HKOI 2016/17
SQ4 – Magic Triangle II

Alex Tung

21/1/2017

Problem description

- Given a N-layered triangular grid with numbers
- Want to make it ‘K-magical’

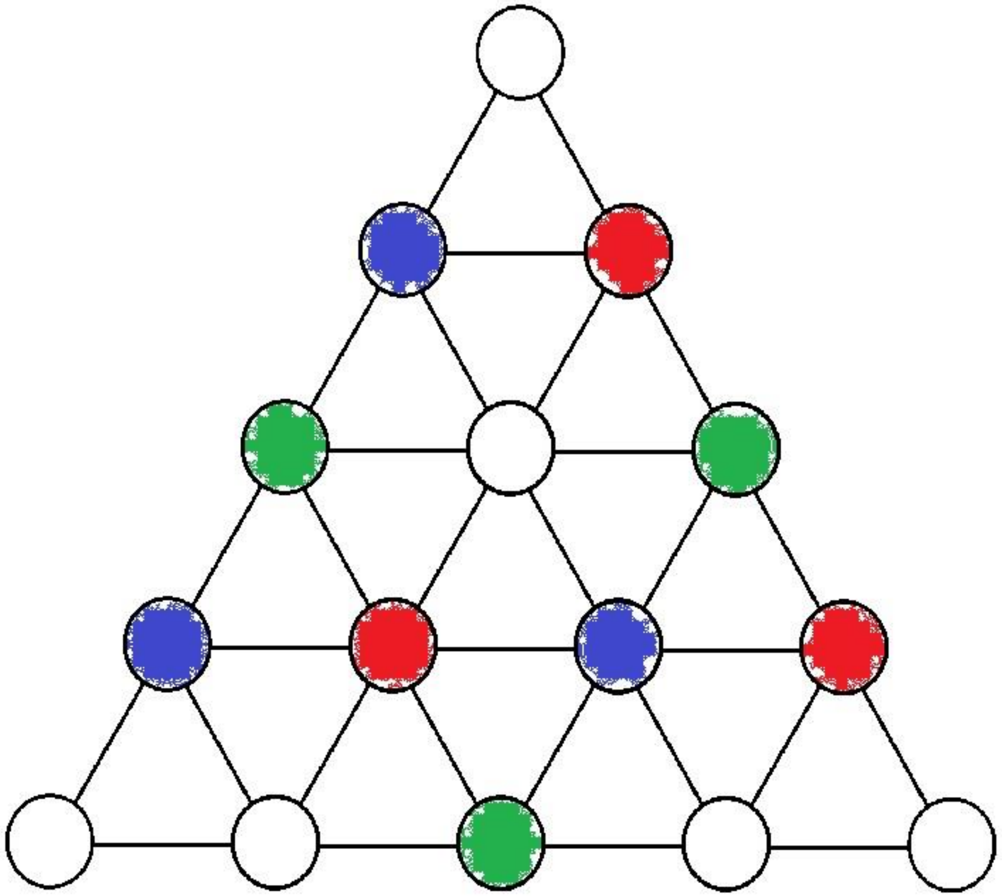
- +1: cost = A
- -1: cost = B

- Output minimal cost and final configuration

Meaning of ‘K-magical’

- For example, if 2-magical,

  sum(**B**) = sum(**R**) = sum(**G**)



SUBTASKS

For all cases: $1 \leq K < N \leq 80$, $1 \leq A, B \leq 50$

	Points	Constraints
1	11	$N = 3$ $K = 1$ $A = B = 1$
2	15	$N = 3$ $K = 1$
3	10	$N \leq 6$ $K = 1$
4	18	$K = 1$
5	25	$N \leq 10$
6	21	No additional constraints

INPUT

The first line of input consists of four integers N, A, B, and K.

For the next N lines, the i^{th} line consists of i integers, representing the initial numbers on the i^{th} layer, from left to right. The initial numbers are between 1 and 128 (inclusive).

OUTPUT

Output $N + 1$ lines in total.

On the first line, output a single integer, the minimal cost to make the grid K-magical.

On the next N lines, output a final configuration of the grid, using the same format as that for the input. All numbers on the vertices must be between 1 and 512 (inclusive).

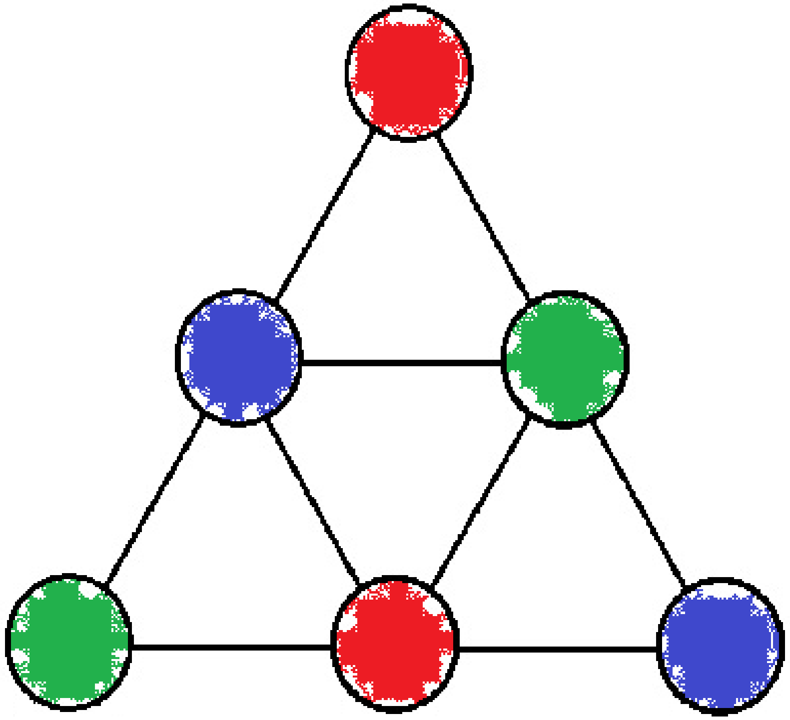
Statistics

Attempts	Max	Mean	Std Dev	Subtasks					
21	54	10.714	19.761	11: 6	15: 5	10: 3	18: 3	25: 0	21: 0

Subtasks 1, 2 (N = 3, K = 1)

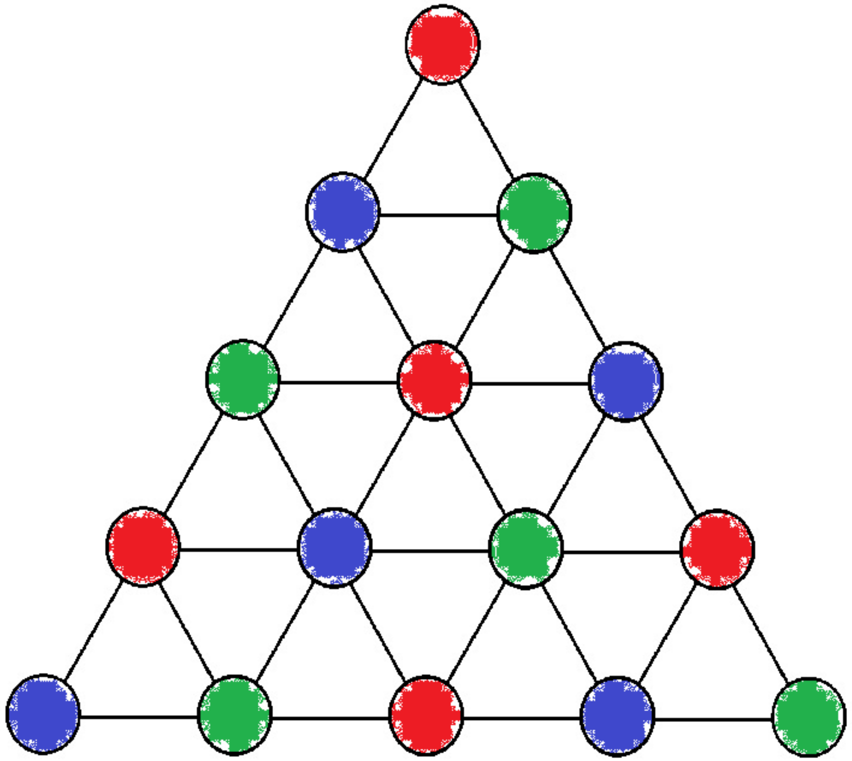
- Key Observation:

  Same numbers within a group



Subtasks 3, 4 (K = 1)

- We can divide the vertices into three small groups in similar manner



Solving for each group

- Note that the three small groups are **independent**
- For each target value V (between 1 and 128), calculate the total cost to make **all** numbers in a group = V

- Time complexity: $O(N^2R)$
  - R: range (= 128)

Subtask 5 (N small)

- K > 1 ... so what?

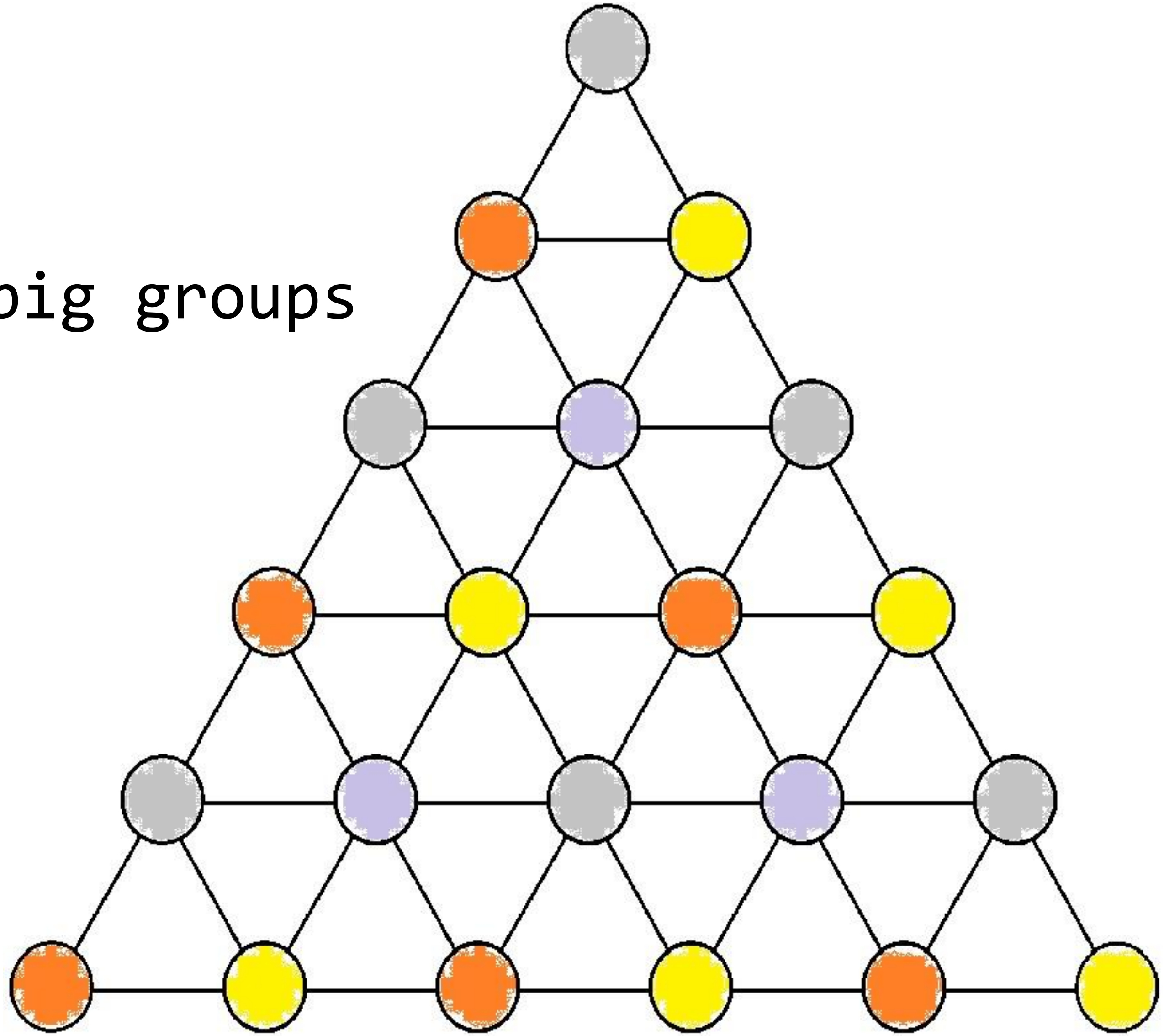
Basic idea:

- Break grid into ‘big groups’
- Solve each big group using algorithm for K = 1

- It’s more complicated than that!

‘Big groups’

- Example: $N = 6$, $K = 2$
- There are roughly K^2 big groups



Why more complicated?

- Need to make sure triangles in different **big** groups have the same sum

Algorithm

- Step 1: Break the grid into big groups
- Step 2: For each big group, calculate cost[S], the minimal cost to make triangle sum = S
- Step 3: The desired triangle sum, S_{opt}, is the one which minimizes sum(cost[S]). Output the cost and the grid.

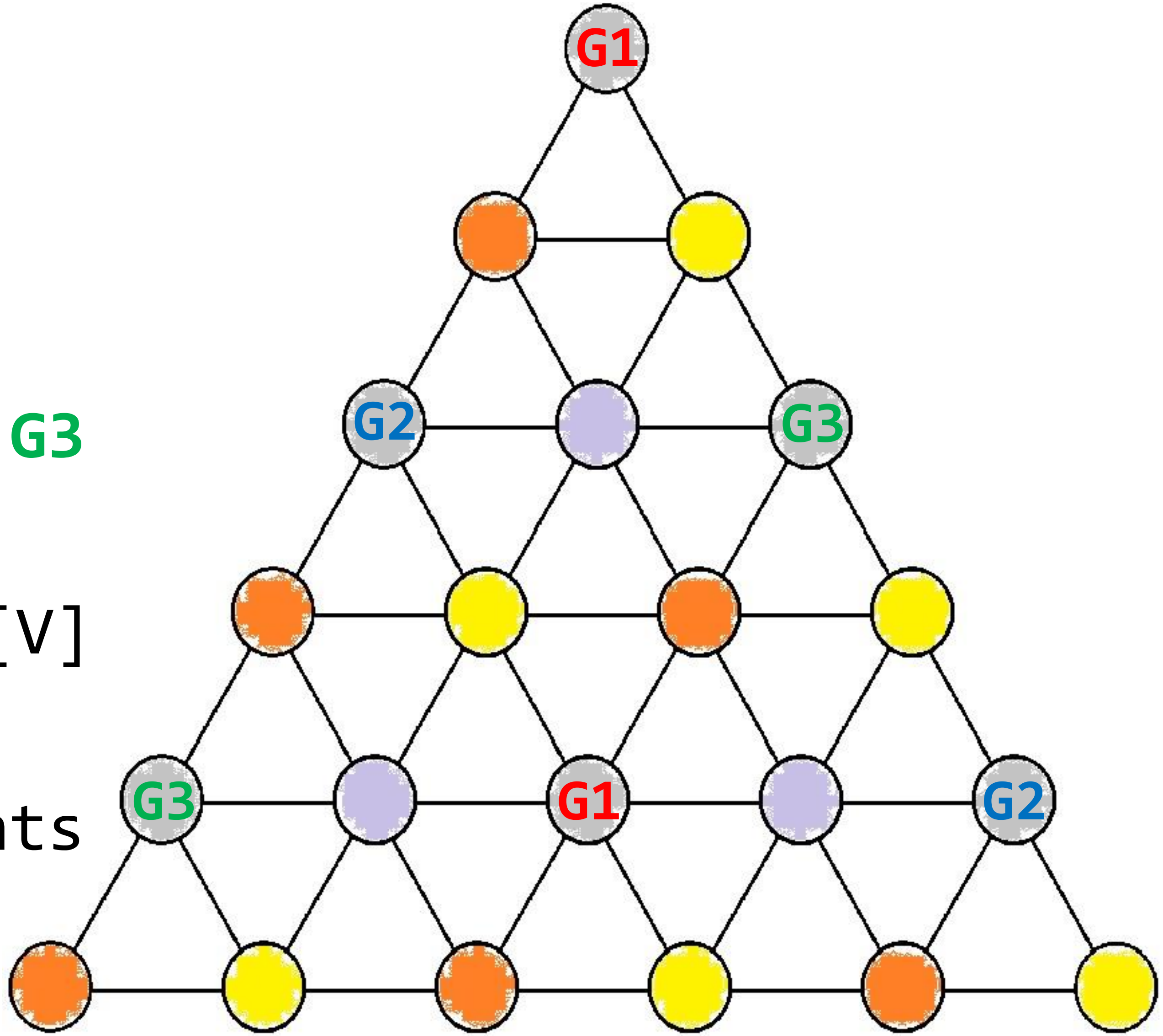
- Step 2: For each big group, calculate cost[S], the minimal cost to make triangle sum = S

- How to calculate?

- Say G is a big group with three small groups G1, G2, G3

- Calculate c1[V], c2[V], c3[V]

c1[V]: cost to change elements of G1 to V



- $cost[S] = min(c1[V_1] + c2[V_2] + c3[V_3] \mid V_1 + V_2 + V_3 = S)$

Time complexity analysis

- Calculate c1[V], c2[V], c3[V]: $O((N/K)^2 * R)$
- Calculate cost[S]: $O(R^3)$
- Time complexity: $O(K^2 * ((N/K)^2 * R + R^3))$

- $O(N^2R + K^2R^3)$, which solves subtasks 1 - 5

Full solution

- Calculate cost[S]: $O(R^3)$ <- Too slow!
- Can be optimized to $O(R^2)$

- Then, time complexity: $O(K^2 * ((N/K)^2 * R + R^2))$

- $O(N^2R + K^2R^2)$, which will get 100 points

The final optimization

- Have c1[V], c2[V], c3[V]
- precost[S'] := min(c1[V_1] + c2[V_2] | $V_1 + V_2 = S'$)
- cost[S] = min(precost[S'] + c3[V_3] | $S' + V_3 = S$)
- Each part is $O(R^2)$

The End

- Any questions?