动态规划

可以将一个大问题分为若干个小问题，但与分治法不同的是，这个方法需要去记录子问题的解，进行记录，从而在之后使用。

矩阵连乘

问题

输入：n个矩阵A1,A2,...,An，其中Ai的维数为pi-1×pi
Ai 和Ai+1是可乘的

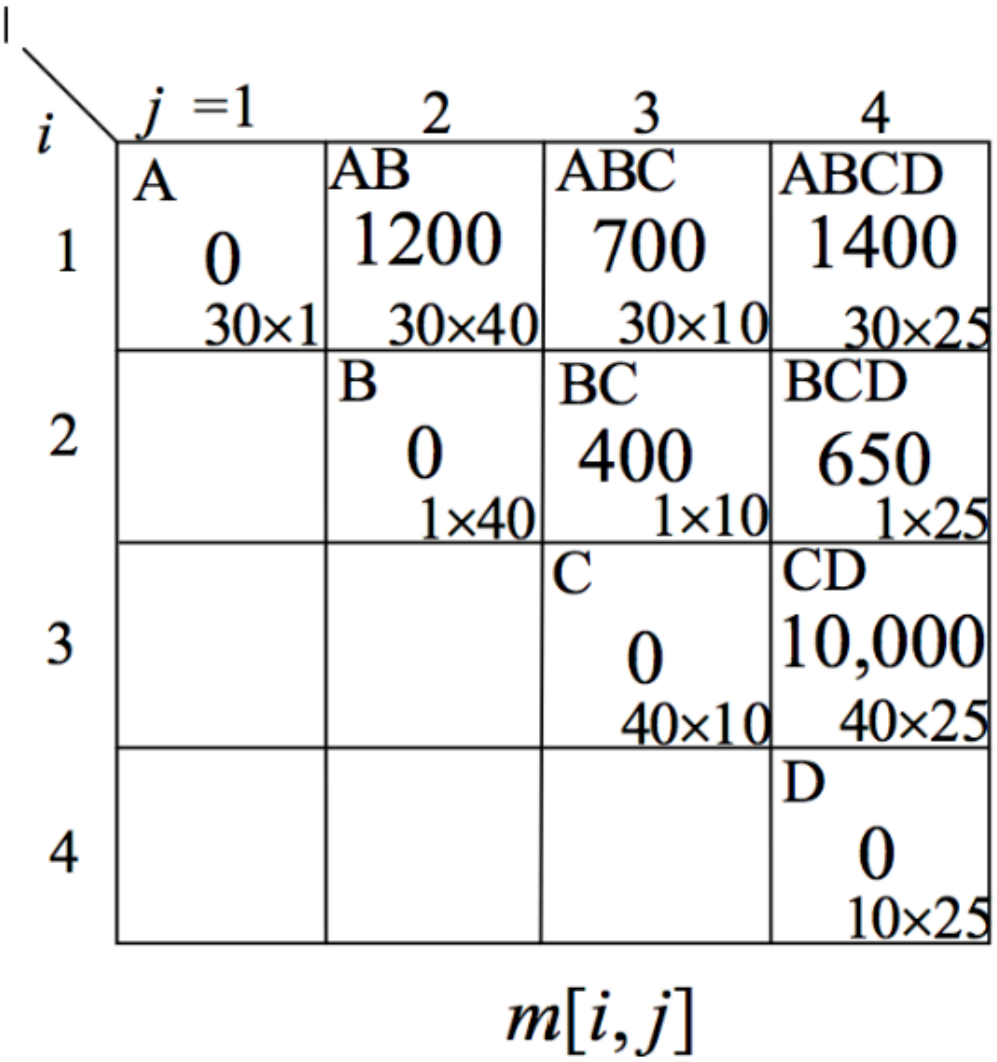
输出：连乘积A1A2A3...An

优化目标：最小计算代价（最优的计算次序）

矩阵乘法的代价：乘法次数
若A 是p ×q 矩阵，B 是q ×r 矩阵，则A ×B 的代价是pqr
因为矩阵乘法满足结合律，因此矩阵连乘可以由不同的计算次序，这种计算次序可以用加括号来表示。

思路：

最优结构多个划分，对于m(i,j)来说，可分为m（i,k)和m(k+1,j)两个矩阵相乘，那我们分别算出两个矩阵内部划分出的最优乘数最少的解最终再加上A(i-1) * A(k) * A(j)就得到了其中一个解，然后按k从i到j遍历就可以得到所有解，然后找到最小解就是答案



输入

```
6
30
35
15
10
20
25
```

代码

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

const int N=100;
int A[N];//矩阵
int m[N][N];//最优解
int s[N][N];

void MatrixChain(int n)
{
    //初始化对角线
    for(int i=0;i<=n;i++)
```

```
    {
        m[i][i]=0;
    }
    //n个矩阵存在r-1个空隙，现在我们的目的就是，计算按每个空隙劈成两半的解，再找最优
    for(int r=2;r<=n;r++)
    {
        //从后面开始切
        for(int i=1;i<=n-r+1;i++)
        {
            int j=i+r-1;
            //[i,i]+[i,j]
            m[i][j]=m[i][i]+m[i+1][j]+A[i-1]*A[i]*A[j];
            s[i][j]=i;
            //变换分割位置
            for(int k=i+1;k<j;k++)
            {
                int t =m[i][k]+m[k+1][j]+A[i-1]*A[k]*A[j];
                //如果变换后的位置最优
                if(t<m[i][j])
                {
                    m[i][j]=t;
                    s[i][j]=k;
                }
            }
        }
    }
}
void print(int i,int j)
{
    if(i==j)
    {
        cout<<"A["<<i<<"]";
        return ;
    }
    cout<<"(";
    print(i,s[i][j]);
    print(s[i][j]+1,j);
    cout<<")";
}
int main()
{
    int n;//n个矩阵
    cin>>n;
    for(int i=0;i<=n;i++)//从0开始，，放入你+1个数据
    {
        cin>>A[i];
    }
    MatrixChain(n);
    cout << "最佳添加括号的方式为：";
    print(1, n);
    cout << "\n最小计算量的值为：" << m[1][n] << endl;
    return 0;
}
```

输出

```
最佳添加括号的方式为：((A[1](A[2]A[3]))((A[4]A[5])A[6]))
最小计算量的值为：15125
```

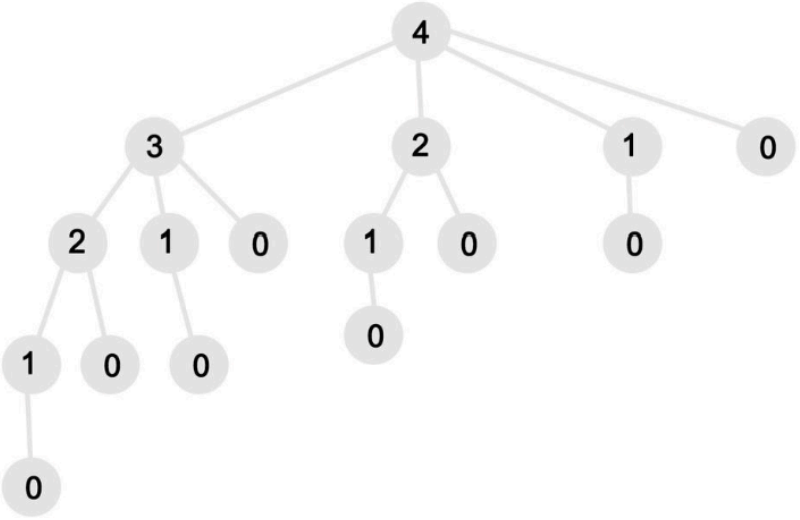
时间复杂度O(n^2)

钢条切割问题

思路：

对于钢条进行切割时，用r[n]表示的切割的利润，

$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i})$$



而如图指的是n,n=4就要算切一下，剩余3,2,1,0的情况

代码

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

int result[11]={0};
//自上而下
int UpDown(int n,int*p)
{
    if(n==0)
    {
        return 0;
    }
    //输出返回记录的结果
    if(result[n]!=0)
    {
        return result[n];
    }
    int t=0;
```

```
    for(int i=1;i<=n;i++)
    {
        //切一下，剩余多少的情况
        int max=p[i]+UpDown(n-i,p);
        if(max>t)
        {
            t=max;
        }
    }
    return t;

}
//自下而上
int result1[11]={0};
int BottomUp(int n,int*p)
{

    for(int i=1;i<=n;i++)
    {
        int t=0;
        for(int j=1;j<=i;j++)
        {
            int max=p[j]+result1[i-j];
            if(max>t)
            {
                t=max;
            }
        }
        result1[i]=t;

    }
    return result1[n];

}
int main()
{
    int p[11]{ 0, 1, 5, 8, 9, 10, 17, 17, 20, 24, 30 };//索引代表 钢条的长度，值代表价格
    cout << UpDown(4,p) <<endl;
    cout << BottomUp(4,p);
}
```

最长公共子序列

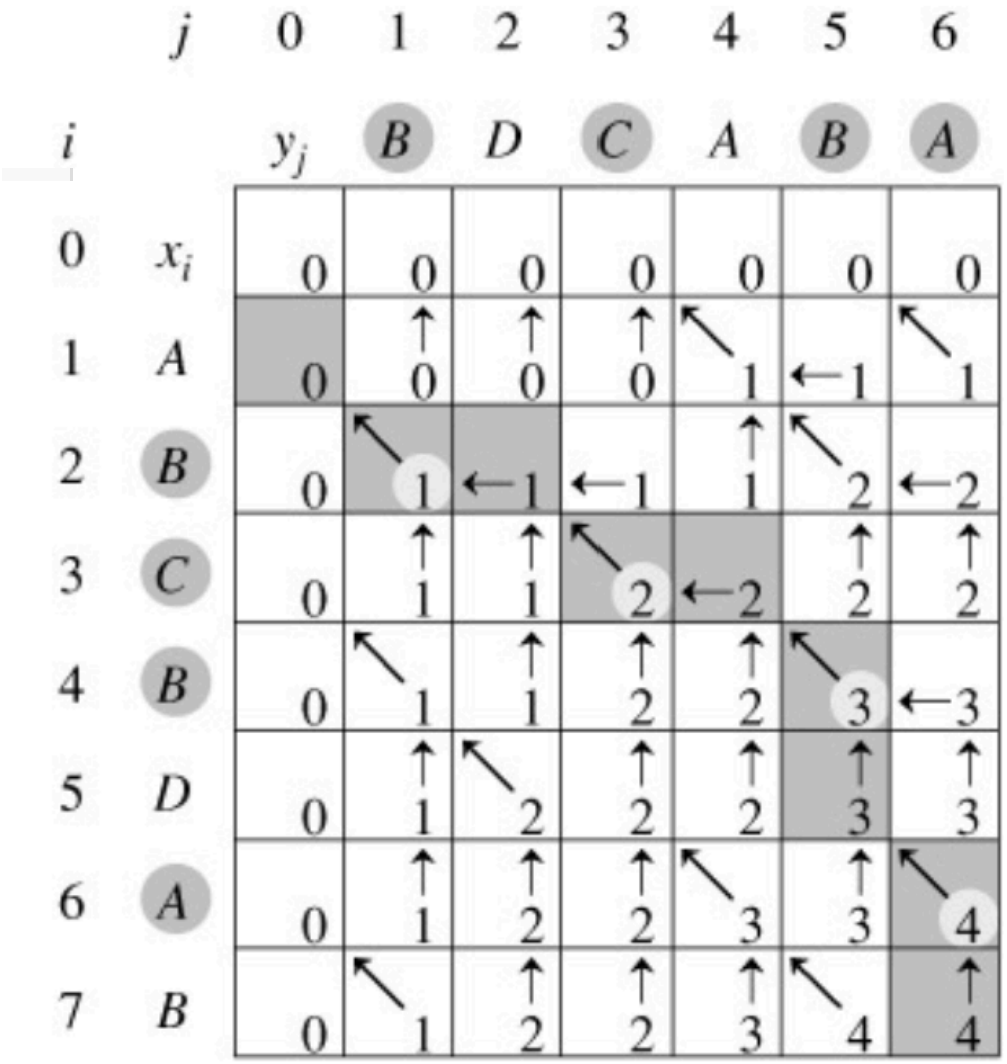
题目

给定两个长度分别为 NN 和 MM 的字符串 AA 和 BB，求既是 AA 的子序列又是 BB的子序列的字符串 **长度最长** 是多少。

思路

当a[i]==b[i]时，问题转化为f(i,j)=f(i-1,j-1);

当a[i]!=b[i]时，看有没有做出贡献




```cpp
#include<iostream>
#include<algorithm>
using namespace std;

const int N=1010;
int n,m;
char a[N],b[N];
int f[N][N];
int main()
{
    cin>>n>>m>>a+1>>b+1;
    for(int i=1;i<=n;i++)
    {
        for(int j=1;j<=m;j++)
        {
            f[i][j]=max(f[i-1][j],f[i][j-1]);
            {
                if(a[i]==b[i])
                {
                    f[i][j]=f[i-1][j-1]+1;
                }
            }
        }
    }
```

```
    printf("%d",f[n][m]);
}
```

子集和问题

f(i,j)记录的是在子集和为j的情况下是否选择W[i]这个数

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

bool subsetsum(vector<int>&nums,int target)
{
    int n=nums.size();
    vector<vector<bool>>dp(n+1,vector<bool>(target+1,false));
    for(int i=0;i<=n;++i)
    {
        dp[i][0]=true;
    }
    for(int i=1;i<=n;i++)
    {
        //j指的是子集和
        for(int j=1;j<=target;j++)
        {
            if(nums[i-1]>j)
            {
                dp[i][j]=dp[i-1][j];
            }
            else
            {
                dp[i][j]=dp[i-1][j]||dp[i-1][j-nums[i-1]];
            }
        }
    }
    return dp[n][target];
}
int main() {
    vector<int> nums = {3, 34, 4, 12, 5, 2};
    int target = 9;

    if (subsetsum(nums, target)) {
       cout << "存在一个子集，其元素之和等于 " << target <<endl;
    } else {
       cout << "不存在这样的子集" << endl;
    }

}
```

01背包问题

- N个物品，容量V的背包(上限)，W_i表示物品的体积,V_i表示价值
  如何组装背包，在VV的上限限制情况下，使得价值最大,求最大值。
  总结：每个物品只有11个，可以选或不选，求在容量限制下的价值最大值。

```cpp
#include<iostream>
using namespace std;

const int N = 1010;
int n, m;
int f[N][N];

int mian()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        int v,w;
        cin>>v>>w;
        for(int j=1;j<=m;j++)
        {
            f[i][j]=f[i-1][j];
            if(j>=v)
            {
                f[i][j]=max(f[i][j], f[i - 1][j - v] + w);
            }
        }
    }
     printf("%d\n", f[n][m]);
}
```