# Counting Flags

Timothy Gao

You are given a $N(1 \leq N \leq 1000)$ by $M(1 \leq M \leq 1000)$ grid of binary 0/1 values. Define a ”flag” as an right triangle with equal legs. For example,

images here

Count the number of flags made by 1s in the grid.

## 1 Initial Observations

Define angle point as the cell where two legs of the flag intersect.

Let's start with a brute force solution. We iterate over all angle points, trying all lengths $0...N$, then for each length we scan in $N^2$ whether angle point extending out by length makes a flag. This runs for $\mathcal{O}(N^5)$, but it's a good starting point.

One way we to optimize this brute force solution is by noticing that as we iterate through the lengths, we don't need to scan the entire area but only the new cells added.

[insert image]

With this, we can optimize our brute force down to $\mathcal{O}(N^4)$. The key idea here is that instead of counting all cells for each new angle point and length, we can utilize the results of our previous length and only count at most $N$ new cells. Let's see how we can extend this idea of ”building on previous results” further.

If we know that a triangle exists with side length $x$, then all triangles with lengths $1...x-1$ also exist with the same angle point.

## 2 Arriving at the Solution

Let dp[i][j] be the side length of the largest flag with angle point $(i, j)$.

$dp[i][j] = min(dp[i-1][j], dp[i][j-1])$

Note: An alternate solution exists for this problem utilizing a similar approach to The Largest Rectangle Problem, by employing binary search or prefix sums on each row as we iterate down.

## 3 Code

```
//@timothyg

#include <bits/stdc++.h>

using namespace std;

typedef long long ll;
typedef pair<int, int> pii;

#define pb push_back

#define f first
#define s second

const int maxn = 2005;
int matrix[maxn][maxn];
```

```cpp
int triangle[maxn];

int main(){
    ios_base::sync_with_stdio(false); cin.tie(0);

    int N, M, ones = 0; cin >> N >> M;
    for(int i = 0; i<N; i++){
        for(int j = 0; j<<M; j++){
            cin >> matrix[i][j];
            ones += matrix[i][j];
        }
    }

    vector<vector<int>> dp(N, vector<int>(M));
    for(int i = 0; i<N; i++){
        for(int j = 0; j<M; j++){
            if(matrix[i][j] == 0) continue;
            dp[i][j] = min(
                i == 0 ? 0 : matrix[i-1][j],
                j == 0 ? 0 : matrix[i][j-1]
            );
            dp[i][j]++;
            triangle[dp[i][j]]++;
        }
    }

    int Q; cin >> Q;
    while(Q--){
        int x; cin >> x;
        cout << triangle[x] << '\n';
    }

    return 0;
}
```