



# SHAPE Final Project "Cupboard Jumps"

8.16.24

Dexter Theisen, Siddharth Nair



01

# Problem Statement

What are we solving?!

# "Cupboard Jumps" – 3500\* rated

From Moscow-based **CodeForces Round 707**

**\*3500** is the highest rating a problem can receive

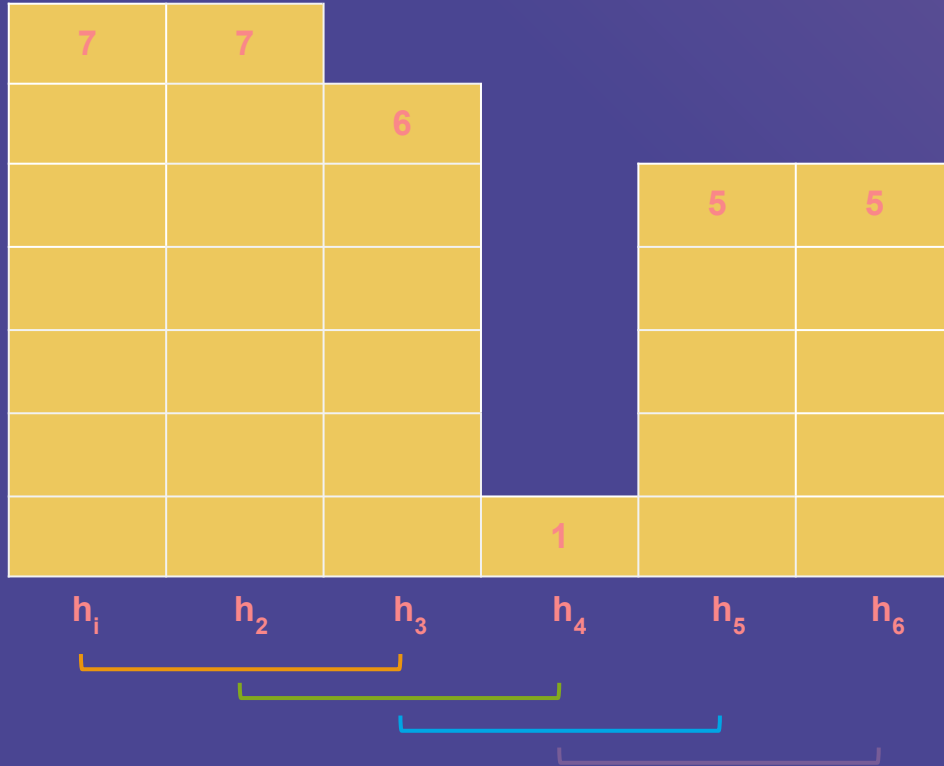
In the house where Krosh used to live, he had  $n$  cupboards standing in a line, the  $i$ -th cupboard had the height of  $h_i$ . Krosh moved recently, but he wasn't able to move the cupboards with him. Now he wants to buy  $n$  new cupboards so that they look as similar to old ones as possible.

Krosh does not remember the exact heights of the cupboards, but instead for every three consecutive cupboards he remembers the height difference between the tallest and the shortest of them. In other words, if the cupboards' heights were  $h_1, h_2, h_3, \dots, h_n$ , then Krosh remembers the values  $w_i = \max(h_i, h_{i+1}, h_{i+2}) - \min(h_i, h_{i+1}, h_{i+2})$  for all  $1 \leq i \leq n-2$ .

Krosh wants to buy such  $n$  cupboards that all the values  $w_i$  remain the same. Help him determine the required cupboards' heights, or determine that he remembers something incorrectly and there is no suitable sequence of heights.

**Note:** all cupboards must have a height greater than or equal to 0.

# Example Diagram



In this example, the cupboards have heights: 7, 7, 6, 1, 5, 5. This means that:

$$w_1 = \max(7, 7, 6) - \min(7, 7, 6) = 7 - 6 = 1$$

$$w_2 = \max(7, 6, 1) - \min(7, 6, 1) = 7 - 1 = 6$$

$$w_3 = \max(6, 1, 5) - \min(6, 1, 5) = 6 - 1 = 5$$

$$w_4 = \max(1, 5, 5) - \min(1, 5, 5) = 5 - 1 = 4$$

# Example Input/Output

Input:

$n$     $C$   
6   9

$w_1$     $w_2$     $w_3$     $w_4$   
1   6   5   4

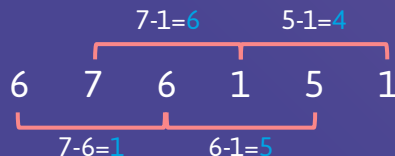
Here,  $n$  denotes the number of cupboards Krosh has. Each  $w_i$  denotes respectively the range for every 3 cupboards. And  $C$  represents an upper bound on  $w_i$  (used primarily in the code and not the logic.)

Output:

An example output may look like:

$h'_1$     $h'_2$     $h'_3$     $h'_4$     $h'_5$     $h'_6$   
6   7   6   1   5   1

where  $h'_i$  denotes the  $i$ -th cupboard in Krosh's new house. Note that there can exist more than one solution for any given  $w_i$ , but it is our job only to output one.



It can be easily seen that our solution yields the correct  $w_i$ :

1   6   5   4

A decorative graphic in the top-left corner consisting of a network of thin, light-blue lines that resemble a circuit board or a neural network, with small dots at various nodes.

# 02

# Brainstorming

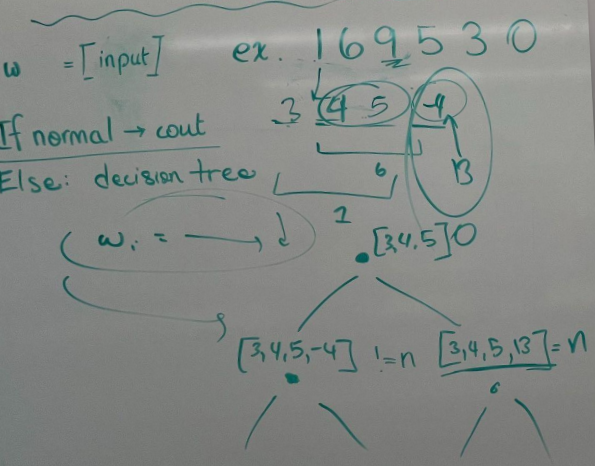
Okay great! What do we do now?

A decorative graphic in the bottom-right corner consisting of a network of thin, light-blue lines that resemble a circuit board or a neural network, with small dots at various nodes.

possible  $w_i$ :  $w_i = \text{range}(h_i, h_{i-1}, h_{i+1})$   
 mean in terms of constraints

① ②  
 $n-2$  5 7 2 3 4 5 2 1 8  
 $n-2$   $w_1, w_2, \dots, w_{n-2}$  ( $0 \leq w_i \leq$ )

Pseudocode for dynamic programming



T2:  
 $h'' = 1 \ 1 \ 20 \ 2 \ 3 \ 7$   
 $w = 19, 19, 18, 5, 10,$

T3:  
 $h'' = 12 \ 0 \ 11 \ 0 \ 10 \ 0$   
 $w = 12, 11, 11, 10, 10,$

T4:  
 $h'' = 2 \ 7 \ 3 \ 4 \ 5 \ 6 \ 2$   
 $w = 5, 4, 2, 2, 4, 5, 4,$

T5:  
 $h'' = 1 \ 3 \ 5 \ 7 \ 9 \ 2 \ 4$   
 $w = 4, 4, 4, 7, 7, 2,$

T6: (no sol.)  
 $w = 1 \ 7 \ 13 \ 2 \ 1 \ 1$

T7: (no sol.)  
 $w = 0 \ 0 \ 4 \ 5 \ 6 \ 1 \ 8$

max (pr...)

$n=6$

3 4 5 4  
 1 3 4 0  
 a b c d e

Does's conjecture:  
 for all  $v_j, w_j, v_{j+2}$  where  $0 \leq j \leq n-2$ :  
 if:  $v_{j+1} \geq v_j$  and  $w_j \geq w_{j+2}$  and  $(w_j + w_{j+2}) \geq w_{j+1}$   
 there exists some solution  $S$  for  $h_n$ .

Whiteboarding!!!!

max  
 $w_i = (\text{pairwise } d_i)$

Theorem's Algorithm:  
 Init:  
 $h_1 = 0$   
 $h_2 = \max(0, v_1 - v_2)$   
 $h_3 = v_1$   
 from here:  
 if  $w_{i-2} \geq |h_{i-1} - h_{i-2}|$ :  
 let  $a = \min(h_{i-1}, h_{i-2}) + w_{i-2}$   
 let  $b = \max(h_{i-1}, h_{i-2}) - w_{i-2}$   
 if  $|h_{i-1} - a| \leq |h_{i-1} - b|$ :  
 else:  
 $h_i = a$   
 else:  
 $h_i = b$   
 else:  
 cut cc - No ascend;  
 break;  
 "once the for loop (to calculate all  $h_i$ ) is done  
 cut cc - No ascend;  
 for loop to print all  $h_i$   
 return 0; // :)

$d_1 = [1, 3]$   
 $d_2 = [0, 2]$   
 $d_3 = [1, 3]$   
 $d_4 = [1, 3]$   
 $d_5 = [1, 3]$   
 $d_6 = [1, 3]$   
 $d_7 = [1, 3]$   
 $d_8 = [1, 3]$   
 $d_9 = [1, 3]$   
 $d_{10} = [1, 3]$   
 $d_{11} = [1, 3]$   
 $d_{12} = [1, 3]$   
 $d_{13} = [1, 3]$   
 $d_{14} = [1, 3]$   
 $d_{15} = [1, 3]$   
 $d_{16} = [1, 3]$   
 $d_{17} = [1, 3]$   
 $d_{18} = [1, 3]$   
 $d_{19} = [1, 3]$   
 $d_{20} = [1, 3]$   
 $d_{21} = [1, 3]$   
 $d_{22} = [1, 3]$   
 $d_{23} = [1, 3]$   
 $d_{24} = [1, 3]$   
 $d_{25} = [1, 3]$   
 $d_{26} = [1, 3]$   
 $d_{27} = [1, 3]$   
 $d_{28} = [1, 3]$   
 $d_{29} = [1, 3]$   
 $d_{30} = [1, 3]$   
 $d_{31} = [1, 3]$   
 $d_{32} = [1, 3]$   
 $d_{33} = [1, 3]$   
 $d_{34} = [1, 3]$   
 $d_{35} = [1, 3]$   
 $d_{36} = [1, 3]$   
 $d_{37} = [1, 3]$   
 $d_{38} = [1, 3]$   
 $d_{39} = [1, 3]$   
 $d_{40} = [1, 3]$   
 $d_{41} = [1, 3]$   
 $d_{42} = [1, 3]$   
 $d_{43} = [1, 3]$   
 $d_{44} = [1, 3]$   
 $d_{45} = [1, 3]$   
 $d_{46} = [1, 3]$   
 $d_{47} = [1, 3]$   
 $d_{48} = [1, 3]$   
 $d_{49} = [1, 3]$   
 $d_{50} = [1, 3]$   
 $d_{51} = [1, 3]$   
 $d_{52} = [1, 3]$   
 $d_{53} = [1, 3]$   
 $d_{54} = [1, 3]$   
 $d_{55} = [1, 3]$   
 $d_{56} = [1, 3]$   
 $d_{57} = [1, 3]$   
 $d_{58} = [1, 3]$   
 $d_{59} = [1, 3]$   
 $d_{60} = [1, 3]$   
 $d_{61} = [1, 3]$   
 $d_{62} = [1, 3]$   
 $d_{63} = [1, 3]$   
 $d_{64} = [1, 3]$   
 $d_{65} = [1, 3]$   
 $d_{66} = [1, 3]$   
 $d_{67} = [1, 3]$   
 $d_{68} = [1, 3]$   
 $d_{69} = [1, 3]$   
 $d_{70} = [1, 3]$   
 $d_{71} = [1, 3]$   
 $d_{72} = [1, 3]$   
 $d_{73} = [1, 3]$   
 $d_{74} = [1, 3]$   
 $d_{75} = [1, 3]$   
 $d_{76} = [1, 3]$   
 $d_{77} = [1, 3]$   
 $d_{78} = [1, 3]$   
 $d_{79} = [1, 3]$   
 $d_{80} = [1, 3]$   
 $d_{81} = [1, 3]$   
 $d_{82} = [1, 3]$   
 $d_{83} = [1, 3]$   
 $d_{84} = [1, 3]$   
 $d_{85} = [1, 3]$   
 $d_{86} = [1, 3]$   
 $d_{87} = [1, 3]$   
 $d_{88} = [1, 3]$   
 $d_{89} = [1, 3]$   
 $d_{90} = [1, 3]$   
 $d_{91} = [1, 3]$   
 $d_{92} = [1, 3]$   
 $d_{93} = [1, 3]$   
 $d_{94} = [1, 3]$   
 $d_{95} = [1, 3]$   
 $d_{96} = [1, 3]$   
 $d_{97} = [1, 3]$   
 $d_{98} = [1, 3]$   
 $d_{99} = [1, 3]$   
 $d_{100} = [1, 3]$   
 $d_{101} = [1, 3]$   
 $d_{102} = [1, 3]$   
 $d_{103} = [1, 3]$   
 $d_{104} = [1, 3]$   
 $d_{105} = [1, 3]$   
 $d_{106} = [1, 3]$   
 $d_{107} = [1, 3]$   
 $d_{108} = [1, 3]$   
 $d_{109} = [1, 3]$   
 $d_{110} = [1, 3]$   
 $d_{111} = [1, 3]$   
 $d_{112} = [1, 3]$   
 $d_{113} = [1, 3]$   
 $d_{114} = [1, 3]$   
 $d_{115} = [1, 3]$   
 $d_{116} = [1, 3]$   
 $d_{117} = [1, 3]$   
 $d_{118} = [1, 3]$   
 $d_{119} = [1, 3]$   
 $d_{120} = [1, 3]$   
 $d_{121} = [1, 3]$   
 $d_{122} = [1, 3]$   
 $d_{123} = [1, 3]$   
 $d_{124} = [1, 3]$   
 $d_{125} = [1, 3]$   
 $d_{126} = [1, 3]$   
 $d_{127} = [1, 3]$   
 $d_{128} = [1, 3]$   
 $d_{129} = [1, 3]$   
 $d_{130} = [1, 3]$   
 $d_{131} = [1, 3]$   
 $d_{132} = [1, 3]$   
 $d_{133} = [1, 3]$   
 $d_{134} = [1, 3]$   
 $d_{135} = [1, 3]$   
 $d_{136} = [1, 3]$   
 $d_{137} = [1, 3]$   
 $d_{138} = [1, 3]$   
 $d_{139} = [1, 3]$   
 $d_{140} = [1, 3]$   
 $d_{141} = [1, 3]$   
 $d_{142} = [1, 3]$   
 $d_{143} = [1, 3]$   
 $d_{144} = [1, 3]$   
 $d_{145} = [1, 3]$   
 $d_{146} = [1, 3]$   
 $d_{147} = [1, 3]$   
 $d_{148} = [1, 3]$   
 $d_{149} = [1, 3]$   
 $d_{150} = [1, 3]$   
 $d_{151} = [1, 3]$   
 $d_{152} = [1, 3]$   
 $d_{153} = [1, 3]$   
 $d_{154} = [1, 3]$   
 $d_{155} = [1, 3]$   
 $d_{156} = [1, 3]$   
 $d_{157} = [1, 3]$   
 $d_{158} = [1, 3]$   
 $d_{159} = [1, 3]$   
 $d_{160} = [1, 3]$   
 $d_{161} = [1, 3]$   
 $d_{162} = [1, 3]$   
 $d_{163} = [1, 3]$   
 $d_{164} = [1, 3]$   
 $d_{165} = [1, 3]$   
 $d_{166} = [1, 3]$   
 $d_{167} = [1, 3]$   
 $d_{168} = [1, 3]$   
 $d_{169} = [1, 3]$   
 $d_{170} = [1, 3]$   
 $d_{171} = [1, 3]$   
 $d_{172} = [1, 3]$   
 $d_{173} = [1, 3]$   
 $d_{174} = [1, 3]$   
 $d_{175} = [1, 3]$   
 $d_{176} = [1, 3]$   
 $d_{177} = [1, 3]$   
 $d_{178} = [1, 3]$   
 $d_{179} = [1, 3]$   
 $d_{180} = [1, 3]$   
 $d_{181} = [1, 3]$   
 $d_{182} = [1, 3]$   
 $d_{183} = [1, 3]$   
 $d_{184} = [1, 3]$   
 $d_{185} = [1, 3]$   
 $d_{186} = [1, 3]$   
 $d_{187} = [1, 3]$   
 $d_{188} = [1, 3]$   
 $d_{189} = [1, 3]$   
 $d_{190} = [1, 3]$   
 $d_{191} = [1, 3]$   
 $d_{192} = [1, 3]$   
 $d_{193} = [1, 3]$   
 $d_{194} = [1, 3]$   
 $d_{195} = [1, 3]$   
 $d_{196} = [1, 3]$   
 $d_{197} = [1, 3]$   
 $d_{198} = [1, 3]$   
 $d_{199} = [1, 3]$   
 $d_{200} = [1, 3]$   
 $d_{201} = [1, 3]$   
 $d_{202} = [1, 3]$   
 $d_{203} = [1, 3]$   
 $d_{204} = [1, 3]$   
 $d_{205} = [1, 3]$   
 $d_{206} = [1, 3]$   
 $d_{207} = [1, 3]$   
 $d_{208} = [1, 3]$   
 $d_{209} = [1, 3]$   
 $d_{210} = [1, 3]$   
 $d_{211} = [1, 3]$   
 $d_{212} = [1, 3]$   
 $d_{213} = [1, 3]$   
 $d_{214} = [1, 3]$   
 $d_{215} = [1, 3]$   
 $d_{216} = [1, 3]$   
 $d_{217} = [1, 3]$   
 $d_{218} = [1, 3]$   
 $d_{219} = [1, 3]$   
 $d_{220} = [1, 3]$   
 $d_{221} = [1, 3]$   
 $d_{222} = [1, 3]$   
 $d_{223} = [1, 3]$   
 $d_{224} = [1, 3]$   
 $d_{225} = [1, 3]$   
 $d_{226} = [1, 3]$   
 $d_{227} = [1, 3]$   
 $d_{228} = [1, 3]$   
 $d_{229} = [1, 3]$   
 $d_{230} = [1, 3]$   
 $d_{231} = [1, 3]$   
 $d_{232} = [1, 3]$   
 $d_{233} = [1, 3]$   
 $d_{234} = [1, 3]$   
 $d_{235} = [1, 3]$   
 $d_{236} = [1, 3]$   
 $d_{237} = [1, 3]$   
 $d_{238} = [1, 3]$   
 $d_{239} = [1, 3]$   
 $d_{240} = [1, 3]$   
 $d_{241} = [1, 3]$   
 $d_{242} = [1, 3]$   
 $d_{243} = [1, 3]$   
 $d_{244} = [1, 3]$   
 $d_{245} = [1, 3]$   
 $d_{246} = [1, 3]$   
 $d_{247} = [1, 3]$   
 $d_{248} = [1, 3]$   
 $d_{249} = [1, 3]$   
 $d_{250} = [1, 3]$   
 $d_{251} = [1, 3]$   
 $d_{252} = [1, 3]$   
 $d_{253} = [1, 3]$   
 $d_{254} = [1, 3]$   
 $d_{255} = [1, 3]$   
 $d_{256} = [1, 3]$   
 $d_{257} = [1, 3]$   
 $d_{258} = [1, 3]$   
 $d_{259} = [1, 3]$   
 $d_{260} = [1, 3]$   
 $d_{261} = [1, 3]$   
 $d_{262} = [1, 3]$   
 $d_{263} = [1, 3]$   
 $d_{264} = [1, 3]$   
 $d_{265} = [1, 3]$   
 $d_{266} = [1, 3]$   
 $d_{267} = [1, 3]$   
 $d_{268} = [1, 3]$   
 $d_{269} = [1, 3]$   
 $d_{270} = [1, 3]$   
 $d_{271} = [1, 3]$   
 $d_{272} = [1, 3]$   
 $d_{273} = [1, 3]$   
 $d_{274} = [1, 3]$   
 $d_{275} = [1, 3]$   
 $d_{276} = [1, 3]$   
 $d_{277} = [1, 3]$   
 $d_{278} = [1, 3]$   
 $d_{279} = [1, 3]$   
 $d_{280} = [1, 3]$   
 $d_{281} = [1, 3]$   
 $d_{282} = [1, 3]$   
 $d_{283} = [1, 3]$   
 $d_{284} = [1, 3]$   
 $d_{285} = [1, 3]$   
 $d_{286} = [1, 3]$   
 $d_{287} = [1, 3]$   
 $d_{288} = [1, 3]$   
 $d_{289} = [1, 3]$   
 $d_{290} = [1, 3]$   
 $d_{291} = [1, 3]$   
 $d_{292} = [1, 3]$   
 $d_{293} = [1, 3]$   
 $d_{294} = [1, 3]$   
 $d_{295} = [1, 3]$   
 $d_{296} = [1, 3]$   
 $d_{297} = [1, 3]$   
 $d_{298} = [1, 3]$   
 $d_{299} = [1, 3]$   
 $d_{300} = [1, 3]$   
 $d_{301} = [1, 3]$   
 $d_{302} = [1, 3]$   
 $d_{303} = [1, 3]$   
 $d_{304} = [1, 3]$   
 $d_{305} = [1, 3]$   
 $d_{306} = [1, 3]$   
 $d_{307} = [1, 3]$   
 $d_{308} = [1, 3]$   
 $d_{309} = [1, 3]$   
 $d_{310} = [1, 3]$   
 $d_{311} = [1, 3]$   
 $d_{312} = [1, 3]$   
 $d_{313} = [1, 3]$   
 $d_{314} = [1, 3]$   
 $d_{315} = [1, 3]$   
 $d_{316} = [1, 3]$   
 $d_{317} = [1, 3]$   
 $d_{318} = [1, 3]$   
 $d_{319} = [1, 3]$   
 $d_{320} = [1, 3]$   
 $d_{321} = [1, 3]$   
 $d_{322} = [1, 3]$   
 $d_{323} = [1, 3]$   
 $d_{324} = [1, 3]$   
 $d_{325} = [1, 3]$   
 $d_{326} = [1, 3]$   
 $d_{327} = [1, 3]$   
 $d_{328} = [1, 3]$   
 $d_{329} = [1, 3]$   
 $d_{330} = [1, 3]$   
 $d_{331} = [1, 3]$   
 $d_{332} = [1, 3]$   
 $d_{333} = [1, 3]$   
 $d_{334} = [1, 3]$   
 $d_{335} = [1, 3]$   
 $d_{336} = [1, 3]$   
 $d_{337} = [1, 3]$   
 $d_{338} = [1, 3]$   
 $d_{339} = [1, 3]$   
 $d_{340} = [1, 3]$   
 $d_{341} = [1, 3]$   
 $d_{342} = [1, 3]$   
 $d_{343} = [1, 3]$   
 $d_{344} = [1, 3]$   
 $d_{345} = [1, 3]$   
 $d_{346} = [1, 3]$   
 $d_{347} = [1, 3]$   
 $d_{348} = [1, 3]$   
 $d_{349} = [1, 3]$   
 $d_{350} = [1, 3]$   
 $d_{351} = [1, 3]$   
 $d_{352} = [1, 3]$   
 $d_{353} = [1, 3]$   
 $d_{354} = [1, 3]$   
 $d_{355} = [1, 3]$   
 $d_{356} = [1, 3]$   
 $d_{357} = [1, 3]$   
 $d_{358} = [1, 3]$   
 $d_{359} = [1, 3]$   
 $d_{360} = [1, 3]$   
 $d_{361} = [1, 3]$   
 $d_{362} = [1, 3]$   
 $d_{363} = [1, 3]$   
 $d_{364} = [1, 3]$   
 $d_{365} = [1, 3]$   
 $d_{366} = [1, 3]$   
 $d_{367} = [1, 3]$   
 $d_{368} = [1, 3]$   
 $d_{369} = [1, 3]$   
 $d_{370} = [1, 3]$   
 $d_{371} = [1, 3]$   
 $d_{372} = [1, 3]$   
 $d_{373} = [1, 3]$   
 $d_{374} = [1, 3]$   
 $d_{375} = [1, 3]$   
 $d_{376} = [1, 3]$   
 $d_{377} = [1, 3]$   
 $d_{378} = [1, 3]$   
 $d_{379} = [1, 3]$   
 $d_{380} = [1, 3]$   
 $d_{381} = [1, 3]$   
 $d_{382} = [1, 3]$   
 $d_{383} = [1, 3]$   
 $d_{384} = [1, 3]$   
 $d_{385} = [1, 3]$   
 $d_{386} = [1, 3]$   
 $d_{387} = [1, 3]$   
 $d_{388} = [1, 3]$   
 $d_{389} = [1, 3]$   
 $d_{390} = [1, 3]$   
 $d_{391} = [1, 3]$   
 $d_{392} = [1, 3]$   
 $d_{393} = [1, 3]$   
 $d_{394} = [1, 3]$   
 $d_{395} = [1, 3]$   
 $d_{396} = [1, 3]$   
 $d_{397} = [1, 3]$   
 $d_{398} = [1, 3]$   
 $d_{399} = [1, 3]$   
 $d_{400} = [1, 3]$   
 $d_{401} = [1, 3]$   
 $d_{402} = [1, 3]$   
 $d_{403} = [1, 3]$   
 $d_{404} = [1, 3]$   
 $d_{405} = [1, 3]$   
 $d_{406} = [1, 3]$   
 $d_{407} = [1, 3]$   
 $d_{408} = [1, 3]$   
 $d_{409} = [1, 3]$   
 $d_{410} = [1, 3]$   
 $d_{411} = [1, 3]$   
 $d_{412} = [1, 3]$   
 $d_{413} = [1, 3]$   
 $d_{414} = [1, 3]$   
 $d_{415} = [1, 3]$   
 $d_{416} = [1, 3]$   
 $d_{417} = [1, 3]$   
 $d_{418} = [1, 3]$   
 $d_{419} = [1, 3]$   
 $d_{420} = [1, 3]$   
 $d_{421} = [1, 3]$   
 $d_{422} = [1, 3]$   
 $d_{423} = [1, 3]$   
 $d_{424} = [1, 3]$   
 $d_{425} = [1, 3]$   
 $d_{426} = [1, 3]$   
 $d_{427} = [1, 3]$   
 $d_{428} = [1, 3]$   
 $d_{429} = [1, 3]$   
 $d_{430} = [1, 3]$   
 $d_{431} = [1, 3]$   
 $d_{432} = [1, 3]$   
 $d_{433} = [1, 3]$   
 $d_{434} = [1, 3]$   
 $d_{435} = [1, 3]$   
 $d_{436} = [1, 3]$   
 $d_{437} = [1, 3]$   
 $d_{438} = [1, 3]$   
 $d_{439} = [1, 3]$   
 $d_{440} = [1, 3]$   
 $d_{441} = [1, 3]$   
 $d_{442} = [1, 3]$   
 $d_{443} = [1, 3]$   
 $d_{444} = [1, 3]$   
 $d_{445} = [1, 3]$   
 $d_{446} = [1, 3]$   
 $d_{447} = [1, 3]$   
 $d_{448} = [1, 3]$   
 $d_{449} = [1, 3]$   
 $d_{450} = [1, 3]$   
 $d_{451} = [1, 3]$   
 $d_{452} = [1, 3]$   
 $d_{453} = [1, 3]$   
 $d_{454} = [1, 3]$   
 $d_{455} = [1, 3]$   
 $d_{456} = [1, 3]$   
 $d_{457} = [1, 3]$   
 $d_{458} = [1, 3]$   
 $d_{459} = [1, 3]$   
 $d_{460} = [1, 3]$   
 $d_{461} = [1, 3]$   
 $d_{462} = [1, 3]$   
 $d_{463} = [1, 3]$   
 $d_{464} = [1, 3]$   
 $d_{465} = [1, 3]$   
 $d_{466} = [1, 3]$   
 $d_{467} = [1, 3]$   
 $d_{468} = [1, 3]$   
 $d_{469} = [1, 3]$   
 $d_{470} = [1, 3]$   
 $d_{471} = [1, 3]$   
 $d_{472} = [1, 3]$   
 $d_{473} = [1, 3]$   
 $d_{474} = [1, 3]$   
 $d_{475} = [1, 3]$   
 $d_{476} = [1, 3]$   
 $d_{477} = [1, 3]$   
 $d_{478} = [1, 3]$   
 $d_{479} = [1, 3]$   
 $d_{480} = [1, 3]$   
 $d_{481} = [1, 3]$   
 $d_{482} = [1, 3]$   
 $d_{483} = [1, 3]$   
 $d_{484} = [1, 3]$   
 $d_{485} = [1, 3]$   
 $d_{486} = [1, 3]$   
 $d_{487} = [1, 3]$   
 $d_{488} = [1, 3]$   
 $d_{489} = [1, 3]$   
 $d_{490} = [1, 3]$   
 $d_{491} = [1, 3]$   
 $d_{492} = [1, 3]$   
 $d_{493} = [1, 3]$   
 $d_{494} = [1, 3]$   
 $d_{495} = [1, 3]$   
 $d_{496} = [1, 3]$   
 $d_{497} = [1, 3]$   
 $d_{498} = [1, 3]$   
 $d_{499} = [1, 3]$   
 $d_{500} = [1, 3]$   
 $d_{501} = [1, 3]$   
 $d_{502} = [1, 3]$   
 $d_{503} = [1, 3]$   
 $d_{504} = [1, 3]$   
 $d_{505} = [1, 3]$   
 $d_{506} = [1, 3]$   
 $d_{507} = [1, 3]$   
 $d_{508} = [1, 3]$   
 $d_{509} = [1, 3]$   
 $d_{510} = [1, 3]$   
 $d_{511} = [1, 3]$   
 $d_{512} = [1, 3]$   
 $d_{513} = [1, 3]$   
 $d_{514} = [1, 3]$   
 $d_{515} = [1, 3]$   
 $d_{516} = [1, 3]$   
 $d_{517} = [1, 3]$   
 $d_{518} = [1, 3]$   
 $d_{519} = [1, 3]$   
 $d_{520} = [1, 3]$   
 $d_{521} = [1, 3]$   
 $d_{522} = [1, 3]$   
 $d_{523} = [1, 3]$   
 $d_{524} = [1, 3]$   
 $d_{525} = [1, 3]$   
 $d_{526} = [1, 3]$   
 $d_{527} = [1, 3]$   
 $d_{528} = [1, 3]$   
 $d_{529} = [1, 3]$   
 $d_{530} = [1, 3]$   
 $d_{531} = [1, 3]$   
 $d_{532} = [1, 3]$   
 $d_{533} = [1, 3]$   
 $d_{534} = [1, 3]$   
 $d_{535} = [1, 3]$   
 $d_{536} = [1, 3]$   
 $d_{537} = [1, 3]$   
 $d_{538} = [1, 3]$   
 $d_{539} = [1, 3]$   
 $d_{540} = [1, 3]$   
 $d_{541} = [1, 3]$   
 $d_{542} = [1, 3]$   
 $d_{543} = [1, 3]$   
 $d_{544} = [1, 3]$   
 $d_{545} = [1, 3]$   
 $d_{546} = [1, 3]$   
 $d_{547} = [1, 3]$   
 $d_{548} = [1, 3]$   
 $d_{549} = [1, 3]$   
 $d_{550} = [1, 3]$   
 $d_{551} = [1, 3]$   
 $d_{552} = [1, 3]$   
 $d_{553} = [1, 3]$   
 $d_{554} = [1, 3]$   
 $d_{555} = [$

# First Thoughts

After a few hours of whiteboarding, we had derived some key insights (not all of which did we strictly implement):

- Transform the  $w_i$  into pairwise differences,  $d_i = h'_{i+1} - h'_i$ , and subsequently rewrite  $w_i$  as  $\max(|d_i|, |d_{i+1}|, |d_i + d_{i+1}|)$ .
- **WLOG\***, assume we can fix  $h'_1$  to 0, as even if we end up with some negative  $h'_i$ , we know there must be some  $k$  such that all  $h'_i + k \geq 0$ .
- We need to recursively define  $h'_{n+1}$  from  $h'_n$ ,  $h'_{n-1}$ , and  $w_{i+1}$  – but how? There are multiple ideas here, in minimizing vs maximizing  $|h'_{n+1} - h'_n|$  (or doing neither).
- Dex's Conjecture: if there exists some  $i$  such that  $w_i > w_{i+1} + w_{i-1}$ , then there exists no  $h'_i$  that satisfy the given  $w_i$ . This can be proven mathematically by assigning ranges to each  $h'_i$ .
- Direction (ie increasing vs decreasing) of the  $w_i$  affects how to define  $h_{n+1}$ .
- At the start of the sequence,  $w_1$  will be in increasing order.
- Set  $h'_3$  to  $w_1$ , and define  $h'_2$  such that  $h_1 \leq h_2 \leq h_3$ .
- Utilize some type of dynamic programming or tree system to generate  $h'_{n+1}$  faster

# Some cases

1

## The case of 0

Key points: all  $h'_i$  need to be equivalent, meaning for the next  $w_i$  there is some constraint/forced  $h'_i$ .

2

## Alternating $w_i$

Key points: very hard to know what value to set intermediate  $h'$  values to, because they affect 3  $w_i$ .

3

## No solution

Key points: Dex's conjecture only finds about 95% of these cases, meaning our main algo will have to check for impossible cases as well.

The background of the slide is a solid dark blue color. Overlaid on this background are faint, light blue circuit board traces. These traces are most prominent in the top-left and bottom-right corners, where they form a complex web of lines and small circles, resembling a printed circuit board (PCB) layout. The traces in the top-left corner are more dense and intricate, while those in the bottom-right corner are more sparse and follow a more linear path.

# 03

# Implementation

How exactly did we solve this problem?!

# Full implementation



Algo

```
#include <iostream>
#include <deque>
using namespace std;
```

1. define basic variables, arrays, and references

```
typedef pair<long, long> pll;
#define MP make_pair
#define fi first
#define se second

const int mn = 1000005;
int n;
long __, w[mn], a[mn], v1[mn], d[mn][3];
bool g[mn], rev[mn];
deque<pll> q;
```

```
bool solve() {
    long a = 1, b = 0;
    q.emplace_back(0, 1, w[1]);
    for (int i = 1; i <= n; i++) {
        if (a == 1) {
            while (!q.empty() && q.back().se + b > w[i]) {
                pll p = q.back();
                q.pop_back();
                if (p.fi + b <= w[i]) {
                    q.emplace_back(p.fi, w[i] - b);
                    break;
                }
            }
            if (q.empty()) return false;
            if (q.back().se + b == w[i]) {
                g[i] = true;
                while (!q.empty()) q.pop_back();
                q.emplace_back(0, w[i]), a = 1, b = 0;
            } else {
                v1[i] = q.back().se + b;
                a = -1, b = w[i] - b;
                q.emplace_front(b - w[i], b - w[i]);
            }
        }
    }
}
```

3. iterate over  $w_i$  and store ranges for pairwise differences  $d_i$  in array  $d$

```
else {
    while (!q.empty() && -q.front().fi + b
> w[i]) {
        pll p = q.front();
        q.pop_front();
        if (-p.se + b <= w[i]) {
            q.emplace_front(b - w[i],
p.se);
            break;
        }
    }
    if (q.empty()) return false;
    if (-q.front().fi + b == w[i]) {
        g[i] = true;
        while (!q.empty()) q.pop_back();
        q.emplace_back(0, w[i]), a = 1, b
= 0;
    } else {
        v1[i] = -q.front().fi + b;
        a = 1, b = w[i] - b;
        q.emplace_back(w[i] - b, w[i] -
b);
    }
}
if (q.empty()) return false;
d[n][2] = a * q.back().se + b;
for (int i = n; i; i--) {
    if (i < n) d[i][2] = d[i + 1][0];
    if (d[i][2] != w[i]) {
        if (g[i]) d[i][0] = w[i], d[i][1] =
w[i] - d[i][2];
        else d[i][1] = w[i], d[i][0] = w[i] -
d[i][2];
    } else if (g[i]) d[i][0] = w[i], d[i][1] =
0;
    else d[i][0] = v1[i], d[i][1] = w[i] -
v1[i];
}
```

4. backpropagate final ranges in array  $d$  that satisfy all  $w_i$

```
return true;
}
```

```
int main() {
    cin >> n >> __;
    n--;
    for (int i = 1; i <= n; i++) cin >> w[i];
    if (solve()) {
        a[1] = 0, a[2] = d[1][0];
        for (int i = 1; i <= n; i++) {
            if (w[i] == d[i][0] && !rev[i - 1])
d[i][2] = -d[i][2], rev[i] = true;
            else if (w[i] == d[i][1] && rev[i -
1]) d[i][2] = -d[i][2], rev[i] = true;
            else if (w[i] == d[i][2] && !rev[i -
1]) d[i][2] = -d[i][2], rev[i] = true;
            a[i + 2] = a[i + 1] + d[i][2];
        }
        long dlt = 0;
        for (int i = 1; i <= n; i++) dlt =
min(dlt, a[i]);
        cout << "YES" << endl;
        for (int i = 1; i <= n; i++) cout <<
a[i] - dlt << " ";
    } else { cout << "NO"; }
    return 0;
}
```

5. grab inputs  $n$ ,  $C$ , and  $w_i$

6. iterate over array  $d$  and finalize  $h_i$  values

7. ensure all values are nonnegative

8. print out either YES followed by  $h_i$  or NO

# Thank you!

Contest status ☰

#	When	Who	Problem	Lang	Verdict	Time	Memory
<a href="#">276797632</a>	Aug/16/2024 10:54 <sup>UTC-4</sup>	dexrey4	<a href="#">1500F - Cupboards Jumps</a>	C++14 (GCC 6-32)	Accepted	874 ms	62200 KB

Special thanks to our SSLs Abhishek and Courtney for their support