## 1. 最短路径数量

假设棋子不能“回头”从而满足最短路径要求。不妨令棋子起点为棋盘左上角、终点为棋盘的右下角。

(1) **动态规划**

① 子问题：要求移动到另一对角的最短路径数量，需要得到之前到达每个位置的最短路径数量，因此我们利用一个和棋盘大小相同的二维数组记录每个位置的最短路径数，即 dp[i][j]表示由起点到达棋盘位置(i, j)的最短路径数量，并且满足无后效性。

② 递推关系：对于除了位于第零行和第零列的方格，其他方格的最短路径数量可以通过该方格上一个和左边两个方格的最短路径数量加合求得，即 dp[i][j] = dp[i - 1][j] + dp[i][j - 1]。

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 3 | 6 | 10 | 15 | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |
| 1 | | | | | | | |

以此类推，直到求出目标行列的最短路径数量。

③ 初始化：对于第零行和第零列的方格方法数量初始化为 1。

④ 实现代码：

```
int minPathNum(int row, int col) {
    vector<vector<int>> dp(row, vector<int>(col, 0));
    // 初始化第一行第一列
    for (int i = 0; i < row; ++i)
        dp[i][0] = 1;
    for (int j = 0; j < col; ++j)
        dp[0][j] = 1;
    // 填表
    for (int i = 1; i < row; ++i)
        for (int j = 1; j < col; ++j)
            dp[i][j] = dp[i - 1][j] + dp[i][j - 1];
    return dp[row - 1][col - 1];
```

}

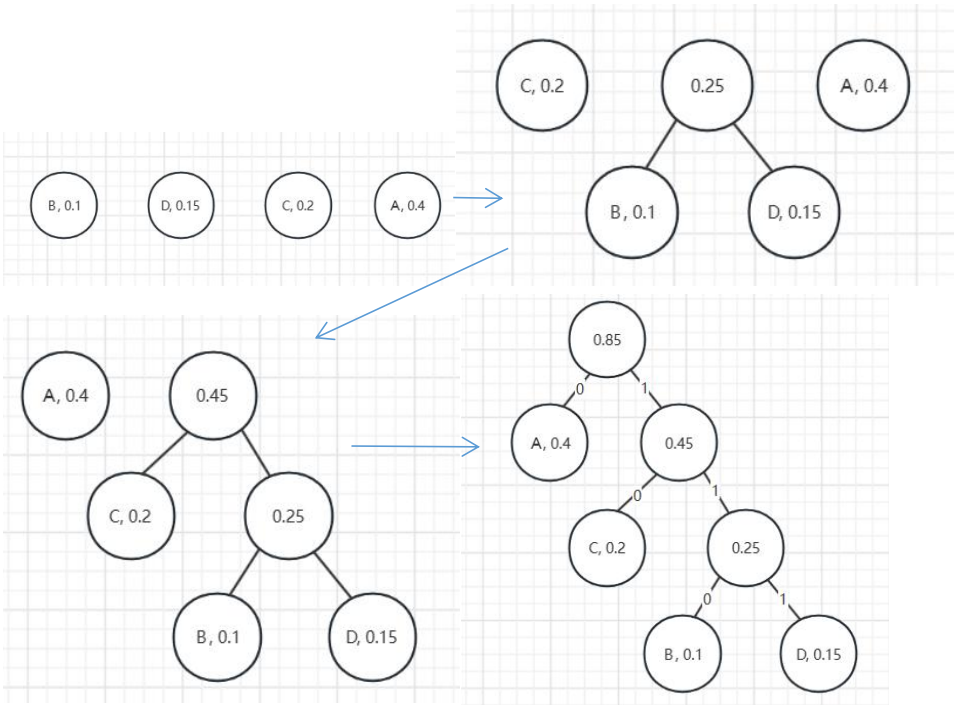
(2) **排列组合**

不妨设棋盘大小为 m 和 n（m 行 n 列），从左上角开始，从一个对角到达另一个对角需要走的步数为 m + n - 2，同时在棋子移动的过程中，选择 m - 1 步向下——因此共有 $C_{m+n-2}^{m-1}$ 种选择方法。

## 2. 哈夫曼编码

### (1) 哈夫曼编码

通过哈夫曼树构建哈夫曼编码，过程如下：




由此可以得出哈夫曼编码：

A: 0

B: 110

C: 10

D: 111

对文本 ABACABAD 进行编码

结果为：0-110-0-10-0-110-0-111，即 011001001100111

(2) **解码**

10-0-0-10-111-0-0-10-10

解码结果为：CAACDAACC