实验舱
青少年编程
走近科学 走进名校
LAB

提高算法班

启发式搜索

Mas

盲目搜索

盲目搜索的基本思路：生成和测试

将状态看作节点,遍历该节点的所有转移,将转移到的节点记录等待展开

称未展开或未完全展开的节点为开节点

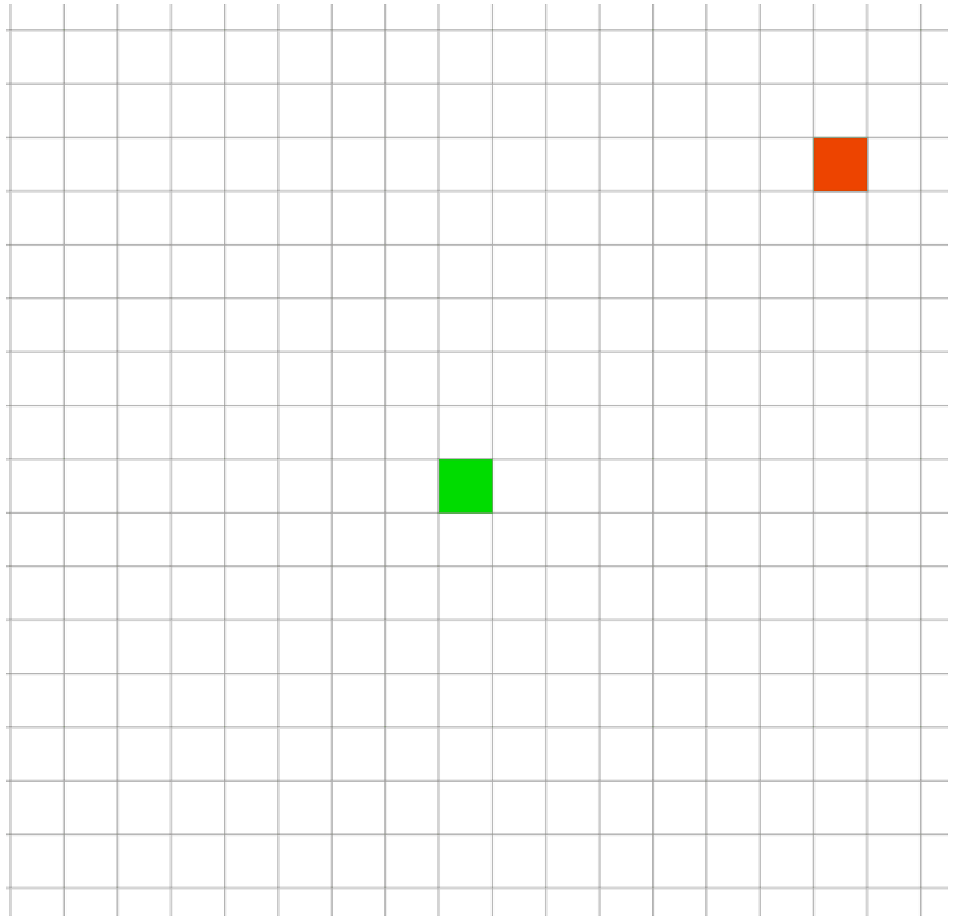
已完全展开的节点为闭节点

搜索就是展开节点的过程,搜索在找到目标或没有开节点时结束

BFS：用队列存储开节点 FIFO

DFS：用栈存储开节点 LIFO

称 BFS 和 DFS 为盲目搜索，因其未使用状态转移之外的信息



启发式搜索

启发式搜索（heuristic search）是利用问题的启发信息引导搜索的一种算法

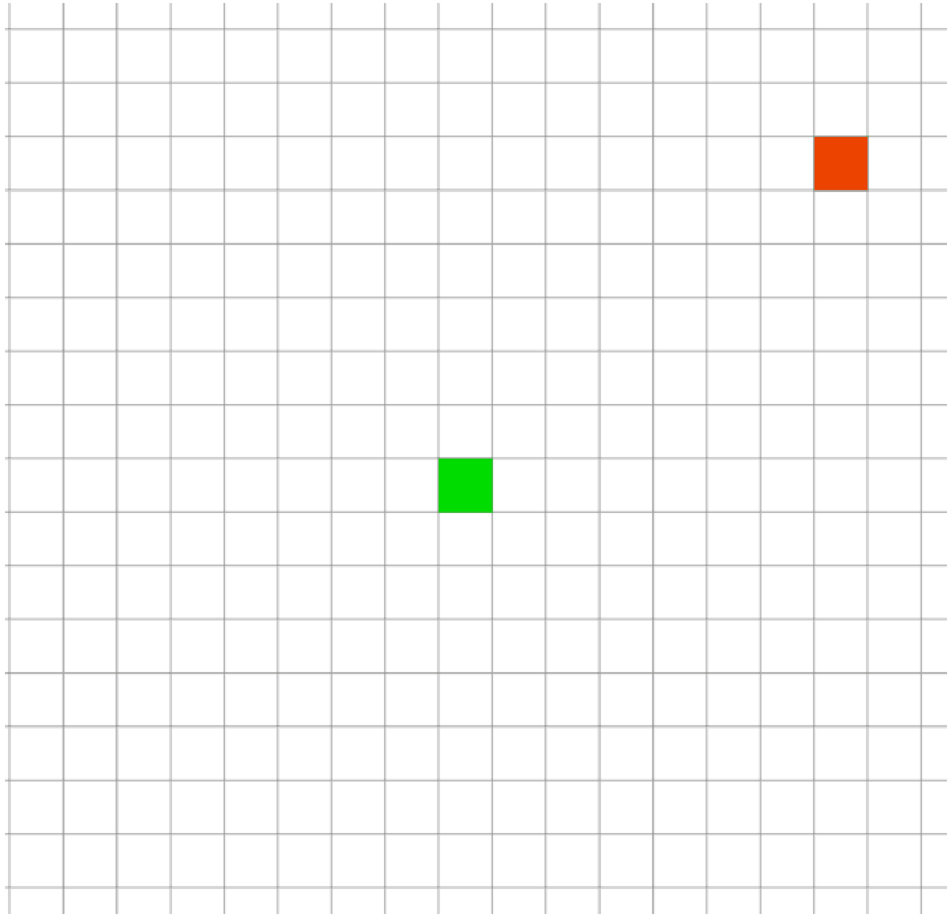
复杂度往往不好准确描述

启发式搜索中最重要的一个概念是一个状态的估价函数,常见形式为

$$f(x) = g(x) + h(x)$$

其中 $g(x)$ 为到达状态 x 已经付出的实际代价

$h(x)$ 为 x 到目标状态还需要的代价的**预估值**



A*

A * (A * search algorithm) 是一种带有估价函数对 BFS 的改进算法

在搜索过程中维护一个优先队列

不断取出 **当前代价+未来预估** 的最小状态扩展

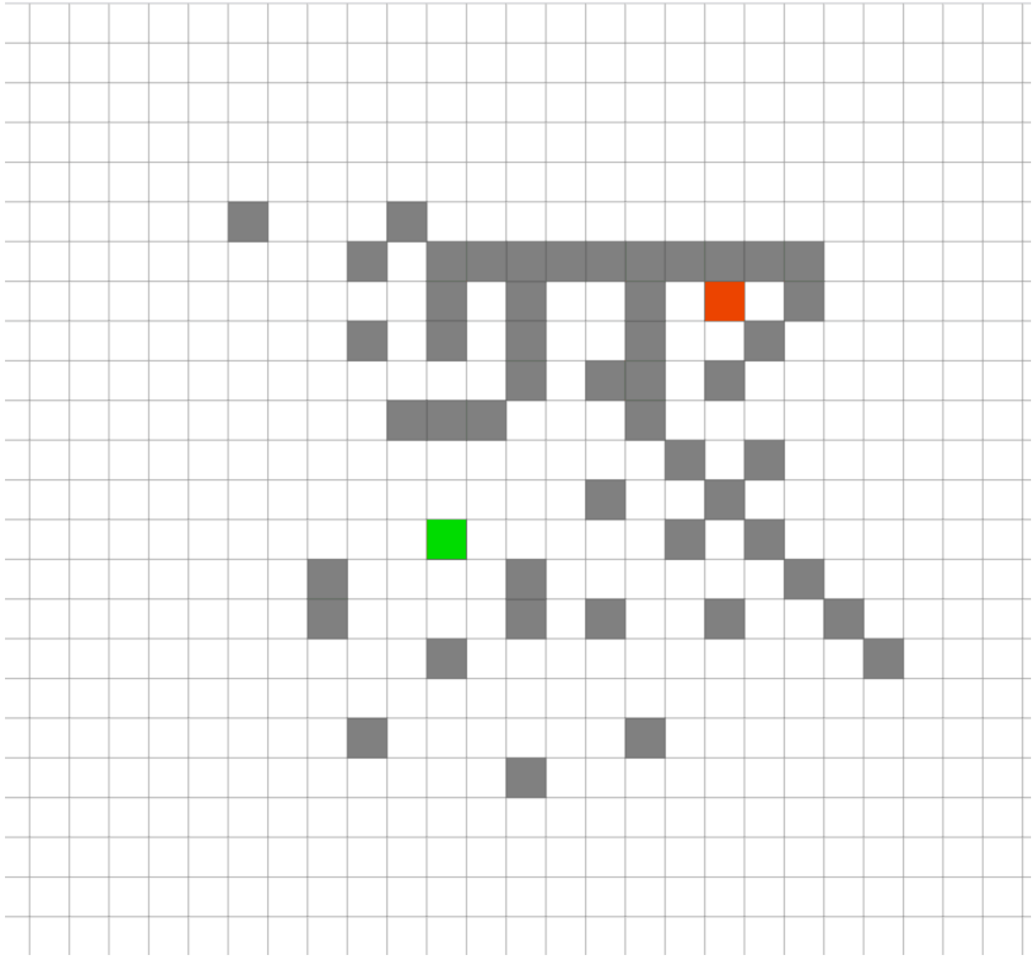
为了满足第一次从优先队列中取出目标状态即为最优解

估价函数 $f(s)$ 不超过真实最小代价

若预估值大于未来实际代价

将导致某最优解搜索路径上的状态被估计较大的代价，可能无法及时取出

导致非最优解路径上的状态不断扩展，直至在目标状态上产生错误答案



估价函数

对于启发函数需满足

$$0 \leq h(x) \leq h^*(x)$$

$g(x)$ 为起点到 x 的真实最小代价

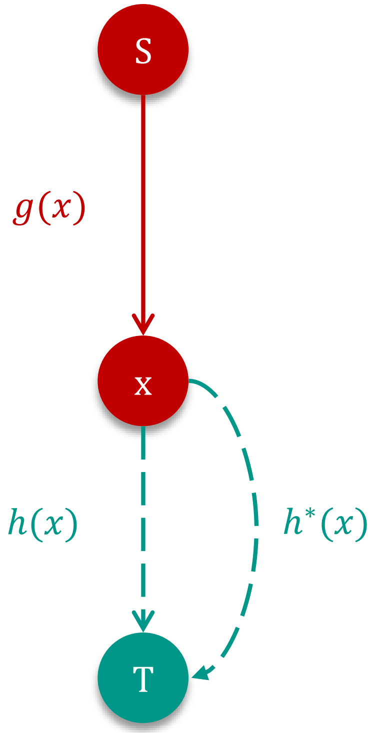
$h^*(x)$ 为 x 到目标状态的真实最小代价，$h(x)$ 为 x 到目标状态的预估代价

考虑启发函数的影响，大致有以下几种情况:

- $0 \leq h(x) < h^*(x)$ ，搜索的点数多搜索范围大、效率低,但能得到最优解

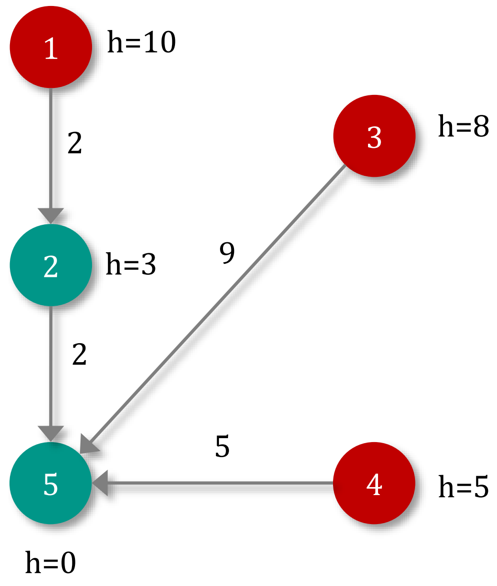
  $h(x) = 0$ 时即为 Dijkstra

- $h(x) = h^*(x)$,搜索严格沿着最短路径进行,此时的搜索效率是最高的
- $h(x) > h^*(x)$,搜索的点数少,搜索范围小,效率高,但**不能保证得到最优解**



迷宫最小步数问题应该如何选取合适估价函数?

A*



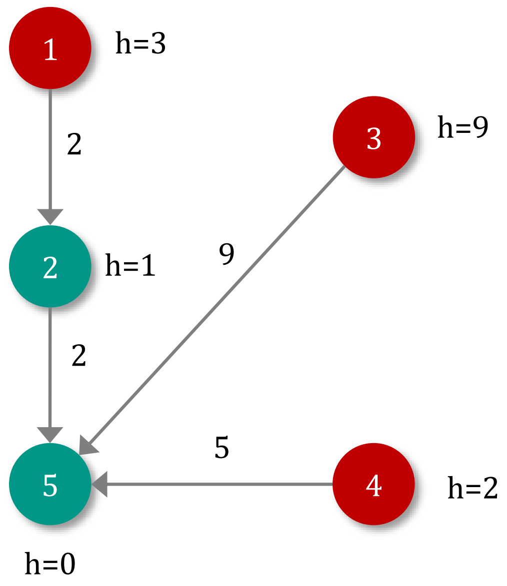
初始时优先队列内存在

$4(0+5), 3(0+8), 1(0+10)$

取出 $4(0+5)$ 进行扩展得目标状态

此时最优解为 5

答案错误



初始时优先队列内存在

$4(0+2), 1(0+3), 3(0+9)$

取出 $4(0+2)$ 进行扩展

优先队列内存在

$1(0+3), 5(5+0), 3(0+9)$

取出 $1(0+3)$ 进行扩展

优先队列内存在

$2(2+1), 5(5+0), 3(0+9)$

取出 $2(2+1)$ 进行扩展

优先队列内存在

$5(4+0), 5(5+0), 3(0+9)$

取出 $5(4+0)$ 得到正确答案

A*

与 Dijkstra 类似需要保证 **转移代价/预估代价** 非负且 $0 \le h(x) \le h^*(x)$

当问题存在解时，队中至少存在一个节点为最优解路径上的节点(如起点)

记最优解路径上的某个节点为 u 目标节点为 t 其真实代价分别为 dis_u 和 dis_t

最优解路径上的点的代价不超过终点代价

即 $dis_u \le dis_t$

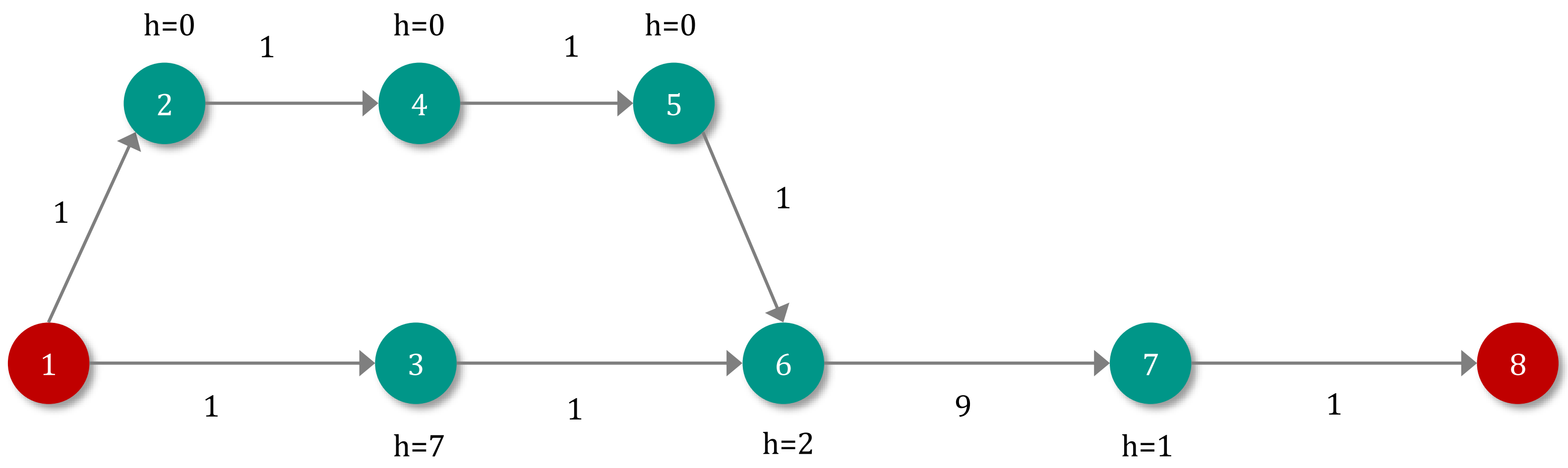
若 t 首次出队时 dis_t 并非最优解，存在更优解 dis'_t

$$dis'_t = dis_u + h^*(u) < dis_t$$

$$dis_t \ge dis_u + h(u)$$

与条件矛盾

A*



对于上图满足 $h(u) \le h^*(u)$

路径 $1 \to 2 \to 4 \to 5 \to 6$ 使得 6 首次出队时最小距离为 3

所以出队即为最小距离在 $A*$ 算法中仅**对目标节点**成立

同时也应当允许节点被多次更新扩展

#2989、又是八数码

题目描述

在一个 3×3 的网格中, $1 \sim 8$ 这 8 个数字和一个 `x` 恰好不重不漏地分布在这 3×3 的网格中

例如:

```
1 2 3
x 4 6
7 5 8
```

在游戏过程中,可以把 `x` 与其上、下、左、右四个方向之一的数字交换(如果存在)

我们的目的是通过交换,使得网格变为如下排列(称为正确排列):

```
1 2 3
4 5 6
7 8 x
```

例如,示例中图形就可以通过让 `x` 先后与右、下、右三个方向的数字交换成功得到正确排列

交换过程如下:

```
1 2 3   1 2 3   1 2 3   1 2 3
x 4 6   4 x 6   4 5 6   4 5 6
7 5 8   7 5 8   7 x 8   7 8 x
```

把 `x` 与上下左右方向数字交换的行动记录为 `u` 、 `d` 、 `l` 、 `r`

现在,给你一个初始网格,请你通过最少的移动次数,得到正确排列

输入格式

输入占一行,将 3×3 的初始网格描绘出来
例如,如果初始网格如下所示:

```
1 2 3
x 4 6
7 5 8
```

则输入为: `123x46758`

输出格式

输出占一行,包含一个字符串,表示得到正确排列的完整行动记录
如果答案不唯一,输出任意一种合法方案即可
如果不存在解决方案,则输出 `unsolvable`

#2989、又是八数码

将数值去掉空格后作为一个新的序列 S

若 S 逆序对数量与目标状态逆序对奇偶性不一致，此时无解 *（仅讨论必要性，可尝试证明充分性）*

- 当空格进行水平移动时 S 不发生变化，不改变逆序对数量
- 当空格进行竖直移动时，相当将某个元素 t 移动至 0 前

对于 n 为奇数时 $n \times n$ 数码问题(八数码为 $n = 3$ 时情况)，不难发现一次操作不影响其它元素逆序对数量

设 t 后 $n-1$ 个元素中有 x 个元素大于 t，y 个元素小于 t 那么 $n-1 = x+y$

移动后将会增加 x 个逆序对，减少 y 个逆序对，逆序对改变数量为

$$x - y = x - (n - 1 - x) = 2x - (n - 1)$$

$2x - (n-1)$ 显然为偶数，即一次交换不改变逆序对奇偶性

#2989、又是八数码

若无解 A＊ 求解效率极低(低于朴素 BFS)

不妨根据上述性质，提前特判无解

对于估价函数 $f(S)$

可统计 S 中各个位置与目标状态中对应数值的曼哈顿距离

与目标状态不同的格子数

优先队列中根据各状态的 **实际步数+预估代价** 进行排序

对于各个状态可使用 $hash$ 表记录最小步数

#2523、第K短路

题目描述

给定一张 N 个点(编号 $1 \sim N$), M 条边的有向图,求从起点 S 到终点 T 的第 K 短路的长度,路径允许重复经过点或边

> *每条最短路中至少要包含一条边*

输入格式

第一行包含两个整数 N 和 M

接下来 M 行,每行包含三个整数 A, B 和 L ,表示点 A 与点 B 之间存在有向边,且边长为 L

最后一行包含三个整数 S, T 和 K ,分别表示起点 S ,终点 T 和第 K 短路

输出格式

输出占一行,包含一个整数,表示第 K 短路的长度,如果第 K 短路不存在,则输出 `-1`

数据范围

对于全部的数据 $1 \leq S, T \leq N \leq 1000, 0 \leq M \leq 10^5, 1 \leq K \leq 1000, 1 \leq L \leq 100$

#2523、第K短路

对于一次 Dijkstra 当节点第一次出队时此时距离为最短距离

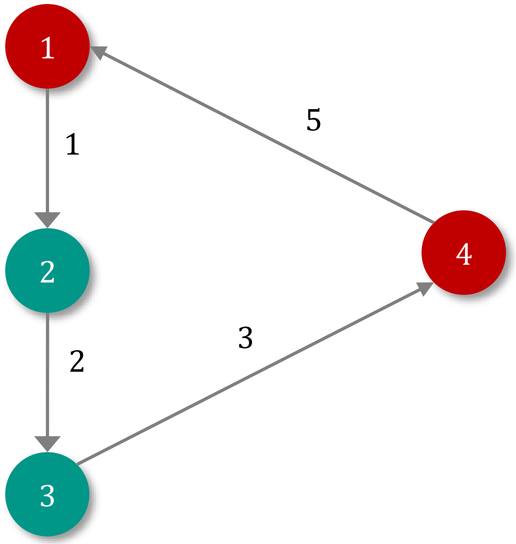
不妨允许多次入队多次出队,当终点第 k 次出队时即为所求

考虑使用 $A*$ 优化

对于估价函数 $f(u)$ ，可选取 $u \rightarrow t$ 的最短路距离

可建反图从t跑一遍最短路即可

根据题意若 $s = t$ 需要求第 $k+1$ 短路



对于 $A*$ 解法若精心构造数据 时/空复杂度不优

可求出 Shortest Path Tree 并使用可持久化可并堆优化，时间复杂度 $O\big((n+m)\log m + k\log\ k\big)$

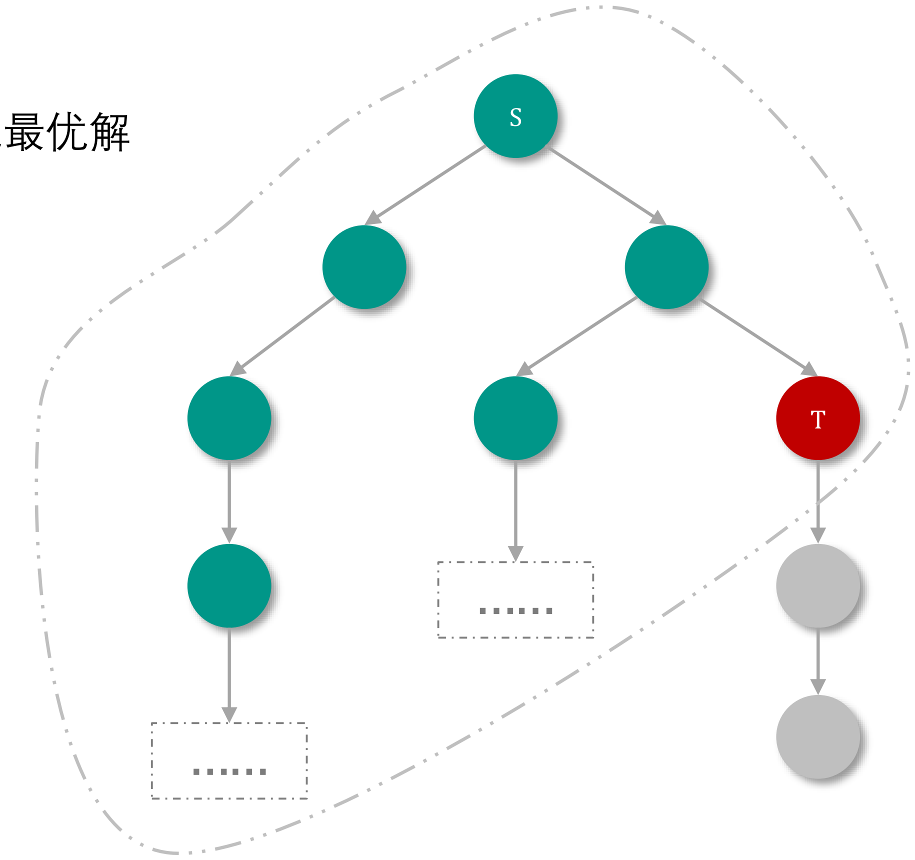
迭代加深搜索

当搜索树的分支数目特别多而答案在较浅的节点，若开始选错了分支,会在深层次浪费时间

迭代加深搜索 (iterative deepening search) 的本质还是深度优先搜索

限制搜索的深度为 dep，若搜索深度达到限定值 dep 时直接返回，一般用于找最优解

不断增加 dep，则只要答案深度有限，当 dep 增加到答案深度时一定能搜索到



为什么不使用 BFS?

是否需要记忆化避免重复搜索?

#629、Addition Chains

实验舱 青少年编程 走近科学 走进名校

题目描述

已知一个数列 a_0， $a_1 \ldots a_m$

其中 $a_0 = 1, a_m = n, a_0 < a_1 < a_2 < \cdots < a_{m-1} < a_m$

对于每个 k ，需要满足 $a_k = a_i + a_j$ ($0 \leq i, j \leq k-1$,这里 i 与 j 可以相等)

现给定 n 的值,要求 m 的最小值(并不要求输出),及这个数列每一项的值(可能存在多个数列,只输出任一个满足条件的就可以了)

输入格式

多组数据，每行给定一个正整数 n
输入以 `0` 结束

输出格式

对于每组数据,输出满足条件的长度最小的数列

数据范围与提示

对于全部的数据 $1 \leq n \leq 100, 1 \leq k \leq m$

样例输入

```
5
7
12
15
77
0
```

样例输出

```
1 2 4 5
1 2 4 6 7
1 2 4 8 12
1 2 4 5 10 15
1 2 4 8 9 17 34 68 77
```

#629、Addition Chains

根据题意 m 不会超过 10，即深度不超过 10

但是搜索分支较多(解空间较大)，可限制深度进行 IDDFS

为了让最后一个数尽快接近 n，可以从大到小枚举每一层的数

枚举每一层的数，每一层由前面的某两层相加得到

可行性剪枝

　　每一层的数不能超过 n

　　序列必须递增

重复性剪枝

　　每一层出现过的数，不能再次出现

```cpp
bool dfs(int deep, int limit)
{
  if (deep > limit) //超过限制深度
    return false;
  if (ans[deep - 1] == n)
    return true;
  bool vis[101] = {false};
  for (int i = deep - 1; i >= 1; i--)
    for (int j = i; j >= 1; j--)
    {
      int x = ans[i] + ans[j];
      if (x > n
          || ans[deep - 1] >= x //非递增
          || vis[x])                      //排除同一层出现过的数
        continue;
      vis[x] = true, ans[deep] = x;
      if (dfs(deep + 1, limit))
        return true;
    }
  return false;
}
```

#630、埃及分数

题目描述

在古埃及,人们使用单位分数的和（形如 $\frac{1}{a}$ 的, a 是自然数）表示一切有理数

如：$\frac{2}{3}=\frac{1}{2}+\frac{1}{6}$,但不允许 $\frac{2}{3}=\frac{1}{3}+\frac{1}{3}$,因为加数中有相同的

对于一个分数 $\frac{a}{b}$,表示方法有很多种,但是哪种最好呢？首先,加数少的比加数多的好,其次,加数个数相同的,最小的分数越大越好。如：

$$\frac{19}{45}=\frac{1}{3}+\frac{1}{12}+\frac{1}{180}$$
$$\frac{19}{45}=\frac{1}{3}+\frac{1}{15}+\frac{1}{45}$$
$$\frac{19}{45}=\frac{1}{3}+\frac{1}{18}+\frac{1}{30}$$
$$\frac{19}{45}=\frac{1}{4}+\frac{1}{6}+\frac{1}{180}$$
$$\frac{19}{45}=\frac{1}{5}+\frac{1}{6}+\frac{1}{18}$$

最好的是最后一种,因为 $\frac{1}{18}$ 比 $\frac{1}{180},\frac{1}{45},\frac{1}{30},\frac{1}{18}$ 都大。 注意,可能有多个最优解。

如：

$$\frac{59}{211}=\frac{1}{4}+\frac{1}{36}+\frac{1}{633}+\frac{1}{3798}$$
$$\frac{59}{211}=\frac{1}{6}+\frac{1}{9}+\frac{1}{633}+\frac{1}{3798}$$

由于方法一与方法二中,最小的分数相同,因此二者均是最优解

给出 a,b ,编程计算最好的表达方式。保证最优解满足：最小的分数 $\geq \frac{1}{10^7}$

输入格式

一行两个整数,分别为 a 和 b 的值

输出格式

输出若干个数,自小到大排列,依次是单位分数的分母

样例输入

```
19 45
```

样例输出

```
5 6 18
```

数据范围与提示

对于全部的数据, $0<a<b<1000$

#630、埃及分数

若朴素 DFS 递归分支深度可能为无限大，朴素 BFS 每层可扩展的分支无限多

应当考虑 IDDFS

若 剩余要凑出的分数为 $\frac{a}{b}$（若进入子问题时都约分，那么 $a \perp b$），当前枚举的分母为 p

剩余子问题规模为 $\frac{a}{b}-\frac{1}{p}=\frac{ap-b}{bp}=\frac{ap-\frac{b}{g}}{\frac{bp}{g}}$ 其中 $g = (ap-b, bp)$

可限制 序列长度 L 以及 最大分母的上界 S

- L 从 1 开始每次迭代后 $L \leftarrow L+1$
- S 从 1000 开始每次迭代后 $S \leftarrow S \times 10$，直到 $S >$ 10^7

假设当前枚举到第 i 层

考虑剪枝

#630、埃及分数

- **Optimization 1**

  若枚举完前 $L-1$ 个分母

  最后一个分母无需枚举，仅需验证剩余分子是否为 1

- **Optimization 2**

  以 $\frac{a}{b}=\frac{1}{x}+\frac{1}{y}+\frac{1}{z}$ 为例，通分后

  $$\frac{a}{b}=\frac{xy+xz+yz}{xyz}$$

  即 $ag=xy+xz+yz$，$bg=xyz$ 其中 $g=(xy+xz+yz,xyz)$

  由于限制最大分母不超过 S

  若 $a>3S^2$ 或 $b>S^3$ 此时无解

  类似若枚举到第 i 个分母剩余问题为 $\frac{a}{b}$

  若 $a>(L-i+1)\times S^{L-i}$ 或 $b>S^{L-i+1}$ 此时无解

#630、埃及分数

- **Optimization 3**

由于分母递增，若前一分母为 p' 需满足 $p > p'$

又由于当前单位分数不能超过剩余部分，所以有

$$\frac{1}{p} \le \frac{a}{b} \Rightarrow p \ge \frac{b}{a}$$

为了能在 L 层全部分解完，需满足

$$(L - i + 1) \times \frac{1}{p} \ge \frac{a}{b}$$

即

$$p \le \frac{b(L - i + 1)}{a}$$

#630、埃及分数

考虑进一步缩紧下界

若 $i+1=L$ (除第 i 层外仅剩一层 剩余部分为 $\frac{a}{b}-\frac{1}{p}$) 为保证下一层分母($\frac{1}{\frac{a}{b}-\frac{1}{p}}$)依然存在，需满足

$$\frac{1}{\frac{a}{b}-\frac{1}{p}} \le S \Rightarrow \frac{bp}{ap-b} \le S \Rightarrow p \ge \frac{bS}{Sa-b}$$

类似的在第 i 层需满足

$$p \ge \frac{bS}{Sa-(L-i)b}$$

综上

$$p \in \left[\max\left(p', \left\lceil\frac{b}{a}\right\rceil, \frac{bS}{Sa-(L-i)b}\right), \min\left(S, \frac{b(L-i+1)}{a}\right)\right]$$

#630、埃及分数

- **Optimization 4**

当 $i+1=L$ 时 (含当前层仅需枚举两个分母)

设剩余两个分母分别为 x,y 那么

$$\frac{a}{b}=\frac{1}{x}+\frac{1}{y} \Longrightarrow \begin{cases} az=x+y \\ bz=xy \end{cases} \Longrightarrow x^2-azx-bz=0$$

考虑枚举 z

当 $\Delta=(az)^2-4bz>0$ 时存在 $x\neq y$ 的解

当 $\Delta=0$ 时 $x=y=\frac{az}{2}$ 与规则相悖

不妨令

$$x=\frac{az-\sqrt{\Delta}}{2} \quad y=\frac{az+\sqrt{\Delta}}{2}$$

#630、埃及分数

在求解时需验证 Δ 是否为平方数 以及 是否满足 $2 \mid az-\sqrt{\Delta}$

考虑 z 的枚举上下界

由于 $x,y \le \ s$ 那么

$$az = x + y \le 2S \ \Rightarrow z \le \frac{2S}{a} \qquad\qquad bz = xy \le S^2 \ \Rightarrow z \le \frac{S^2}{b}$$

同时在求解时 $\Delta > 0$ 那么

$$\Delta = (az)^2 - 4bz > 0 \Rightarrow z > \frac{4b}{a^2}$$

综上

$$z \in \left[\left\lfloor\frac{4b}{a^2}\right\rfloor + 1, \min\left(\frac{2S}{a}, \frac{S^2}{b}\right)\right]$$

IDA*

A * 在搜索过程中队列中维护了过的的节点，空间开销较大

IDA * 为采用了迭代加深算法的 A * 算法,本质上是一种借助了A * 的 IDDFS

优点

- 空间开销小
- 利于深度剪枝

缺点

- 重复搜索

搜索时对每个结点进行估价,如果 **当前代价+预估代价** 仍然大于层数限制 dep ,则直接回溯

#2524、骑士精神

题目描述

在一个 5×5 的棋盘上有 12 个白色的骑士和 12 个黑色的骑士,且有一个空位

在任何时候一个骑士都能按照骑士的走法移动到空位上

> *它可以走到和它横坐标相差为 1 ,纵坐标相差为 2 或者横坐标相差为 2 ,纵坐标相差为 1 的格子*

给定一个初始的棋盘,怎样才能经过移动变成如下目标棋盘：为了体现出骑士精神,他们必须以最少的步数完成任务

输入格式

第一行有一个正整数 T ,表示一共有 T 组数据

接下来有 T 个 5×5 的矩阵, `0` 表示白色骑士, `1` 表示黑色骑士, `*` 表示空位

两组数据之间没有空行

输出格式

每组数据输出占一行

如果能在 15 步以内(包括 15 步)到达目标状态,则输出步数,否则输出 `-1`

数据范围

对于全部的数据 $1 \le T \le 10$

若直接 BFS

最大步数 15 ,每一步可以尝试八个方向

转移的状态接近 8^{15} 个，内存将会超限

以与目标状态的差异格子数量为估价函数

当前限制深度为 dep

若 当前深度+预估步数超过 dep 直接返回

不断增加限制深度

若预估步数为 0，代表已搜得答案

也可双向 BFS 解决

#2530、旋转游戏

题目描述

如下图所示,有一个 `#` 形的棋盘,上面有 $1,2,3$ 三种数字各 8 个

给定 8 种操作,分别为图中的 $A \sim H$

这些操作会按照图中字母和箭头所指明的方向,把一条长为 7 的序列循环移动 1 个单位

例如下图最左边的 `#` 形棋盘执行操作 A 后,会变为下图中间的 `#` 形棋盘,再执行操作 C 后会变成下图最右边的 `#` 形棋盘

给定一个初始状态,请使用最少的操作次数,使 `#` 形棋盘最中间的 8 个格子里的数字相同

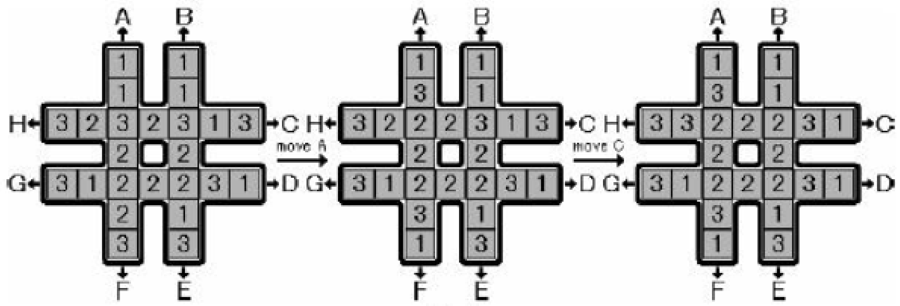


Fig.1

输入格式

输入包含多组测试用例

每个测试用例占一行,包含 24 个数字,表示将初始棋盘中的每一个位置的数字,按整体从上到下,同行从左到右的顺序依次列出

输入样例中的第一个测试用例,对应上图最左边棋盘的初始状态

当输入只包含一个 0 的行时,表示输入终止

输出格式

每个测试用例输出占两行。

第一行包含所有移动步骤,每步移动用大写字母 $A \sim H$ 中的一个表示,字母之间没有空格
如果不需要移动则输出 `No moves needed`

第二行包含一个整数,表示移动完成后,中间 8 个格子里的数字

如果有多种方案,则输出字典序最小的解决方案

#2530、旋转游戏

将所有数字编号，将每个操作移动位置打表记录

估价函数

中间出现次数最多的数数量记为 t

预估步数为 $8-t$

每一步枚举可能的操作

从小到大枚举可以保证字典序最小

相邻两个操作不能是互逆的，可以此剪枝

实验舱
青少年编程
走近科学 走进名校

谢谢观看