只使用标准库，实现一个双子塔建筑（分别为A塔，B塔）内的最短路径搜索系统，满足

- 使用Dijkstra算法计算两点间最短路径
- 动态处理电梯权重参数以适应不同场景
- 支持跨塔连接和特殊楼层结构

只需要写出 `reference.h` ，提供一个函数的实现

```
int FindShortestPath(
    std::string start,
    std::string end,
    std::vector<std::string>& path,
    int** eWeights  // 电梯权重矩阵
);
```

其中 `start` ， `end` 是起终点的房间号； `path` 会记录沿途的节点（含起终点），`ewights` 这是一个4×m矩阵，包含每层楼电梯节点的出弧权重。m是楼层数。矩阵的第一行 `eWeights[0][n-1]` 包含电梯AEn1的出弧权重，n是楼层号；第二行 `eWeights[1][n-1]` 是针对AEn2，第三行 `eWeights[2][n-1]` 是针对BEn1，第四行 `eWeights[3][n-1]` 是针对BEn2。在特定电梯的特定楼层，电梯节点的权重可能会非常大，以至于可以通过例如**横梁**(见下文说明)找到最短路径。

返回值是全程权重的总和，也就是距离。

测试函数会负责初始化双子塔的楼高，电梯的权重。在计算完距离之后，提取 `path` 里面的记录观察算法的执行。测试函数同时还会记录算法的执行时间，**请追求最小的时间复杂度并通过内存的操作尽量加速**。

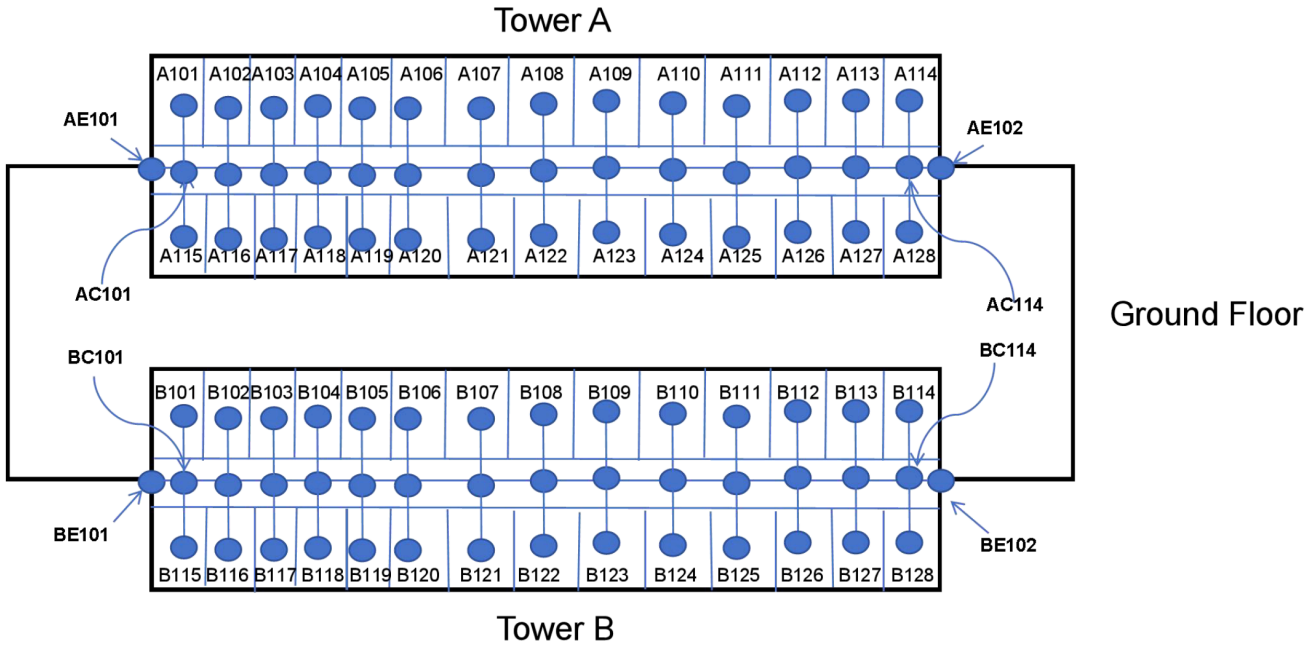
节点定义

每一层楼固定为28个房间，出发点和到达点是房间，其他时候要么处于电梯，要么处于走廊。

办公室	[塔楼][楼层][房间号]	A125，B923

走廊	[塔楼]C[楼层][节点号]	AC1109
电梯	[塔楼]E[楼层][电梯编号]	BE1101

相对位置如下，以一楼为例



边权重规则

A-B塔连接（仅1楼）	固定100
电梯跨楼层连接	由输入参数eWeights动态决定
电梯↔走廊连接	固定8
普通走廊连接	固定5
每10层特殊**横梁**连接	等同对应楼层电梯AE101的权重

连接相邻楼层之间两个电梯节点的边的权重是作为输入参数的变量。电梯节点的连接只能跨越一层楼。（意思就是上02→04楼要吃两次权重而不是一次）

特殊结构：在A塔每10层(如第10、20、40层...)，都有一条双向横梁连接A(n)22和B(n-1)08，另一条双向横梁连接A(n)23和B(n+1)09。这些横梁的权重与电梯AE101相同。