[14]搜索

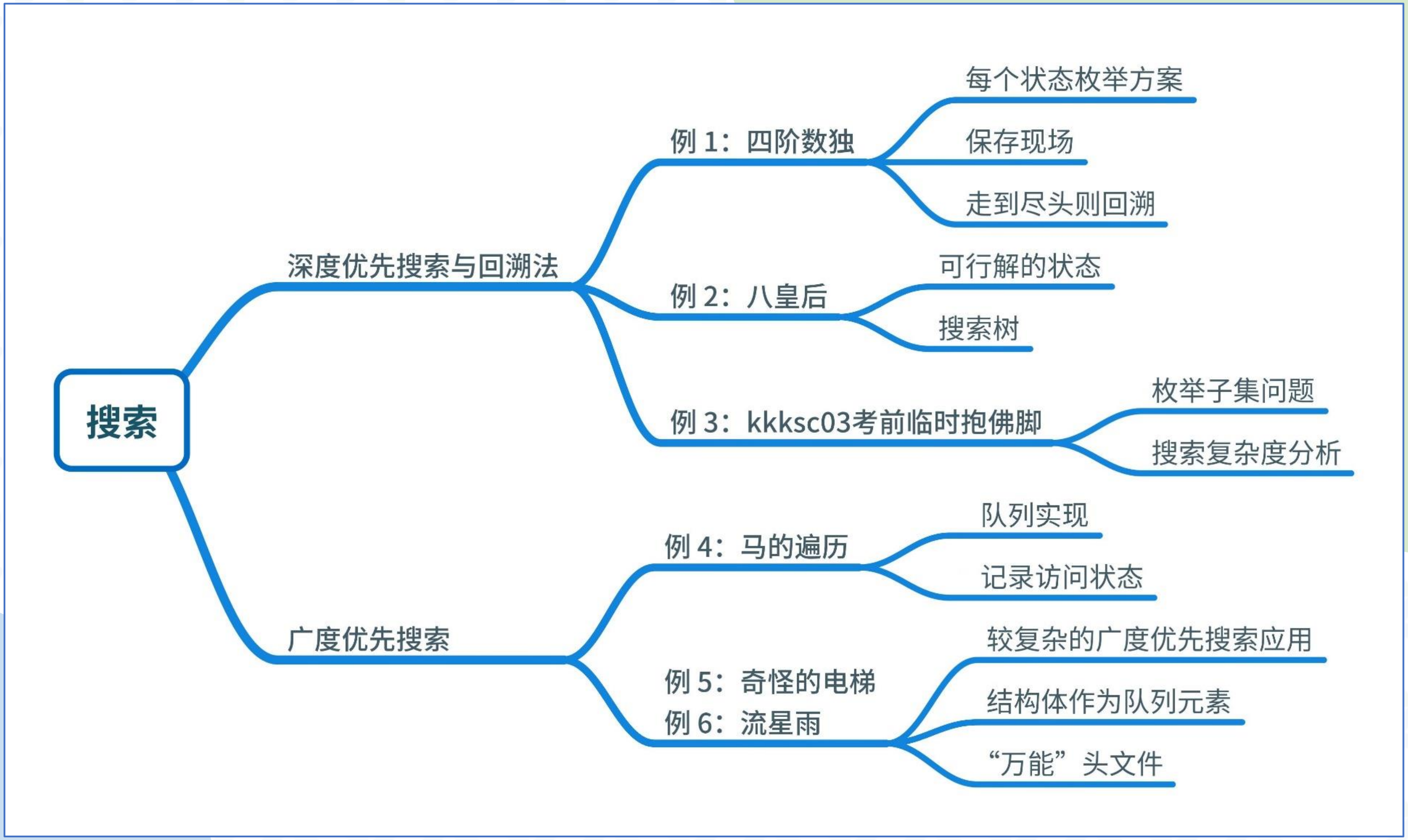
深入浅出程序设计竞赛
第 2 部分 – 初涉算法
V 2021-04

洛谷
www.luogu.com.cn

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图



搜索
深度优先搜索与回溯法
例 1：四阶数独
每个状态枚举方案
保存现场
走到尽头则回溯
例 2：八皇后
可行解的状态
搜索树
例 3：kkksc03考前临时抱佛脚
枚举子集问题
搜索复杂度分析
广度优先搜索
例 4：马的遍历
队列实现
记录访问状态
例 5：奇怪的电梯
例 6：流星雨
较复杂的广度优先搜索应用
结构体作为队列元素
“万能”头文件

第 14 章 搜索

[深度优先搜索与回溯法]

[广度优先搜索与洪泛法]

[课后习题与实验]

回顾：枚举法

在第 10 章中，我们学习了枚举法，即按照一定顺序，不重复、不遗漏地逐个尝试。我们掌握了：

- 使用多重循环的经典枚举手段
- 通过数学工具“剪枝”降低枚举规模
- 基于二进制“状态压缩”表达子集状态

例如，迷宫坐标

在更复杂的问题中，状态无法简单用单一变量表示。

同时，当数据范围扩大时，即便问题本质上仍为子集（指数）枚举或排列（阶乘）枚举，但很多状态是无效的。

搜索，是一种具有策略的枚举法。它许利用数据结构表示状态，并借此更有针对性地剪枝，从而求解传统枚举难以解决的问题。

深度优先搜索与回溯法

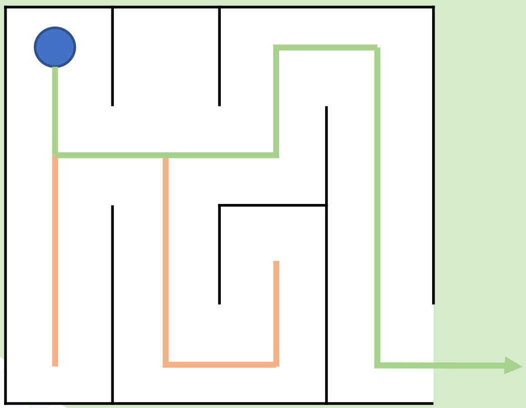
如果你走迷宫时进入了死胡同，是不是应该退回来继续走呢？

请翻至课本 P186

回溯法

递归，即函数调用自身，以逐步减小问题的规模。但在一些问题中，并不是所有的递归路径都是有效的。

如图所示迷宫，很可能会进入橙色所标识的“死胡同”，只能回到之前的路径，直到找到绿色的解为止。



这种方法被称为回溯法。

回溯法往往会尝试一条尽可能深而完整的搜索路线，直至完全无法继续递归时才回溯，因而需要用深度优先搜索（DFS）实现。

四阶数独

例 14.1

这里讨论一种简化的数独——四阶数独。

给出一个 4×4 的格子，每个格子只能填写 1 到 4 之间的整数，要求每行、每列和四等分更小的正方形部分都刚好由 1 到 4 组成。

右图是一个合法的四阶数独的例子。

2	4	1	3
1	3	2	4
4	2	3	1
3	1	4	2

给出空白的方格，请问一共有多少种合法的填写方法？

四阶数独

解法 0：使用暴力枚举法。复杂度$O(n^{n^2})$。

一共有4×4=16个空格。使用16层循环。

每个空格可填入1~4共4个选项。总情况数 $4^{16}= 4294967296$。

$$4^{16} = (2^2)^{16} = 2^{32}$$

即比32位无符号整数的最大值max_unsigned_int恰好多1。

目前计算机一秒约可以处理10^7（一千万）次有效计算。

总之肯定是不行了。

四阶数独

解法 1：使用排列枚举法。复杂度$O(n^2(n!)^n)$

由于已经知道每一行内的数字不可能出现重复，每一行均可视为 1~4的一个排列。因此可以使用排列枚举。

1	2	3	4

1	2	4	3

1	3	2	4

使用类似于计数排序的思路，使用数组维护每一列、每一个子正方形中 1、2、3、4 是否出现，可以 O(1) 完成合法性判断。

总情况数$(4!)^2=24^4=331776$。

总运算次数$331776\times 16=5308416$，勉强压入1秒。

提问：为什么不需要判断每一行？

洛谷

四阶数独

解法 2: 使用回溯法。

1			
~~1~~	~~1~~		

第一行选择第一列为 1，则:

第 2 行将 1 置入第一列，将会违反列规则和子正方形规则。

第 2 行将 1 置入第二列，将会违反子正方形规则。

然而，由于使用循环进行排列枚举，不得不生成后方的所有排列之后，才能进行判断是否合法，产生了大量的浪费。

我们可以总结得到经验：减少浪费 = 尽早剪枝!

四阶数独

本例中，最基本的行为单元为“填上单个数字”。

因此使用回溯法时，每填入一个数字，都立刻进行判断并剪枝。

```
void dfs(int x) { // 第x个空填什么
    if (x > n) { // 如果所有空已经填满
        ans++; // 增加结果数量
        return;
    }
    // 根据x计算出所在行、列、小块编号
    int row = (x - 1) / 4 + 1; // 横行编号
    int col = (x - 1) % 4 + 1; // 竖排编号
    int block = (row - 1) / 2 * 2 + (col - 1) / 2 + 1; // 小块编号
    // 枚举所填内容为i
    for (int i = 1; i <= 4; i++)
        if (b1[row][i] == 0 && b2[col][i] == 0 && b3[block][i] == 0) {  // 合法性判断
            a[x] = i; // 记录放置位置
            b1[row][i] = 1; b2[col][i] = 1; b3[block][i] = 1; // 占位
            dfs(x + 1); // 下一层递归
            b1[row][i] = 0; b2[col][i] = 0; b3[block][i] = 0; // 取消占位
        }
}
```

四阶数独：详细分析

细节 1：如何保证放置方法合法？

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

编号x
(a)

1	1	1	1
2	2	2	2
3	3	3	3
4	4	4	4

row=(x-1)/4+1
(b)

1	2	3	4
1	2	3	4
1	2	3	4
1	2	3	4

col=(x-1)%4+1
(c)

1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

block=(row-1)/2*2+(col-1)/2+1
(d)

小提示：从 0 开始编号有惊喜~

如图，可以构造方格编号与行列号之间的关系。

使用三个数组b1、b2、b3。b1[i][j] 表示第 i 行是否已经存在 j。

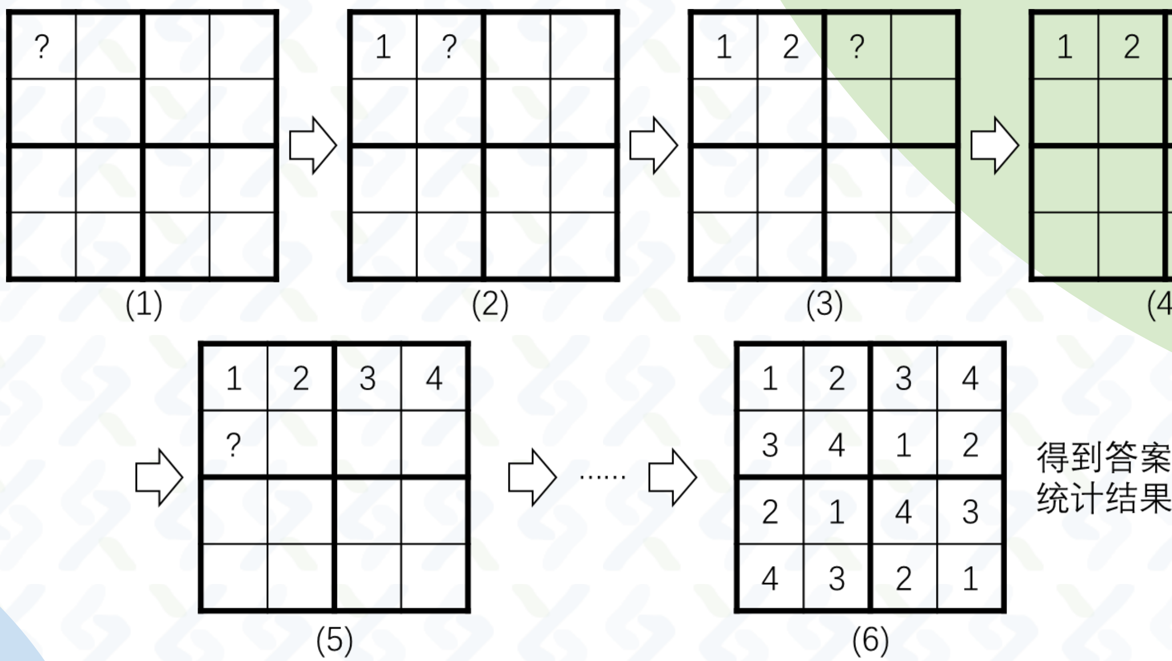
类似地，b2、b3分别表示第 i 列、第 i 块是否已经存在 j。

方案合法，当且仅当新数与现有的 b1、b2、b3 不冲突！

洛谷

四阶数独：详细分析

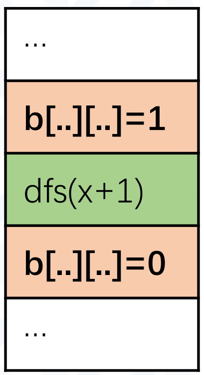
细节 2：递归是如何进行的？观察前几步的操作：请翻至课本 P188

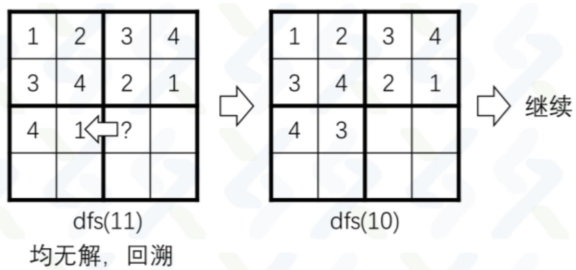


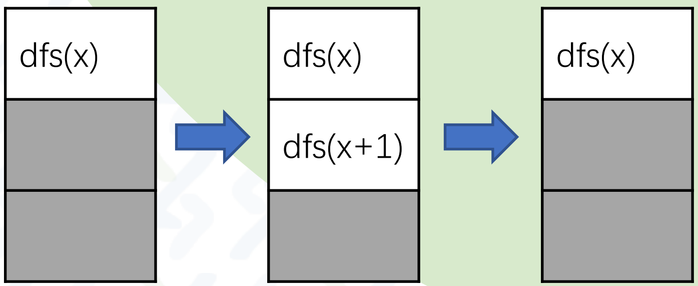
四阶数独：详细分析

细节 3：为什么函数返回后就能回到上一状态?

如果所有方法枚举完毕，则这条路已经死胡同，需要回溯!







计算机运行函数时，为每一个子函数都分配了一片栈空间。

当回溯到上层时，下层的内容会离开栈，从而恢复上层的状态。

注意 b 是全局数组，并不能随着出栈恢复，需要手动清理!

四阶数独

引入递归法后，需要枚举状态数量级别，相对于枚举排列并没有实质性的提升。因为都是指数级（每次选择都有 4 个）。

但是由于删除了大量无需计算的状态，运行时间会有明显的提升。

回溯法的基本模板如下：

```
void dfs(int k) { // k代表递归层数，或者说要填第几个空
    if (所有空已经填完了) {
        判断最优解/记录答案;
        return;
    }
    for (枚举这个空能填的选项)
        if (这个选项是合法的) {
            记录下这个空（保存现场）;
            dfs(k + 1);
            取消这个空（恢复现场）;
        }
}
```

算法竞赛中，如果无法找到高效求解的方法（如贪心、递推、动态规划、公式推导等），使用搜索也可以解决一些规模较小的情况。

但不管怎么说，时间复杂度往往是指数级别的，效率相比于多项式时间复杂度还是要低。

八皇后

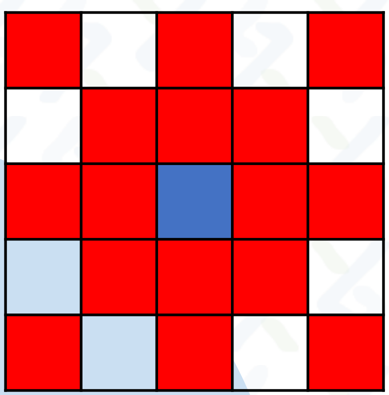
例14.2 （洛谷 P1219，USACO Training）

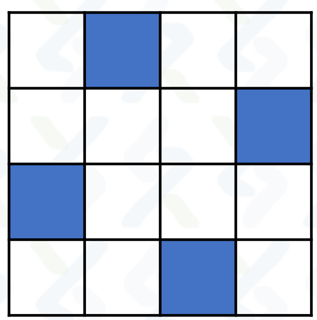
在 n×n 的国际象棋棋盘上放置 n 个皇后使得她们互不攻击。皇后的攻击范围是同一行、同一列、在同斜线上的其他棋子。

n 不超过 13，求方案数，并输出前 3 种放法。

```
6
```

```
2 4 6 1 3 5
3 6 2 5 1 4
4 1 5 2 6 3
4
```





左图是皇后的影响范围示例；
右图是一种四皇后的放法。

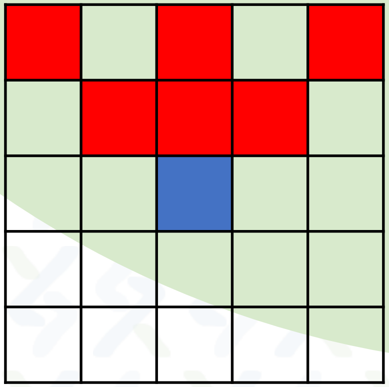
n皇后

本题是经典的搜索回溯例题。和例 1 一样套用回溯法基本模板。

如何表达状态，便于判定皇后的合法性，也就是不会互相攻击。

只需要考虑上方的是否重复。为什么？

- 行：因为按行枚举，所以每一行显然只能有一个皇后。
- 列/斜角：循环枚举判断是否有重复列或者斜对角？可行，但要增加 O(n) 的复杂度，用于遍历上方所有皇后。



能否像例 1 一样，使用一个占位标记实现这一判定？

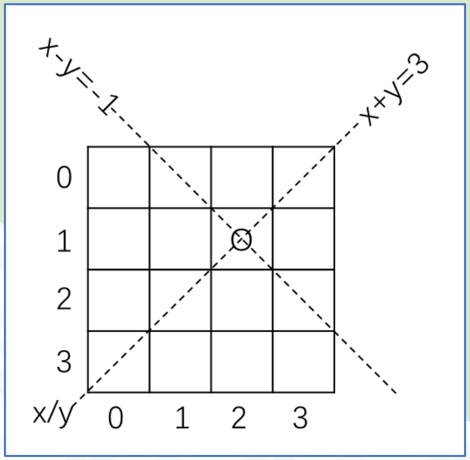
n皇后

需要判定：正上方、左对角、右对角上没有皇后。

正上方容易判断，直接为每一列上一个标记是否存在即可。

对角线上，可以发现一些性质：

- 左下右上斜线：
  这些格子，（横坐标+纵坐标）的值相同
- 左上右下斜线：
  这些格子，（横坐标-纵坐标）的值相同



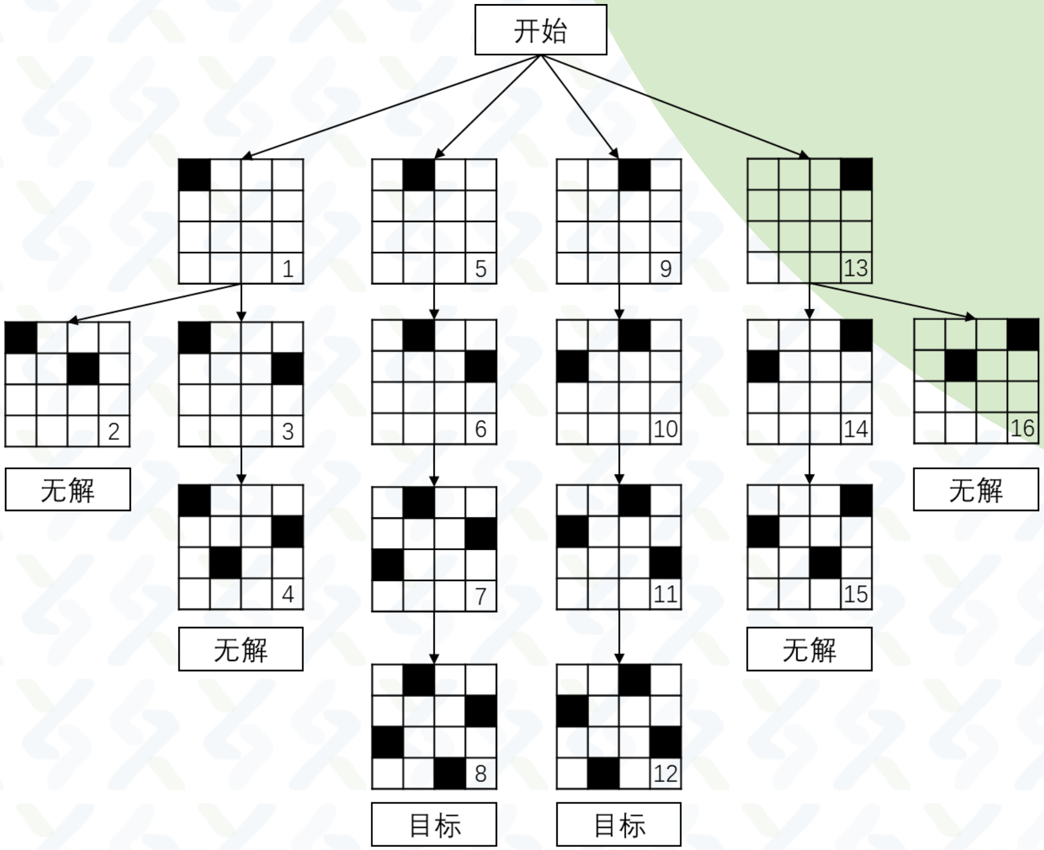
所以就建立三个数组表示列、两个斜线的状态。

（横坐标-纵坐标）可能是负数，加上 n 即可非负。

其实就是一次函数表达式啦

n皇后

观察当 n=4 时的搜索过程：



n皇后

最终可得代码。建立好标记数组；枚举各状态；然后恢复现场。

```
int a[maxn], n, ans = 0;
int b1[maxn], b2[maxn], b3[maxn];
// 分别记录y,x+y,x-y+15是否被占用
void dfs(int x) { // 第x行的皇后放哪儿
    if (x > n) { // 如果所有皇后已经放置
        ans++; // 增加结果数量
        if (ans <= 3) { // 输出前三种答案
            for (int i = 1; i <= n; i++)
                printf("%d ", a[i]);
            puts("");
        }
        return;
    }
    for (int i = 1; i <= n; i++)
        if (b1[i] == 0 && b2[x + i] == 0 && b3[x - i + 15] == 0) {
            a[x] = i; // 记录放置位置
            b1[i] = 1; b2[x + i] = 1; b3[x - i + 15] = 1; // 占位
            dfs(x + 1); // 下一层递归
            b1[i] = 0; b2[x + i] = 0; b3[x - i + 15] = 0; // 取消占位
        }
}
```

kkksc03考前临时抱佛脚

例 14.3 （洛谷 P2392）

有四个科目的作业，每个科目有不超过 20 题，解决每道题都需要一定的时间。kkksc03 可以同时处理同一科目的两道不同的题。

求他完成所有题目所需要的时间。

```
1 2 1 3
5
4 3
6
2 4 3
```

```
20
```

这题在“暴力枚举”中习题中见过！
可以枚举子集！

kkksc03考前临时抱佛脚

学科之间无关，可以单独考虑每个学科。将每一学科的作业分为两组，使得两者分别的时间开销之和尽可能接近。

1 1 4 5 2 4 第1组：1+2+4=7 第2组：1+4+5=10	1 1 4 5 2 4 第1组：1+1+2+4=8 第2组：4+5=9

- 第 1 组：少于一半，但尽可能大。
- 第 2 组：剩下的一半，总时间以这个为准。

使用子集枚举方法，要枚举共2^n种可能，还需要额外$O(n)$的时间计算每组的和。考虑使用回溯法进行优化，去除无效状态。

- Case 1：当前的子集之和已经大于总和的一半，后面都无效！
- Case 2：如果当前子集已经恰好等于总和的一半，最优解！

kkksc03考前临时抱佛脚

虽然没有使用枚举子集，但是利用了搜索的有序性，不重复、不遗漏地对每一项作业决定，是否需要加入当前集合。

```
void dfs(int x){
    if(x > maxdeep){ // 所有作业枚举完毕，达到了最大递归层数
        maxtime = max(maxtime, nowtime); // 如果解更优，更新答案。
        return;
    }
    if(nowtime + a[x] <= sum / 2){ // 如果放入这个作业是合法的，选择它
        nowtime += a[x]; // 增加子集中这道题目的时间
        dfs(x + 1); // 下一层递归
        nowtime -= a[x]; // 去除掉子集中这道题目的时间
    }
    dfs(x + 1); // 不选这个题目，直接进行下一层递归
}
```

可以换位思考：不仅可以为每一个集合选择元素，亦可以为每一个元素选择集合。

广度优先搜索与洪泛法

百一定律：百人中可能只有一人会带来突破性进展；但由于预先并不知道是谁，所以决策者需要雇佣全部100个人，以期其中包含正确的选择。

请翻至课本 P93

洪泛法

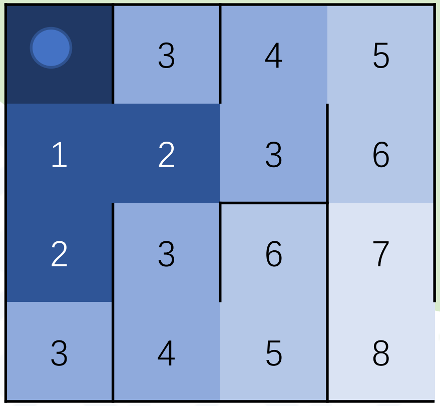
深搜会尽快完成一个可行的解，再回溯尝试其他的可能性。

当解相对稀疏，或问题很大时，深搜可能陷入过深、过窄的“陷阱”。

考虑另一种思路，从起点出发，类似于泼水一般，让水流顺着多个方向同时蔓延。

这种方法被称为洪泛法。

洪泛法会扩展相同层更多的可能性以拓宽广度，往往会使用广度优先搜索（BFS）实现。



马的遍历

广度优先搜索的基本模板：

```
Q.push(初始状态); // 将初始状态入队
while (!Q.empty()) {
    State u = Q.front(); // 取出队首
    Q.pop();//出队
    for (枚举所有可扩展状态) // 找到u的所有可达状态v
        if (是合法的) // v需要满足某些条件，如未访问过、未在队内等
            Q.push(v); // 入队（同时可能需要维护某些必要信息）
}
```

广度优先搜索可以保证在求解最近、最短、最快等一类问题时，搜索到的首个解就是最优解。

队列： 暂时还没有讲到。可以认为是一个容器，先进入的元素可以先被取出。具体可以看课本 P211。

马的遍历

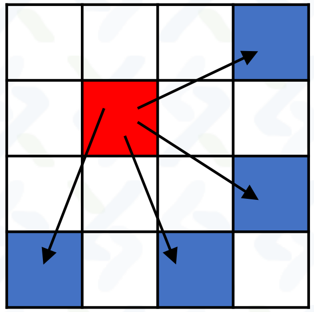
例 14.4 （洛谷P1443）

有一个 $n \times m$ 的棋盘 $(1 < n, m \le 400)$，在某个点上有一个马，要求你计算出马到达棋盘上各个点最少要走几步。

如果到不了就输出 -1。

```
3 3 1 1
```

```
0    3    2
3    -1   1
2    1    4
```



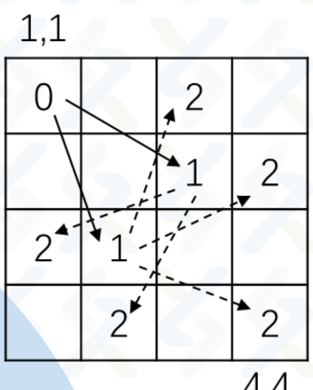
马的遍历

求“最少”步数，使用洪泛法。广度优先搜索使用队列实现。

每次从队首取出元素，将该元素所能扩展到的结果插入队尾。

这样即可保证，在同一层的其他元素均被取出之前，不会访问到下层的新元素。

例如，当输入为4 4 1 1时（马位于4x4棋盘的左上角）：



(a)

q[]	0	1	2	3	4	5	6	7
信息	d=0 (1,1)	d=1 (3,2)	d=1 (2,3)	d=2 (4,4)	d=2 (2,4)	d=2 (1,3)	d=2 (3,1)	d=2 (4,2)

(b)

马的遍历

使用 STL 的 queue 实现队列。建立结构体数组存储扩展的结点。

让起点入队，然后在队列逐个扩展。每个点被扩展到时步数最小。

```
struct coord { //一个结构体存储x,y两个坐标
    int x, y;
};
queue<coord> Q;//队列
int ans[maxn][maxn];//记录答案，-1表示未访问
int walk[8][2] = {{2, 1}, {1, 2}, {-1, 2}, {-2, 1},
    {-2, -1}, {-1, -2}, {1, -2}, {2, -1}
```

```
coord tmp = {sx, sy};
Q.push(tmp); // 使起点入队扩展
ans[sx][sy] = 0;
while (!Q.empty()) { // 循环直到队列为空
    coord u = Q.front();  // 拿出队首以扩展
    int ux = u.x, uy = u.y;
    Q.pop();
    for (int k = 0; k < 8; k++) {
        int x = ux + walk[k][0], y = uy + walk[k][1];
        int d = ans[ux][uy];
        if (x < 1 || x > n || y < 1 || y > m || ans[x][y] != -1)
            continue;  // 若坐标超过地图范围或者该点已被访问过则无需入队
        ans[x][y] = d + 1;  // 记录答案，是上一个点多走一步的结果。
        coord tmp = {x, y};
        Q.push(tmp);
    }
}
```

因为每个点只被扩展一次，故复杂度是 O(mn) 。

奇怪的电梯

例14.5 （洛谷P1135）

$N(N \leq 200)$ 层大楼，有一部奇怪的电梯。

大楼的每层楼都可以停电梯，而且第 $i(1 \leq i \leq N)$ 层楼上有一个数字 $K_i(0 \leq K_i \leq N)$。

电梯只有两个按钮：上，下。上下的层数为当前楼层上的数字。如果不能满足要求，相应的按钮就会失灵。

从 A 楼到 B 楼至少要按几次按钮呢？

```
5 1 5
3 3 1 2 5
```

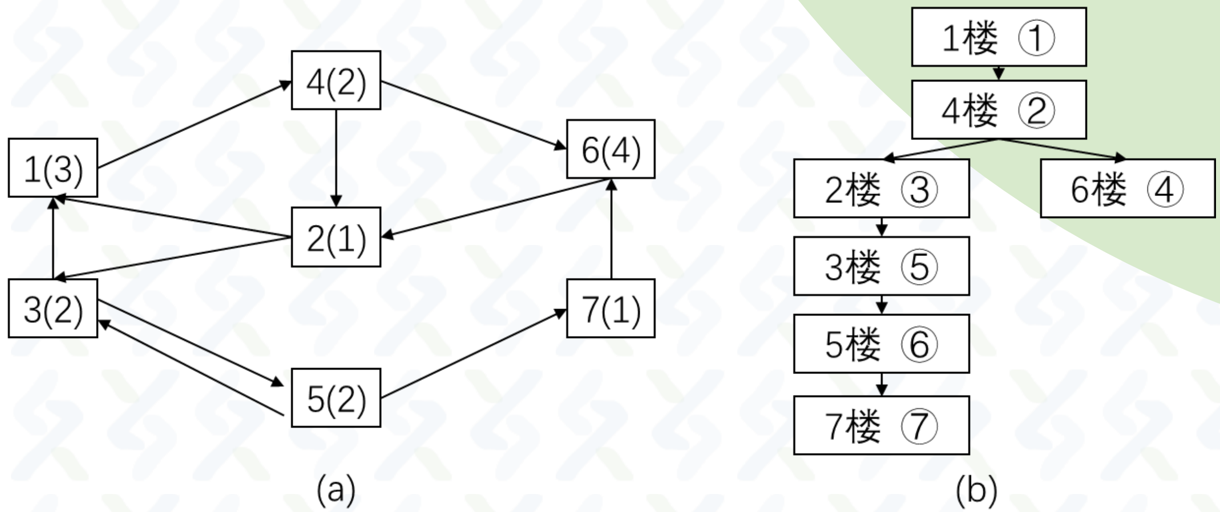
```
3
```

奇怪的电梯

本题的基本思路和上题完全一致。仅仅是空间由二维变为一维，同时移动规则由马步变为给定的数字。

```
5 1 5
3 3 1 2 5
```

可以将模型转化为直观的图：



其中a为可达关系（注意图中的边是有向的），b为访问顺序。

奇怪的电梯

代码和之前的差不多，也是使用队列来扩展。

```
Q.push((node){a, 0});  //将初始元素加入到队列
vis[a] = 1;  //记录初始楼层已访问过
node now;
while (!Q.empty()) {
  now = Q.front();
  Q.pop();
  if (now.floor == b) break;  // 找到目标解
  for (int sign = -1; sign <= 1; sign += 2) { // sign枚举-1和1
    int dist = now.floor + k[now.floor] * sign;  // 目标楼层，sign为1是上
    if (dist >= 1 && dist <= n && vis[dist]==0) {
    // 如果按按钮能到达的楼层有效并且未访问过该楼层
    Q.push((node){dist, now.d + 1});
    vis[dist] = 1;  // 该楼层为已访问过
    }
  }
}
if (now.floor == b) // 找到目标解
cout << now.d << endl;
else // 无法到达
cout << -1 << endl;
```

```
struct node {
    int floor, d;  //队列中记录的层数和按钮次数
};
queue<node> Q; // 广度优先搜索的队列
```

小技巧：万能头文件

当需要使用的头文件较多时，我们也有办法一次性导入C++标准中的所有的头文件。只需引用这个，就不需引用其它头文件啦：

```
#include<bits/stdc++.h>
```

可以明确的是，目前在NOI系列比赛中，该头文件可以使用。

注意：并不是所有的环境都支持。请谨慎确认。

若该文件和 using namespace std; 一同使用，则很多可用标识符（即变量/函数名称）已经被标准库使用，会导致编译错误。

例如，全局变量的 y1 等。

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P199

复习

回溯法/深度优先搜索（右图 a）

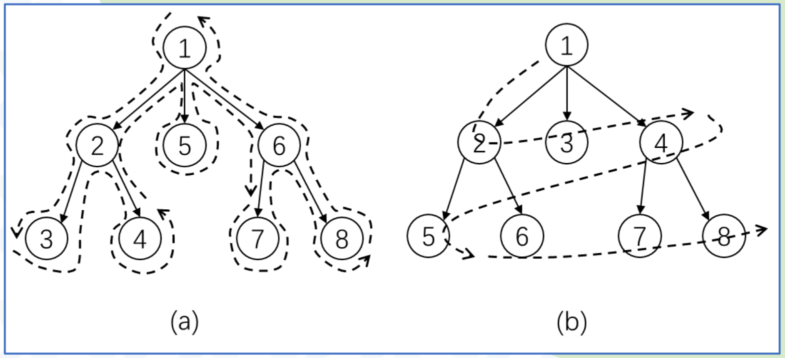
快速构造解，使用递归。不撞南墙心不死。

但进入死路就回头了。

洪泛法/广度优先搜索（右图 b）

寻找最优解，使用队列

从起点开始，逐层往外扩展。



优化技巧：将不可能的解提前剪掉。

万能头文件：

```
#include<bits/stdc++.h>
```

作业

习题 14.1：考察以下任务（“暴力枚举”的题目），用回溯搜索实现。

选数（洛谷 P1036，NOIP2002 普及组）
Perket（洛谷 P2036）
吃奶酪（洛谷 P1433）

习题 14.2：迷宫（洛谷 P1605）

给定一个 $N \times M(1 \leq N, M \leq 5)$ 方格的迷宫，迷宫里有 T 处障碍，障碍处不可通过。给定起点坐标和终点坐标。在迷宫中移动有上下左右四种方式，每次只能移动一个方格。

问：每个方格最多经过 1 次，有多少种从起点坐标到终点坐标的方案。数据保证起点上没有障碍。

作业

习题 14.3 单词接龙（洛谷P1019，NOIP2000 提高组）

已知一组不超过 20 个单词，且给定一个开头的字母，要求出以这个字母开头的最长的“龙”（每个单词都最多在“龙”中出现两次）。

在两个单词相连时，其重合部分合为一部分，例如 beast 和 astonish，如果接成一条龙则变为 beastonish。

另外相邻的两部分不能存在包含关系，例如 at 和 atide 间不能相连。输出以此字母开头的最长的“龙”的长度。

```
5
at
touch
cheat
choose
tact
a
```

```
23

//可以组成
atoucheatactactouchoose
```

作业

习题 14.5 自然数的拆分问题 （洛谷P2404）

给出一个自然数 $n(n \leq 8)$，求出 n 的拆分成一些数字的和。每个拆分后的序列中的数字从小到大排序。

输出这些序列，其中字典序小的序列需要优先输出。

```
7
```

```
1+1+1+1+1+1+1
1+1+1+1+1+2
1+1+1+1+3
1+1+1+2+2
1+1+1+4
1+1+2+3
1+1+5
1+2+2+2
1+2+4
1+3+3
1+6
2+2+3
2+5
3+4
```

作业

习题 14.6 湖计数（洛谷 P1596，USACO 2010 October）

由于降雨，雨水汇集在田地。用 $N \times M(1 \leq N, M \leq 100)$ 网格图表示。每个网格中有水(W) 或是旱地(.)。

一个网格与周围八个网格相连，而一组相连的网格视为一个水坑。

给出约翰田地的示意图，确定当中有多少水坑。

提示：请尝试分别用深度优先搜索和广度优先搜索解决这个问题。

作业

习题 14.7 填涂颜色（洛谷 P1162）

由数字 0 组成的方阵中，有一任意形状闭合圈，闭合圈由数字 1 构成，围圈时只走上下左右 4 个方向。

要求把闭合圈内的所有空间都填写成 2 。

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 0 0 1
1 1 0 0 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

```
0 0 0 0 0 0
0 0 1 1 1 1
0 1 1 2 2 1
1 1 2 2 2 1
1 2 2 2 2 1
1 1 1 1 1 1
```

参考阅读材料

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

P97 例 14.6：较复杂的广度优先搜索应用

习题 14.4、14.8、14.9。

另外，综合深度优先和广度优先的长处，还有：

- 迭代加深深搜：行为上类似广搜，但仅需要深搜的空间；代价是较浅的层需要被反复搜索。
- 迭代增广广搜：行为上类似深搜，但如果策略合理，有几率快速寻找到最优答案；代价是策略失效时需要更多的查找。