1219 Path With Maximum Gold

执行结果: 通过 显示详情 >

执行用时: 72 ms , 在所有 C++ 提交中击败了 47.18% 的用户

内存消耗: 7.3 MB , 在所有 C++ 提交中击败了 40.11% 的用户

炫耀一下:

```cpp
class Solution {
public:
    vector<vector<int>> dir{{-1, 0},{0, -1}, {1, 0}, {0, 1}};
    int row, col;
    int res;
    int getMaximumGold(vector<vector<int>>& grid) {
        row = grid.size();
        col = grid[0].size();
        res = 0;

        for(int i = 0; i < row; i++){
            for(int j = 0; j < col; j++){
                if(grid[i][j] != 0)
                    dfs(i, j, grid, 0);
            }
        }

        return res;
    }

    void dfs(int i, int j, vector<vector<int>>& grid, int sum){
        int val = grid[i][j];
        grid[i][j] = 0;
        sum += val;

        res = max(res, sum);

        for(int k = 0; k < dir.size(); k++){
            int newX = i + dir[k][0];
            int newY = j + dir[k][1];

            if(isInRange(newX, newY) && grid[newX][newY] != 0){

                dfs(newX, newY, grid, sum);
            }
        }

```

```
        grid[i][j] = val;
//        sum -= val;
    }

    bool isInRange(int i, int j){
        return i >= 0 && j >= 0 && i < row && j < col;
    }
};
```

1235 Maximum Profit in Job Scheduling

```
    public int jobScheduling(int[] startTime, int[] endTime, int[] profit) {
        int len = startTime.length;
        int[][] time = new int[len][4];

        for(int i = 0; i < len; i++){
            time[i][0] = startTime[i];
            time[i][1] = endTime[i];
            time[i][2] = profit[i];
            time[i][3] = profit[i];
        }

        Arrays.sort(time, (o1, o2) -> (o1[1] - o2[1]));
        int maxProfit = 0;

        for(int i = 1; i < len; i++){
            time[i][3] = Math.max(time[i][2], time[i - 1][3]);
            for(int j = i  - 1; j >= 0; j--){
                if(time[j][1] <= time[i][0]){
                    time[i][3] = Math.max(time[i][3], time[j][3] + time[i][2]);
                    break;
                }
            }

            maxProfit = Math.max(maxProfit, time[i][3]);
        }

        return maxProfit;
    }

```

```java
//26 out of 27
public int jobScheduling(int[] startTime, int[] endTime, int[] profit) {
        int len = startTime.length;
        int[][] time = new int[len][4];

        for(int i = 0; i < len; i++){
            time[i][0] = startTime[i];
            time[i][1] = endTime[i];
            time[i][2] = profit[i];
            time[i][3] = profit[i];
        }

        Arrays.sort(time, (o1, o2) -> (o1[1] - o2[1]));
        int maxProfit = 0;

        for(int i = 1; i < len; i++){
            for(int j = 0; j < i; j++){
                if(time[j][1] <= time[i][0]){
                    time[i][3] = Math.max(time[i][3], time[j][3] + time[i][2]);
                }else{
                    break;
                }
            }

            maxProfit = Math.max(maxProfit, time[i][3]);
        }

        return maxProfit;
    }


```

1236 Web Crawler

```cpp
class Solution {
public:
    vector<string> crawl(string startUrl, HtmlParser htmlParser) {
        string region = "";
        for(int i = 7; i < startUrl.size(); i++){
```

```
            if(startUrl[i] == '/')
                break;
            region += startUrl[i];
        }

        unordered_set<string> set;
        queue<string> que;
        que.push(startUrl);
        set.insert(startUrl);
        vector<string> res;

        while(!que.empty()){
            int size = que.size();
            for(int i = 0; i < size; i++){
                string cur = que.front();
                que.pop();
                res.push_back(cur);

                vector<string> strs = htmlParser.getUrls(cur);
                for(string& str : strs){
                    int pos = str.find('/', 7);
                    if(set.count(str) == 0 && str.substr(7, pos - 7) == region){
                        que.push(str);
                        set.insert(str);
                    }
                }
            }
        }


        return res;
    }
};
```

1254 Number of Closed Islands

Runtime: 16 ms, faster than 30.17% of C++ online submi
Number of Closed Islands.

Memory Usage: 10.3 MB, less than 17.85% of C++ onlin
for Number of Closed Islands.

Next challenges:

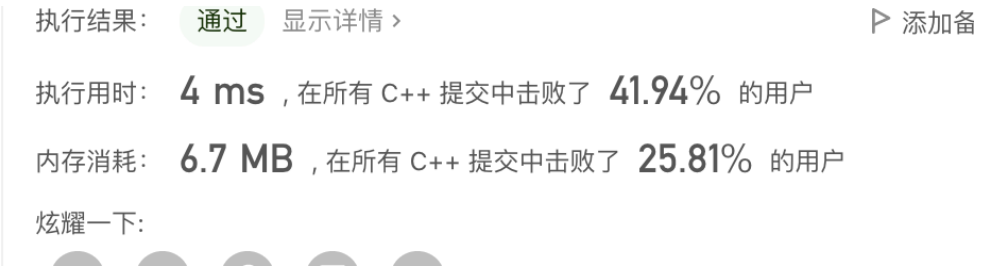
```
/*
```

```cpp
  Author: guoguo
  采用并查集，把周围边缘的点连接起来

  其他的点进行 dfs
*/
class WeightedUnionFind{
public:
    vector<int> id;
    vector<int> sz;

    WeightedUnionFind(int N){
        id = vector<int>(N, 0);
        sz = vector<int>(N, 1);
        for(int i = 0; i < N; i++){
            id[i] = i;
        }
    }

    void uni(int p, int q){
        int pRoot = find(p);
        int qRoot = find(q);

        if(pRoot == qRoot)
            return;

        if(sz[pRoot] > sz[qRoot]){
            sz[pRoot] += sz[qRoot];
            id[qRoot] = id[pRoot];
        }else{
            sz[qRoot] += sz[pRoot];
            id[pRoot] = id[qRoot];
        }
    }

    bool isConnected(int p, int q){
//        cout << p << " " << q << endl;
//        cout << sz.size() << endl;
        return find(p) == find(q);
    }

    int find(int p){
        while(p != id[p]){
            id[p] = id[id[p]];
            p = id[p];
        }

        return p;
    }
};
```

```
51
52  class Solution {
53  public:
54      int row;
55      int col;
56      vector<vector<int>> dir{{-1, 0}, {0, -1}, {1, 0}, {0, 1}};
57      int closedIsland(vector<vector<int>>& grid) {
58          row = grid.size();
59          col = grid[0].size();
60          WeightedUnionFind wuf = WeightedUnionFind((row + 1)* col + 1);
61
62          for(int i = 0; i < row; i++){
63              for(int j = 0; j < col; j++){
64                  if(grid[i][j] == 0 && (i == 0 || j == 0 || i == row - 1 || j ==
    col - 1)){
65  //                    cout << i << " " << j << endl;
66                      dfs(wuf, i, j, grid);
67                  }
68              }
69          }
70
71  //        cout << " " << endl;
72
73          int finalPoint = getCoor(row, col);
74          int count = 0;
75          for(int i = 0; i < row; i++){
76              for(int j = 0; j < col; j++){
77                  if(grid[i][j] == 0 && !wuf.isConnected(finalPoint, getCoor(i, j)))
    {
78                      dfs(grid, i, j) ;
79                      count++;
80                  }
81              }
82          }
83
84          return count;
85      }
86
87      void dfs(vector<vector<int>>& grid, int i, int j){
88          grid[i][j] = -1;
89
90          for(int k = 0; k < 4; k++){
91              int newX = i + dir[k][0];
92              int newY = j + dir[k][1];
93
94              if(isInRange(newX, newY) && grid[newX][newY] == 0)
95                  dfs(grid, newX, newY);
96          }
97      }
```

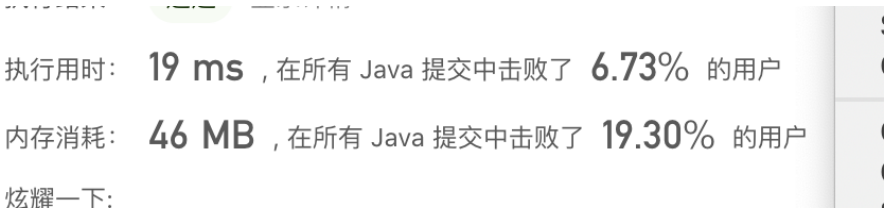
```
98
99      int getCoor(int i, int j){
100         return i * col + j;
101     }
102
103     void dfs(WeightedUnionFind& wuf, int i, int j, vector<vector<int>>& grid){
104 //        cout << "IN DFS" << i << " " << j << endl;
105         if(wuf.isConnected(getCoor(row, col), getCoor(i, j)))
106             return;
107
108
109
110         wuf.uni(getCoor(row, col), getCoor(i, j));
111
112         for(int k = 0; k < 4; k++){
113             int newX = i + dir[k][0];
114             int newY = j + dir[k][1];
115
116             if(isInRange(newX, newY) && grid[newX][newY] == 0){
117                 dfs(wuf, newX, newY, grid);
118             }
119         }
120     }
121
122     bool isInRange(int i, int j){
123         return i >= 0 && j >= 0 && i < row && j < col;
124     }
125 };
126
```

1265 Print Immutable Linked List in Reverse



```cpp
class Solution {
public:
    void printLinkedListInReverse(ImmutableListNode* head) {
        if(head == nullptr)
            return;

        printLinkedListInReverse(head->getNext());
        head->printValue();
    }
};
```

1267 Count Servers That Communicate




```java
    public int countServers(int[][] grid) {
        Set<String> set = new HashSet<>();
        int row = grid.length;
        int col = grid[0].length;
        for(int i = 0; i < row; i++){
            Set<String> temp = new HashSet<>();
            for(int j = 0; j < col; j++){
                if(grid[i][j] == 1)
                    temp.add(i + "@" + j);
            }

            if(temp.size() >= 2)
                set.addAll(temp);
        }

        for(int j = 0; j < col; j++){
            Set<String> temp = new HashSet<>();
            for(int i = 0; i < row; i++){
                if(grid[i][j] == 1)
                    temp.add(i + "@" + j);
            }

```

```
            if(temp.size() >= 2)
                set.addAll(temp);
        }

        return set.size();
    }
```

1278 Palindorme Patitioning III

1278. Palindrome Partitioning III

难度 困难 82

You are given a string `s` containing lowercase letters and an integer `k`. You need to :

- First, change some characters of `s` to other lowercase English letters.
- Then divide `s` into `k` non-empty disjoint substrings such that each substring is a palindrome.

Return *the minimal number of characters that you need to change to divide the string*.

Example 1:

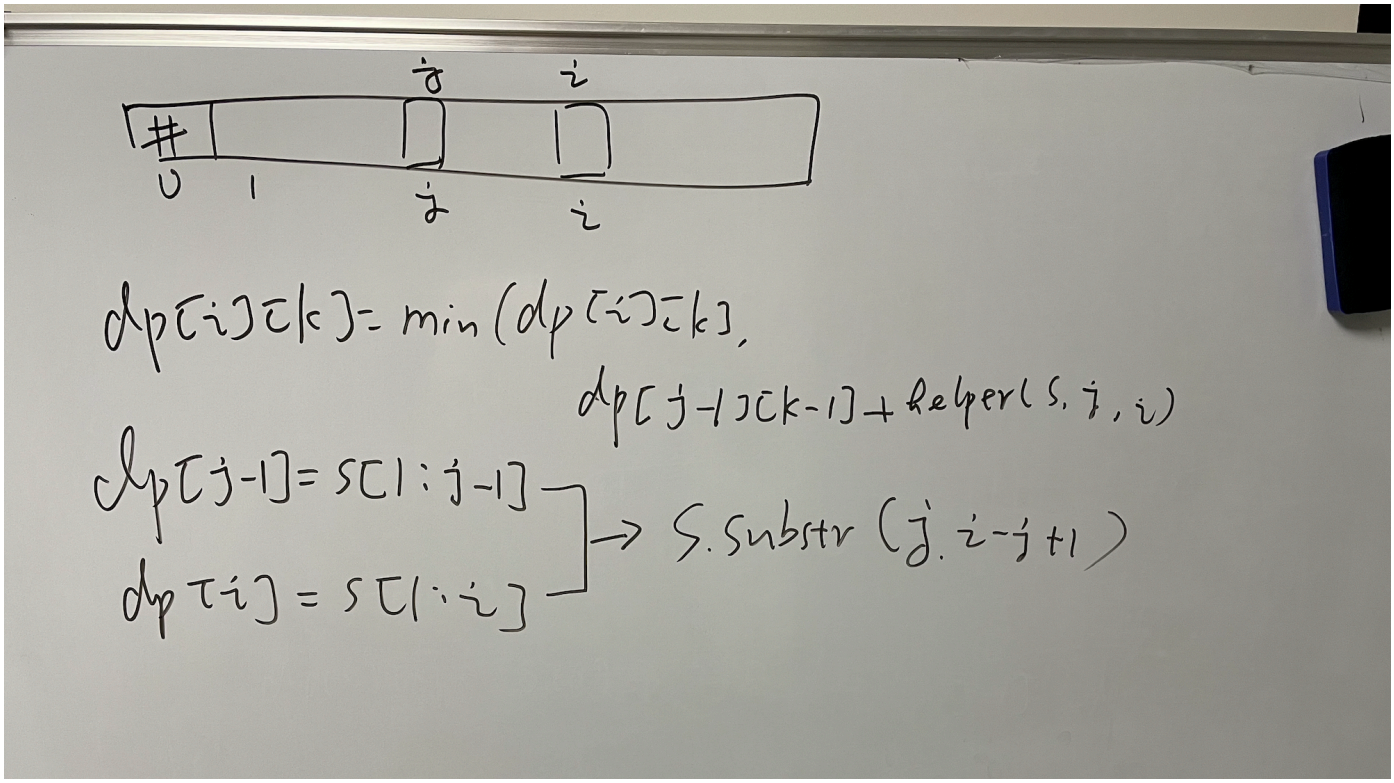
```
Input: s = "abc", k = 2
Output: 1
Explanation: You can split the string into "ab"
and "c", and change 1 character in "ab" to make it
palindrome.
```

Example 2:

```
Input: s = "aabbc", k = 3
Output: 0
Explanation: You can split the string into "aa",
"bb" and "c", all of them are palindrome.
```

```
/*
    dp 当中又嵌套 dp
*/
class Solution {
```

```
public:
    int palindromePartition(string s, int k) {
        int n = s.size();
        s = "@" + s;

        int K = k;
        auto dp = vector<vector<int>>(n + 1, vector<int>(k + 1, INT_MAX / 2));
        dp[0][0] = 0;
        for(int i = 1;  i <= n; i++){
            for(int k = 1; k <= min(K, i); k++){
                for(int j = k; j <= i;j++){
                    dp[i][k] = min(dp[i][k], dp[j - 1][k - 1] + helper(s, j, i));
                }
            }
        }

        return dp[n][K];
    }

    int helper(string& s, int i, int j){
        int res = 0;
        while(i < j){
            if(s[i] != s[j])
                res++;
            i++;
            j--;
        }

        return res;
    }
};
```




```cpp
/*
ref:
https://github.com/wisdompeak/LeetCode/tree/master/Dynamic_Programming/1278.Palindrome-Partitioning-III
    典型问题， 题目明确说需要 划分为 若干个连续子区间

    针对这种类型题目的套路，就是 设计 dp[i][k] 将 s[0:i] 分隔为 k个连续的区间

    实际上我个人认为， 对于这道，可以理解为 dp[i][k] means s[1:i]
*/
class Solution {
public:
    int palindromePartition(string s, int k) {
        int n = s.size();
        s = "@" + s;

        int K = k;
      //这个是典型的区间 DP， 快速拿到最少的数目转换
        auto count = vector<vector<int>>(n + 1, vector<int>(n + 1));
        for(int len = 1; len <= n; len++){
            for(int i = 1; i + len - 1 <= n; i++){
                int j = i + len - 1;
                if(len == 1)
                    count[i][j] = 0;
                else
                    count[i][j] = (s[i] == s[j] ? 0 : 1) + count[i + 1][j - 1];
            }
        }

```

```
        auto dp = vector<vector<int>>(n + 1, vector<int>(k + 1, INT_MAX / 2));
        dp[0][0] = 0;
      //因为 s 扩展了一个字符
        for(int i = 1;  i <= n; i++){
          //分隔的区间,  注意这里 最多也就 i 个,  因为总共就i个字符
            for(int k = 1; k <= min(K, i); k++){
              //注意这里的划分点,  参考下图
                for(int j = k; j <= i;j++){
                    dp[i][k] = min(dp[i][k], dp[j - 1][k - 1] + count[j][i]);
                }
            }
        }

        return dp[n][K];
    }

    int helper(string& s, int i, int j){
        int res = 0;
        while(i < j){
            if(s[i] != s[j])
                res++;
            i++;
            j--;
        }

        return res;
    }
};
```

1281 Subtract the product and Sum of Digits

执行用时: 5 ms , 在所有 Java 提交中击败了 6.13% 的用户

内存消耗: 35.4 MB , 在所有 Java 提交中击败了 16.80% 的用户

```
class Solution {
    public int subtractProductAndSum(int n) {
        String str = n + "";

        int res1 = 1, res2 = 0;
        for(char ch : str.toCharArray()){
            res1 *= (ch - '0');
            res2 += (ch - '0');
```

```
        }

        return res1 - res2;
    }
}

func subtractProductAndSum(n int) int {
  res1 := 1
  res2 := 0

  for ;n != 0; {
    digit := n % 10

    res1 *= digit
    res2 += digit

    n /= 10
  }

  return res1 - res2
}

```