NOIP

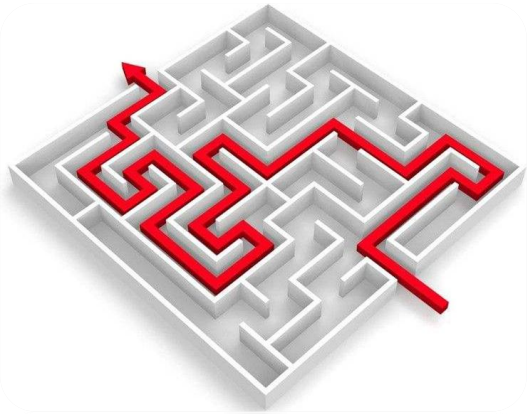
搜 索

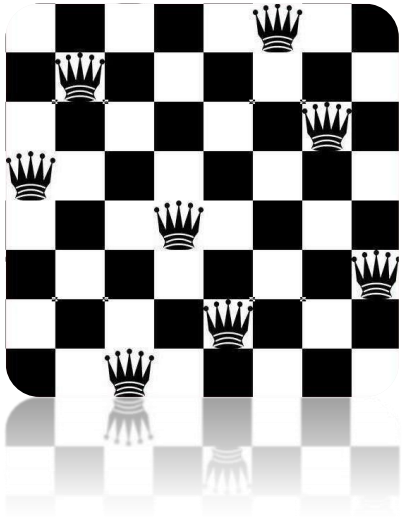
搜索及其优化

江苏省扬州中学 刘超

前言

当我们面对一个算法问题时，如果能找到数学方法（如递推法、构造法）或者类似贪心、动态规划求最优值的方法时，那么对于这道题而言，已经基本解决。如果没有找到行之有效的方法，搜索便成了不错的选择。常见的搜索手段有：穷举、深度优先搜索、宽度优先搜索等。





搜索是万能的，但没有优化的搜索却万万不能。

目录
Index

OI生涯需要掌握的搜索算法（建议顺序）

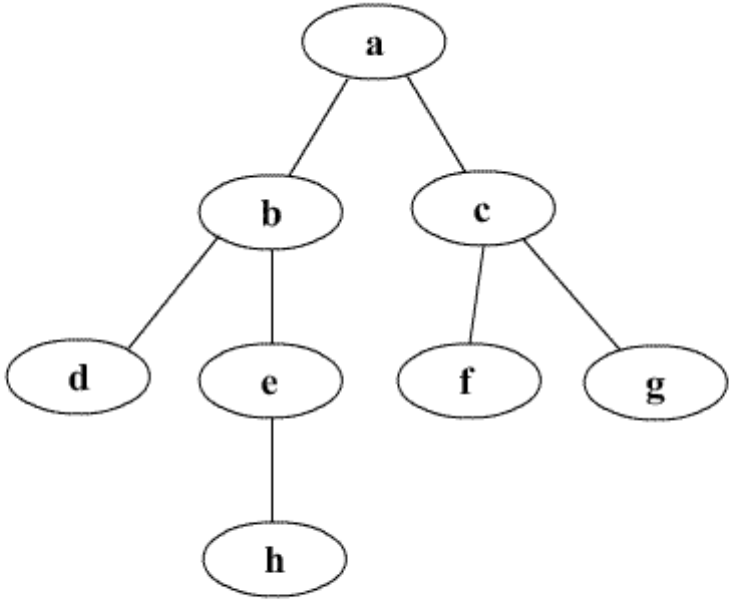
- 深搜、宽搜（特点、结构及应用）
- 深搜优化：剪枝、记忆化、迭代加深搜索等
- 宽搜优化：双向宽搜、哈希判重等

搜索的状态

- 状态：决定了我们的搜索对象。影响搜索的效率、思维复杂度和代码实现复杂度~
- 如何定义搜索状态：经验 经验 经验！
- 做题百道，其义自见！

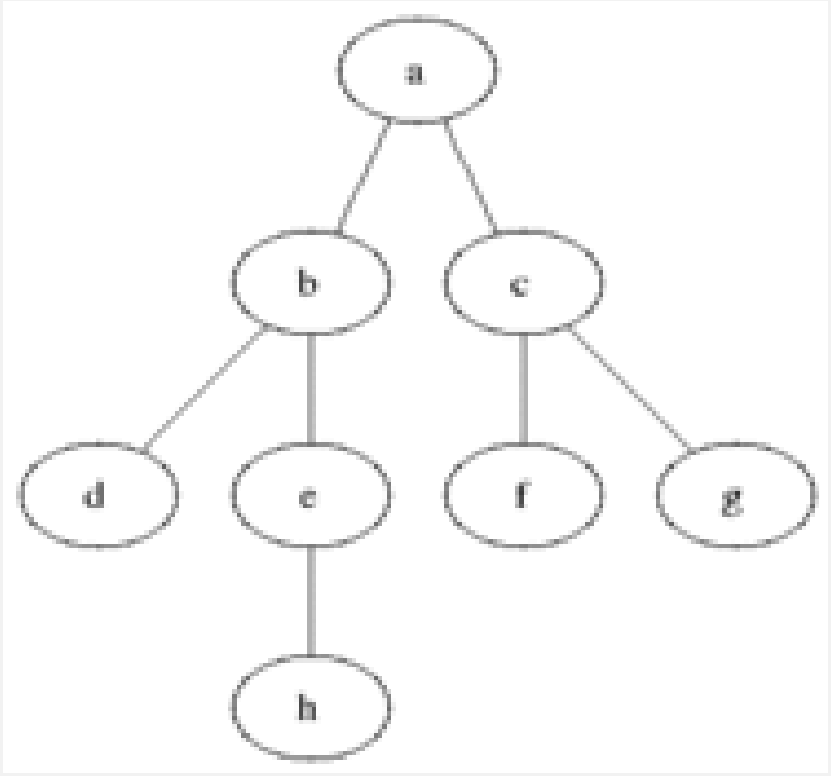
—— 杭二 李建

搜索的过程（搜索树）



dfs:深度优先

适合：求可行解或全部解问题



bfs：广度优先

适合：求最优解问题

搜索的实现（dfs）

```
void dfs(int dep,[参数表])
{
    if(当前是目标状态) 输出解或者作计数、评价处理;
    else for(int i = 1; i <= 状态的拓展可能数; ++i)
    {
        保存现场(断点,维护参数表);
        if(第i种状态拓展可行) dfs(dep+1,[参数表]);
        //恢复现场(回溯,回到上一个断点继续执行);
    }
}
```

搜索的实现（bfs）

```
初始状态入队;
while(队列非空)
 {
      取队首元素;   队首出队;
      if(当前是目标状态) 输出解或者作计数、评价处理;
      else for(int i = 1; i <= 状态的拓展可能数; ++i)
      {
           if(!判重 && 第i种状态拓展可行) 新状态入队;
       }
}
```

NOIP 小题大做

先来看个问题：POJ3278 抓住那头牛(cow)

【问题描述】 农夫知道一头牛的位置，想要抓住它。农夫和牛都位于数轴上，农夫起始位于点N(0<=N<=100000)，牛位于点K(0<=K<=100000)。

农夫有两种移动方式：

1、从X移动到X-1或X+1，每次移动花费一分钟

2、从X移动到2*X，每次移动花费一分钟

假设牛没有意识到农夫的行动，站在原地不动。

问：农夫最少需要花多少时间才能逮住那头懵牛？

输入样例：
5 17
输出样例：
4

NOIP 小题大做

问题分析：抓住那头牛(cow)

状态表示 （当前位置,已花费的时间）

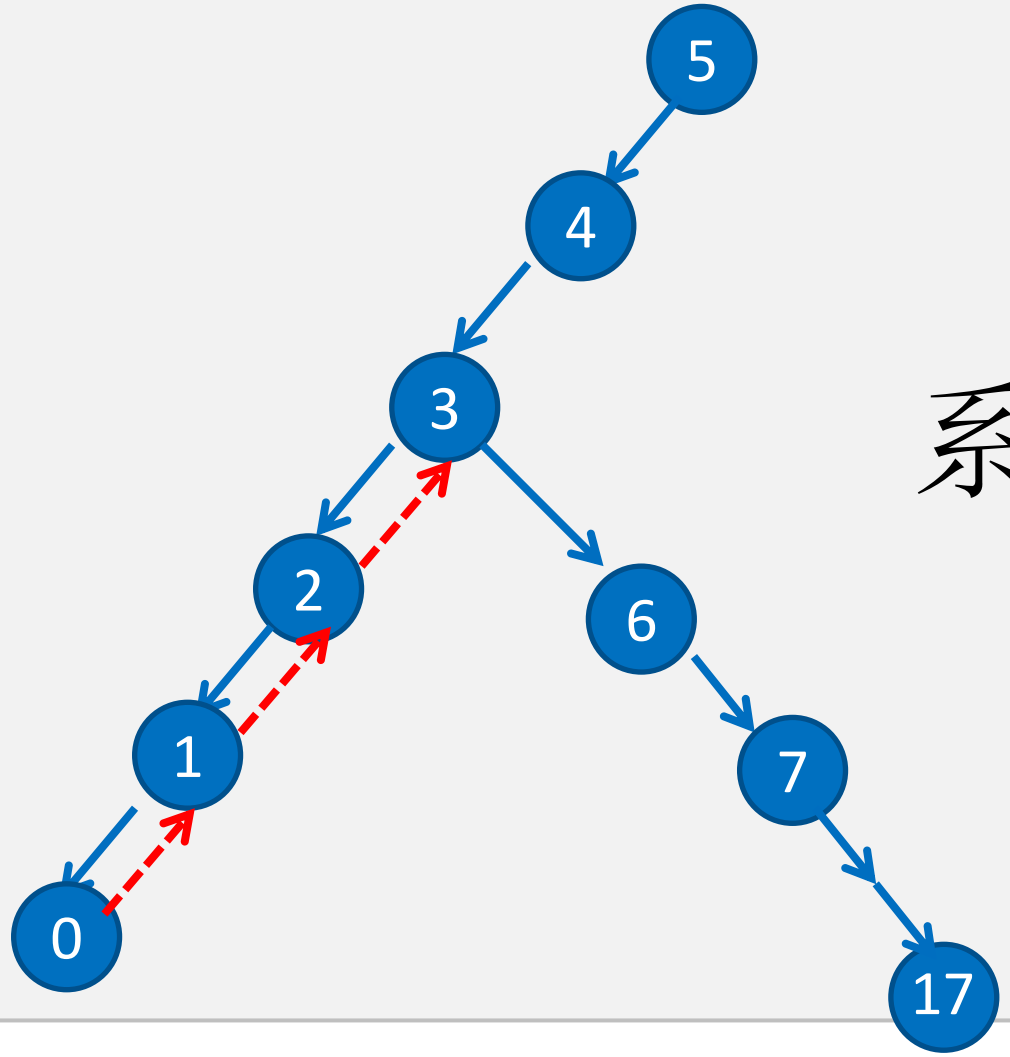
状态转移规则

- 规则1： x → x - 1
- 规则2： x → x + 1
- 规则3： x → 2 * x

约束条件： 0 <= x <=100000

NOIP 裸搜dfs

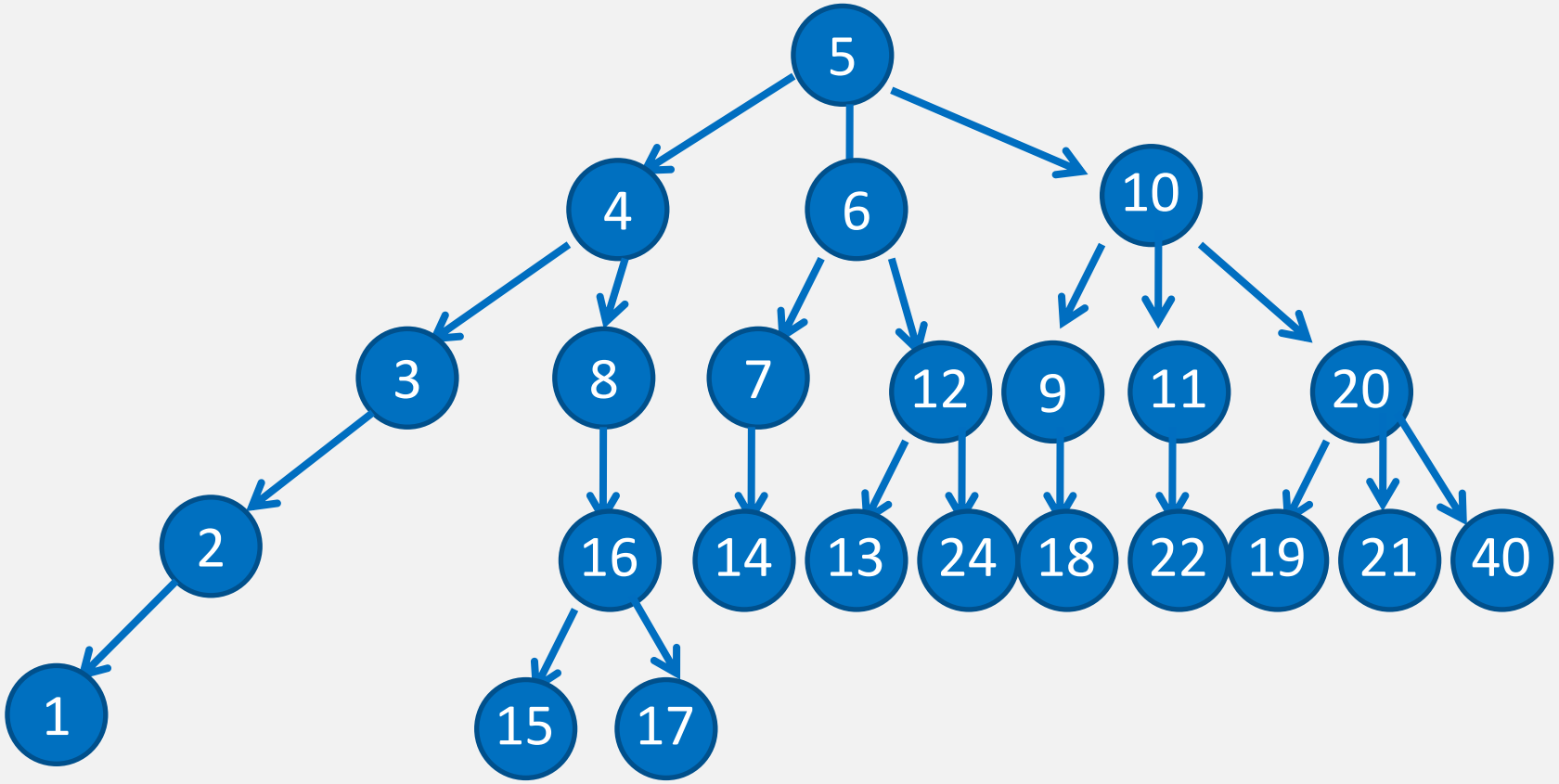
搜索树



系统栈BOOM

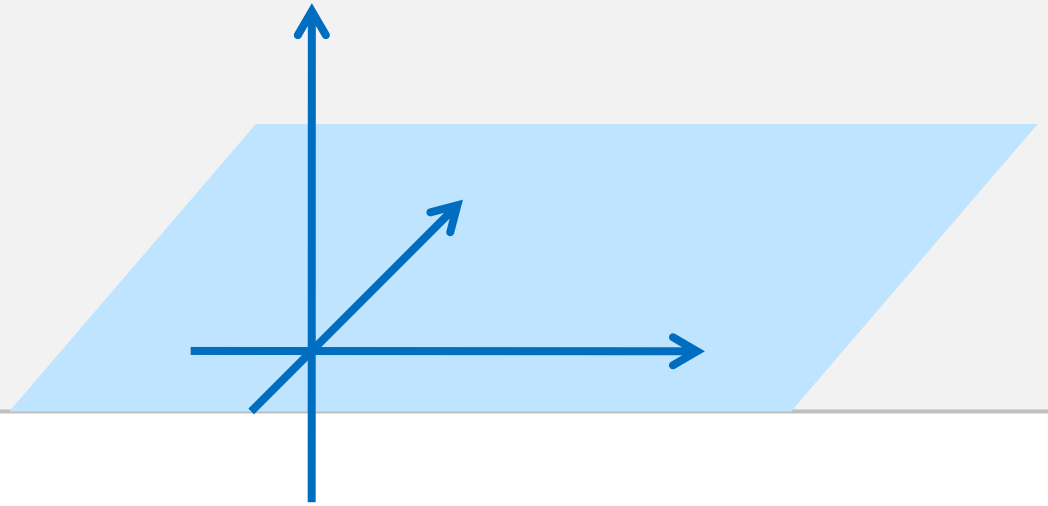
NOIP 裸搜bfs

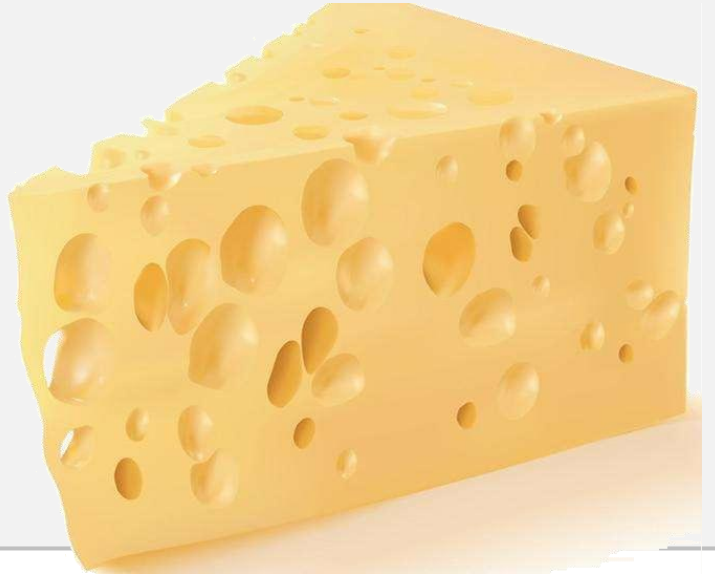
搜索树



例1：NOIP2017-提高组-day2T1 奶酪(cheese)

[问题描述] 现有一块大奶酪，高度为 h ，长度和宽度假设无限大。奶酪中间有许多半径相同的球形空洞。我们可以在这块奶酪中建立空间坐标系，在坐标系中，奶酪的下表面为 z = 0 ，奶酪的上表面为 z = h 。





NOIP 经典诠释

现在，奶酪的下表面有一只小老鼠 Jerry ，它知道奶酪中所有空洞的球心坐标。如果两个空洞相切或是相交，则 Jerry 可以从其中一个空洞跑到另一个空洞。

特别地，如果一个空洞与下表面相切或是相交， Jerry 则可以从奶酪下表面跑进空洞；如果一个空洞与上表面相切或是相交，Jerry 则可以从空洞跑到奶酪上表面。

位于奶酪下表面的 Jerry 想知道，在不破坏奶酪的情况下，能否利用已有的空洞跑到奶酪的上表面去？

空间内两点$P1(x1, y1, z1)$、$P2(x2, y2, z2)$的距离公式如下：

$$\mathrm{dist}(P_1, P_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$$

输入格式

- 输入格式：包含多组数据。

第一行，包含一个正整数 T，代表所含的数据组数。

接下来是 T 组数据，每组数据的格式如下：

第一行包含三个正整数 n，h 和 r，分别代表奶酪中空洞的数量，奶酪的高度和空洞的半径。

接下来的 n 行，每行包含三个整数 x、y、z，表示空洞球心坐标。

输出格式

- 输出格式：

输出包含T行，分别对应T组数据的答案，如果在第i组数据中，Jerry能从下表面跑到上表面，则输出Yes，如果不能，则输出No（均不包含引号）。

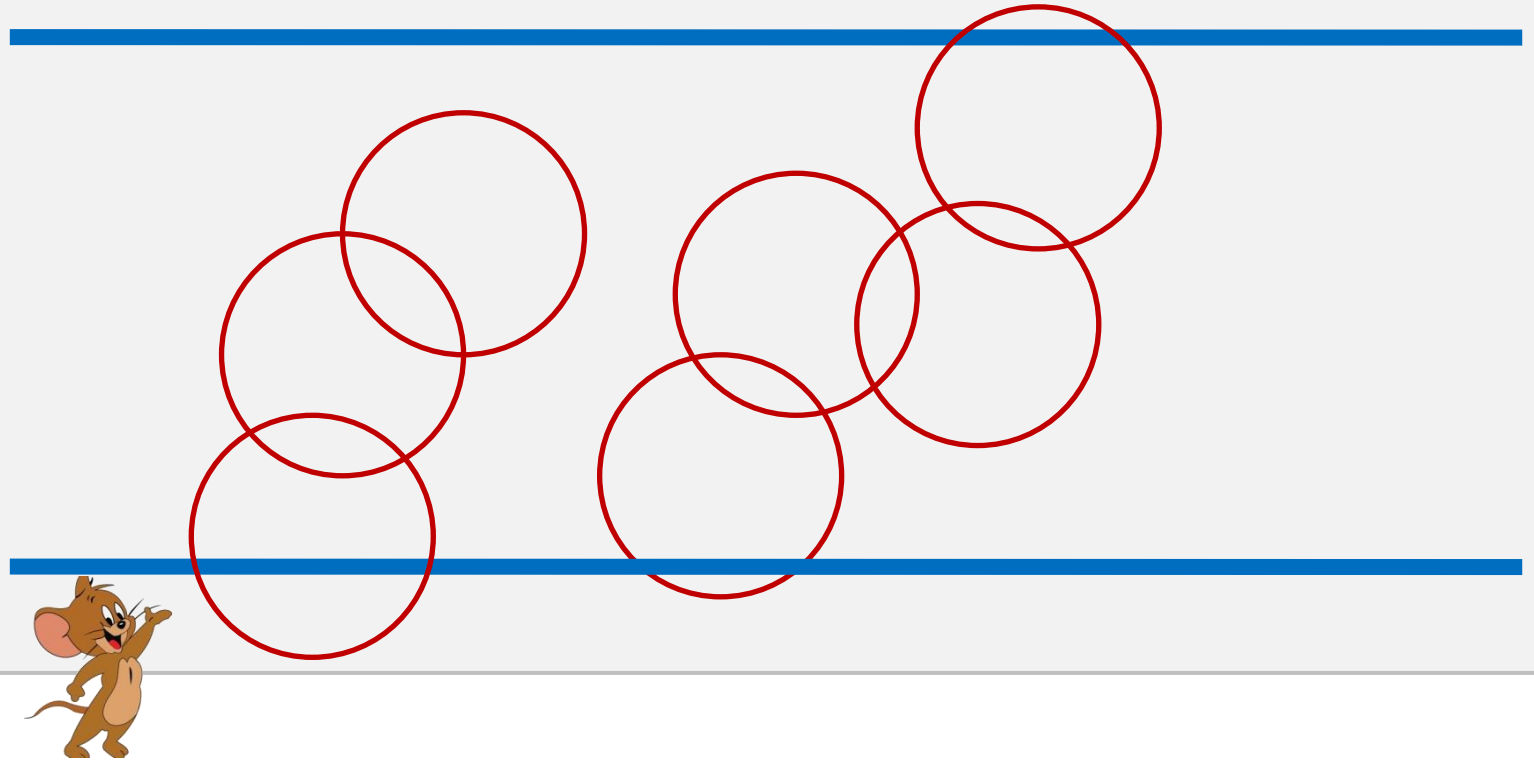
[输入样例]	[输出样例]
3	Yes
2 4 1	No
0 0 1	Yes
0 0 3	
2 5 1	
0 0 1	
0 0 4	
2 5 2	
0 0 2	
2 0 4	

数据规模与约定

- 对于 20%的数据，n = 1，1 ≤ h，r ≤ 10,000，坐标的绝对值不超过 10,000。
- 对于 40%的数据，1 ≤ n ≤ 8， 1 ≤ h，r ≤ 10,000，坐标的绝对值不超过 10,000。
- 对于 80%的数据，1 ≤ n ≤ 1,000，1 ≤ h，r ≤ 10,000，坐标的绝对值不超过 10,000。
- 对于 100%的数据，1 ≤ n ≤ 1,000，1 ≤ h，r ≤ 1,000,000,000，T ≤ 20，坐标的绝对值不超过 1,000,000,000。

问题分析：奶酪

- 大意：判断Jerry能否利用现有空洞从奶酪下表面跑到上表面？
- 假设你是Jerry，你怎么跑?

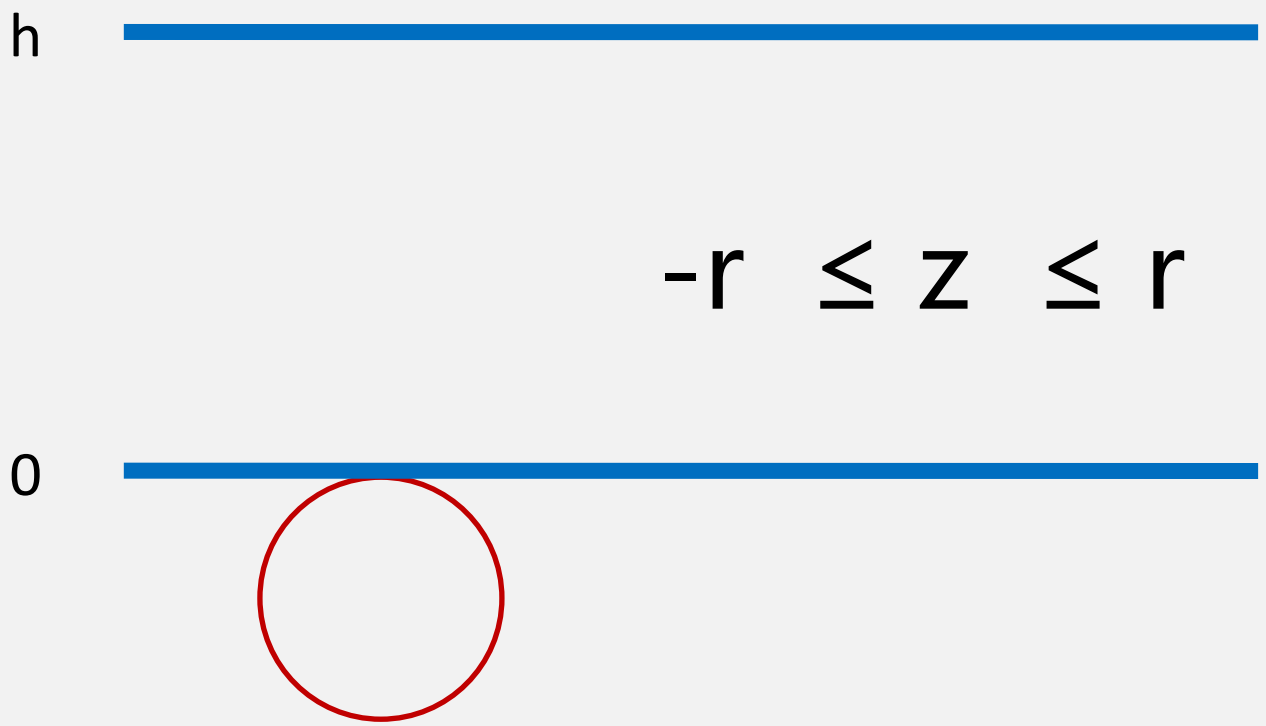


问题分析：奶酪

- 本质：判连通性问题（染色）
- 状态表示： 当前到达的空洞编号
- 初始状态？目标状态？

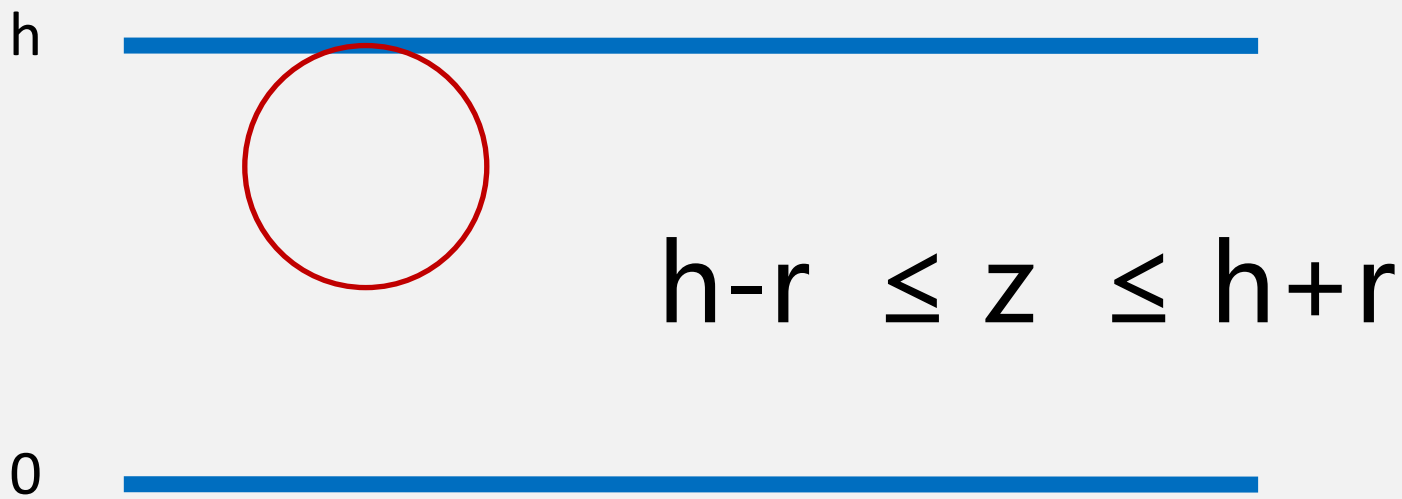
问题分析：奶酪

- 初始状态： 下表面可以直接到达的空洞



问题分析：奶酪

- 目标状态： 有空洞可以直接到达上表面



问题分析：奶酪

- 状态转移：两个空洞相通 — 相交或相切

$$0 \le dist(P1,P2) \le 2r$$

特别提醒

- 注意：利用公式计算空间两点间距离时，请不要使用sqrt，以免精度失真（80分）。
- 解决：两边平方，但注意数据范围统一用long long，否则会溢出。

例2： NOIP2014-提高组-day2T2寻找道路(road)

[问题描述]

在有向图 G 中，每条边的长度均为 1 ，现给定起点和终点，请你在图中找一条从起点到终点的路径，该路径满足以下条件：

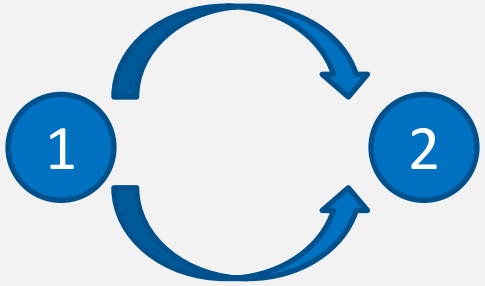
（1）路径上的所有点的出边所指向的点都直接或间接与终点连通。

（2）在满足条件 1 的情况下使路径最短。

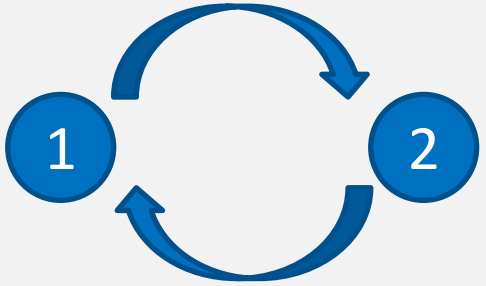
有向图：单向图　　我 —认识→ 习大大

NOIP 经典诠释

注意：图 G 中可能存在重边和自环，题目保证终点没有出边。请你输出符合条件的路径的长度。



重边



自环

输入输出格式

输入格式：

第一行有两个整数 n 和 m ，表示图有 n 个点和 m 条边。

接下来的 m 行每行 2 个整数 x,y ，表示有一条边从点 x 指向点 y 。

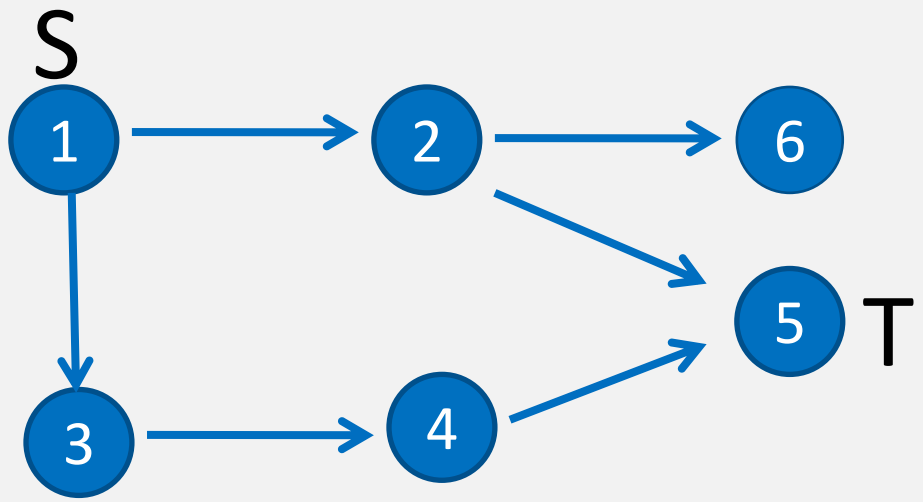
最后一行有两个整数 s, t，表示起点为 s ，终点为 t 。

输出格式：

输出只有一行，包含一个整数，表示最短路径的长度。如果不存在，输出 −1 。

寻找道路(road)

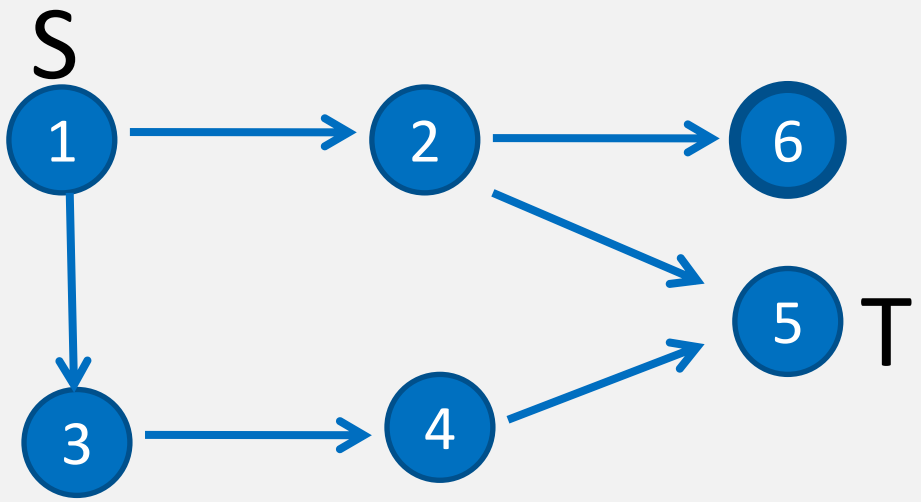
[输入样例]	[输出样例]
6 6	3
1 2	
1 3	
2 6	
2 5	
4 5	
3 4	
1 5	



点 2 不能在答案路径中，因为点 2 连了一条边到点 6 ，而点 6 不与终点 5 连通。

寻找道路(road)

```
[输入样例]  [输出样例]
6 6         3
1 2
1 3
2 6
2 5
4 5
3 4
1 5
```



数据规模

对于 30% 的数据： $0 < n \le 10$ ， $0 < m \le 20$;

对于 60% 的数据： $0 < n \le 100$ ， $0 < m \le 2000$;

对于 100% 的数据：$0 < n \le 10000$, $0 < m \le 200000$,

$0 < x,y,s,t \le n$, $s \neq t$ 。

问题分析：寻找道路

第一步、图的存储

对于 100% 的数据：$0 < n \le 10000, 0 < m \le 200000$

稀疏图

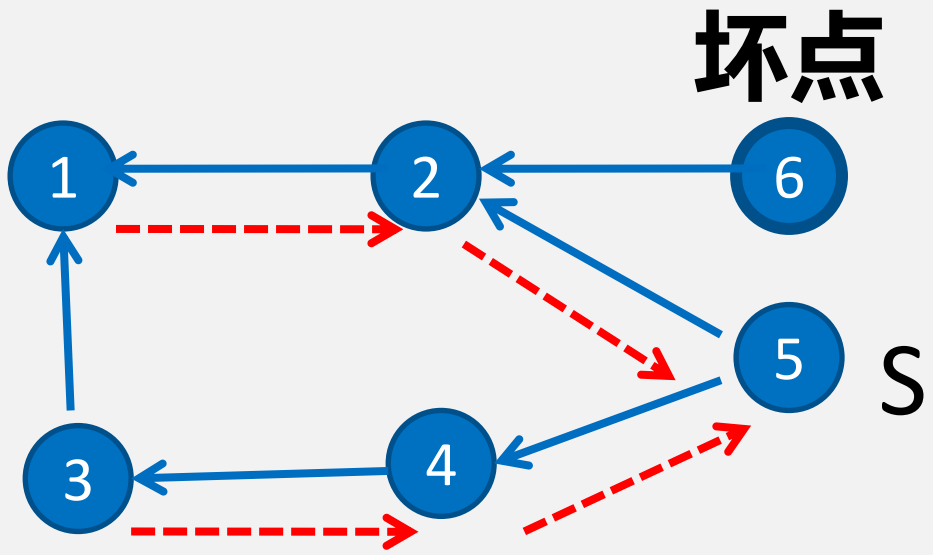
有重边

适合用vector

问题分析：寻找道路

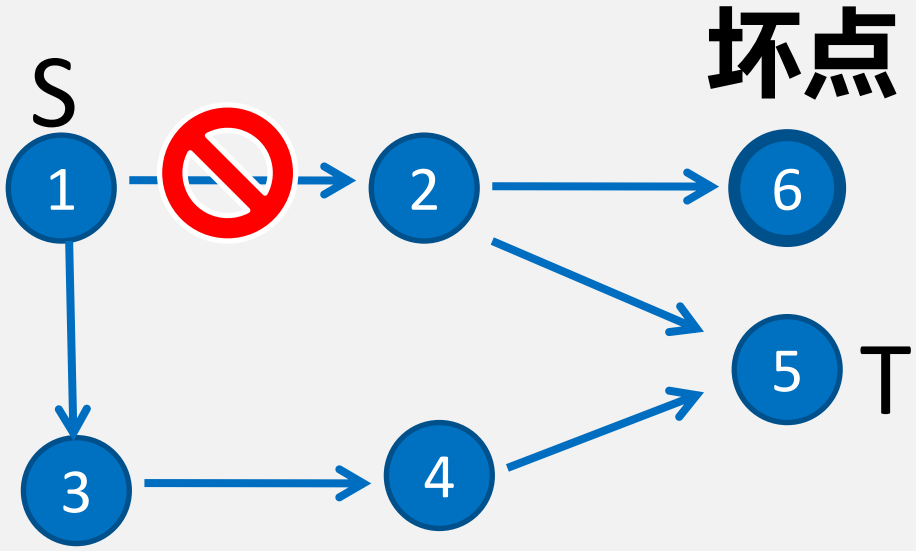
第二步、如何保证路径上的所有点的出边所指向的点都直接或间接与终点连通。

实现：在反向图中，利用dfs从终点出发，标记终点是否可直接或间接到达的点。



问题分析：寻找道路

第三步、在正向图中，利用bfs从起点出发寻找约束条件（下一步即将访问的点不能与“坏点”有染）下可以到达终点的最短路。



特别提醒

- 注意：数据量较大，用cin读入最后一个点超时
- 解决：改用scanf、printf

NOIP 小题大做

还是这个问题：POJ3278 抓住那头牛(cow)

【问题描述】 农夫知道一头牛的位置，想要抓住它。农夫和牛都位于数轴上，农夫起始位于点N(0<=N<=100000)，牛位于点K(0<=K<=100000)。

农夫有两种移动方式：

1、从X移动到X-1或X+1，每次移动花费一分钟

2、从X移动到2*X，每次移动花费一分钟

假设牛没有意识到农夫的行动，站在原地不动。

问：农夫最少需要花多少时间才能逮住那头懵牛？

输入样例：
5 17
输出样例：
4

深搜真的一无是处吗？

NOIP 深搜优化

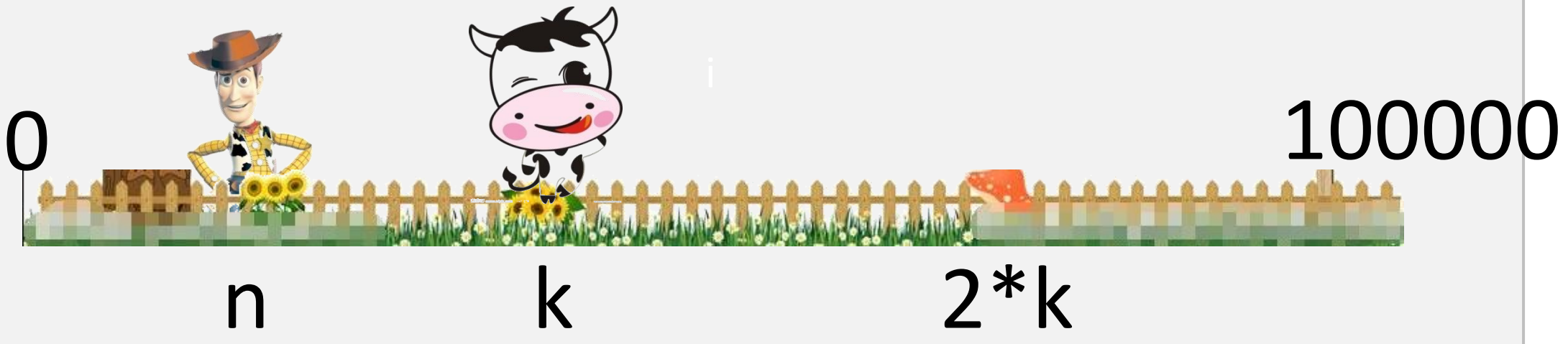
常见的剪枝策略

- dfs优化： “剪枝”
- 可行性剪枝
- 最优性剪枝
- 处处最优性剪枝（记忆化剪枝）
- 搜索顺序选择

NOIP 深搜优化

常见的剪枝策略

- **可行性剪枝**：如果当前条件不合法就不再继续搜索，直接return。这是非常好理解的剪枝。

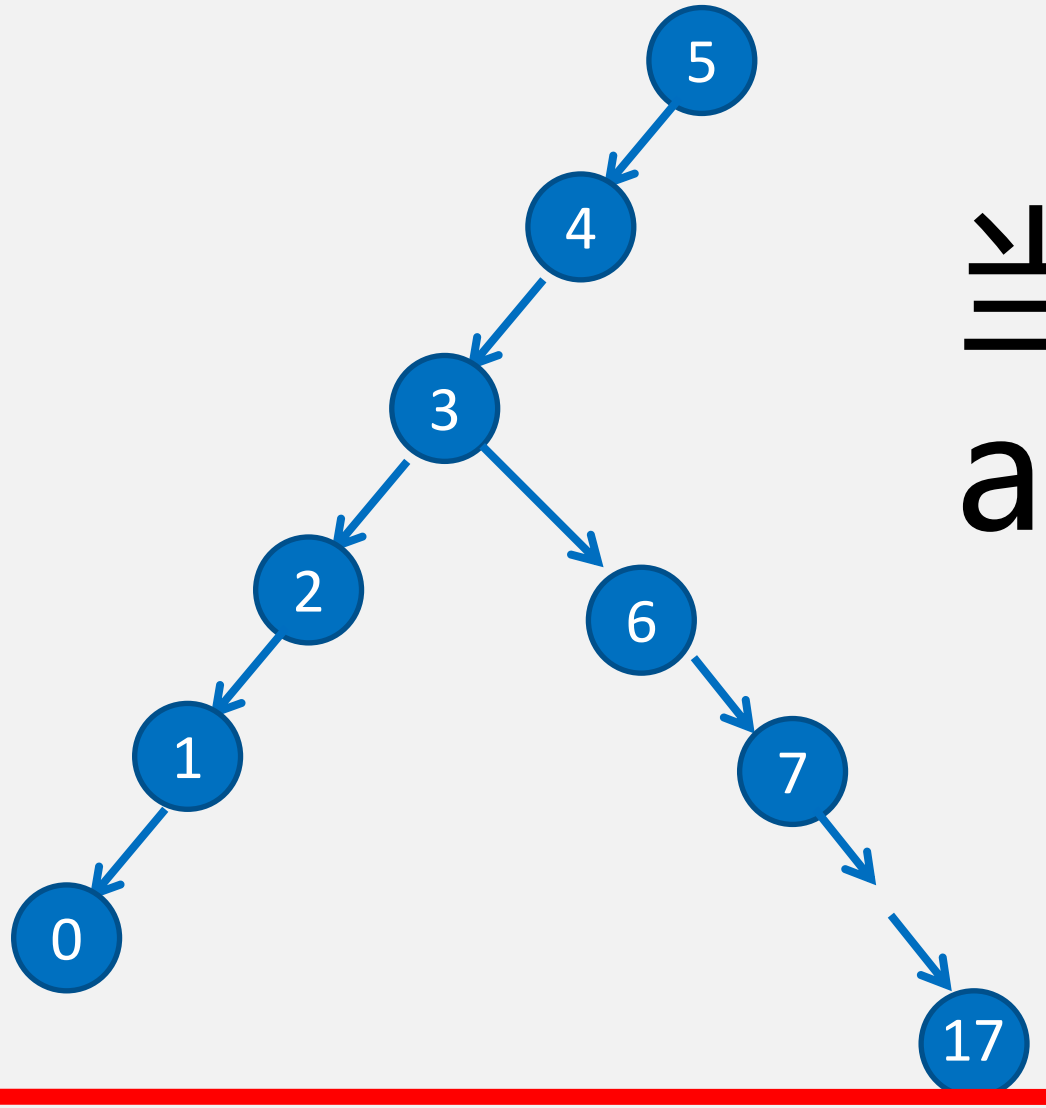


深搜优化

常见的剪枝策略

- **最优性剪枝**：找到一个成功方案作为当前最优解，下次在搜索其他方案的过程中发现当前花费已不如当前最优解，则return，否则更新当前最优解。

NOIP dfs+剪枝

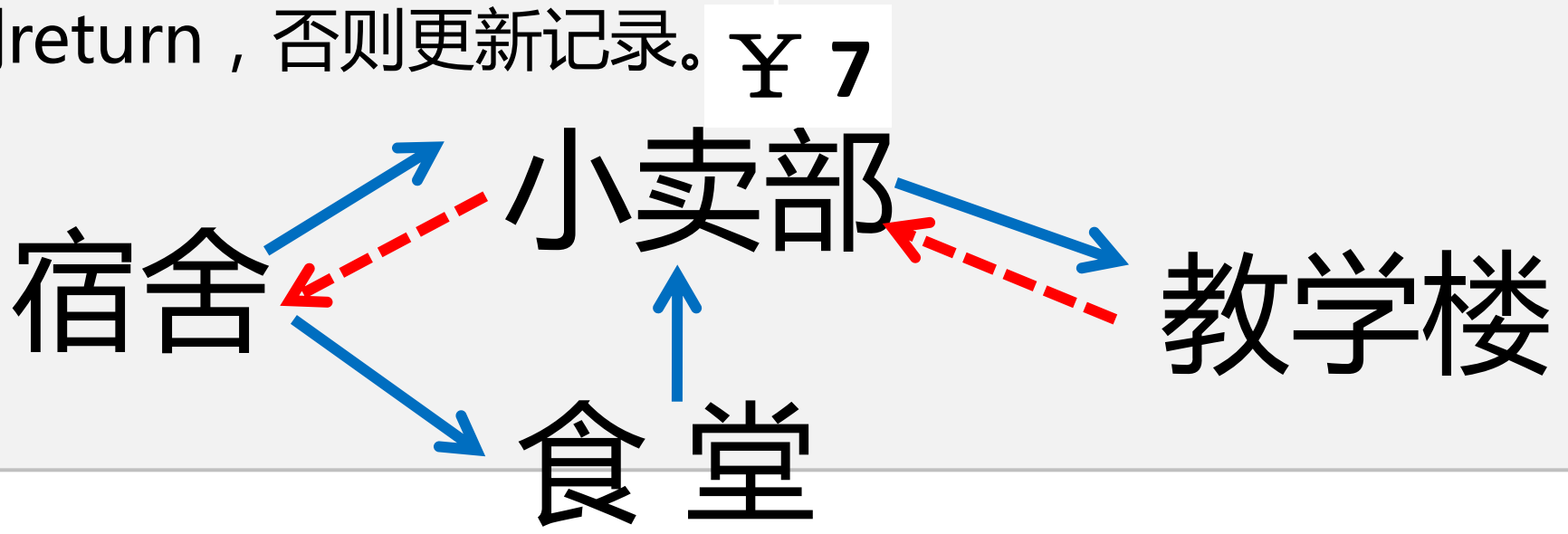


当前最优
ans = 14

ans = 14

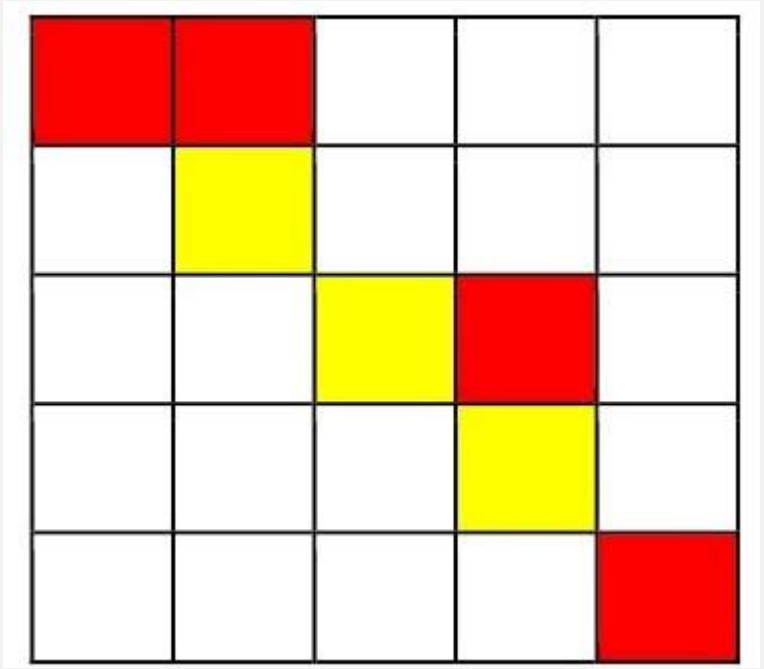
常见的剪枝策略

- **处处最优性（记忆化）剪枝**：到达每个状态，即记录下从初始状态到达该状态的当前最优解。下次在搜索其他方案再次走到该状态时，则先判断当前花费是否比原先记录下的“当前最优解”优，不如它则return，否则更新记录。

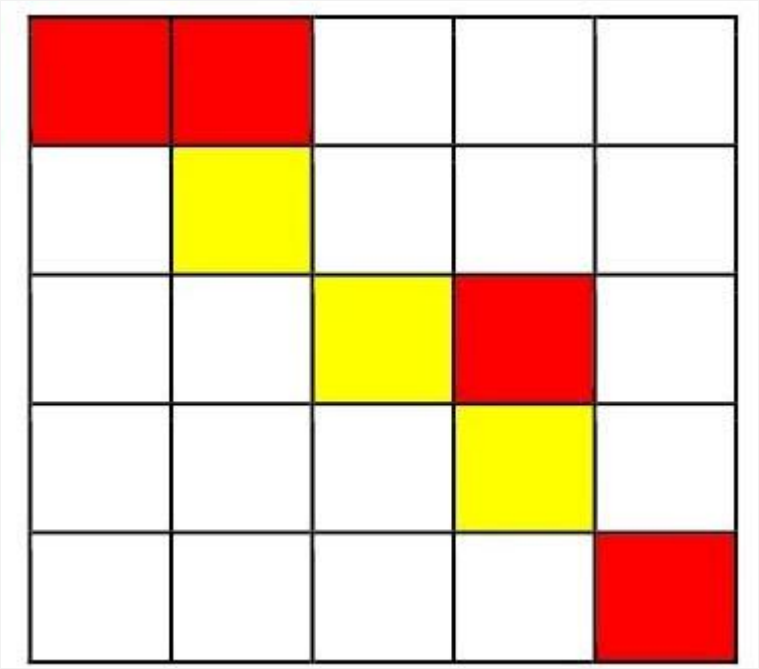


例3： NOIP2017-普及组-T3棋 盘(chess)

有一个m × m的棋盘，棋盘上每一个格子可能是红色、黄色或无色。现在你要从棋盘的左上角走到棋盘的右下角。



任何时刻，你所在的位置必须是有颜色的（不能是无色的），你只能向上、下、左、右四个方向前进。当你从一个格子走向另一个格子时，如果两个格子的颜色相同，那你不需要花费金币；如果不同，则需要花费1 个金币。



另外，你可以花费2个金币施展魔法让下一个无色格子暂时变为你指定的颜色。但这个魔法不能连续使用，而且这个魔法的持续时间很短，也就是说，如果你使用魔法走到这个暂时有颜色的格子上，你就不能继续使用魔法；只有当你离开这个位置，走到一个本来就有颜色的格子上的时候，你才能继续使用这个魔法，而当你离开了这个位置（施展魔法使得变为有颜色的格子）时，这个格子恢复为无色。现在你要从棋盘的左上角，走到棋盘的右下角，求花费的最少金币是多少？

NOIP 经典诠释

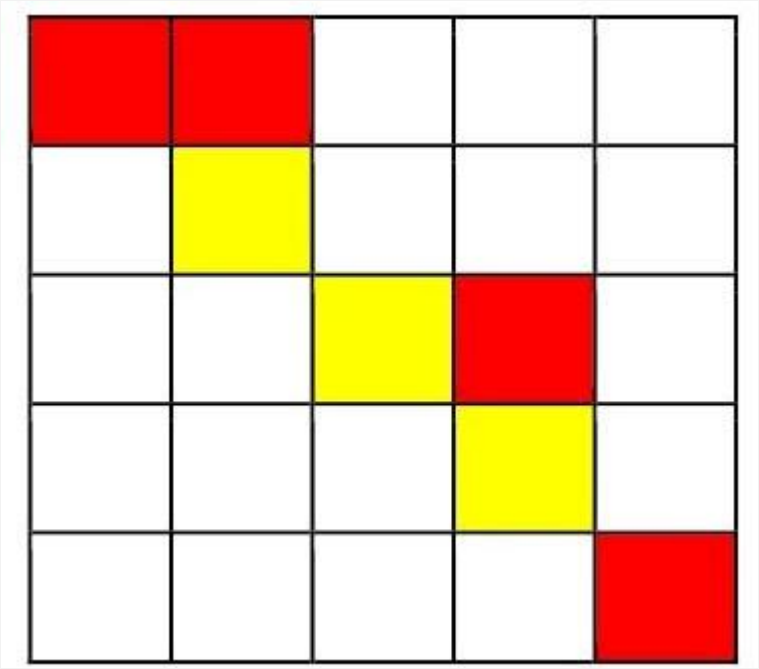
输入格式

第一行包含两个正整数m，n，以一个空格分开，分别代表棋盘的大小，棋盘上有颜色的格子的数量。

接下来的 n 行，每行三个正整数x，y，c，分别表示坐标为（x，y）的格子有颜色c。 其中c=1 代表黄色，c=0 代表红色。相邻两个数之间用一个空格隔开。棋盘左上角的坐标为（1,1），右下角的坐标为（m, m）。棋盘上其余的格子都是无色。保证棋盘的左上角，也就是（1，1）一定是有颜色的。

输出格式

一行，一个整数，表示花费的金币的最小值，如果无法到达，输出-1。



【输入样例】
```
5 7
1 1 0
1 2 0
2 2 1
3 3 1
3 4 0
4 4 1
5 5 0
```

【输出样例】
```
???
```

样例解释

从 (1,1)开始，走到 (1,2) 不花费金币

从 (1,2)向下走到(2,2) 花费 1 枚金币

从 (2,2) 施展魔法，将(2,3) 变为黄色，花费 2 枚金币

从 (2,2) 走到 (2,3) 不花费金币

从 (2,3) 走到 (3,3)不花费金币

从 (3,3) 走到 (3,4) 花费 1 枚金币

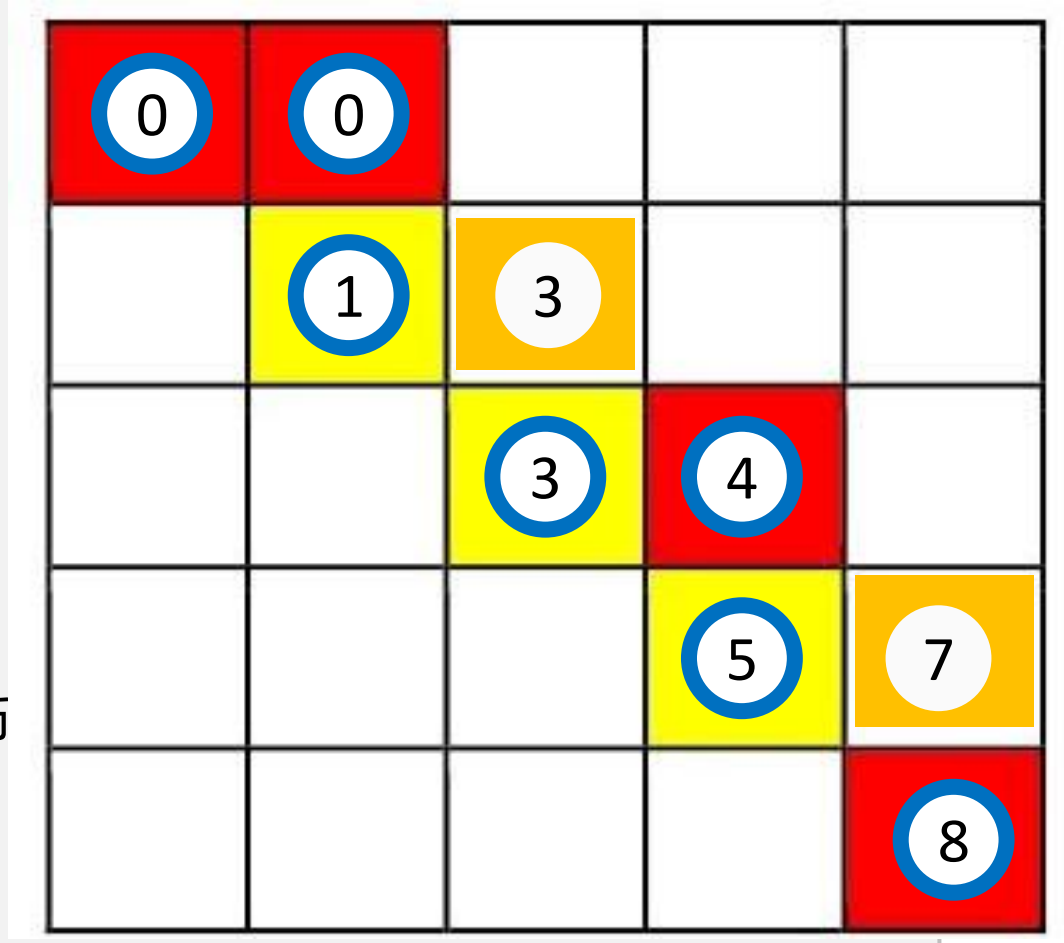
从 (3,4) 走到 (4,4) 花费 1 枚金币

从 (4,4) 施展魔法，将 (4,5) 变为黄色，花费 2 枚金币

从 (4,4) 走到 (4,5) 不花费金币

从 (4,5) 走到 (5,5) 花费 1 枚金币

共花费 8 枚金币。



数据规模与约定

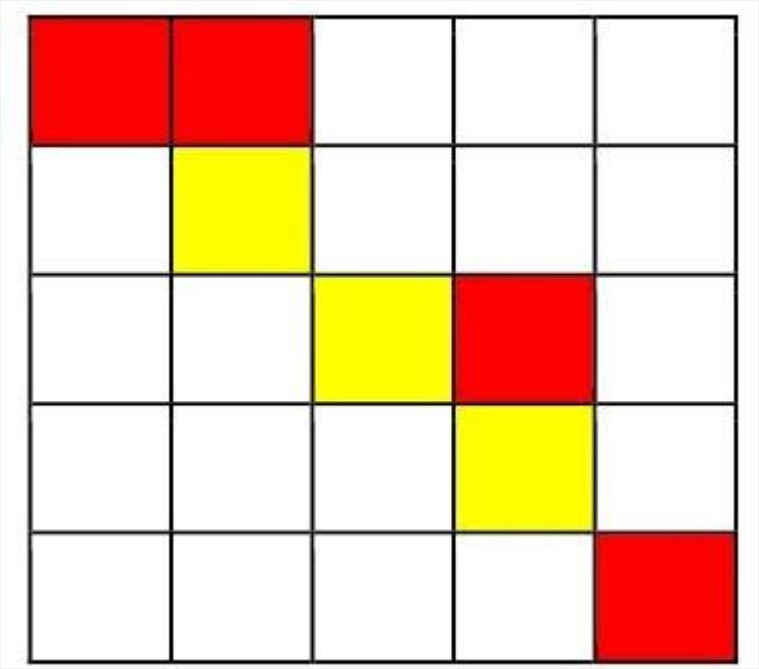
对于 30% 的数据： $1 \le m \le 5$ ， $1 \le n \le 10$ ；

对于 60% 的数据： $1 \le m \le 20$ ， $1 \le n \le 200$ ；

对于 100% 的数据：$1 \le m \le 100$ ， $1 \le n \le 1000$

问题分析：棋盘

棋盘保存　c=1 代表黄色，c=0 代表红色，无色如何表示?



【输入样例】

```
5 7
1 1 0
1 2 0
2 2 1
3 3 1
3 4 0
4 4 1
5 5 0
```

【输出样例】

```
8
```

问题分析：棋盘

状态表示：

当前格子的坐标位置（x,y）

当前花费的金币

当前是否能够使用魔法

问题分析：棋盘

状态转移：

四个方向：上下左右

分情况讨论：下一步格子是否有颜色

Case1：下一步试探的格子有颜色：

（1）与当前格子同色，则dfs(nx,ny,tot,true);

（2）与当前格子异色，则dfs(nx,ny,tot+1,true);

问题分析：棋盘

状态转移：

Case2：下一步试探的格子无色

如果可以使用使用魔法，则

让下一个无色格子暂时变为你指定的颜色（贪心）

```
dfs(nx,ny,tot+2,false);  //魔法暂时失效
```

返回寻找其他路径时，恢复为无色

问题分析：棋盘

裸的dfs：40分

优化：

最优性剪枝：60分

记忆化剪枝：100分

例4： NOIP2009-提高组-T4 靶型数独(sudoku)

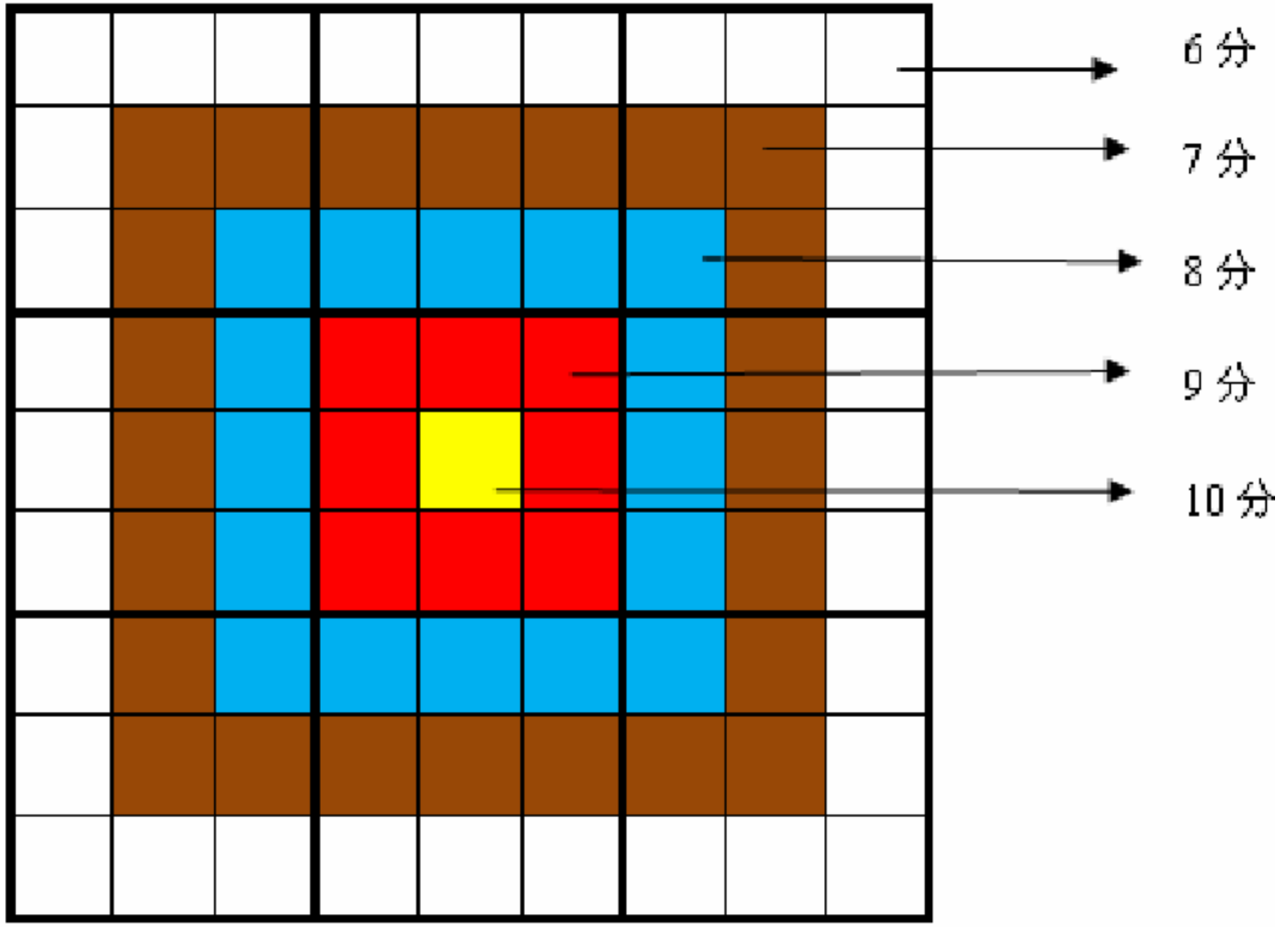
[问题描述]

小城和小华都是热爱数学的好学生，最近，他们不约而同地迷上了数独游戏，好胜的他们想用数独来一比高低。但普通的数独对他们来说都过于简单了，于是他们向 Z 博士请教，Z 博士拿出了他最近发明的“靶形数独”，作为这两个孩子比试的题目。

NOIP 经典诠释

靶形数独的方格同普通数独一样，在 9 ×9 的大九宫格中有 9 个 3 ×3 高的小九宫格（用粗黑色线隔开的）。在这个大九宫格中，有一些数字是已知的，根据这些数字，利用逻辑推理，在其他的空格上填入 1 到 9 的数字。每个数字在每个小九宫格内不能重复出现，每个数字在每行、每列也不能重复出现。但靶形数独有一点和普通数独不同，即每一个方格都有一个分值，而且如同一个靶子一样，离中心越近则分值越高。

NOIP 经典诠释

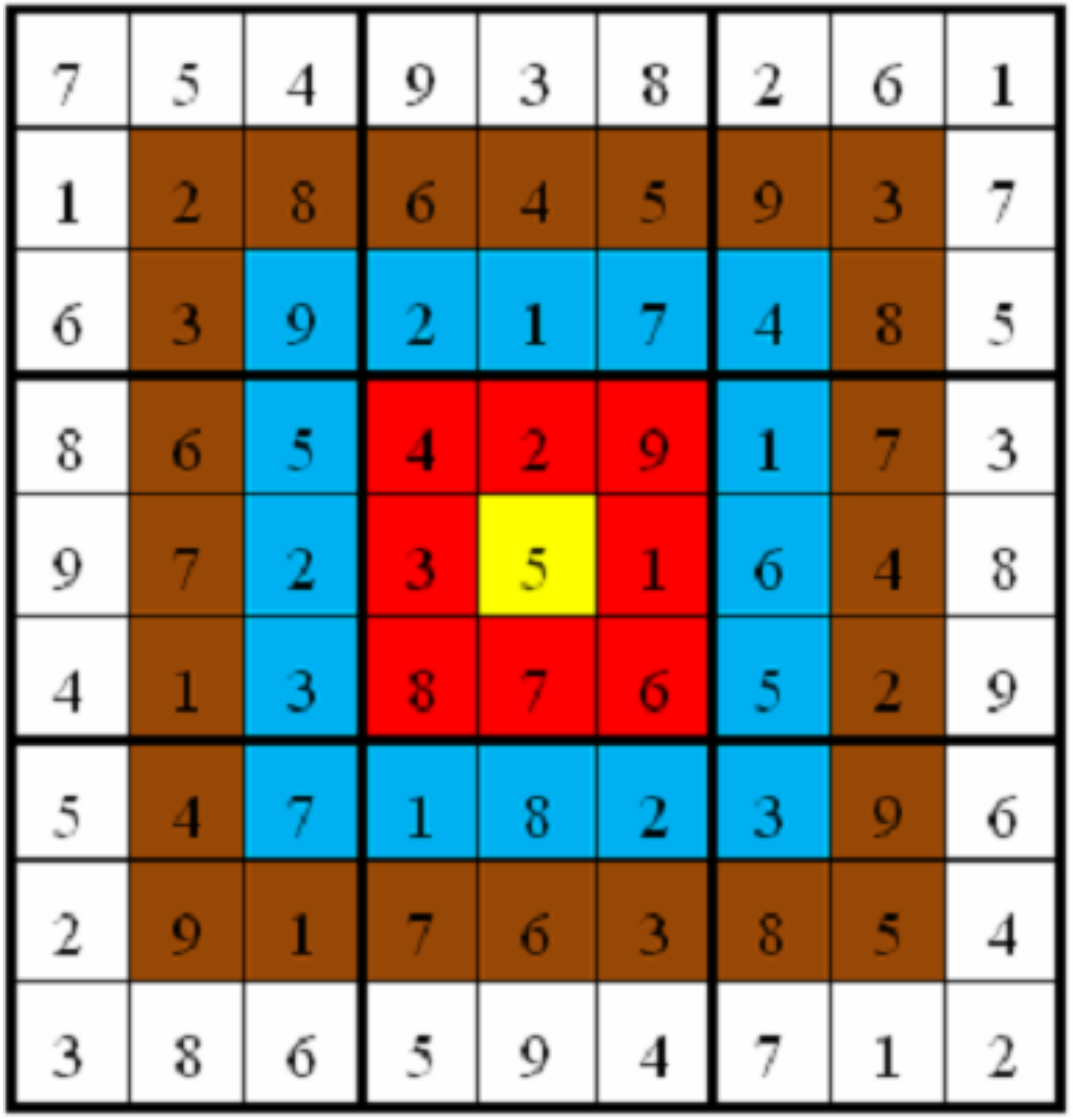


NOIP 经典诠释

比赛的要求是：每个人必须完成一个给定的数独（每个给定数独可能有不同的填法），而且要争取更高的总分数。而这个总分数即每个方格上的分值和完成这个数独时填在相应格上的数字的乘积的总和。

NOIP 经典诠释

如图，在以下的这个已经填完数字的靶形数独游戏中，总分数为 2829。游戏规定，将以总分数的高低决出胜负。



输入输出格式

【输入格式】

一共 9 行。每行 9 个整数（每个数都在 0—9 的范围内），“0”表示一个尚未填满的数独方格。每两个数字之间用一个空格隔开。

【输出格式】

输出可以得到的靶形数独的最高分数。如果无解，则输出-1。

输入输出样例

【输入样例】

```
700900001
100005900
000200080
005020003
000000648
413000000
007002090
201060804
080504012
```

【输入样例】

```
2829
```

数据规模

40%的数据，数独中非 0 数的个数不少于 30。

80%的数据，数独中非 0 数的个数不少于 26。

100%的数据，数独中非 0 数的个数不少于 24。

问题分析：靶型数独

- 40分算法：暴力dfs 从第一个空白位置开始利用dfs填数
- 75分算法：暴力dfs + 一点点预处理
- 预处理1：把所有没填的数放到一个数组里，方便查找
- 预处理2：数字判重（参考算法竞赛入门经典中“八皇后问题”处理策略）

问题分析：靶型数独

- 100分算法：暴力dfs + 一点点预处理 +一点点优化：

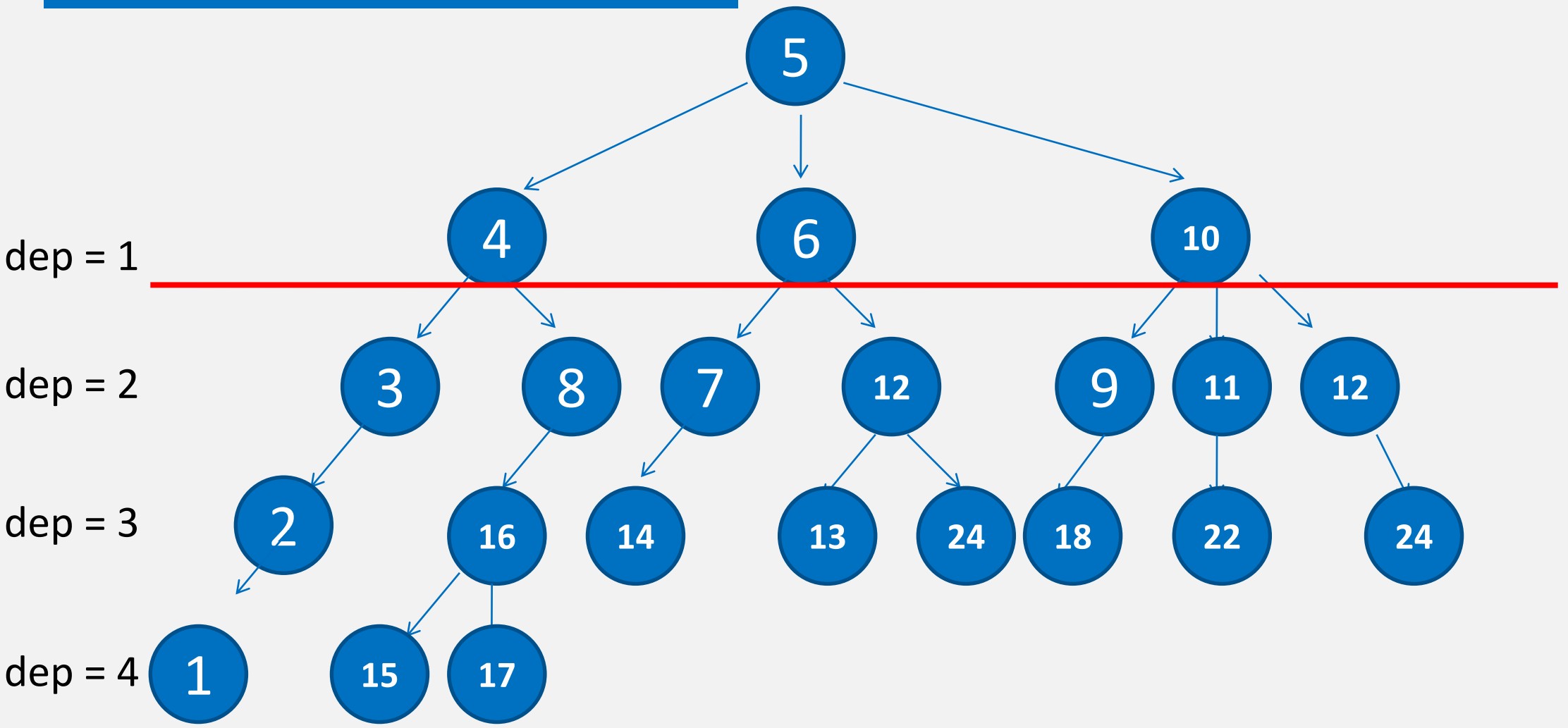
  搜索顺序优化：每次先选“可选性最小”的格子填

NOIP 小结

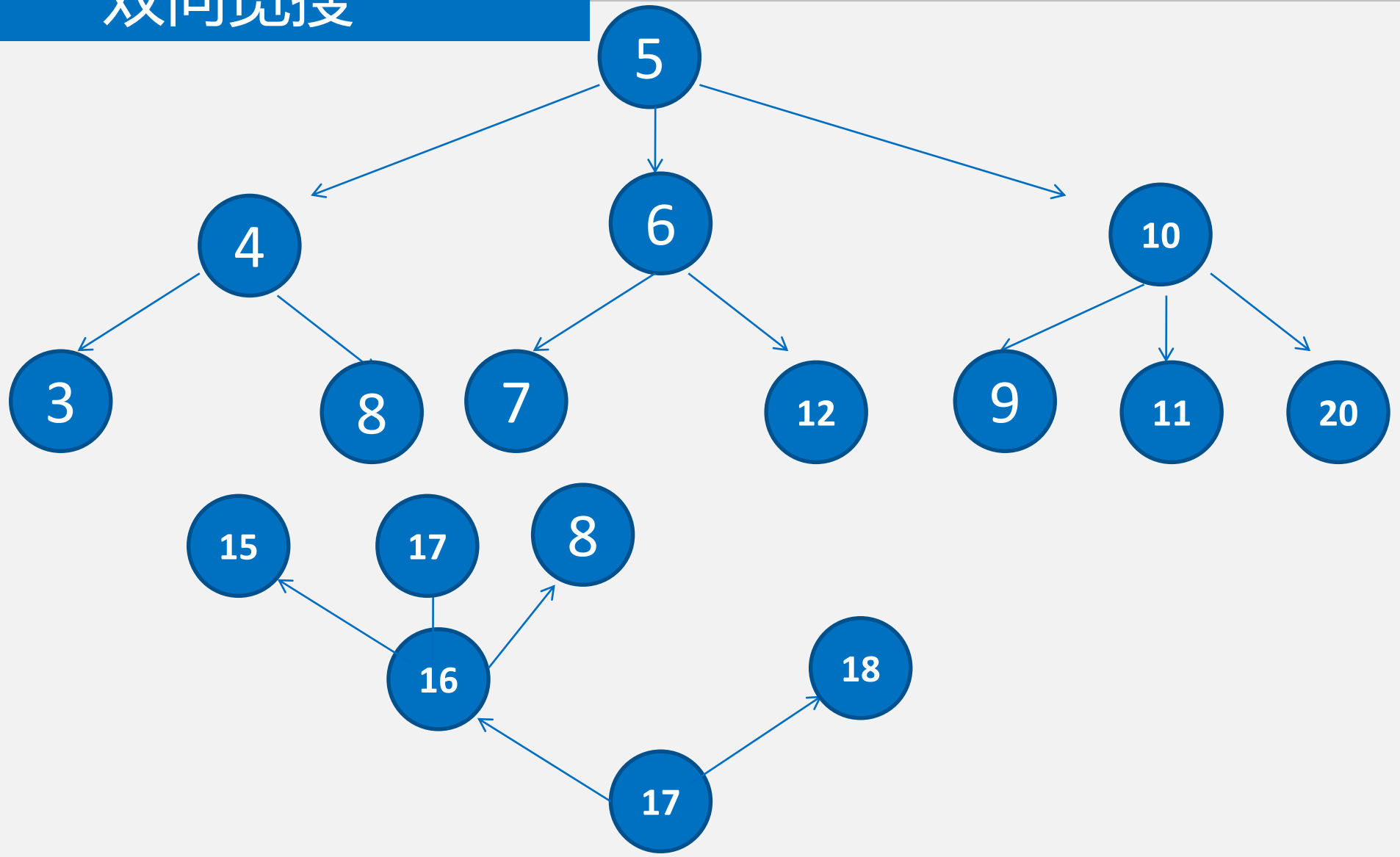
深搜剪枝

- **可行性剪枝**
  - 每次放最小的都放不下
  - 每次放最大的也放不满
- **最优性剪枝**
  - 如果当前搜索结果已经比之前保存的最优值差，继续搜也只会越来越差，则直接退出
- **find every next state u -> v**
  - 搜索顺序~全靠出题人良心，详见NOIP2009靶形数独

迭代加深搜索


dep = 1
dep = 2
dep = 3
dep = 4
5
4
6
10
3
8
7
12
9
11
12
2
16
14
13
24
18
22
24
1
15
17

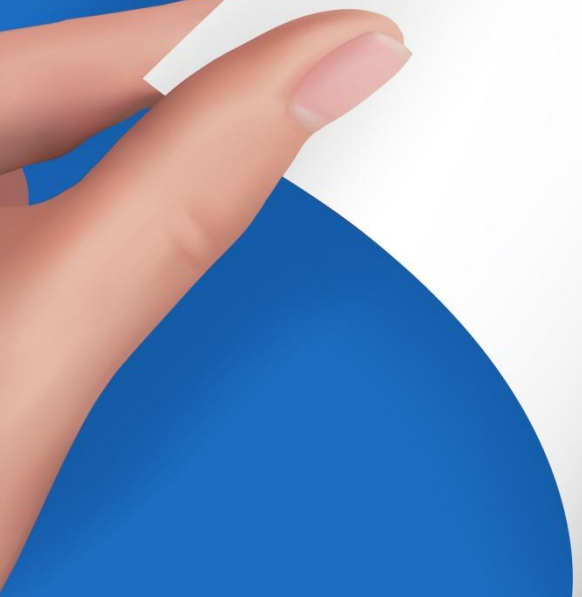
双向宽搜



上机习题
Practice

- 奶酪（裸的dfs）
- 飞越原野（bfs+记忆化）
- 寻路道路（dfs+bfs）
- 棋 盘（dfs+记忆化）
- 靶型数独（ dfs+搜索顺序）
- 小木棍（dfs+剪枝）

JSOI



谢谢阅读

请提出您的宝贵意见