1514 Path With Maximum Probability

```cpp
//超时
class Solution {
public:
    double res = 0;
    bool seen = false;
    double maxProbability(int n, vector<vector<int>>& edges, vector<double>& succProb, int start, int end) {
        queue<int> myQueue;
        vector<vector<pair<int, double>>> adj(n);

        for(int i = 0; i < edges.size(); i++){
            adj[edges[i][0]].push_back(make_pair(edges[i][1], succProb[i]));
            adj[edges[i][1]].push_back(make_pair(edges[i][0], succProb[i]));
        }

        set<int> visited;
        dfs(start, end, start, adj, 1, visited);

        return seen  ? res : 0;
    }

    void dfs(int start, int end, int cur, vector<vector<pair<int, double>>>& adj, double prob, set<int>& visited){
        if(cur == end){
            seen = true;
            res = res > prob ? res : prob;
            return;
        }

        for(pair<int, double> p : adj[cur]){
            if(visited.find(p.first) == visited.end()){
                visited.insert(p.first);

                dfs(start, end, p.first, adj, prob * p.second, visited);

                visited.erase(p.first);
            }
        }
    }
};

```

1547 Minimum Cost to Cut a Stick

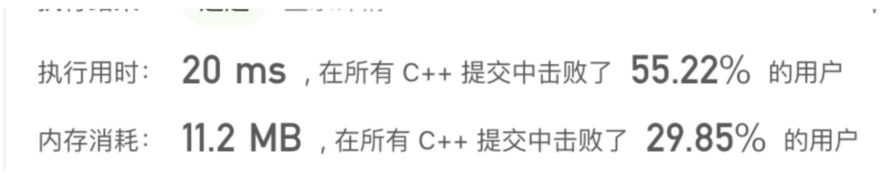
```
/*
    注意这里的 DP 代表的时候 dp[i][j] i ~ j 之间有几个切分点
    len 不代表长度，代表切分点个数
*/
class Solution {
public:
    int minCost(int n, vector<int>& cuts) {
        cuts.insert(cuts.begin(), 0);
        cuts.push_back(n);
        sort(cuts.begin(), cuts.end());
        int m = cuts.size();
        auto dp = vector<vector<int>>(m, vector<int>(m, INT_MAX));

        for(int i  = 0; i + 1 < m; i++)
            dp[i][i + 1] = 0;

        for(int len = 3; len <= m; len++){
            for(int i = 0; i + len - 1 < m; i++){
                int j = i + len - 1;
                for(int k = i + 1; k < j; k++){
                    dp[i][j] = std::min(dp[i][j], cuts[j] - cuts[i] + dp[i][k] +
dp[k][j]);
                }
            }
        }

        return dp[0][m - 1];
    }
};
```

1522 Diameter of N-Ary Tree



执行用时： 20 ms ，在所有 C++ 提交中击败了 55.22% 的用户

内存消耗： 11.2 MB ，在所有 C++ 提交中击败了 29.85% 的用户

```cpp
class Solution {
public:
    int res = 0;
    int diameter(Node* root) {
        dfs(root);
        return res;
    }

    int dfs(Node* root){
        if(root == nullptr || root->children.size() == 0)
            return 0;

        int curMax = 0;
        priority_queue<int, vector<int>, greater<>> pq;

        for(Node* child : root->children){
            int childLen = dfs(child);

            pq.push(childLen);
            if(pq.size() > 2)
                pq.pop();
        }

        if(pq.size() == 1){
            res = std::max(res, pq.top() + 1);
            return pq.top() + 1;
        }


        int first  = pq.top();    pq.pop();
        int second = pq.top();    pq.pop();
        res = std::max(res, first + second + 2);

        return std::max(first, second) + 1;
    }
};
```

1592 Rearrange Spaces Between Words

执行结果： 通过 显示详情 >　　　　　　　　　　添加

执行用时： 0 ms ，在所有 C++ 提交中击败了 100.00% 的用户

内存消耗： 6 MB ，在所有 C++ 提交中击败了 88.84% 的用户

炫耀一下：

```cpp
class Solution {
public:
    string reorderSpaces(string text) {
        int spaces = 0;

        vector<string> myStr;

        for(int i = 0; i < text.size();){
            if(text[i] == ' ') {
                while (i < text.size() && text[i] == ' ') {
                    spaces++;
                    i++;
                }
            }else{
                int right = i;
                while(right < text.size() && text[right] != ' '){
                    right++;
                }

                myStr.push_back(text.substr(i, right - i));
                i = right;
            }
        }



        int numOfWords = myStr.size();
        int numOfSpaceToAssign = 0;
        if(numOfWords != 1)
            numOfSpaceToAssign = spaces / (numOfWords - 1);
        else{
            numOfSpaceToAssign = spaces;
            string res = myStr[0];
            for(int j = 0; j < numOfSpaceToAssign; j++)
                res += ' ';
            return res;
        }

        string res;
        int index = 0;
        for(int i = 0; i < numOfWords; i++){
            res += myStr[index++];
            if(index == numOfWords)
                continue;

            for(int j = 0; j < numOfSpaceToAssign; j++)
                res += ' ';
        }

        for(int i = 0; i < spaces - (numOfWords - 1) * numOfSpaceToAssign; i++){
            res += ' ';
        }

        return res;
    }
};
```