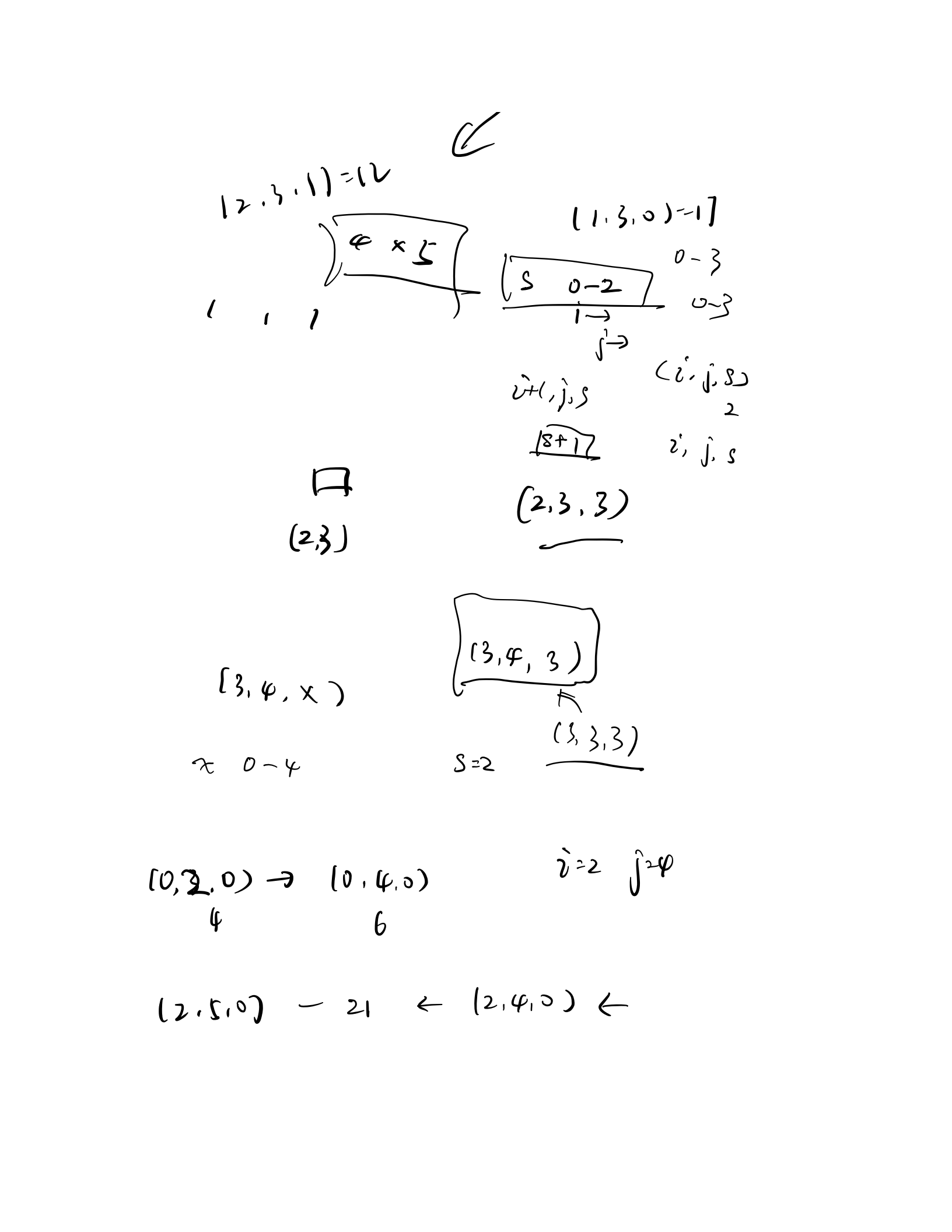
i=0 j=3

(2, 3, 1) = 13

δ=1

(0, 5, 1) = 16

i=1 j=3 δ=1


(2,3,1)=12
(1,3,0)=1
0-3
0-3
0-2
(2,3,3)
(3,4,3)
(3,3,3)
[3,4,x)
x 0-4
S=2
[0,2,0) → [0,4,0)
4 6
i=2 j=4
[2,5,0) — 21 ← [2,4,0) ←

1. 3037534676 Shenghan Zheng

$\sum_{g \in G} \sum_{h \in H} f(N-1, g, h)$

H = given g the possible set for h

add 2 empty rows at the bottom

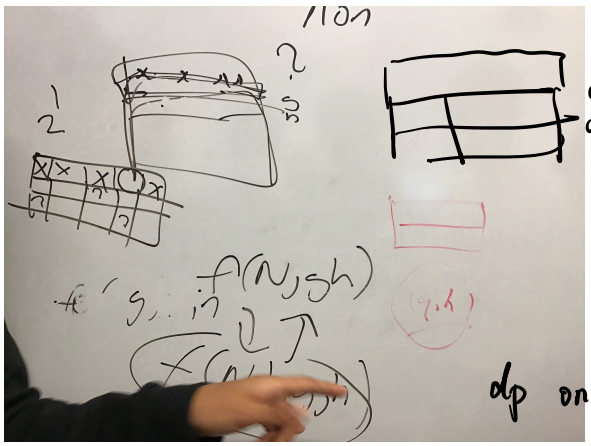
$f(1, \phi, \phi) = 2^M$

$f(2, g, \phi) = \cdots$

given g, h of last 2 rows

$N(2^M)^2$

c




(g, h) → can determine their above row's possibilities

dp on each 2 last rows

## 3 Knightmare

Give a dynamic programming algorithm to find the number of ways you can place knights on an $N$ by $M$ ($M < N$) chessboard such that no two knights can attack each other (there can be any number of knights on the board, including zero knights). Clearly describe your algorithm and prove its correctness. Your algorithm's runtime can be exponential in $M$ but should be polynomial in $N$.

**(Please provide a 4-part DP solution as defined at the top of this homework)**

## 4 Modeling: Tricks of the Trade

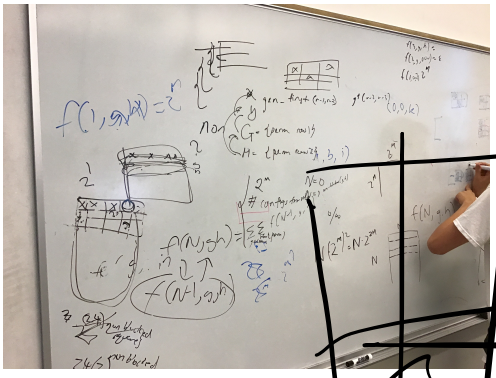
One of the most important problems in the field of *statistics* is the *linear regression problem*. Roughly speaking, this problem involves fitting a straight line to statistical data represented by points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ on a graph. Denoting the line by $y = a + bx$, the objective is to choose the constants $a$ and $b$ to provide the "best" fit according to some criterion. The criterion usually used is the *method of least squares*, but there are other interesting criteria where linear programming can be used to solve for the optimal values of $a$ and $b$.

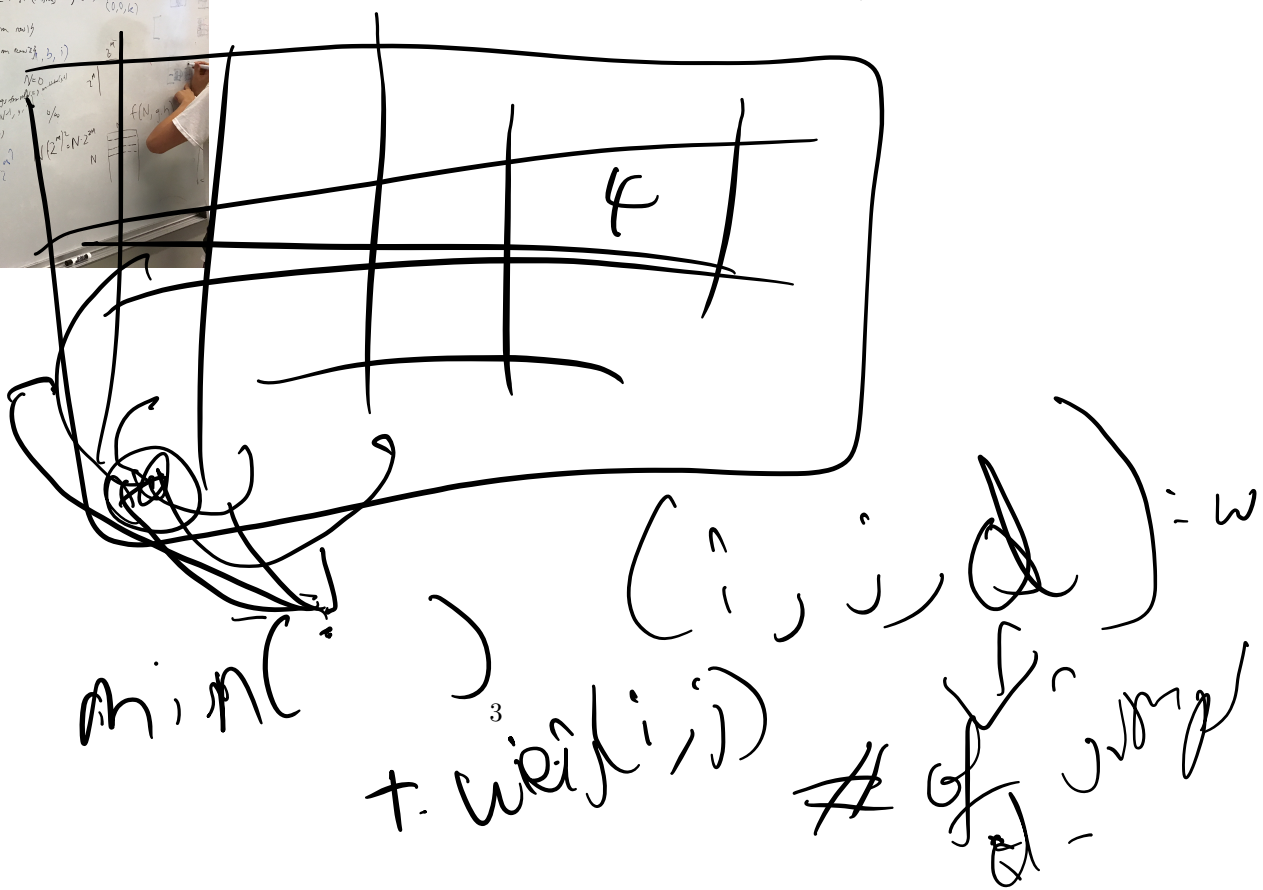
Suppose instead we wish to minimize the sum of the absolute deviations of the data from the line, that is,

$$\min \sum_{i=1}^{n} |y_i - (a + bx_i)|$$

Write a linear program with variables $a, b$ to solve this problem.

*Hint: Create a new variable $z_i$ that will equal $|y_i - (a + bx_i)|$ in the optimal solution.*



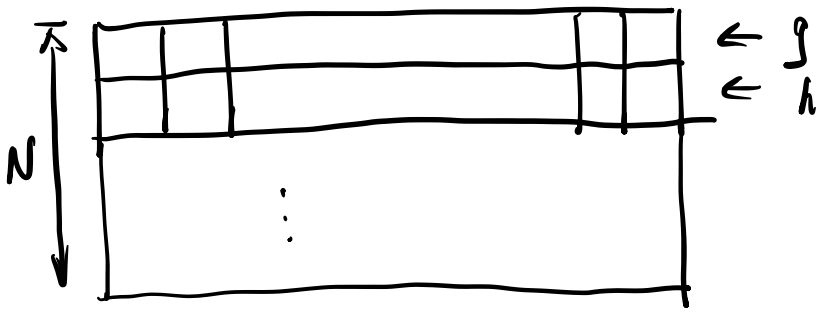


3.

1. define a function

input : $N$ , $g$, $h$ ($g$, $h$ are the permutation of the 2 rows at the top of the table with $N$ rows)

output : number of possibilities




2. Base case , Recurrence Relation

insert 2 empty rows under the matrix

$$f(1, g, 0) = 1$$

$$f(2, g, h) = 1$$

$$f(N, g, h) = \sum_{s \in S} f(N-1, h, s)$$

$S = \{$all possible permutation of $N-3$th row given $h$, $g\}$

given g, h

3. Correctness

From the rules of the knight, if we start from bottom to up, the 1st row is only affected by it's underneath 2 rows. Thus, given g, h we can find all permutations for 1st row

Induction:

base case: if there is only 1 row, and the row is given, then the number of possibility is 1. If there are 2 rows, the number of possibility is also 1

for N

to find f(N, g, h) we need to find how many permutations for the N-2 and N-3 row to make g valid which is $\sum_{s \in S} f(N-1, h, s)$. By induction, it will return the correct value of sum of all possibilities of the placement that makes g valid.

proved!

4.

Space complexity

Since each $(g,h)$'s permutation is at most $\left(2^m\right)^2$

$\Rightarrow O\left(\frac{N}{2}\left(2^m\right)^2\right) = O\left(N\left(2^m\right)^2\right)$

Runtime complexity

for each $(g,h)$, we need compute the number of possible permutation which is at most $2^m$

thus runtime complexity $= O\left(N\left(2^m\right)^2 \cdot 2^m\right)$

$= O\left(N\left(2^m\right)^3\right)$

bottom up can't improve space complexity because we traverse all possible permutations of $(g,h)$

3.

$(i, j)$

key $(i, j, \text{jump}) : w$

coordinate — weight

base case $\hat{j} = 0$ $w$ = value on that grid

4.

Let $z_i = |y_i - (a+bx_i)| \quad (i = 1, \cdots, n)$

Thus, we get $\quad -z_i \le y_i - (a+bx_i) \le z_i$

$$\Rightarrow \quad \begin{aligned} z_i + y_i - bx_i &\ge a \\ z_i - y_i + bx_i &\ge -a \end{aligned} \qquad (1) \qquad x = \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}$$

$$y = u - v \quad u, v \ge 0 \qquad x = p - q \quad p, q \ge 0$$

$$(1) \Leftrightarrow \quad \begin{aligned} z_i + u_i - v_i - bp_i + bq_i &\ge a \\ z_i - u_i + v_i + bp_i - bq_i &\ge -a \end{aligned}$$

$$\hat{A} = \begin{pmatrix} \vec{e} & \vec{e} & -\vec{e} & -\vec{b} & \vec{b} \\ \vec{e} & -\vec{e} & \vec{e} & \vec{b} & -\vec{b} \end{pmatrix} \qquad \hat{b} = \begin{pmatrix} \vec{a} \\ -\vec{a} \end{pmatrix}$$

$$\hat{x} = \begin{bmatrix} z \\ u \\ v \\ p \\ q \end{bmatrix} \qquad \vec{e} = \begin{bmatrix} 1 \\ \vdots \\ 1 \end{bmatrix} \Big\}n \quad \vec{b} = \begin{bmatrix} b \\ \vdots \\ b \end{bmatrix} \Big\}n \quad z = \begin{bmatrix} z_1 \\ \vdots \\ z_n \end{bmatrix} \quad \vec{a} = \begin{bmatrix} a \\ \vdots \\ a \end{bmatrix} \Big\}n$$

$$\hat{c} = \begin{bmatrix} \vec{e} \\ \vec{0} \end{bmatrix} \qquad u = \begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} \quad v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \quad p = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} \quad q = \begin{bmatrix} q_1 \\ \vdots \\ q_n \end{bmatrix}$$

LP:

$$\hat{A}\hat{x} \geq \hat{b}$$

$$\hat{x} \geq 0$$

$$\hat{c}^T\hat{x} = \min$$