一、简答

1. 在图的搜索中，求最短路径，无边权时使用队列，考虑边权时要用优先队列。一般来说，按照先进先出考虑的问题直接使用队列，出队时考虑某个权重时用优先队列。优先队列是基于堆实现的，插入是 $O(logn)$ 的时间复杂度，时间上相较队列的 $O(1)$ 差些。
2. 通过如最优化剪枝、可行性剪枝和记忆化的方法跳过冗余的搜索，使得搜索能在规定时间内完成。

二、实验E3

B.跳一跳

题目大意

“跳一跳”游戏

解法

模拟

时间复杂度

$O(n)$, n 为输入规模

代码

```
void solve() {
    ll score = 0;
    int a, pre = 2;

    while (cin >> a) {
        if (!a) break;

        if (a == 2) {
            score += pre;
            pre += 2;
        }
        else if (a == 1) {
            score += 1;
            pre = 2;
        }
    }

    cout << score << '\n';
}
```

C. 奇怪的电梯

题目大意

坐电梯，每层电梯可以坐到指定的楼层，问是否可以从A层坐到B层且最少坐几次电梯?

解法

BFS，从 A 点出发bfs，看是否可以搜索到 B，使用 $pair<int,int>$ 存储楼层和当前已乘坐几次电梯（使用数组存储后者也可）。

时间复杂度

$O(n)$

代码

```cpp
int N, A, B;
int K[205];
bool vis[205];
queue<pair<int, int> > Q;

void solve() {
    cin >> A >> B;

    for (int i = 1; i <= N; i++) {
        cin >> K[i];
    }

    if (A == B) {
        cout << "0\n";
        return;
    }

    while (!Q.empty())
        Q.pop();

    vis[A] = 1;
    Q.push({A, 0});


    while (!Q.empty()) {
        auto Now = Q.front();
        Q.pop();

        int f = Now.first;

        if (f + K[f] == B || f - K[f] == B) {
            cout << Now.second + 1 << '\n';
            return;
        }

        // 将合理以及未访问过的楼层加入队列
        if (f - K[f] >= 1 && !vis[f - K[f]]) {
            vis[f - K[f]] = 1;
            Q.push({f - K[f], Now.second + 1});
        }
        if (f + K[f] <= N && !vis[f + K[f]]) {
            vis[f + K[f]] = 1;
            Q.push({f + K[f], Now.second + 1});
        }
    }

    cout << "-1\n";
}

int main() {
```

```
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    while (cin >> N) {
        if (!N) break;
        memset(vis, false, sizeof vis);
        solve();
    }

    return 0;
}
```