青少年软件编程（C 语言）等级考试试卷（六级）

古董

题目描述

拍卖师准备了 N*N* 件古董进行拍卖。这些古董有以下重要特性：

- **陈列规则**：所有古董按 1,…,N1,…,*N* 编号陈列在一条长廊中，每天拍卖师只能从长廊**任一端**取出一件古董拍卖。
- **升值效应**：古董拍卖顺序直接影响成交价。若第 i*i* 件古董在第 a*a* 天拍卖，成交价为 Vi×a*Vi*×*a*（Vi*Vi* 为初始估价）。
- **价值分布**：第 i*i* 件古董的初始估价 Vi*Vi*取决于其陈列位置——从入口端开始，第 i*i* 个展柜内的古董估价为 Vi*Vi*。

拍卖师需要制定最优拍卖顺序，最大化总成交额。请帮助他计算出古董全部售出后的最大收益。

输入格式

- 第一行：古董数量 N*N*
- 接下来 n*n* 行：古董初始估价序列 V1～VN*V1*～*VN*

输出格式

- 一行整数表示最大总收益

输入样例

```
5
1
3
1
5
2
```

输出样例

```
43
```

说明提示

【数据范围】

$1 \le N \le 2000$ $1 \le N \le 2000$

$1 \le Vi \le 1000$ $1 \le V_i \le 1000$

限制

时间限制：1000ms

内存限制：256MiB

参考代码：

#include<iostream>

int n;

long long v[2001];

```
long long s[2001];
long long mem[2001][2001];
bool filled[2001][2001];

long long solve(int begin, int end)
{
   int length = end - begin + 1;
   if (length == 0) return 0;
       if  (filled[begin][end])  return  mem[begin]
[end];
   long long head = solve(begin+1, end);
   long long tail = solve(begin, end-1);

   filled[begin][end] = true;
    return mem[begin][end]=std::max(head, tail)
+ s[end] - s[begin-1];
}

int main()
{
   std::cin >> n;
   for (int i = 1; i <= n; ++i)
```

```
    {
        std::cin >> v[i];
        s[i] = s[i-1] + v[i];
    }
    std::cout << solve(1, n) << "\n";
}
```

括号

题目描述

给定一个整数 nn。生成所有长度为 nn 的**合法括号序列**，并按字典序升序输出。

合法括号序列定义：

1. 空字符串是合法的。
2. 若字符串 ss 合法，则 (+s+)(+s+) 合法。
3. 若字符串 ss 和 tt 合法，则 s+ts+t 合法。

输入格式

输入一个整数 nn

输出格式

每行输出一个合法括号序列（按字典序升序）。若无解则不输出。

输入样例#1

```
2
```

输出样例#1

```
()
```

输入样例#2

```
4
```

输出样例#2

```
(())
()()
```

输入样例#3

```
6
```

输出样例#3

```
((()))
(()())
(())()
()(())
()()()
```

说明提示

【数据范围 】

- $1 \le N \le 20$

- N∧ 是偶数

限制

时间限制：1000ms

内存限制：256MiB

参考代码

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int INF = 1e9;

int n, a[100010], sum, pd;

int main(){

    cin >> n;
    for(int i = 0; i < 1 << n; i++){
        sum = 0, pd = 0;
        for(int j = 0; j < n; j++)
            if((i >> j) & 1) sum++;
            else{
                if(!sum) pd = 1;
```

```
            sum--;
        }
    if(!pd && !sum){
        for(int j = n - 1; j >= 0; j--)
            putchar((i >> j) & 1 ? ')' : '(');
        puts("");
    }
  }

  return 0;
}
```

搬运水果

题目描述

在果园里，nn 堆果实排成一个环形，第 ii 堆果实的重量为 aia_i。果农需要将所有果实合并成一堆。合并规则如下：

- 每次只能合并相邻的两堆，新堆的重量为两堆重量之和
- 每次合并消耗的体力等于新堆的重量
- 合并后新堆与剩余堆仍保持环形排列

请设计合并顺序，求出合并全过程消耗的最小总体力与最大总体力。

输入格式

第一行：整数 nn，表示果实堆数

第二行：nn 个整数 a1,a2,…,an$a_1,a_2,\ldots,a_n$，表示每堆果实的重量

输出格式

第一行：最小总体力消耗

第二行：最大总体力消耗

输入样例

```
4
4 5 9 4
```

输出样例

```
43
54
```

说明提示

【数据范围】

1≤n≤1001≤n≤100
1≤ai≤10001≤a_i≤1000

限制

时间限制：1000ms

内存限制：256MiB

参考代码

```
#include <iostream>
#include <cstring>
#include <climits>
using namespace std;

int n, a[205], s[205], f[205][205], g[205][205];
int ans1 = INT_MAX, ans2 = INT_MIN;

int main()
{
    memset(f, 0x3f, sizeof(f));
    cin >> n;
    for (int i = 1; i <= n; i++)
    {
        cin >> a[i];
        a[i + n] = a[i];
    }
    for (int i = 1; i <= 2 * n; i++)
    {
```

```
        s[i] = s[i - 1] + a[i];
        f[i][1] = g[i][1] = 0;
    }
    for (int len = 2; len <= n; len++)
        for (int i = 1; i + len - 1 <= 2 * n; i++)
            for (int k = 1; k < len; k++)
            {
                f[i][len] = min(f[i][len], f[i][k] + f[i +
k][len - k] + s[i + len - 1] - s[i - 1]);
                g[i][len] = max(g[i][len], g[i][k] + g[i +
k][len - k] + s[i + len - 1] - s[i - 1]);
            }
    for (int i = 1; i <= n; i++)
    {
        ans1 = min(ans1, f[i][n]);
        ans2 = max(ans2, g[i][n]);
    }
    cout << ans1 << endl
         << ans2 << endl;
    return 0;
}
```

金字塔

题目描述

给定长度为 nn 的字符串 ss。

从 ss 中提取子序列，组成 pyramid$pyramid$ 字符串的方法有多少种？

答案需对 $109+710^9+7$ 取模后输出。

输入格式

第一行：一个整数 nn

第二行：一个字符串 ss

输出格式

输出组成 pyramid$pyramid$ 的方法数（取模 $109+710^9+7$）。

输入样例#1

```
5
pxxxx
```

输出样例#1

```
0
```

输入样例#2

```
10
pyyradmiid
```

输出样例#2

```
4
```

说明提示

【数据范围】

- $1 \le N \le 1051 \le N \le 10^5$
- s 仅包含小写英文字母

限制

时间限制：1000ms

内存限制：256MiB

参考代码

```
#include <iostream>
#include <string>
using namespace std;
const int MOD = 1e9 + 7;

const int MAXN = 100010;
int dp[MAXN][10];
string t = "pyramid";
```

```
int main() {
    int n;
    string s;
    cin >> n >> s;

    for(int i = 0; i <= n; i++) {
        for(int j = 0; j < 7; j++) {
            dp[i][j] = 0;
        }
    }

    for(int i = 1; i <= n; i++) {
        for(int j = 0; j < 7; j++) {
            dp[i][j] = dp[i-1][j];
        }
        char c = s[i-1];
        if(c == t[0]) {
            dp[i][0] = (dp[i][0] + 1) % MOD;
        }
        for(int j = 1; j < 7; j++) {
            if(c == t[j]) {
                    dp[i][j] = (dp[i][j] + dp[i-1][j-1]) %
```

```
MOD;
            }
        }
    }
    cout << dp[n][6] << endl;
    return 0;
}
```