计算机实验

解答. [0/1 背包问题] 设 `dp[i][j]` 表示从前 i 个物品中，使用容量为 j 的背包所能获得的最大价值. 则状态状态转移方程如下

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

其中 max 中的第一项 $dp[i-1][j-w[i]]+v[i]$ 表示在剩余容量为 j 的前提下，选择第 i 个物品所获得的最大价值，则可以从前 $i-1$ 个物品容量为 $j-w[i]$ 所能获得的最大价值加上当前物品的价值 $v[i]$ 得到；max 的第二项 $dp[i-1][j]$ 表示不选取第 i 件物品，则直接从前 $i-1$ 件物品容量为 j 的最大价值直接转移得到.

若 $dp[i][j]$ 是从 max 的第一项转移得到，则选择了当前物品 i；否则，从 max 的第二项转移得到，则没有选择物品 i. 通过记录 $dp[i][j]$ 从谁转移得到的，即可得知最终选取的物品是哪些. 最终输出可逆向枚举得到.

代码如下：

```cpp
#include <iostream>
using namespace std;
const int N = 1001;
int w[N], v[N];
int dp[N][N];
bool use[N][N];
int main() {
    int tot, n;
    cin >> tot >> n;
    for (int i = 1; i <= n; i++) cin >> w[i];
    for (int i = 1; i <= n; i++) cin >> v[i];
    for (int i = 1; i <= n; i++) {  // 枚举第 i 个物品
        for (int j = w[i]; j <= tot; j++) {  // 枚举当前背包容量 j
            dp[i][j] = dp[i-1][j];  // 默认不选择当前物品
            if (dp[i-1][j-w[i]] + v[i] > dp[i][j]) {  // 若价值更高, 则选择当前物品
                dp[i][j] = dp[i-1][j-w[i]] + v[i];
                use[i][j] = 1;  // 选择当前物品
            }
        }
    }
    cout << " 能获得的最大价值为: " << dp[n][tot] << '\n';
    int now = n, less = tot;
    cout << " 选取物品编号: ";
```

```
    while (now) {
        if (use[now][less]) {
            cout << now << ' ';
            less -= w[now];
        }
        now--;
    }
    cout << '\n';
        return 0;
}

#if 0
输入样例:
10
5
2 2 6 5 4
6 3 5 4 6
输出:
能获得的最大价值为: 15
选取物品编号: 5 2 1
#endif
```

解答. [猴子吃香蕉] 设 `dp[i][j]` 表示当前 i 棵树上用掉 j 次跳跃所能吃到最多的香蕉个数，则状态转移方程如下

$$dp[i][j] = \max_{1 \leqslant k < i} \{dp[k][j-1] + a[i] : b[i] - b[k] \leqslant d\}$$

上述方程表示，可以从第 k 棵树跳转到第 i 棵树（前提是 $b[i] - b[k] \leqslant d$，即两棵树之间的距离在最大跳跃距离内），所以从第 k 棵树使用 $j-1$ 次跳跃所能获得的最大香蕉数加上当前第 i 棵树上的香蕉数 $a[i]$ 就是当前所能获得的最大香蕉数目.

`dp[][]` 数组默认初始化为负无穷，`dp[0][0]=a[0]` 表示当前猴子在第 0 棵树上，无需花费步数能吃到 `a[0]` 个香蕉.

代码如下:

```
#include <iostream>
#include <cstring>
using namespace std;
const int N = 1000;
```

```cpp
int n, m, d, a[N], b[N], dp[N][N], ans;
int main() {
    memset(dp, 128, sizeof(dp));  // 初始化 dp 为负最大值
    cin >> n >> d >> m;
    for (int i = 0; i < n; i++) cin >> a[i] >> b[i];
    dp[0][0] = a[0];
    for (int j = 1; j <= m; j++) {  // j 为当前跳跃次数
        for (int i = 1; i < n; i++) {  // i 为当前位于树的编号
            for (int k = i-1; k >= 0 && b[i] - b[k] <= d; k--) {  // 枚举从第 k 棵
↪ 树上跳过去
                dp[i][j] = max(dp[k][j-1] + a[i], dp[i][j]);
                ans = max(ans, dp[i][j]);
            }
        }
    }
    cout << ans << '\n';
        return 0;
}
#if 0
输入:
6 8 2
3 0
6 5
1 6
2 7
7 10
4 11
输出:
16
#endif
```