Date: / /

动态规划 (DP)

基本思想：

·动态规划方法：处理分段过程最优化问题的一类极其有效的方法。

·多阶段的决策过程：

- 问题的活动过程可以分段若干个阶段
- 在任一阶段后的行为依赖于该阶段的状态
- 与该阶段之前的过程是如何达到这种状态的方式无关

·最优化原则：多阶段过程的最优决策序列应当具有如下性质

- 无论过程的初始状态和初始决策是什么，
- 其余的决策都必须相对于初始决策所产生的状态构成一个最优决策序列。
- 即最优子结构性质

基本步骤：

· 研究最优解的性质，并刻画其结构特征

· 递归地定义最优值

· 以自底向上的方式计算出最优值

· 根据计算最优值时得到的信息，构造最优解

基本要素：最优子结构性质；重叠子问题性质

基本思想：保存已解决的子问题的解，避免重复计算

案例分析1：最大子段和

·问题描述：给定长度为n的整数序列，a[1...n]，求[1,n]某个子区间[i,j]，使得a[i]+...+a[j]和最大，并求出最大的这个和。

例如：(-2,11,-4,13,-5,2)

最大子段和为20，所求子区间为[2,4]

提炼问题三要素

Date: / /

-输入：a[1...n]

-输出：[1,n]中某个子区间[i,j]

-约束条件：a[i]+…+a[j]和最大

方法1：穷举法：

穷举所有的[1,n]之间的区间，用两重循环，可遍历所有子区间。

```
int start=0; //起始位置
int end=0; //结束位置
int max=0;
for (int i=1; i<=n; ++i)
{
    for (int j=1; j<=n; ++j)
    {
        int sum=0;
        for(int k=i; k<=j; ++k)
            sum+=a[k];
        if(sum>max)          O(n^3)
        {
            start=i;
            end=j;
            max=sum;
        }
    }
}
```

穷举法改进：$\sum_{k=1}^{j} a_k = a_j + \sum_{k=1}^{j-1} a_k$ 避免重复计算：$O(n^2)$

Date: / /

方法2：分治法

递归复杂分析

$$T(n)=\begin{cases} O(1) & n\le c \\ 2T(n/2)+O(n) & n>c \end{cases}$$

$T(n)=O(n\log n)$

方法3：动态规划法

步骤1——分析最优子结构性质

步骤2——建立 b[j] 的递推关系

$b[j]=\max[b[j-1]+a[j],a[j]]$

步骤3——从自底向上的方式计算各个 b[j]

步骤4——构造最优解

```
int max=0;
int b[n+1];
int start=0;
int end=0;
memset(b,0,n+1);
for(int i=1;i<=n;++i)
{
  if(b[i-1]>0)
  {
    b[i]=b[i-1]+a[i];          O(n)
  }
  else
    b[i]=a[i];
  if(b[i]>max)
    max=b[i];
}
```

案例分析2：最长公共子序列

问题描述：

给定2个序列$X=[x_1,x_2,\cdots,x_m]$和$Y=[y_1,y_2,\cdots,y_n]$，找出X和Y的最长公共子序列。

提炼问题三要素

- 输入：$X=(x_1,x_2,\cdots,x_m)$，$Y=(y_1,y_2,\cdots,y_n)$
- 输出：Z= X与Y的最长公共子序列
- 约束：无

- 相关概念：第i前缀
- 公共子序列

1. 结构分析：2个序列的最长公共子序列包含了这2个序列的前缀的最长公共子序列。因此，最长公共子序列问题具有最优子结构性质。

2. 子问题的递归结构：

$$c[i][j]=\begin{cases}0 & i=0,j=0\\ c[i-1][j-1]+1 & i,j>0;x_i=y_j\\ \max[c[i][j-1],c[i-1][j]] & i,j>0;x_i\neq y_j\end{cases}$$

算法复杂性

- 时间复杂性
  - 计算代价的时间
    - (i,j)两层循环，i循环m步，j循环n步
    - O(mn)
  - 构造最优解的时间：O(m+n)
  - 总时间复杂性为：O(mn)
- 空间复杂性
  - 使用数组C和B
  - 需要空间O(mn)

Date: / /

案例分析3：矩阵链乘法

问题描述：

给定n个矩阵{$A_1, A_2, \cdots, A_n$}，其中A_i与A_{i+1}是可乘的，$i=1, 2 \cdots, n-1$。如何确定计算矩阵连乘积的计算次序，使得依此次序计算矩阵连乘积需要的数乘次数最少。

提炼问题三要素：

-输入：$<A_1, A_2, \cdots, A_n>$，A_i是矩阵

-输出：计算$A_1 \times A_2 \times \cdots \times A_n$的最优计价方法

-约束条件：A_i与A_{i+1}是可乘的，$i=1, 2, \cdots, n-1$

方法1：穷举法

方法2：动态规划法

步骤1：分析最优解的结构

步骤2：建立递归关系

$$m[i,j]=\begin{cases} 0 & i=j \\ \min\limits_{i\le k<j}\{m[i,k]+m[k+1,j]+P_{i-1}P_kP_j\} & i<j \end{cases}$$

上面位置只有$j-i$种可能

步骤3：自底向上计算最优值

步骤4：构造最优解

算法复杂性

·时间复杂性

-计算计价的时间

·(1,i,k)三层循环，每层至多n-1步

·$O(n^3)$

-构造最优解的时间：$O(n)$

-总时间复杂性为：$O(n^3)$

·空间复杂性

Date: / /

- 用数组 m 和 S
- 需要空间 $O(n^2)$

算法分析：0-1背包问题

$$m(i,j)=\begin{cases}\max\{m(i+1,j),\ m(i+1,j-w_i)+v_i\} & j\geq w_i\\ m(i+1,j) & 0\leq j<w_i\end{cases}$$

$$m(n,j)=\begin{cases}v_n & j\geq w_n\\ 0, & 0\leq j<w_n\end{cases}$$