764. Largest Plus Sign

DescriptionHintsSubmissionsDiscussSolution

- Difficulty:Medium
- Total Accepted:1.1K
- Total Submissions:3.5K
- Contributor:awice
-
- Subscribe to see which companies asked this question.

Related Topics Dynamic Programming

In a 2D `grid` from (0, 0) to (N-1, N-1), every cell contains a `1`, except those cells in the given list `mines` which are `0`. What is the largest axis-aligned plus sign of `1`s contained in the grid? Return the order of the plus sign. If there is none, return 0.

An "*axis-aligned plus sign of `1`s* of order **k**" has some center `grid[x][y] = 1` along with 4 arms of length `k-1` going up, down, left, and right, and made of `1`s. This is demonstrated in the diagrams below. Note that there could be `0`s or `1`s beyond the arms of the plus sign, only the relevant area of the plus sign is checked for 1s.

Examples of Axis-Aligned Plus Signs of Order k:

```
Order 1:
000
010
000

Order 2:
00000
00100
01110
00100
00000

Order 3:
0000000
0001000
0001000
0111110
0001000
0001000
0000000
```

Example 1:

```
Input: N = 5, mines = [[4, 2]]
Output: 2
Explanation:
11111
11111
11111
11111
```

```
11011
In the above grid, the largest plus sign can only be order 2.  One of them is
marked in bold.
```

Example 2:

```
Input: N = 2, mines = []
Output: 1
Explanation:
There is no plus sign of order 2, but there is of order 1.
```

Example 3:

```
Input: N = 1, mines = [[0, 0]]
Output: 0
Explanation:
There is no plus sign, so return 0.
```

Note:

1. `N` will be an integer in the range `[1, 500]`.
2. `mines` will have length at most `5000`.
3. `mines[i]` will be length 2 and consist of integers in the range `[0, N-1]`.
4. *(Additionally, programs submitted in C, C++, or C# will be judged with a slightly smaller time limit.)*

Intuition

How can we improve our bruteforce? One way is to try to speed up the inner loop involving k, the order of the candidate plus sign. If we knew the longest possible arm length (Lu,Ll,Ld,Lr) in each direction from a center, we could know the order min(Lu,Ll,Ld,Lr) of a plus sign at that center. We could find these lengths separately using dynamic programming.

Algorithm

For each (cardinal) direction, and for each coordinate (r, c) let's compute the count of that coordinate: the longest line of '1's starting from (r, c) and going in that direction. With dynamic programming, it is either 0 if grid[r][c] is zero, else it is 1 plus the count of the coordinate in the same direction. For example, if the direction is left and we have a row like 01110110, the corresponding count values are 01230120, and the integers are either 1 more than their successor, or 0. For each square, we want dp[r][c] to end up being the minimum of the 4 possible counts. At the end, we take the maximum value in dp.

```
int orderOfLargestPlusSign(int N, vector<vector<int>>& mines) {
    unordered_set<int> bannedMines;
    for(vector<int> elem:mines)
    {
            int pos = elem[0]*N+elem[1];
            bannedMines.insert(pos);
      }
```

```
int dp[N][N];
memset(dp,0,sizeof(dp));
int ans = 0; int count = 0;
for(int r=0;r<N;r++)
{
      count=0;
      for(int c=0;c<N;++c)
      {
            int tmp = r*N+c;
            count = bannedMines.count(tmp)?0:count+1;
            dp[r][c]=count;
      }
      count=0;
      for(int c=N-1;c>=0;c--)
      {
            int tmp = r*N+c;
            count = bannedMines.count(tmp)?0:count+1;
            dp[r][c]=min(dp[r][c],count);
      }
}

for(int c=0;c<N;c++)
{
      count = 0;
      for(int r=0;r<N;++r)
      {
            int tmp = r*N+c;
            count = bannedMines.count(tmp)?0:count+1;
            dp[r][c]=min(dp[r][c],count);
      }
      count=0;
      for(int r=N-1;r>=0;r--)
      {
            int tmp = r*N+c;
            count = bannedMines.count(tmp)?0:count+1;
            dp[r][c]=min(dp[r][c],count);
            ans = max(ans,dp[r][c]);
      }
```

```
        }
        return ans;
    }
```