实验舱
青少年编程
走近科学 走进名校

提高算法班

搜索剪枝优化

Mas

剪枝

搜索的进程可以看作是从树根出发,遍历一棵倒置的树——搜索树的过程

所谓剪枝,就是通过某种判断,避免不必要的遍历过程,形象的说,就是剪去搜索树中的某些“枝条”,故称剪枝

正确性

剪枝能优化执行效率,是因为它“剪去”搜索树中的一些“枝条”

但在剪枝时,将有所需解的枝条剪掉,一切优化将无意义

准确性

在保证了正确性的基础上,对剪枝判断的第二个要求就是准确性,即能够尽可能多的剪去不能通向正解的枝条

高效性

利用出解的“必要条件”进行判断,会有不含正解的枝条没被剪枝

这些情况下的剪枝判断操作,对程序的效率的提高有副作用为了减少剪枝判断的副作用,除了要改善判断的准确性外,还需要提高判断操作本身的时间效率

剪枝/优化

优化搜索顺序

一些搜索问题的搜索树的各层次、分支间的顺序不固定不同的搜索顺序产生不同的搜索树形态,规模大小也不同

重复性剪枝

若能判定从搜索树的当前节点上沿着不同分支到达的子树是等效的,那么只需要对其中的一条分支搜索

可行性剪枝

在搜索过程中,如果发现分支已经无法到达递归边界,就执行回溯

某些题目条件的范围限制是一个区间,此时可行性剪枝也被称为“上下界剪枝”

最优性剪枝

在最优化问题的搜索过程中,如果当前代价已经超过了历史最优解,此时可以停止对当前分支的搜索,执行回溯

记忆化

可以记录每个状态的搜索结果,在重复遍历一个状态时直接检索并返回

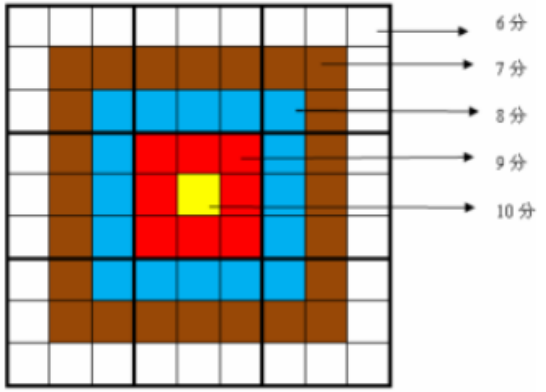
#568、靶形数独

题目描述

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

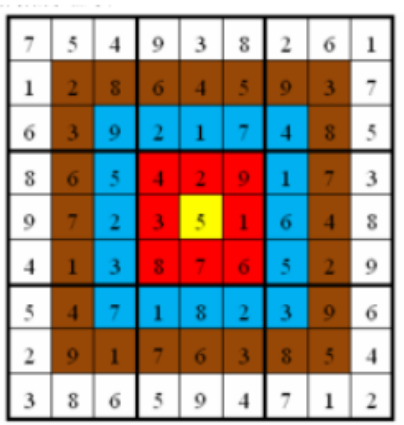
靶形数独的方格同普通数独一样,在 9 格宽× 9 格高的大九宫格中有 9 个 3 格宽 $\times 3$ 格高的小九宫格(用粗黑色线隔开的)。在这个大九宫格中,有一些数字是已知的,根据这些数字,利用逻辑推理,在其他的空格上填入 1 到 9 的数字

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。(如图)



上图具体的分值分布是:最里面一格(黄色区域)为 10 分,黄色区域外面的一圈(红色区域)每个格子为 9 分,再外面一圈(蓝色区域)每个格子为 8 分,蓝色区域外面一圈(棕色区域)每个格子为 7 分,最外面一圈(白色区域)每个格子为 6 分,如上图所示。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

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。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



由于求胜心切,小城找到了善于编程的你,让你帮他求出,对于给定的靶形数独,能够得到的最高分数

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),表示一个尚未填满的数独方格,未填的空格用 0 表示

每两个数字之间用一个空格隔开

输出格式

输出共 1 行。输出可以得到的靶形数独的最高分数

如果这个数独无解,则输出整数 -1

数据规模

对于 40% 的数据,数独中非 0 数的个数不少于 30

对于 80% 的数据,数独中非 0 数的个数不少于 26

对于 100% 的数据,数独中非 0 数的个数不少于 24

#568、靶形数独

需 DFS 搜索所有可行的放置方案

可行性剪枝

每 行 / 列 / 宫 不能填入重复的数字

可使用数组标记

调整搜索顺序

当一行能够被确定时，剩余的可行方案将会被现有局面限制

尽可能多确定一些行，即从空格较少的行开始搜索

更高效的求解数独的方法 Dancing Links X

#1416、还是N皇后

题目描述

N 皇后问题是指将 N 个皇后放置在 $N \times N$ 棋盘上

使得皇后不能相互攻击到,即任意两个皇后都不能处于同一行、同一列或同一斜线上

现在增加棋盘的限制，请你求出放入 N 个皇后的方案数

输入格式

第一行输入一个整数 N

接下来输入 $N \times N$ 的字符矩阵

其中 $*$ 表示可放 $.$ 表示不可放

输出格式

输出一个整数表示方案数

数据规模

对于全部的数据 $1 \leq N \leq 14$

由于枚举可放置的列时做了大量无意义的尝试，朴素 DFS 会超时

使用整数 S_1 表示列状态

使用整数 S_2 表示主对角线状

使用整数 S_3 表示副对角线状态

同时令 ST_{ij} 初始时各行列的限制状态

上述状态各二进制位为 1 表示该列无法选取

如

$S_1 = (0101)_2$ 表示 1，3 列已放置皇后

$S_2 = (0101)_2$ 进入下一行后令 $S_2 = (0011)_2$ 表示 1,2 列不能放置

$S_3 = (0110)_2$ 进入下一行后令 $S_3 = (1100)_2$ 表示 3，4 列不能放置

#1416、还是N皇后

第 i 行 j 的列若不能选取则（$S_1 \mid S_2 \mid S_3 \mid ST_i$）第 j 个二进制位为 1

换言之能放置的列其二进制位为 0，考虑快速求出可放置的位置

令 $P = \sim(S_1 \mid S_2 \mid S_3 \mid ST_i)$

问题转而求出 P 中所有二进制位中的 1 所在位置

在 C++ 中计算机里有符号数是以 **补码** 的形式存储

在补码表示中 x 的相反数 $-x = \sim x + 1$

令 $-P \,\&\, P$ 即可求出 P 的最低且数位为 1 的二进制位对应权

再令 $P \leftarrow P - (-P \,\&\, P)$

不断迭代即可求出 P 中所有二进制位中的 1 所在位置

求出一个合法二进制位的代价 $O(1)$

一次操作也被称为 lowbit 运算

原 01001000

反 10110111

补 10111000

00001000

#628、小木棍

题目描述

乔治有一些同样长的小木棍,他把这些木棍随意砍成几段,直到每段的长都不超过 50

现在,他想把小木棍拼接成原来的样子,但是却忘记了自己开始时有多少根木棍和它们的长度

给出每段小木棍的长度,编程帮他找出原始木棍的最小可能长度

输入格式

第一行为一个单独的整数 N 表示砍过以后的小木棍的总数

第二行为 N 个用空格隔开的正整数,表示 N 根小木棍的长度

输出格式

输出仅一行,表示要求的原始木棍的最小可能长度

样例输入

```
9
5 2 1 5 2 1 5 2 1
```

样例输出

```
6
```

数据范围与提示

对于全部的数据 $1 \leq N \leq 60$

设所有木棍长度之和为sum,搜索的合法长度为len

需要拼凑出 $cnt = \frac{sum}{len}$ 根长度为 len 的木棍

枚举 $len = \max(a_i) \sim sum$

搜索 len 能否凑出 $\frac{sum}{len}$ 根

第一个满足条件的 len 即为答案

#628、小木棍

在下列论述中，称每个小单位为“小木棍”，正在拼的长度为 len 的单位为 “大木棒”

不难发现对于长度为 len 的大木棒的拼凑方案中，小木棍的顺序对结果不产生影响

- ***Optimization* 1**

  合法的 len 只可能存在于小木棍最大值 到 sum 之间，且 sum 必为 len 的倍数

- ***Optimization* 2**

  若当前枚举的小木棍之和已经超过 len，不再继续搜索

- ***Optimization* 3**

  若优先使用长度较小木棍，会使得搜索树深度更容易加深，不利于后续剪枝，应当优先使用较长的木棍

  对所有小木棍从大到小排序，再进行搜索

- ***Optimization* 4**

  拼当前大木棒时，使用了第 i 根小木棍后，下一层从第 $i+1$ 根小木棍开始尝试（i 之前的木棍已考虑过）

#628、小木棍

- ***Optimization 5***

  若长度为 t 的小木棍被确定无法成功参与拼凑当前大木棒，应跳过所有长度为 t 的小木棍

- ***Optimization 6***

  每次拼新的大木棒时，第一根用的小木棍可以直接确定：就是当前剩下木棍里**长度最大**的那一根

  无论它属于哪一根大木棒，它在拼成该大木棒的小木棍里都是最靠前的

  而后续待凑出的木棒都是等价的

  若第一根小木棍选取就失败了，那么不再继续搜索

- ***Optimization 7***

  若不存在一种组成当前大木棒的小木棍拼法，后续搜索无意义

  若使用更短的木棍进行拼凑，即使够拼凑完整

  此时局面中更灵活的较短小木棍数量减少

#627、生日蛋糕

题目描述

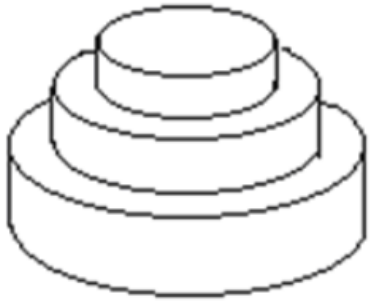
Mr.W 要制作一个体积为 $N\pi$ 的 M 层生日蛋糕,每层都是一个圆柱体

设从下往上数第 i 蛋糕是半径为 R_i ,高度为 H_i 的圆柱

当 $i < M$ 时,要求 $R_i > R_{i+1}$ 且 $H_i > H_{i+1}$

由于要在蛋糕上抹奶油,为尽可能节约经费,我们希望蛋糕外表面(最下一层的下底面除外)的面积 Q 最小

令 $Q = S\pi$,请编程对给出的 N 和 M ,找出蛋糕的制作方案(适当的 R_i 和 H_i 的值),使 S 最小(除 Q 外,以上所有数据皆为正整数)



输入格式

第一行为 N ,表示待制作的蛋糕的体积为 $N\pi$

第二行为 M ,表示蛋糕的层数为 M

输出格式

输出仅一行,一个整数 S (若无解则 $S = -1$)

提示

圆柱相关公式:

体积 $V = \pi R^2 H$

侧面积 $S' = 2\pi RH$

底面积 $S = \pi R^2$

数据范围

对于全部数据, $1 \le N \le 6 \times 10^4, 1 \le M \le 15$

从上到下每一层蛋糕编号为 $1 \sim m$

除侧面积外的其他面积为R_m^2

搜索时仅考虑每层侧面积即可

记搜索中第 i 层蛋糕半径和高度为 R_i, H_i

当前体积表面积为 V, S

#627、生日蛋糕

预处理出前 i 层蛋糕的最小体积 $minV_i$ 和最小面积 $minS_i$

$$minV_i = minV_{i-1} + i^3$$

$$minS_i = minS_{i-1} + 2 \times i^2$$

- ***Optimization* 1**

  考虑从最底层开始搜索,每一层直径和高度从大到小枚举

- ***Optimization* 2**

$$R_i \in \left[i, \min\left(R_{i+1} - 1, \sqrt{\frac{n - V - minV_{i-1}}{i}} \right) \right]$$

$$H_i \in \left[i, \min\left(H_{i+1} - 1, \frac{n - V - minV_{i-1}}{R_i^2} \right) \right]$$

- ***Optimization* 3**

  若 $V + minV_i > n$ 不再继续搜索

#627、生日蛋糕

- ***Optimization* 4**

  记当前最小表面积为ans

  若 $S + minS_i \geq ans$ 不再继续搜索

- ***Optimization* 5**

  前i层蛋糕的面积为

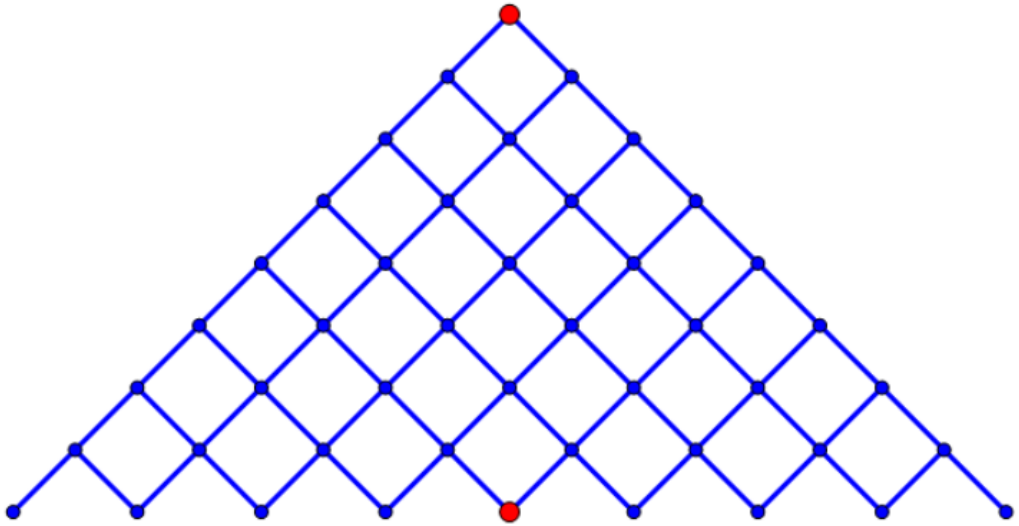
$$\sum_{j=1}^{i} 2 \times R_j \times H_j = \frac{2}{R_{i+1}} \sum_{j=1}^{i} R_j \times H_j \times R_{i+1} > \frac{2}{R_{i+1}} \sum_{j=1}^{i} R_j^2 \times H_j = \frac{2 \times (n - V)}{R_{i+1}}$$

  若

$$S + \frac{2 \times (n - V)}{R_{i+1}} \geq ans$$

  不再继续搜索

meet-in-middle

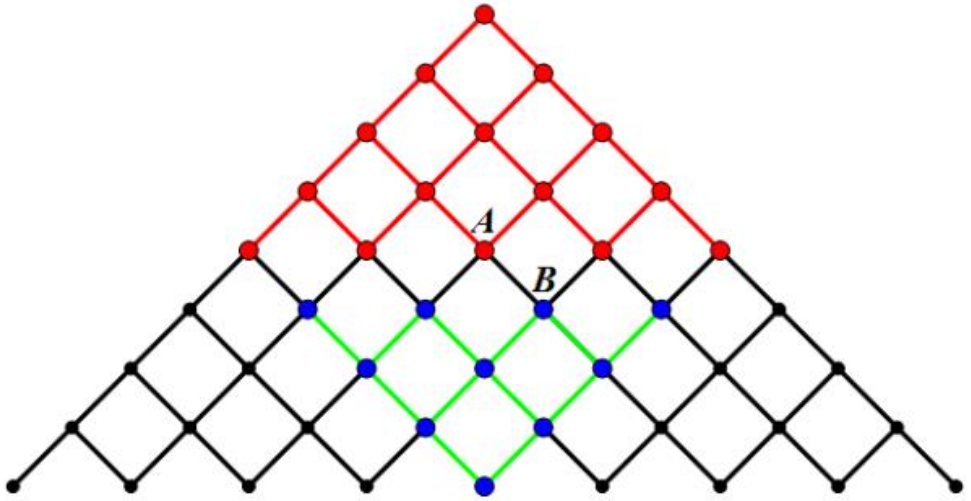


深度优先搜索（depth – first – search）

按照深度优先的方式进行搜索

搜索是一种穷举的方式,把所有可行的方案都列出来

不断尝试,直到找到问题的解



中途相遇（meet in the middle）

将搜索过程分成两半分别搜索,最后将两半结果合并

搜索的复杂度往往为指数级,中途相遇可使指数减半

再其它策略合并处理结果

#2537、方程的解数

题目描述

已知一个 n 元高次方程：

$$k_1X_1^{p_1}+k_2X_2^{p_2}+\ldots+k_nX_n^{p_n}=0$$

其中：$x_1,x_2,\ldots,x_n$ 是未知数，$k_1,k_2,\ldots,k_n$ 是系数，$p_1,p_2,\ldots p_n$ 是指数。且方程中的所有数均为整数

假设未知数 $x_i\in[1,m]\ (i\in[1,n])$ ，求这个方程的整数解的个数

输入格式

第一行一个正整数 n ，表示未知数个数

第二行一个正整数 m 。接下来 n 行，每行两个整数 k_i,p_i

输出格式

输出一行一个整数，表示方程解的个数

数据范围

对于 100% 的数据，$1\le n\le 6,1\le m\le 150$ ，且

$$\sum_{i=1}^{n}|k_im^{p_i}|<2^{31}$$

答案不超过 $2^{31}-1,p_i\in\mathbb{N}^*$

直接 DFS

每一层 m 种可能，一共 n 层

时间复杂度 $O(m^n)$ 无法通过

Meet in middle

先处理前 $\frac{n}{2}$ 层统计各 sum 出现次数

再处理后 $\frac{n}{2}$ 层统计各 $-sum$ 出现次数

时间复杂度 $O(m^{\frac{n}{2}}+km^{\frac{n}{2}})$

BFS

若将问题状态空间看成一张图，那么广度优先搜索就相当于对这张图的广度优先遍历

借助一个队列来实现广度优先搜索，初始时队列中仅包含起始状态

在广度优先搜索的过程中

- 不断从队头取出状态
- 对于该状态面临的所有分支,把沿着每条分支到达的下一个状态(若尚未访问)插入队尾
- 重复执行上述过程直到队列为空

对于最小步数模型的 BFS 队列满足两个性质

两段性

队列内节点状态权值(步数)差值至多为 1

单调性

队列内状态权值单调非降

BFS的优化

对于最小步数模型的BFS，队列内节点状态权值(步数)差值至多为1，队列内步数一定是 **单调** 的

初始状态下队列为空,满足单调性,设某一次队内步数满足单调性 $\{X, X, X, X+1, X+1, X+1\}$

对于下一次更新第一个 X 先出队，扩展得到 $X+1$ 入队，依然满足单调性(对于 $\{X, X, X, X\}$ 情况类似)

设队首元素为 x 当前步数为 dis_x，队列后续某一个元素y

若 y 能够将 dis_x 变得更小（y 能够通过一些点到达 x) 那么 BFS 最小步数无法保证正确性

由于队列的单调性满足 $dis_y \geq dis_x$ 则

$$dis'_x = dis_y + z < dis_x \Rightarrow z < 0$$

对于 BFS 最小步数模型而言，并不存在步数为负，与条件矛盾

同理可证明 $Dijkstra$

#2527、矩阵距离

题目描述

给定一个 N 行 M 列的 01 矩阵 A

$A_{i,j}, A_{k,l}$ 之间的曼哈顿距离定义为:

$$dist(A_{i,j}, A_{k,l}) = |i-k| + |j-l|$$

输出一个 N 行 M 列的整数矩阵 B,其中:

$$B_{i,j} = min_{\forall 1 \le x \le N, 1 \le y \le M, A_{x,y}=1}\{dist(A_{i,j}, A_{k,l})\}$$

输入格式

第一行两个整数 N, M 。

接下来一个 N 行 M 列的 01 矩阵,数字之间没有空格

输出格式

一个 N 行 M 列的矩阵 B,相邻两个整数之间用一个空格隔开

数据范围

对于全部的数据 $1 \le N, M \le 1000$

输入样例

```
3 4
0001
0011
0110
```

输出样例

```
3 2 1 0
2 1 0 0
1 0 0 1
```

多次 BFS 更新最小值

最坏情况下时间复杂度$O(n^2m^2)$

多源 BFS

将所有 1 加入队列,进行BFS

每个位置仅更新一次

时间复杂度 $O(nm)$

正确性能否证明?

#2462、大选演说

题目描述

普朗特为了准备 4202 年的大选,要到全国的各个州给自己做演说

这个国家有 n 个州, m 条连接州与州之间的道路

原本在普朗特的上一个任期中,他下令对任何跨州道路都修边境墙

但是因为普朗特连任失败了,修墙计划被迫搁置,只有一部分的墙修好了

普朗特现在希望从 1 州尽快到达 n 州,边境墙会对通行造成一定的阻碍,因此经过的墙越少越好

给定这些修墙的情况和出行计划,请计算经过的边境墙数目最少是多少?

输入格式

第一行两个整数 n 和 m 表示州和道路的数目

接下来 m 行,每行三个数 x, y, z ,表示一条连接 x 和 y 的双向道路

z 为 1 的时候,这条道路会经过边境墙,否则没有墙

输出格式

一行,表示 1 到 n 经过的最小墙数

输入样例

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

输出样例

```
0
```

数据范围

对于 30% 的数据, $1 \leq n \leq 300$ 。

对于 70% 的数据, $1 \leq n \leq 10^4$ 。

对于 100% 的数据, $1 \leq n \leq 2 \times 10^5, 1 \leq m \leq 4 \times 10^5$

01BFS(双端队列BFS)

朴素 BFS 每次沿着分支的扩展都记为“一步”，通过逐层搜索，解决了求从起始状态到每个状态的最少步数的问题

这等价于在一张边权均为 1 的图上执行广度优先遍历，求出每个点相对于起点的最短距离(层次)

并且每个状态在第一次被访问并入队时,计算出的步数即为所求

然而,如果图上的边权不全是1呢?我们先考虑图上的边权要么是1、要么是0的情况。

对于每一次扩展节点

如果边权为1，将节点插入队尾

如果边权为0，将节点插到队首

正确性如何证明?

是否可以推广到所有仅有两种边权(负边权、非负边权)?

双向BFS

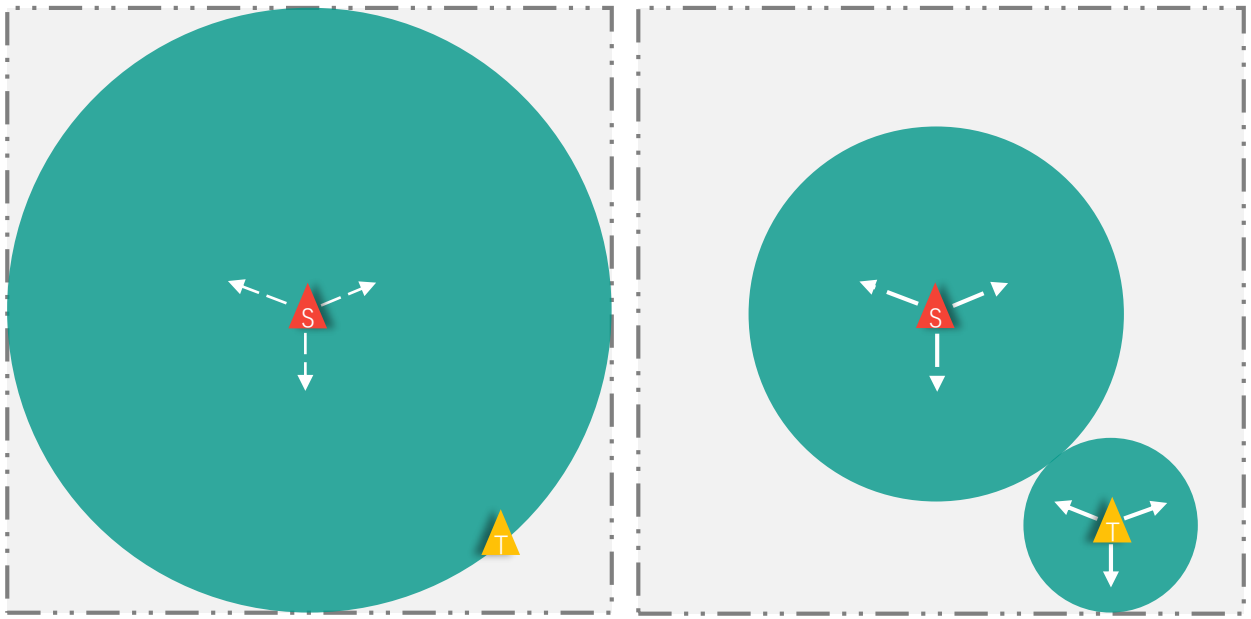
从正反两个方向进行宽度优先搜索,可以大大减少搜索量所需空间,提高搜索效率

从初始状态和目标状态两个方向同时进行扩展,如果两棵解答树在某个节点第一次发生重合,即可终止此搜索过程。

双向宽度优先搜索通常有两种搜索方法:

两个方向交替扩展

选择结点个数较少的那个方向先扩展



#1193、字符变换

题目描述

已知有两个字串 A, B 及一组字串变换的规则(至多 6 个规则):

$$A_1 \to B_1$$

$$A_2 \to B_2$$

规则的含义为:在 A 中的子串 A_1 可以变换为 B_1、A_2 可以变换为 B_2 ...

例如: $A=$ `abcd` , $B=$ `xyz`

变换规则为:

$$abc \to xu$$

$$ud \to y$$

$$y \to yz$$

则此时, A 可以经过一系列的变换变为 B ,其变换的过程为:

$$abcd \to xud \to xy \to xyz$$

共进行了三次变换,使得 A 变换为 B

输入格式

输入的第一行输入两个字符串 A 和 B

接下来若干行输入变换规则:

$A\ B$.
$A_1\ B_1$.
$A_2\ B_2$.
. . . . . .

所有字符串长度的上限为 20

输出格式

对于每组输入数据,若在 10 步(包含 10 步)以内能将 A 变换为 B ,则输出最少的变换步数
否则输出 `NO ANSWER!`

#1193、字符变换

枚举在原字符串中使用替换规则的起点，和所使用的的替换规则

假设字符串长度是 L

直接 BFS

最坏情况下,每一个字符都可以使用 K 个规则扩展

每个节点有 LK 个选择

时间复杂度 $O((LK)^{10})$

双向 BFS

在最坏情况下，起点和终点最多扩展 5 步

时间复杂度$O((LK)^{5})$

谢谢观看