Atcoder Educational DP Contest 题解

张晴川

March 12, 2020

Contents

前言

本文为 **Atcoder** 上的 *Educational DP Contest* 的简要题解。该场比赛共有 26 题，从易到难，是练习动态规划的优秀之选。

提交地址为: `https://atcoder.jp/contests/dp`

如发现错误或有修改建议，请联系我 (**qzha536@aucklanduni.ac.nz**)。

A　Frog 1

题意

有 $n(1 \le n \le 10^5)$ 块石头（编号为 1 到 n）排成一排，第 i 块石头的高度为 h_i。青蛙现在站在第 1 块石头上。它可以从第 i 块石头跳到第 $i+1$ 或者 $i+2$ 块石头上。从第 i 块石头跳到第 j 块需要消耗 $|h_j - h_i|$ 点体力。请求出跳到第 n 块石头所需的最小体力值。

题解

设 $dp[i]$ 为跳到第 i 块石头上所需的最小体力值，则 $dp[n]$ 为题之所求。根据题意，不难发现有：

$$dp[i] = \begin{cases} 0 & i = 1 \\ |h_2 - h_1| & i = 2 \\ \min(dp[i-1] + |h_i - h_{i-1}|, dp[i-2] + |h_i - h_{i-2}|) & i \ge 3 \end{cases}$$

$O(N)$ 递推一下即可。

核心代码

```
dp[1] = 0
dp[2] = abs(h[2]-h[1]);
for(int i = 2;i<=n;i++){
    dp[i] = min(dp[i-1]+abs(h[i]-h[i-1]),dp[i-2]+abs(h[i]-h[i-2]));
}
```

B　Frog 2

题意

有 $n(1 \le n \le 10^5)$ 块石头（编号为 1 到 n）排成一排，第 i 块石头的高度为 h_i。青蛙现在站在第 1 块石头上。它可以从第 i 块石头跳到第 $i+1, i+2, \dots, i+k(1 \le k \le 100)$ 块石头上。从第 i 块石头跳到第 j 块需要消耗 $|h_j - h_i|$ 点体力。请求出跳到第 n 块石头所需的最小体力值。

题解

设 $dp[i]$ 为跳到第 i 块石头上所需的最小体力值，则 $dp[n]$ 为题之所求。根据题意，不难发现有：

$$dp[i] = \begin{cases} 0 & i = 1 \\ \min\{dp[j] + |h_i - h_j| \mid i - k \le j < i\} & i \ge 2 \end{cases}$$

$O(nk)$ 递推一下即可。

核心代码

```
dp[1] = 0
for(int i = 2;i<=n;i++){
    for(int j = max(i-k,1);j<i;j++){
        dp[i] = min(dp[i],dp[j]+abs(h[i]-h[j]));
    }
}
```

C Vacation

题意

太郎的暑假有 $n(1 \leq n \leq 10^5)$ 天，第 i 天他可以选择以下三种活动之一：

- 游泳，获得 a_i 点幸福值。
- 捉虫，获得 b_i 点幸福值。
- 写作业，获得 c_i 点幸福值。

但他不能连续两天进行同一种活动，请求出最多可以获得多少幸福值。

题解

设 $A[i], B[i], C[i]$ 分别表示在第 i 天进行三种活动的前提下，前 i 天的最大幸福值。那么可以得到，

$$A[i] = \begin{cases} a_1 & i = 1 \\ \max(B[i-1], C[i-1]) + a_i & i \geq 2 \end{cases}$$

$B[i]$ 和 $C[i]$ 的求法类似，故不赘述。实现时可以开一个大小为 $n \times 3$ 的二维数组。

核心代码

```
//v[d][i] 表示第 d 天进行第 i 项活动获得的幸福值
for(int i = 0;i<3;i++){
    dp[1][i] = v[1][i];
}
for(int day = 2;day<=n;day++){
    for(int cur = 0;cur<3;cur++){
        dp[day][cur] = 0;
        for(int last = 0;last<3;last++){
            if(cur != last){
                dp[day][cur] = max(dp[day][cur],dp[day-1][last] + v[day][cur]);
            }
        }
    }
}
```

D　Knapsack 1

题意

经典背包问题。有 n 个物品，每个物品有重量 w_i 和价值 v_i。背包的容量为 W，换句话说，可以放进背包的物品的重量总和不能超过 W。请最大化背包内物品的价值总和。

数据范围

- $1 \le N \le 100$
- $1 \le W \le 10^5$
- $1 \le v_i \le 10^9$

题解

我们设 $dp[i][j]$ 表示 **"只考虑前 i 个物品的情况下，背包容量是 j 所能凑出的最大价值之和"**。那么在计算 $dp[i][j]$ 的时候，我们可以把所有情况分成两类进行考虑，

1. 拿第 i 件物品，那么现在背包容量只剩下 $j - w_i$，而由于每样物品只能拿一件，所以我们只需要考虑前 $i-1$ 件物品的最优选取方式，即最终价值为 $dp[i-1][j-w_i] + v_i$。
2. 不拿第 i 件物品，那么背包容量仍然是 j。由于我们选择不拿第 i 件物品，现在只需要考虑前 $i-1$ 件物品的最优选取方式，即最终价值为 $dp[i-1][j]$。

所以可以得到转移方程为：

$$dp[i][j] = \begin{cases} 0 & i = 0 \\ \max(dp[i-1][j-w_i] + v_i, dp[i-1][j]) & i \ge 1 \end{cases}$$

递推或记忆化搜索的复杂度均为 $O(nW)$。

核心代码

```cpp
//注意要开 long long
for(int i = 1;i<=n;i++){
    for(int j = 0;j<=W;j++){
        //容量不够
        if(j < w[i]){
            dp[i][j] = dp[i-1][j];
        //容量足够
        }else{
            dp[i][j] = max(dp[i-1][j-w[i]] + v[i], dp[i-1][j]);
        }
    }
}
```

E

题意

题解

核心代码

```
E
```

F

题意

题解

核心代码

```
F
```

G

题意

题解

核心代码

```
G
```

H

题意

题解

核心代码

```
H
```

I

题意

题解

核心代码

```
I
```

J

题意

题解

核心代码

```
J
```

K

题意

题解

核心代码

```
K
```

L

题意

题解

核心代码

```
L
```

M

题意

题解

核心代码

```
M
```

N

题意

题解

核心代码

```
N
```

O

题意

题解

核心代码

```
0
```

P

题意

题解

核心代码

```
P
```

Q

题意

题解

核心代码

```
Q
```

R

题意

题解

核心代码

```
R
```

S

题意

题解

核心代码

```
S
```

T

题意

题解

核心代码

```
T
```

U

题意

题解

核心代码

```
U
```

V

题意

题解

核心代码

```
V
```

W

题意

题解

核心代码

```
W
```

X

题意

题解

核心代码

```
X
```

Y

题意

题解

核心代码

```
Y
```

Z

题意

题解

核心代码

```
Z
```