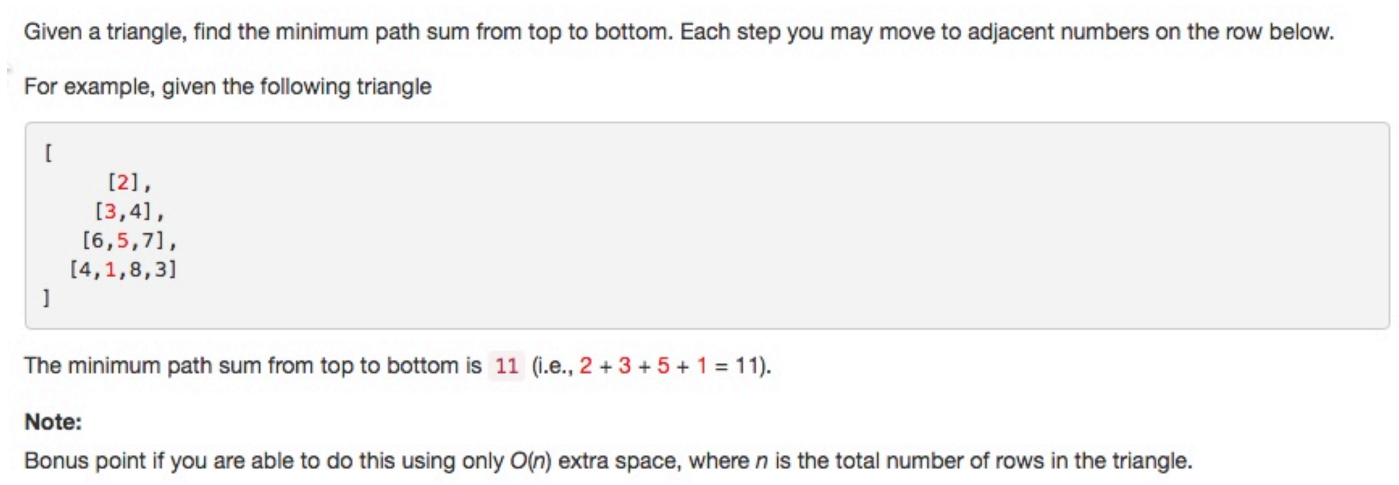
120. Triangle

120 Triangle

- **Dynamic Programming** + array

Description



Given a triangle, find the minimum path sum from top to bottom. Each step you may move to adjacent numbers on the row below.

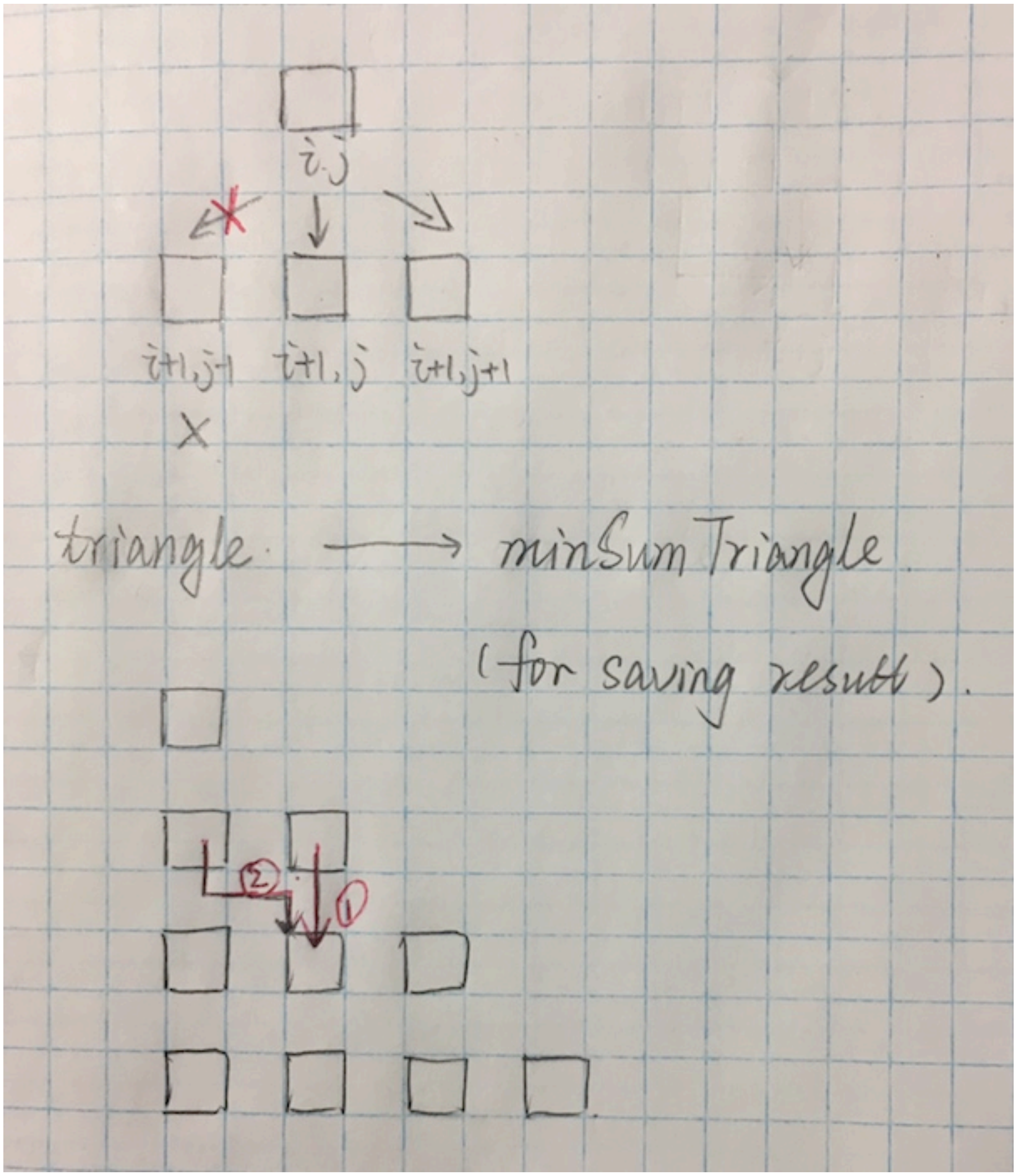
For example, given the following triangle

```
[
     [2],
    [3,4],
   [6,5,7],
  [4,1,8,3]
]
```

The minimum path sum from top to bottom is `11` (i.e., 2 + 3 + 5 + 1 = 11).

Note:
Bonus point if you are able to do this using only *O*(*n*) extra space, where *n* is the total number of rows in the triangle.

1. Thought line



2. Dynamic Programming+ array

```
class Solution {
public:
    int minimumTotal(vector<vector<int>>& triangle) {
        if (triangle.empty()) return 0;
        for (int i = 1; i<=triangle.size()-1; ++i){
            for (int j = 0; j<=i; ++j){
                int a = (i-1>=0 && j-1>=0) ? triangle[i][j] + triangle[i-1][j-1] : INT_MAX;
                int b = (i-1>=0 && j<=i-1) ? triangle[i][j] + triangle[i-1][j] : INT_MAX;
                triangle[i][j] = a<b ? a : b;
            }
        }
        sort(triangle[triangle.size()-1].begin(), triangle[triangle.size()-1].end());
        return triangle[triangle.size()-1][0];
    }
};
```