# 实验二

记f[i][j]为从第i个矩阵到第j个矩阵相乘最小运算次数，则f[i][j]=min(f[i][k]+f[k+1][j]+row[i]\*col[w]\*col[j]),每个矩阵的行列已知，还需要知道规模较小的连续矩阵的最小运算次数，因此从较小的连续矩阵开始运算。

## 动态方程

f[i][j]=min(f[i][k]+f[k+1][j]+row[i]\*col[w]\*col[j])

枚举i,j,k时间复杂度为O(n3)

## 代码

#include <bits/stdc++.h>

typedef long long ll;

using namespace std;

const int maxn=200;

ll a[maxn],dp[maxn][maxn];

string ans;

void dfs(int l,int r){

    if(l+1==r){

        ans+='A';

        ans+=char(r+'0');

        return;

    }

    ans+='(';

    // 反向查找最小的运算次数

    for(int i=l+1;i<r;++i) {

        if(dp[l][r]==dp[l][i]+dp[i][r]+a[l]\*a[i]\*a[r]){

            dfs(l,i);

            dfs(i,r);

            break;

        }

    }

    ans+=')';

}

void solve(){

    int n,t=0;

    while(cin>>n){

        if(n==0) break;

        for(int i=0;i<=n;++i) cin>>a[i];

        //初始化dp

        for(int i=0;i<=n;++i)

        for(int j=0;j<=n;++j)

        dp[i][j]=1e15;

  //临界条件矩阵自身不需要运算

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

        for(int len=2;len<=n;++len){

            for(int i=0;i+len<=n;++i){

                for(int j=i+1;j<i+len;++j){

                    dp[i][i+len]=min(dp[i][i+len],dp[i][j]+dp[j][i+len]+a[i]\*a[j]\*a[i+len]);

                    //对于i-i+len区间 找到最小的预算次数（i-j）+（j-i+len）+a[i]\*a[j]\*a[i+len]

                }

            }

        }

        cout<<"Case "<<++t<<endl;

        cout<<dp[0][n]<<" ";

        ans.clear();

        dfs(0,n);

        for(int i=1;i<ans.length()-1;++i) cout<<ans[i];

        cout<<endl;

    }

}

int main(){

    cin.tie(0)->ios::sync\_with\_stdio(0);

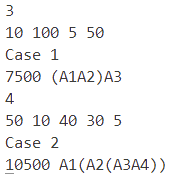
    int T=1;

    solve();

    return 0;

}

## 运行结果



## 遇到的问题

需要去掉最外面的括号，首尾的括号不一定匹配。需要弄清楚a[i]表示那个矩阵的行列。

## 体会

通过反向遍历可以找到最优的运算顺序，需要注意对应矩阵的行列和运算前的初始化问题。设计动态规划的状态和转移方程是解决问题关键步骤。