一、简答

1. 对于 $f[i][j], j > 0$ 代表以 i 为根的子树中选取 j 个关键节点对答案的贡献，应初始化为无穷大。而当 $j = 0$ 时，选取 0 个关键节点贡献为 0 ，初始化为 0 ，同时，当节点 i 为关键节点时，$j = 1$ 也初始为 0 。

二、代码及注释

D. 帝国大厦

题目大意

帝国大厦共有 n 层，LZH 初始时在第 a 层上。
帝国大厦有一个秘密实验室，在第 b 层，这个实验室非常特别，对 LZH 具有约束作用，即若 LZH 当前处于 x 层，当他下一步想到达 y 层时，必须满足 |x−y|<|x−b|，而且由于实验室是不对外开放的，电梯无法停留在第 b 层。
LZH 想做一次旅行，即他想按 k 次电梯，他想知道不同的旅行方案个数有多少个。
两个旅行方案不同当前仅当存在某一次按下电梯后停留的楼层不同。

解法

前缀和优化dp。
定义状态 $f[i][j]$ 代表按第 i 次电梯后停在第 j 层的方案数，显然答案为 $\sum_j (f[k][j])$.
转移方程为

$$f[i][t] = \sum_{j \in [l,r] and j \neq t} (f[i-1][j])$$

其中，当 $t > b$ 时, $l = (t+b)/2 + 1, r = n$; 当 $t < b$ 时, $l = 1, r = (t+b-1)/2$.
求和计算可通过前缀和优化为 $O(1)$.

时间复杂度

$O(k * n)$, 看嵌套循环的层数。

代码

```cpp
#include<iostream>
using namespace std;
using ll = long long;
const int mod = 1e9 + 7;

ll f[5005][2];
ll sum[5005];   // 前缀和优化

int main() {
    ios::sync_with_stdio(false);
    cin.tie(nullptr);
    int n, a, b, k;
    cin >> n >> a >> b >> k;
    f[a][0] = 1;    // 初始化一开始停在a层
    // 预处理前缀和
    for (int j = 1; j <= n; j++) {
        sum[j] = sum[j - 1] + f[j][0];
        sum[j] %= mod;
    }
    for (int i = 1; i <= k; i++) {
        for (int j = 1; j <= n; j++) {
            int l, r;
            if (j > b) {
                l = (j + b) / 2 + 1;
                r = n;
            }
            else if (j < b) {
                l = 1;
                r = (j + b - 1) / 2;
            }
            else continue;
            // 转移方程，通过滚动数组进行空间优化
            f[j][i & 1] = sum[r] - sum[l - 1] - f[j][1 - (i & 1)];
            // 对可能负数取模
            f[j][i & 1] = (f[j][i & 1] + mod) % mod;
            // cout << f[j][i & 1] << ' ';
        }
        // cout << '\n';
        for (int j = 1; j <= n; j++) {
            sum[j] = sum[j - 1] + f[j][i & 1];
            sum[j] %= mod;
        }
    }
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        ans += f[i][k & 1];
        ans %= mod;
    }
    cout << ans;
```

```
    return 0;
}
```