【第二十六周】动态规划算法

120. 三角形最小路径和

给定一个三角形 `triangle` ，找出自顶向下的最小路径和。

每一步只能移动到下一行中相邻的结点上。**相邻的结点** 在这里指的是 **下标** 与 **上一层结点下标** 相同或者等于 **上一层结点下标 + 1** 的两个结点。也就是说，如果正位于当前行的下标 `i` ，那么下一步可以移动到下一行的下标 `i` 或 `i + 1` 。

示例 ：

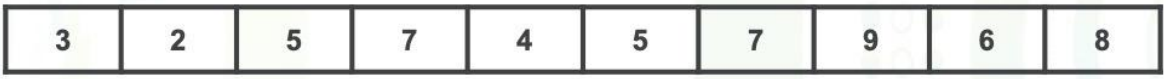
```
输入：triangle = [[2],[3,4],[6,5,7],[4,1,8,3]]
输出：11
解释：如下面简图所示：
2
3 4
6 5 7
4 1 8 3
自顶向下的最小路径和为 11（即，2 + 3 + 5 + 1 = 11）。
```

```
class Solution {
public:
    int minimumTotal(vector<vector<int>>& triangle) {
        int n = triangle.size();
        vector<vector<int>> dp;
        for (int i = 0; i < n; i++) dp.push_back(vector<int>(n));
        for (int i = 0; i < n; i++) {
            for (int j = 0; j <= i; j++) {
                dp[i][j] = INT_MAX;
            }
        }
        dp[0][0] = triangle[0][0];
        for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j <= i; j++) {
                dp[i + 1][j] = min(dp[i + 1][j], dp[i][j] + triangle[i + 1][j]);
                dp[i + 1][j + 1] = min(dp[i + 1][j + 1], dp[i][j] + triangle[i + 1][j
 + 1]);
            }
        }
        int ans = INT_MAX;
        for (auto x : dp[n - 1]) ans = min(ans, x);
        return ans;
    }
};
```

HZOJ44-最长上升子序列

题目描述

有一个数字序列，求其中最长严格上升子序列的长度



输入

输入一个数字n (1≤n≤1000000)(1≤n≤1000000)，代表数字序列的长度。

后跟 nn 个整数，第 ii 个整数 ai(1≤ai≤10000)ai(1≤ai≤10000)，代表数字序列中的第 ii 个值。

输出

输出一个整数，代表所求的最长严格上升子序列的长度。

样例输入

```
10
3 2 5 7 4 5 7 9 6 8
```

样例输出

```
5
```

```
/*************************************************************************
  > File Name: 2.HZOJ-44.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
```

```
#include <set>
#include <vector>
using namespace std;

int main() {
    int n;
    vector<int> val;
    cin >> n;
    for (int i = 0, a; i < n; i++) {
        cin >> a;
        val.push_back(a);
    }
    vector<int> dp(n);
    int ans = 0;
    for (int i = 0; i < n; i++) {
        dp[i] = 1;
        for (int j = 0; j < i; j++) {
            if (val[j] >= val[i]) continue;
            dp[i] = max(dp[i], dp[j] + 1);
        }
        ans = max(ans, dp[i]);
    }
    cout << ans << endl;
    return 0;
}
```

HZOJ45-最长公共子序列

题目描述

给出两个字符串，求其两个的最长公共子序列长度。

输入

第一行输入一个字符串s1，第二行输入一个字符串s2 (字符串长度≤1000)(字符串长度≤1000) ，两个字符串长度可以不相同。

输出

输出一个整数，代表两个字符串的最长公共子序列的长度。

样例输入1

```
sehuaizexi
yhaizeyiux
```

样例输出1

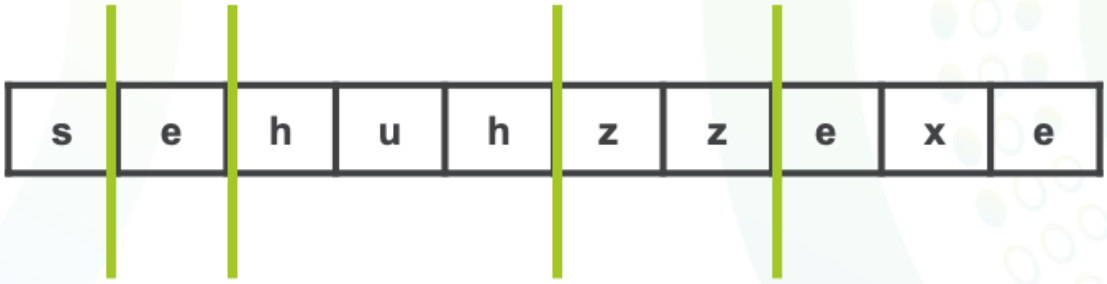
```
6
```

```
/*************************************************************************
  > File Name: 3.HZOJ-45.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
#include <cstring>
using namespace std;

int main() {
    string s1, s2;
    cin >> s1 >> s2;
    int n = s1.size(), m = s2.size();
    int dp[n + 1][m + 1];
    memset(dp, 0, sizeof(dp));
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= m; j++) {
            dp[i][j] = max(dp[i - 1][j], dp[i][j - 1]);
            if (s1[i - 1] == s2[j - 1]) {
                dp[i][j] = max(dp[i][j], dp[i - 1][j - 1] + 1);
            }
        }
    }
    cout << dp[n][m] << endl;
    return 0;
}
```

HZOJ46-切割回文

题目描述

给出一个字符串S，问对字符串S最少切几刀，使得分成的每一部分都是一个回文串（注意：单一字符是回文串）



输入

一个长度为n(1≤n≤500000)(1≤n≤500000)的字符串S，只包含小写字母。

输出

输出一个整数，代表所切的最少刀数。

样例输入

```
sehuhzzexe
```

样例输出

```
4
```

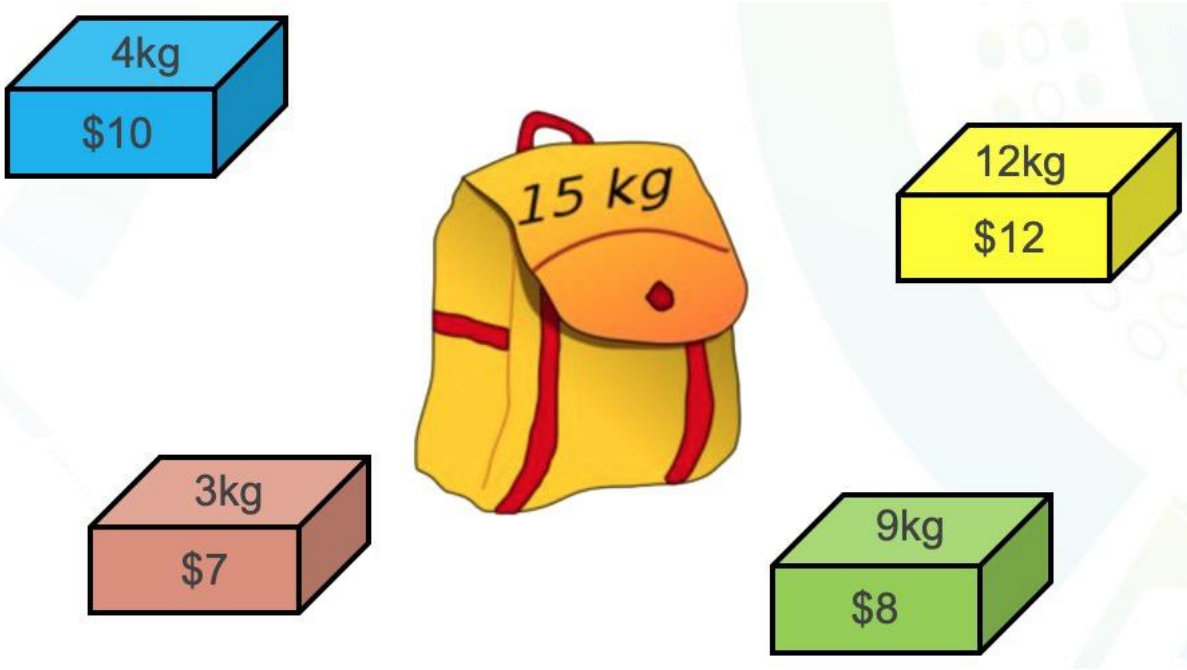
```
/*************************************************************************
  > File Name: 4.HZOJ-46.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
```

```
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

bool is_palindrome(string &s, int i, int j) {
    while (i <= j) {
        if (s[i] - s[j]) return false;
        ++i, --j;
    }
    return true;
}

int main() {
    string s;
    cin >> s;
    int n = s.size();
    vector<int> dp(n + 1);
    dp[0] = 0;
    for (int i = 1; i <= n; i++) {
        dp[i] = i;
        for (int j = 0; j < i; j++) {
            if (is_palindrome(s, j, i - 1)) {
                dp[i] = min(dp[j] + 1, dp[i]);
            }
        }
    }
    cout << dp[n] - 1 << endl;
    return 0;
}
```

HZOJ47-0/1背包

题目描述

给一个能承重VV的背包，和nn件物品，我们用重量和价值的二元组来表示一个物品，第ii件物品表示为（Vi，Wi）（Vi，Wi），问：在背包不超重的情况下，得到物品的最大价值是多少？



输入

第一行输入两个数 V,nV,n，分别代表背包的最大承重和物品数。

接下来nn行，每行两个数Vi,WiVi,Wi，分别代表第i件物品的重量和价值。

(Vi≤V≤10000,n≤100,Wi≤1000000)(Vi≤V≤10000,n≤100,Wi≤1000000)

输出

输出一个整数，代表在背包不超重情况下所装物品的最大价值。

样例输入1

```
15 4
4 10
3 7
12 12
9 8
```

样例输出1

```
19
```

```cpp
/*************************************************************************
  > File Name: 5.HZOJ-47.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

#define MAX_N 100
#define MAX_V 10000
int v[MAX_N + 5], w[MAX_N + 5];
int dp[MAX_N + 5][MAX_V + 5] = {0};

int main() {
    int V, n;
    cin >> V >> n;
    for (int i = 1; i <= n; i++) {
        cin >> v[i] >> w[i];
    }
    for (int i = 1; i <= n; i++) {
        for (int j = 0; j <= V; j++) {
            dp[i][j] = dp[i - 1][j];
            if (j >= v[i]) dp[i][j] = max(dp[i][j], dp[i - 1][j - v[i]] + w[i]);
        }
    }
    cout << dp[n][V] << endl;
    return 0;
}
```

```cpp
/*************************************************************************
  > File Name: 5.HZOJ-47.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
```

```cpp
#include <map>
#include <set>
#include <vector>
using namespace std;

#define MAX_N 100
#define MAX_V 10000
int v[MAX_N + 5], w[MAX_N + 5];
int dp[2][MAX_V + 5] = {0};

int main() {
    int V, n;
    cin >> V >> n;
    for (int i = 1; i <= n; i++) {
        cin >> v[i] >> w[i];
    }
    for (int i = 1; i <= n; i++) {
        int ind = i % 2;
        int pre_ind = !ind;
        for (int j = 0; j <= V; j++) {
            dp[ind][j] = dp[pre_ind][j];
            if (j >= v[i]) dp[ind][j] = max(dp[ind][j], dp[pre_ind][j - v[i]] + w[i]);
        }
    }
    cout << dp[n % 2][V] << endl;
    return 0;
}
```

```cpp
/*************************************************************************
  > File Name: 5.HZOJ-47.cpp
  > Author: huguang
  > Mail: hug@haizeix.com
  > Created Time:
 ************************************************************************/

#include <iostream>
#include <cstdio>
#include <cstdlib>
#include <queue>
#include <stack>
#include <algorithm>
#include <string>
#include <map>
#include <set>
#include <vector>
using namespace std;

#define MAX_V 10000
int dp[2][MAX_V + 5] = {0};
// int dp[MAX_V + 5] = {0};

int main() {
    int V, n, v, w;
    cin >> V >> n;
    for (int i = 1; i <= n; i++) {
```

```
        cin >> v >> w;
        int ind = i % 2;
        int pre_ind = !ind;
        for (int j = 0; j <= V; j++) {
            dp[ind][j] = dp[pre_ind][j];
            if (j >= v) dp[ind][j] = max(dp[ind][j], dp[pre_ind][j - v] + w);
        }
    }
    cout << dp[n % 2][V] << endl;
    return 0;
}
```