Spring 2024
UCF Training Camp Contest #1

Christian Lim
Thursday, March 28, 2024

Difficulty Order, by CodeForces rating

- E: **1300** (8/8, 12 minutes +)
- B: **1800** (7/8, 18 minutes +)
- F: **2300** (5/8, 45 minutes +1)
- C: **2600** (2/8, 65 minutes +)
- D: **2800** (0/8)
- A: **3500** (0/8)

[2800] D: Design Tutorial: Increase the Constraints

- Let's start with a simpler task
  - We have string A and B (|A|, |B| <= n) and q queries.
  - Each query asks the Hamming distance between A and a substring of B with length equals to |A|.
- **How to solve this?**

[2800] D: Design Tutorial: Increase the Constraints

- Let's start with a simpler task
  - We have string A and B (|A|, |B| <= n) and q queries.
  - Each query asks the Hamming distance between A and a substring of B with length equals to |A|.
- How to solve this? **Convolution => FFT**

[2800] D: Design Tutorial: Increase the Constraints

- Let's start with a simpler task
    - We have string A and B (|A|, |B| <= n) and q queries.
    - Each query asks the Hamming distance between A and a substring of B with length equals to |A|.
- How to solve this? Convolution => FFT
    - Use **+1** to replace '1' and use **-1** to replace '0'
    - Do the convolution of A and reverse of B.
    - We can extract the answer of all possible query
- How could we use this to help us?

[2800] D: Design Tutorial: Increase the Constraints

- **Punchline:**

[2800] D: Design Tutorial: Increase the Constraints

- **Punchline: Square Root Decomposition**

[2800] D: Design Tutorial: Increase the Constraints

- **Punchline: Square Root Decomposition**
  - Divide A into L blocks. For each block, we pre-compute the convolution of this part with B. This takes O(L n log n).
  - Now, for each query:
    - Use the pre-calculated results to speedup
    - Each query needs O(L) since each block takes a constant lookup!
  - If n = q then this solution can run in $O((n^* \log n)^{1.5})$.

[2800] D: Design Tutorial: Increase the Constraints

- For this problem, a bit speedup can work:
  - __builtin_popcount() can work, but “cnt[x<<16] + cnt[x>>16]” can work even better!

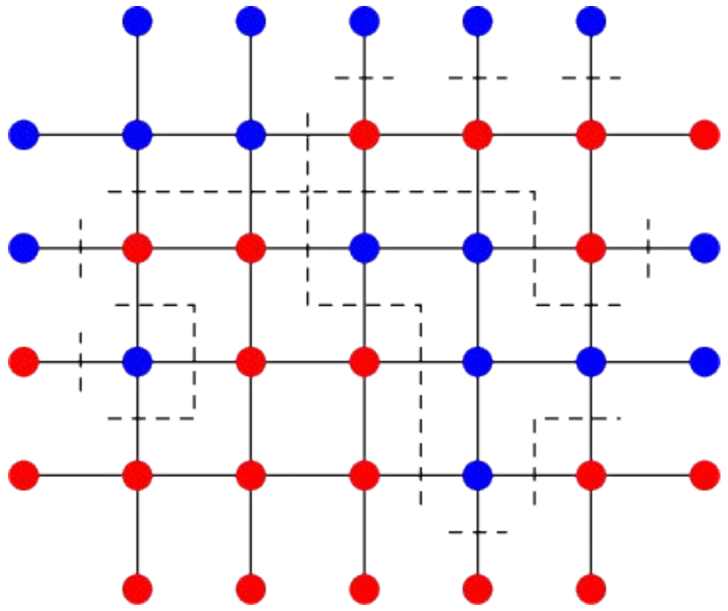
[3500] A: Breadboard Capacity (hard version)

- Find the **maximum flow** value from red ports to blue ports in the grid network. All edges are bidirectional with a capacity 1.

- By Ford-Fulkerson, we know **(maximum flow) = (minimum cut)**
  - The real task is to paint each node inside the grid red or blue so that the number of edges connecting differently colored nodes is as smallest possible.

[3500] A: Breadboard Capacity (hard version)

- For a given coloring, let us construct a cut in the **dual graph** by connecting the faces separated by multi-colored edges.

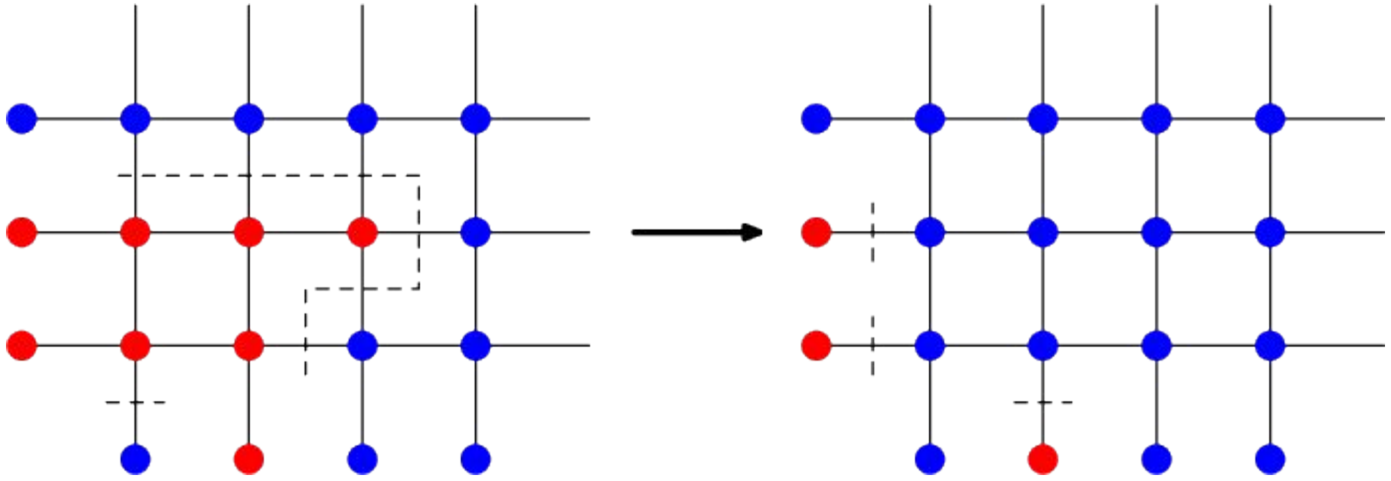


[3500] A: Breadboard Capacity (hard version)

- The following actions modify the cut without increasing its capacity (the number of crossed edges):
  - Interior of any cycle can be recolored, which makes the cycle disappear.
  - If a path connects a pair of adjacent sides, we may get rid of the path and instead cut/uncut ports in the corner separated by the path.
  - A path connecting opposite sides can be transformed into a straight segment, possibly with cutting/uncutting some ports.

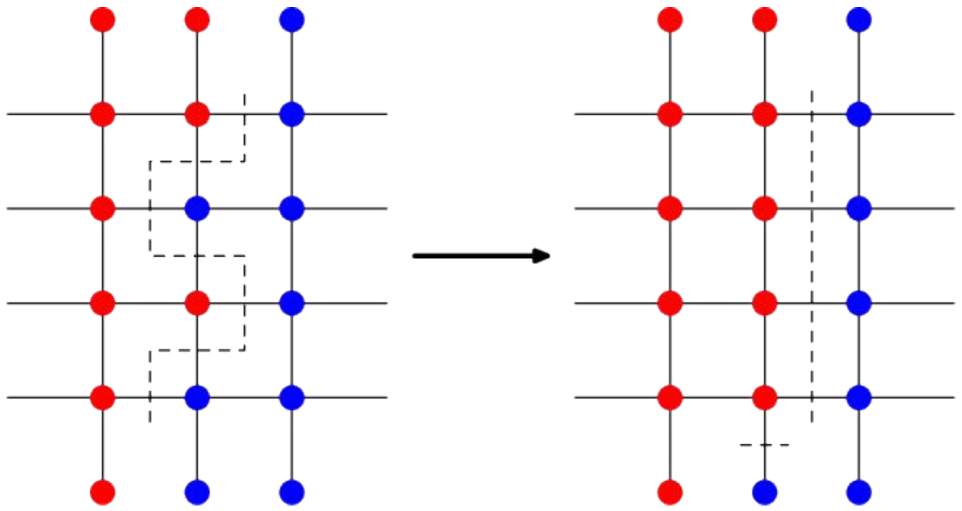
[3500] A: Breadboard Capacity (hard version)

- If a path connects a pair of adjacent sides, we may get rid of the path and instead cut/uncut ports in the corner separated by the path.



[3500] A: Breadboard Capacity (hard version)

- A path connecting opposite sides can be transformed into a straight segment, possibly with cutting/uncutting some ports.



[3500] A: Breadboard Capacity (hard version)

- Applying these operations to any minimum cut, we can obtain a minimum cut that only consists of **port cuts** and **straight segments parallel to one of the sides**.

- **port cut** is equivalent to recoloring of that port; it contributes **1** towards the cut capacity.
- **straight segment** contributes n or m to the cut capacity depending on the orientation.

[3500] A: Breadboard Capacity (hard version)

- For easy version (without modification query), we can just do a **linear DP**.
    - Choose the direction of straight cuts (say, vertical).
    - All ports along each vertical side should be recolored to the same color.
    - Proceeding in the horizontal direction we may decide to recolor ports adjacent to horizontal sides and/or to make a straight vertical cut.
    - Make sure that each connected part between the cuts has uniform color.
    - The only extra parameter is the color immediately behind the current vertical line.

[3500] A: Breadboard Capacity (hard version)

- For hard version, combine this DP with **segment tree with lazy propagation**.
- We will store a separate segment tree for each direction of straight cuts.
- For vertical cuts, a node [L, R) should store costs to properly recolor and/or make straight cuts in the horizontal range [L, R) so that the leftmost/rightmost nodes are red/blue (all four options).
- When calculating the answer, take fixing vertical sides into account.
- Merging the cost values from two halves of a node segment follows directly from the original DP recalculation.

[3500] A: Breadboard Capacity (hard version)

- To be able to make range flips fast enough, we need to store four more values: the answers assuming that the opposite sides take opposite colors instead of the same colors.
- Now to flip a node in the segment tree simply exchange the correct values with the opposite ones.
- Each modification query takes O(log n + log m)!