奇怪的电梯(lift.cpp)　　// 洛谷：P1135

【问题描述】

有一天我做了一个梦，梦见了一种很奇怪的电梯。大楼的每一层楼都可以停电梯，而且第 i 层楼(1<=i<=N)上有一个数字 Ki(0<=Ki<=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。当然，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 Ki(K1=3,K2=3,……)，从一楼开始。在一楼，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】(lift.in)

输入文件共有二行，第一行为三个用空格隔开的正整数，表示 N,A,B(1≤N≤200, 1≤A,B≤N)，第二行为 N 个用空格隔开的正整数，表示 Ki。

【输出】(lift.out)

输出文件仅一行，即最少按键次数,若无法到达，则输出-1。

【输入输出样例】

```
LIFT.IN
   5 1 5
   3 3 1 2 5
LIFT.OUT
   3
```

【参考程序一】

```
#include <iostream>
using namespace std;

struct q_node {
   int floor;
   int num;
};

q_node q[201];

int n, a, b, i, x;

//n 是总层数,a 是起始层,b 是终止层
int k[201];
bool visited[201];

int main()
{
   freopen("lift.in", "r", stdin);
   freopen("lift.out", "w", stdout);

   cin >> n >> a >> b;

   for (i=1; i<=n; i++)   cin >> k[i];
```

```
   if (a==b){
      cout << 0;
      return 0;
   }

   q[1].floor = a;   q[1].num = 0;
   visited[a] = true;
   f = 1;  r = 1;

   while (f<=r) {   //BFS,breadth-first search,宽搜
       temp = q[f].floor + k[q[f].floor];
       if (temp<=n && !visited[temp])
       {
          r++;
          q[r].floor = temp;
          q[r].num = q[f].num + 1;
          visited[temp] = true;
          if (temp==b)   break;
       }

       temp = q[f].floor - k[q[f].floor];
       if (temp>0 && !visited[temp])
       {
          r++;
          q[r].floor = temp;
          q[r].num = q[f].num + 1;
          visited[temp] = true;
          if (temp==b)  break;
       }

       f++;
   }

   if (f<=r) cout << q[r].num << endl;
   else cout << -1 << endl;

   fclose(stdin);
   fclose(stdout);
   return 0;
}
```

【参考程序二】{STL QUEUE}

```
#include <iostream>
#include <queue>
using namespace std;
```

```
struct q_node {
   int floor;
   int num;
};

queue<q_node> q;
int n, a, b, i, x, temp;
q_node x, y;

//n 是总层数,a 是起始层,b 是终止层
int k[201];
bool visited[201];

int main()
{
   cin >> n >> a >> b;

   for (i=1; i<=n; i++)   cin >> k[i];

   if (a==b){
      cout << 0;
      return 0;
   }

   x.floor = a;   x.num = 0;
   q.push(x);
   visited[x.floor] = true;

   while (!q.empty()) {
       y = q.front();
       x.floor = y.floor + k[y.floor];
       x.num = y.num + 1;
       if (x.floor<=n && !visited[x.floor])
       {
          q.push(x);
          visited[x.floor] = true;
          if (x.floor==b)   break;
       }

       x.floor = y.floor - k[y.floor];
       x.num = y.num + 1;
       if (x.floor>=1 && !visited[x.floor])
       {
```

```
            q.push(x);
            visited[x.floor] = true;
            if (x.floor==b)   break;
        }

        q.pop();
    }

    if (!q.empty())
      cout << q.back().num << endl;
    else
      cout << "-1" << endl;

    return 0;
}
```