金币岛问题-贪心与动态规划协同考虑问题

原创 向远洋 LOA算法学习笔记 2021-01-22 15:30

01 题目描述

鲍勃误打误撞进入了一个叫金币岛的地方，这里有n棵会长金币的树从tree[1]、tree[2]...到tree[n]。在第一天的时候，tree[i]有a[i]个金币在上面。如果树i不被砍掉的话，树i每天会长b[i]颗金币在上面。从第一天起到第m天为止，鲍勃每天都可以选择砍下一棵树，并把所有的金币收入囊中。如果他在m天内的某天决定不砍掉树，他之后就不能再砍树了，也就是说，他只能在连续的m天或更少的天数内砍树。给定n、m、a[i]和b[i]，计算鲍勃能拿到的最多的金币数量。

02 基本思路

卜老师在课上常说，拿到一个问题如果不会做，看一看有没有类似的问题我们会做。从的题干上来看，本题其实有一些类似于动态规划中的背包问题，大家可以很快想到，我们是不是可以使用背包问题的解法去解决这道题？然而定睛一看，本题和背包问题又有一些区别，即这个“背包”中的每一件物件都有自己的初始价值与增长价值，换句话说是它们的价值每天都在变化，也就是说增量也是我们在多步决策之外需要考虑的因素。在本题中，我们再来观察其特征，第一方面我们砍掉的树的最多数目是确定的，另一方面我们的目标是对价值的线性加和，因此我们可以考虑使用贪心。我们先给出贪心策略：增长越快的树越晚砍。根据这个策略，我们对树按每日增量进行排序。排完序以后，进入动态规划部分，第i棵树在第j天有两种状态，砍或者不砍。定义表达式dp[i][j]，表示第j天砍第i棵树获得的最大价值，递推表达式如下式所示，这个式子的含义是，第i棵树在第j天没砍获得的价值是dp[i-1][j]，在j天砍了，说明i棵树之前没砍dp[i-1][j-1]+a[i]+b[i]*(j-1)，二者取最大即可。

$$dp[i][j]=\max(dp[i-1][j],\ dp[i-1][j-1]+tree[i].init+(j-1)*tree[i].increment)$$
$$dp[i][0]=dp[0][j]=0,\qquad i,j\in Z,\ 0\le j\le m,\ 0\le i\le n$$

03 伪代码

```
//tree用于表示树的价值tree.first表示初始价值，tree.second表示增长价值
typedef pair<int, int>Tree;
Tree tree[n];
sort(based on tree[n].second);
for(int i=1; i<=n;  i++){
       for(int j=1; j<=m;  j++)
              dp[i][j] = max(dp[i-1][j], dp[i-1][j-1] + trees[i].first + (j-1)*trees[i].second);
}
return dp[n][m];
```

04 正确性证明

1）贪心部分：砍掉e棵树的情况下，假设我们已经得到了最优方案，尽管某个增量大的在某个增量小的树之前被砍。下式列出了我们所收获的硬币数：

$$sum = a[1] + (a[2] + b[2]) + (a[3] + 2*b[3]) + \ldots + a[i] + (j-1)*b[i] + \ldots + a[k] + (j + \triangle j - 1)*b[k] + \ldots$$

按照假设，在第i棵树在第j天砍，第k棵树在第j+Δj天砍，其中b[k]<b[i]，Δj>0。如下式所示，前面一部分是固定值，那么将第i棵树放在第j+Δj天砍，第k棵树放在第j天砍得到的sum会比原来的方案更大。因此，与原方案最优的假设矛盾，故假设不成立。

$$sum = a[1] + a[2] + \ldots + a[i] + \ldots + a[k] + \ldots + b[2] + 2*b[3] + \ldots + (j-1)*b[i] + \ldots + (j + \triangle j - 1)*b[k] + \ldots$$

2）动态规划部分：运用循环不变量，初始化：首先考虑最简单的情况所有的树在第0天时（即dp[i][0]=0），鲍勃能获得收益最大为0，满足。维护：假设已计算出第i-1棵树在第j天与第j-1天得到的最优收益，在第j天决策是否需要砍第i棵树时，如果第i棵树在第j天不砍第i棵树，那么鲍勃口袋中的金币数取决于排序后序号更前的树在当天（第j天）的状态下所得到的最优收益（即dp[i-1][j]），因为根据贪心规则排序更后的应该更后砍，因此我们只需要往前看砍或不砍的状态。否则，我们决定第j天砍第i棵树，也就意味着在第j天不可以砍其他的树，因此鲍勃口袋中的金币数取决于排序后序号更前的树在前一天（第j-1天）与该树在第j天的价值的和值。由于树i在j天只有两种情况，即砍或者不砍，我们将这两种情况获得的收益取最大即是这种情况下的最优，也即第i棵树第j天的决策的最优性得证。终止：我们将维护的证明中参数i、j推至极端，即i=n，j=m，终止情况的最优性也可得证。

综上二者，正确性得到证明。

05 举例模拟

设n=6，m=4，a[i]与b[i]如下表所示

a[1]	a[2]	a[3]	a[4]	a[5]	a[6]
2	8	3	7	4	6
b[1]	b[2]	b[3]	b[4]	b[5]	b[6]
7	2	4	5	3	1

列出dp表格，第一行表示排序后的树的序号，括号中为原序号，第一列表示第几天。

j \ i	0	1(6)	2(2)	3(5)	4(3)	5(4)	6(1)
0	0	0	0	0	0	0	0
1	0	6	8	8	8	8	8
2	0	7	16	16	16	20	20
3	0	8	19	26	27	33	36
4	0	9	22	32	41	49	56

列出回溯路径，即可发现，斜向上的箭头表示砍了树，而水平的箭头表示该天未砍该树，最后结果：第一天砍第6棵树，第2天砍第2棵，第三天砍第4棵，第4天砍第1棵。

j \ i	0	1(6)	2(2)	3(5)	4(3)	5(4)	6(1)
0	0	0	0	0	0	0	0
1	0	6	8	8	8	8	8
2	0	7	16	16	16	20	20
3	0	8	19	26	27	33	36
4	0	9	22	32	41	49	56

06 总结

本题初看上去非常类似于背包问题，但"背包"每一件"物品"的价值在变化，这就相当于给我们的多步决策多引入了一个需要考虑的因素或维度，那我们想着是不是可以将该因素或维度先进行考虑，形成一种多步决策的方向或顺序，接下来的多步决策过程即不需要单独考虑这一维度的因素。因此我们首先运用贪心，得到一个很自然的顺序，即增量大的

需要尽可能晚地砍，那也就是意味着，就算要砍也是要在增量小的某些树先砍之后的某天再砍，因此我们进行决策时的顺序也是按照贪心所给的顺序来进行的。