L1500 ： 搜索2 ☆☆☆

题目描述

有一个奇怪的电梯，大楼有 N 层楼，每层楼有一个数字 K_i（$0 \leq K_i \leq N$）。电梯只有四个按钮：开，关，上，下。上下的层数等于当前楼层上的那个数字。如果不能满足要求，相应的按钮就会失灵。

例如：$3,3,1,2,5$ 代表了 K_i（$K_1=3$，$K_2=3$，……），从1楼开始。在1楼，按”上”可以到4楼，按”下”是不起作用的，因为没有-2楼。

那么，从A楼到B楼至少要按几次按钮呢？

输入输出格式

输入：第一行为三个用空格隔开的正整数，

表示 N, A, B（$1 \leq N \leq 200$，$1 \leq A, B \leq N$）。

第二行为 N 个用空格隔开的非负整数，表示 K_i。

输出：一行，即最少按键次数，若无法到达，则输出 -1。

输入示例	输出示例
5 1 5 3 3 1 2 5	3

样例分析

以输入为例：

楼层	K值
5	5
4	2
3	1
2	3
1	3

最优路径： `1 → 4 → 2 → 5`（3次按键）

1. 第一次按键：从1楼（$K_1=3$）按”上”到4楼
2. 第二次按键：从4楼（$K_4=2$）按”下”到2楼
3. 第三次按键：从2楼（$K_2=3$）按”上”到5楼（目标）

算法分析

1. 问题分析

   这是一个典型的最短路径问题，可以使用广度优先搜索（BFS）来解决。每个楼层是一个状态，每次按按钮是状态转移。

2. **状态转移**
   从当前楼层 u，可以转移到两个可能的楼层：
   - 向上：$u + K[u]$（如果不超过 N）
   - 向下：$u - K[u]$（如果不小于 1）
3. **BFS算法步骤**
   （1） 初始化距离数组，所有楼层设为-1（表示未访问）
   （2） 将起始楼层加入队列，距离设为0
   （3） BFS遍历：
      - 从队列中取出当前楼层
      - 如果到达目标楼层，返回当前距离
      - 否则，尝试向上和向下移动
      - 如果新楼层在范围内且未被访问，更新距离并入队
4. **边界条件处理**
   - 向上移动：检查是否不超过 N
   - 向下移动：检查是否不小于 1
   - 如果起点和目标相同，直接返回0

代码实现

```
while (!q.empty()) {
    int u = q.front();
    if (u == y) {
        cout << cost[u];
        return;
    }
    q.pop();
    for (int i = 0; i < 2; i++) {
        int v = u + a[u][i];
        if (v >= n || v < 0) continue;
        if (cost[v] > cost[u] + 1) {
            cost[v] = cost[u] + 1;
            q.push(v);
        }
    }
}
```

拓展思考

- 如果电梯可以有多个按钮，每个按钮对应不同的移动距离，该如何解决？
- 如果每次移动的代价不同（比如上楼比下楼费更多能量），该如何修改算法？
- 如果要求输出具体的按键序列而不仅仅是最少次数，该如何实现？