哈爾濱工業大學(深圳)

HARBIN INSTITUTE OF TECHNOLOGY, SHENZHEN

《算法设计与分析》
第四章作业

学院: 计算机科学与技术学院

姓名: 周健毅

学号: 220110713

专业: 计算机科学与技术

日期: 2023 年 10 月 14 日

成绩:

一、 第一题

(1) 请写出该问题的递推方程（定义dp[i][j]为第i种物品到第n种物品装进限重为j的背包可获得的最大价值）（10分） $\mathrm{dp}[i][j]=\max\{\mathrm{dp}[i+1][j], d[i+1][j-k*w[i]]+k*v[i]\}$

如果使用该递推方程来求最大价值的话，我们需要进行三重循环，时间复杂度为$O(n^3)$。

我们可以对其进行优化:

将上式中的j替换为$j-w[i]$可以得到:

$$\mathrm{dp}[i][j-w[i]]=\max(f[i+1][j-w[i]], f[i+1][j-2*w[i]]+v[i], ...)$$

对比:

$$\mathrm{dp}[i][j]=\max\{\mathrm{dp}[i+1][j], d[i+1][j-k*w[i]]+k*v[i], ...\}$$

可得新的递推关系式为：

$$\mathrm{dp}[i][j]=\max\{\mathrm{dp}[i+1][j-w[i]]+v[i], \mathrm{dp}[i-1][j]\}$$

使用此递推式来解决完全背包问题，时间复杂度为$O(n^2)$

(2)假设背包容量为5，有4种物品，其重量分别为w=[1,2,3,4]，其价值分别为v=[2,4,4,5]，请写出对应的dp矩阵

i\j	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	2	4	6	8	10
2	0	0	4	4	8	8
3	0	0	0	4	5	5
4	0	0	0	0	5	5

(3) 请写出该问题的伪代码（10分）

```
#include<iostream>
using namespace std;
const int N =1010;
int w[N],v[N],dp[N][N];
int n,m;
int main()
{
    cin>>n>>m;
    for(int i=1;i<=n;i++)
    {
        cin>>w[i]>>v[i];
    }
    for(int i=n;i>=1;i--)
```

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

    cout<<dp[1][m]<<endl;
    return 0;
}
```

(4) 如果上述伪代码是用二维数组实现的，请问是否有空间更优化的的实现版本？提示：可否将2维dp数组降至1维dp数组。 （附加题10 分）

可以优化成一维dp数组：

因为每次更新dp$[i][j]$时，我们只用到了dp$[i+1][j-k*w[i]](k=0,1...)$，即更新第i维的dp数组时，只用到了第$i+1$维的dp数组，因此可以省略掉第一维的数组，新的递推表达式为：

$$\mathrm{dp}[j]=\max(\mathrm{dp}[j],\mathrm{dp}[j-w[i]]+v[i])$$

证明该变形的正确性：

更新后的代码为：

```
for(int i=n;i>=1;i--)
    {
        for(int j=w[i];j<=m;j++)
        {
            dp[j]=max(dp[j],dp[j-w[i]]+v[i]);
            //原代码：dp[i][j]=max(dp[i][j],dp[i][j-w[i]]+v[i]);
        }
    }
```

可以看到每次更新dp$[j]$时，用到的是dp$[j-w[i]]$,因为第二重循环是从小到大遍历的，因此dp$[j-w[i]]$在更新dp$[j]$之前就已经被更新过了，所以此时的dp$[j-w[i]]$代表的是原代码中的dp$[i][j-w[i]]$，因此变形前后的代码是等价的，故可以优化为一维数组。

二、 第二题

II. 若7个关键字的概率如下所示，求其最优二叉搜索树的结构和代价，要求必须写出递推方程。（30分）

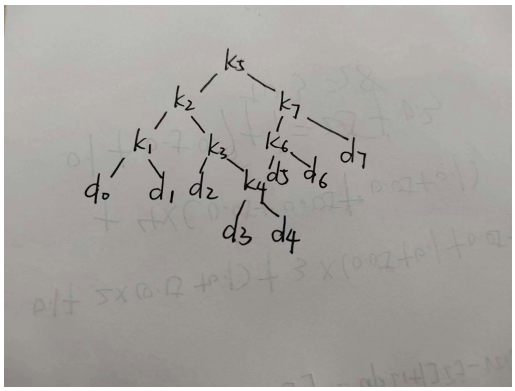

Figure 1: 最优二叉搜索树

递推方程:

$$E(i,j)=\begin{cases}q_{i-1}=q_j \text{ if } j=i-1\\ \min_{i\le r\le j}\{E(i,r-1)+E(r+1,j)+W(i,j) \text{ if } j\ge i\}\end{cases}$$

$$E(1,7)=E(1,4)+E(6,7)+W(1,7)$$

$$=[E(1,1)+E(3,4)+W(1,4)]+[E(6,6)+E(8,8)+W(6,8)]$$

$$=3.12$$

三、 第三题

III. 灭鼠人计划消灭菜田中田鼠，每个鼠洞内都藏有一定数量的田鼠。这个菜田所有的鼠洞不互通且都围成一圈，这意味着第一个鼠洞和最后一个鼠洞是紧挨着的。同时，由于灭鼠动静过大，左右相邻的鼠洞中的田鼠会听见灭鼠动静而全部迅速逃窜，逃窜的田鼠不会去往其他鼠洞，而是逃离这片菜田。

给定一个代表每个鼠洞藏匿老鼠数量的非负整数数组，计算能够消灭田鼠的最大数量。(40分)

思路：将围成圈的鼠洞分成两次计算，第一次只计算$[0,n-2]$的鼠洞的最大灭鼠量，第二次计算$[1,n-1]$鼠洞的最大灭鼠量，两次计算结果的最大值就是答案。 状态转移方程：

$$\mathrm{dp}[i][0]=\max\{\mathrm{dp}[i-1][0],\mathrm{dp}[i-1][1]\}$$

$$\mathrm{dp}[i][1]=\mathrm{dp}[i-1][0]+\mathrm{nums}[i]$$

其中：$\mathrm{dp}[i][1]$表示考虑前i个鼠洞，且选择第i个鼠洞能打到的最大鼠数，$\mathrm{dp}[i][0]$表示考虑前i个鼠洞，且不选择第i个鼠洞能打到的最大鼠数。

优化空间复杂度:

$$\mathrm{dp}[i]=\max\{\mathrm{dp}[i-2]+\mathrm{nums}[i],\mathrm{dp}[i-1]\}$$

其中，$\mathrm{dp}[i-1]$表示不选第i个鼠洞，$\mathrm{dp}[i-2]+\mathrm{nums}[i]$表示选择第i个鼠洞。

实现:

```cpp
#include<iostream>
#include<vector>
#include<cstring>
using namespace std;
const int N = 1010;
int dp[N];
int KillMostMice(vector<int> a,int l,int r)
{
    memset(dp,0,sizeof dp);
    dp[0]=a[0];
    dp[1]=max(a[0],a[1]);
    for(int i=2;i<a.size();i++)
    {
        dp[i]=max(dp[i-1],dp[i-2]+a[i]);
    }
    return dp[r];
}
int main()
{
    vector<int>a;
    int n;
    cin>>n;
    for(int i=0;i<n;i++)
    {
        int x;
        cin>>x;
        a.push_back(x);
    }
    int ans1=KillMostMice(a,0,n-2);
    int ans2=KillMostMice(a,1,n-1);
    cout<<max(ans1,ans2)<<endl;
    return 0;
}
```