Snackdown 2017 Finals Solutions

Maximize Grid Score : Praveen Dhinwa (38 AC)

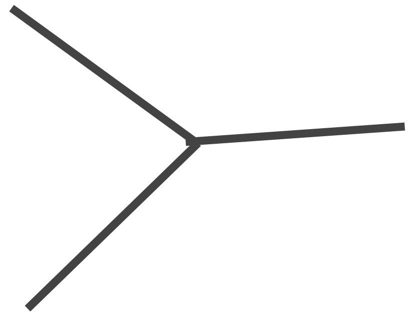
- **Key observation:** Equal characters should be connected in the grid.
- Characters with even number of occurrences are easy to fill
  - aaaa
  - aaaa
- The odd frequency characters can be filled in pairs.
  - aaabb
  - aabbb
- Proof: (due to key observation mentioned above)
- Time complexity : O(2 * n).

Hull Sum: Kamil Debowski (25 AC)

- If we find a valid set of points for N = 50, any of its subsets will be valid answers for smaller N’s.
- So solve for N = 50.

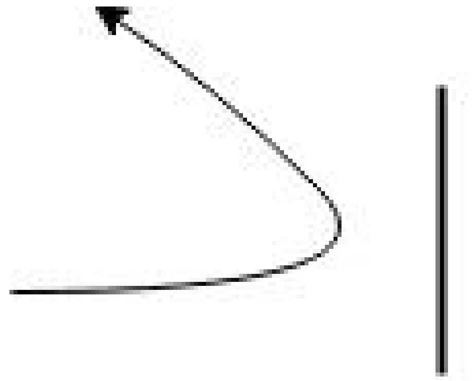
- Estimate the average number of points that should be on the convex hull.
- It will be 4 * 10^15 / (2^50) = 3.55. On average if we manage to get 3 or (sometimes may be 4) on the convex hull’s, we are done .

Hull Sum



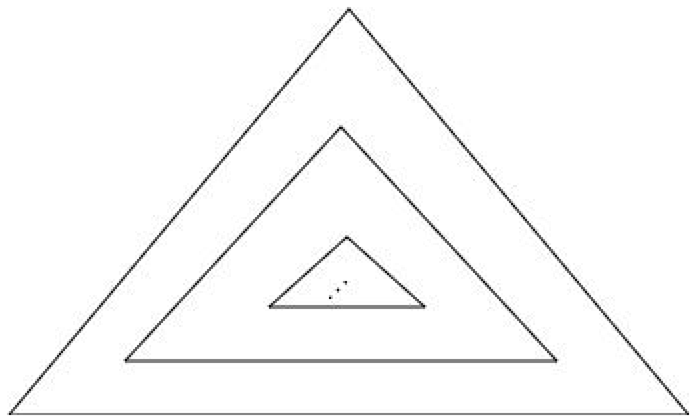
- We have 3 points on the convex hull for each subset of points.
- How to avoid collinearity?
- Perturb the points a little bit one by one so that the points are not collinear and you still have “almost” similar kind of structure.

Hull Sum (Another solution)



- To avoid the collinearity on the left two parts of the triangle, we can convert it to parabola.
- Now perturb the points on the line so as to avoid collinearity.

Hull Sum (Another solution)



- Create nested triangles using the similar perturbing idea.
- One way of doing this to avoid collinearity is to create 3-4 nesting of the triangles and then put the remaining vertices randomly inside the inner triangle..

AND Graph

Alexey Zayakin (9 AC)

- s = 000101, t = 101011
- Let K be the number of leading zeros in s.
  Example path from s to t can be as follows.
  0000101
  0001010
  0100101
  1001010
- You can see that the leading 1 in the number always moves to the left. Or if you go from t to s, it moves to the right (till K steps).
- Thus, length of longest path can be at most K.

AND Graph

- We build the path from t and go to s using dynamic programming idea..
- We go from left to right and build the path from t to s. We maintain the length of our path built till now. Assume we are i-th position, and len denotes the length of the path built till now. We can decide to not create a new vertex in the path and go to dp(i + 1, len).
- The number of ways filling the i-th bit (in all the the vertices added till now) can be found using fibonacci numbers (a bit different initial values).
- dp(i, len): current at i-th position in string s or t, and len denotes the number of edges till now in the path.

AND Graph

- Recurrence Relation:

  ```
  // For i < K.
      We can fill the i-th digit by either 0 or a 1.
      Fill by zero: dp(i, len) -> dp(i + 1, len) * fib[len + 3 - (t[i] == ‘1’);
      Fill by one: dp(i, len) -> dp(i + 1, len + 1) * fib[len + 3 - (s[i] == ‘1’) - 1];

  // For i >= K.
      Now, the len can’t increase.
      dp(i, len) -> dp(i + 1, len) * fib[len + 3 - (s[i] == ‘1’) - (t[i] == ‘1’))
  ```

- Time Complexity and space complexity will be O(N * N) where N = |t|

AND Graph

- If you from s to t, there is an O(n^3) solution using an almost similar dp. In this approach, you will have to kind of know the final length of the path beforehand so that you can find number of ways of filling the i-th digit, because it’s number of ways of filling depend on how many numbers are filled till now, and how many are yet to fill.

Minimax

Kamil Debowski (31 AC)

- $\max(r_i) \leq \min(C_j)$
  - because $r_i \leq t[i][j] \leq C_j$
- The desired equality isn't true only if $\max(r_i) < \min(C_j)$, and then we must increase $\max(r_i)$ or decrease $\min(C_i)$, or do both. In order to increase $\max(r_i)$, it's enough to change elements in one row only (increase r_i for one i). Similarly, to decrease $\min(C_i)$, it's enough to change elements in one column.
- Let's say we know the value $X = \max(r_i) = \min(C_j)$ in the optimal solution. Then we should choose one row and change to X all numbers smaller than X, and similarly choose one column and change to X all numbers greater than X.

Minimax

- Slow approach: iterate over each pair of row and column. For each value X occurring in the row or the column, count the cost of the row as the number of elements smaller than X, and the cost of the column as the number of elements greater than X. Consider the sum of those two costs as the answer.

- Intended $O(n^2 \log(n))$ solution: sort all n^2 values and process them in the increasing order. In each moment, for the current value X, we should know costs of all rows and costs of all columns. The possible answer is the sum of costs of the best row and the best column.

Fusing Weapons

Hasan Jaddouh (17 AC)

- An interval is *good* if it can be changed into a single number.
- Let's prove that there are O(n log(n)) good intervals.
- Let x denote the minimum value in the circle, and let's find any maximal block with x's. Unless the whole circle consists of x's (we'll consider that later), the block is surrounded by bigger numbers from both sides.
  - Lemma: Every good interval contains an even number of x's from the block.
    - Intervals inside the block (we don't care).
    - Intervals outside the block (trivially).
    - Intervals containing the whole block.
    - Intervals containing the prefix or the suffix of the block (and those Containing both prefix and suffix - because we're on a circle).

Fusing Weapons

- Let L denote the length of the block. If L is even, let’s replace the block with L/2 numbers x+1. It turns out that almost all good intervals are still valid and good. We only destroyed some of O(L log(L)) good intervals completely inside the block. The proof uses the previous lemma.
- If L is odd, let’s replace the block with (L-1)/2 numbers x+1 (it might mean 0 numbers x+1). Apart from destroying some of O(L log(L)) good intervals inside the block, we possibly created some new good intervals, but we don’t care about it. We’re showing that there are O(N log(N)) good intervals, even including the new ones created in the process.

Fusing Weapons

- We repeat the process till we get the circle with equal numbers, which means O(L log(L)) good intervals (it's a case mentioned two slides ago). Before that, in each step we decreased the number of elements by some c, while removing O(c log(c)) good intervals. This sums up to O(N log(N)) good intervals in total, which ends the proof.

- We can use the proved fact by iterating over all good intervals. For each value X, we store all good intervals that can be merged to a single X, and then for each such interval [i, j], we brutally merge it with all intervals starting in j+1. The complexity is O(N log(N)).

Racing Refuels

Kamil Debowski (6 AC)

- We don't visit a pit stop if we still have some fuel.
- If we once add X units of fuel, and once Y units of fuel, then we want |X-Y| ≤ 1.
  - Otherwise, e.g. let's say that X ≥ Y + 2. Then we can replace X and Y with X-1 and Y+1.
- If there are R refuels, it's optimal to add floor(K/(R+1)) or ceil(K/(R+1)) units of fuel each time.
- Ternary search the number of refuels R.
  - To prove the correctness, ...

Year 3017 (UNIVERSE)

Hasan Jaddouh (19 AC)

- Let us consider the graph where there is a vertex for each planet in a universe, the edges are the tree edges and the teleports.
- The graph has a lot of vertices and edges. Directly finding shortest path won’t work here.
- We should somehow create a graph that preserves the distances and has fewer vertices and edges

.

Year 3017 (UNIVERSE)

- Let us consider a single universe.
- Let us consider the tree corresponding to planets and their distances in the universe.
- Let us find the set of “good” vertices, the vertices that are part of some teleports, and also included the vertices that are part of the queries.
- We want to build an auxiliary tree over these set of “good” vertices that preserve the distances of this tree.

Year 3017 (UNIVERSE) Building Auxiliary Tree

Computing Vertex Set:

- The auxiliary tree will have all the “good” vertices.
- For each pair of “good” vertices (u, v), we also add the vertex lca(u, v) into the auxiliary tree.

- There can be at most **O(L)** possible LCA’s if there are **O(L)** interesting vertices.
- Sort the "interesting" points into pre order. For each pair of consecutive vertices u, v in the the order, add lca(u, v) into the set of "interesting" points.

Year 3017 (UNIVERSE) Building Auxiliary Tree

Computing Edge Set:

- For each vertex u in the vertex set of auxiliary tree, find the closest ancestor in the original tree that belongs to vertex set of auxiliary tree, and add an edge between them.

- You can see that for each pair of vertices u, v in the “good” vertices, the edge (u, v) have been replaced by (u, lca(u, v)) and (v, lca(u, v)) in the auxiliary tree. As discussed above, the lca(u, v) is part of auxiliary tree for each pair (u, v).

Year 3017 (UNIVERSE)

Final Solution:

- Build the auxiliary trees for each of the universes.
- Make sure to not rebuild auxiliary tree for each of queries.
- Then find the shortest path between given two pair of nodes by using Dijkstra’s algorithm.

BiCycles

Alei Reyes (3 AC)

- Any **d** regular bipartite graph will have **d** disjoint perfect matchings.

  Proof:

  - By using hall's theorem, we say that it should have at least one perfect matching.
  - Remove the edges of the matching, now the remaining graph is d-1 regular.
  - Use this inductive argument over d.

BiCycles

- Find the three perfect matchings. For example, you can use Kuhn’s bipartite matching algorithm.
- Color the edges of the matchings by red, green, blue respectively.

- For each pair of colors, create a graph with these edges only.
- Degree of each vertex is 2 in this graph.
- You can prove that this graph will be a collection of disjoint cycles.
- So, find these cycles and print them

- Overall, you will have 2 disjoint cycles for each edge.

## Matrix Sum		Alexey Zayakin and Evgeny Vihrov (1 AC)

- The answer is at most 3, except for corner cases N = 1 or M = 1.
- The answer is 1 if and only if there is a single CC (connected component) consisting of 1’s.
- The answer can be 2 even if there is a big number of CC’s, for example:

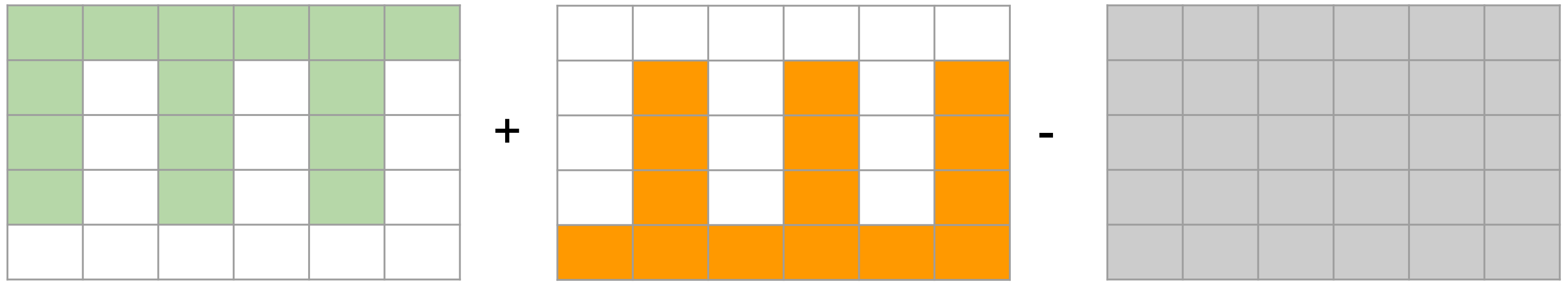
	1				
			1		1
1					
		1		1	

Matrix Sum

- The answer is 2 either for two separate CC’s of 1’s, or for a big CC of 1’s, with subtracted smaller CC, like in the previous slide. To deal with both cases, for each CC we can consider flipping its values (changing 0’s to 1’s or the other way around) and checking whether the new matrix has answer 1. Constraints were small, so you didn’t have to do it very efficiently. The complexity $O((NM)^2)$ was enough.
- Now we will show that for $N \geq 2$ and $M \geq 2$ we can always construct a solution with 3 matrices.

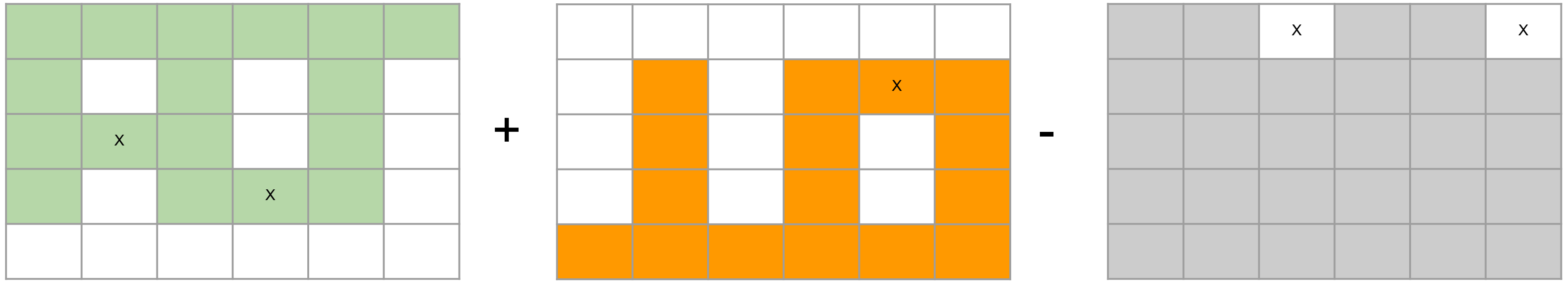
Matrix Sum

The following three matrices produce a matrix with 0’s only, and it’s quite easy to change them to obtain some 1’s in the final matrix.



Matrix Sum

If we want to obtain a 1 in the first or the last row, we should just remove a 1 from the corresponding cell of the third matrix. And if we want to obtain a 1 in some other row, we should just add this cell in one of the first two matrices - the one that doesn't contain it initially.



Minimum of Degrees

Kamil Debowski (0 AC)

The answer is just:

$$\frac{n(n-1)(n-2)}{4}\left(1-\frac{2}{2n-3}\sum_{i=0}^{2n-5}\frac{\binom{n-3}{\lfloor i/2\rfloor}\binom{n-3}{\lceil i/2\rceil}}{\binom{2n-4}{i+1}}\right)$$

Minimum of Degrees

Now more seriously...

- Compute the EV for one edge, and multiply by the number of edges N*(N-1)/2.
- For one fixed edge, we care only about 2N-3 edges incident to the two vertices.
- Equivalent problem: We shuffle letters in a string consisting of N-2 letters ‘a’, N-2 letters ‘b’ and one letter ‘s’. What is the EV of the minimum of numbers of ‘a’ and ‘b’ before the letter ‘s’? For further simplicity, we can remove the letter ‘s’ and say that we’re looking for the EV in the random prefix (or equivalently: the sum of the EV for all prefixes).

Minimum of Degrees

- For simplicity, let’s replace N-2 with N. (We have N letters ‘a’ and N letters ‘b’.)

- For fixed prefix length S, there are C(2N, S) ways to choose which of 2N letters should be in the prefix. Among them, there are C(N, i) * C(N, S-i) ways to do it so that there would be exactly i letters ‘a’. So we want to compute something like:

$$\sum_{i=0}^{S/2} \frac{i \cdot \binom{N}{i} \cdot \binom{N}{S-i}}{\binom{2N}{S}}$$

Minimum of Degrees

- After removing a constant we have:

$$\sum_{i=0}^{S/2} i \cdot \binom{N}{i} \cdot \binom{N}{S-i} = \sum_{i=0}^{S/2} \frac{N!}{(i-1)! \cdot (N-i)!} \cdot \binom{N}{S-i}$$

$$= N \cdot \sum_{i=0}^{S/2} \binom{N-1}{i-1} \cdot \binom{N}{S-i} = N \cdot \sum_{i=0}^{S/2} \binom{N-1}{i-1} \cdot \left(\binom{N-1}{S-i-1} + \binom{N-1}{S-i} \right)$$

Minimum of Degrees

- Now the “half trick”.

$$\sum_{i=0}^{k} \binom{2k}{i} = 2^{2k-1} + \frac{1}{2} \cdot \binom{2k}{k}$$

$$\sum_{i=0}^{k} \binom{2k}{i} \approx 2^{2k-1}$$

$$\sum_{i=0}^{s} \binom{n}{i} \cdot \binom{n}{2s-i} \approx \frac{1}{2} \cdot \binom{2n}{2s}$$