搜索

编程兔教育

主要内容

- dfs和bfs基本应用
- 搜索的最优性,可行性,记忆化剪枝
- 迭代加深搜索
- 二进制搜索
- 折半搜索

搜索算法

- 搜索算法是利用计算机的高性能来有目的的穷举一个问题解空间的部分或所有的可能情况，从而求出问题的解的一种方法。
- 搜索是很重要的基本功。
- OI比赛的目的是“尽量多拿分”，于是搜索就是一个很好的手段。
- 当你不会做题的时候可以暴力搜索，说不定剪剪枝就过了…

比较一下两种搜索算法

- dfs：
- 深度搜索，每次搜完一棵子树，一搜搜到底。
- 往往是把所有可行的情况列举出来了，找一个最优解。
- **易于保存方案**，编码容易，首选的搜索方法。
- 缺点：无法解决优先性质的题，实现会依赖系统栈导致栈溢出。

比较一下目前的两种搜索算法

- bfs：
- 具有**良好的优先性质**。
- 易于实现求最短方案、最少的步骤的搜索问题。
- 缺点：较难统计具体方案。

引例　经典八皇后问题

- 在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。

伪代码

```
void dfs(int dep)
{
        if (dep==n+1) { 输出方案 return;}
        for (int j=1;j<=n;j++)
                if 第dep个可以放置在第dep行第j列
                {
                        vis[dep][j]=1; 记录第dep行第j列放置了一个一个皇后
                        dfs(dep+1);
                        vis[dep][j]=0; 消除影响
                }
}
```

bfs的基本框架

```
初始状态入队列;
while( 队列非空){

    队首元素cur出队;
    if( cur是目标状态 ) return;
    for( int i = 1; i <= 规则数; i++ ){

        由cur产生新状态nxt;

        if( nxt合法且没出现过 ) nxt入队列;

    }

}
```

例1.

给出两个素数p1,p2，x满足如下条件：

(1) x以p1结尾； (2) x是p2的倍数；(3) x是满足条件(1),(2)的所有数中的最小值。

给出t组p1,p2，计算所有x之和。

范围：5 < p1,p2 < 1,000,000; t < 100,000。

例如：p1 = 19， p2 = 23，则x = 1219。

算法一：朴素的枚举（时间复杂度不明确）

```
1  long long calc(int p1, int p2){
2      int t = p1; cnt = 0;
3      while(t){
4          t /= 10; cnt++;
5      }
6      //朴素枚举倍数
7      for(long long x = p2; ; x += p2){
8          if(x % mi[cnt] == p1)
9              return x;
10     }
11 }
```

算法二：稍加改进

- x = p2 * k，显然：x % 10 = p1 % 10 = (p2 % 10) * (k % 10) % 10
- 由此推算k的个位数字y，接下来只需枚举个位数字是y的k。

```
for(;; y += 10){
    if(y * p2 % mi[cnt] == p1)
        return y * p2;
}
```

算法三：继续改进

- 在算法二确定倍数k的个位数字的基础上，x % 100 = p1 % 100 = (p2 % 100) * (k % 100) % 100，由此可以求得k的十位数字，……。主程序调用：dfs(1, y);

```
void dfs(int step, long long cur){
    if(ok) return;
    if(step == cnt){
        ok = true; tmp = cur * p2; return;
    }
    for(int x = 0; x < 10; x++){
        if((cur + x * mi[step]) * p2 % mi[step + 1]
            == p1 % mi[step + 1])
            dfs(step + 1, cur + x * mi[step]);
    }
}
```

例2. 倍数(Multiple), ZOJ1136, POJ1465

【题目描述】

编写程序，实现：给定一个自然数 N， N 的范围为[0，4999]，以及 M 个不同的十进制数字 X1, X2, …, XM （一个，即M≥1），求N 的最小的正整数倍数，满足：N 的每位数字均为 X1, X2, …,XM 中的一个。

【输入格式】

输入文件包含多个测试数据，测试数据之间用空行隔开。每个测试数据的格式为：第 1 行为自然数N；第 2 行为 正整数M；接下来有M 行，每行为一个十进制数字，分别为 X1, X2, …, XM。

【输出格式】

对输入文件中的每个测试数据，输出符合条件的 N 的倍 数；如果不存在这样的倍数，则输出0。

sol

- 把给定的数字按升序排。例如1，2，3，可以组成的数(按从 小到大的顺序)
- 1，2，3
- 11，12，13，21，22，23，
- 31，32，33
- 111，112，113
- 121，122，123
- 131，132，133
- 211，212，213，……
- 显然具有队列的特点。

输入样例	输出样例
22 3 7 0 1 2 1 1	110 0

sol

- 考虑用BFS来按顺序生成正整数，直到能被N整除。
- 如果队列中存生成的数，那么对于无解的情况，没有明显的结束条件。
- 如样例2：n=2，给定的数字只有一个1，可以生成的正整数是1,11,111,……，不存在能被2整除的正整数。

sol

- 改进:
- 由于0<=n<=4999，生成的正整数%n不超过5000。
- 生成的正整数中，如果存在a%n==b%n且a<b，那么没有必要在b的基础上继续搜索。(为什么?)
- 可以在队列中存余数，这样的总状态数不超过5000。

```
struct node{
        int d;          //数字
        int pre;        //前驱
        int yu;         //余数
}
```

```
memset(v,0,sizeof(v));
int head=0,tail=0;
for(int i=0;i<m;i++){
    //最高位不能是0
    if(x[i]==0) continue;
    cur.d=x[i];
    cur.pre=-1;
    cur.yu=x[i]%n;
    q[tail]=cur; tail++;
}
```

```
while(head<tail){
    cur=q[head];
    if(cur.yu==0){   //找到了
        out(head); printf("\n"); return;
    }
    for(int i=0;i<m;i++){
        nxt.d=x[i]; nxt.pre=head;
        nxt.yu=(10*cur.yu+x[i])%n;
        if(!v[nxt.yu]){
            q[tail]=nxt; tail++;
            v[nxt.yu]=true;
        }
    }
    head++;
}
```

例3. 洛谷 P2730 魔板

- https://www.luogu.com.cn/problem/P2730

- bfs
- 只要按照A→C的顺序来扩展新状态，求得的操作序列一定字典序最早。
- 用string类型存储状态，也可以用hash（模大质数）+康拓展开。
- 考虑使用 STL map 来判重。
- 用map存储起点状态到当前状态所需的最少步数，以及到达它的上一个状态与上一个操作代号。

例4. uva10160 服务站

一家公司在N(3<=N<=35)个城市销售个人电脑，城市编号为1、2、……、N。城市间有M条直通航线。公司决定在一 些城市设立服务站，使得对于任意城市X，要么是服务站， 要么它与服务站有直通航线相连。

编写程序，计算公司最少要设立几个服务站。

【输入格式】

包含多组测试数据(不超过10组)，对于每组测试数据：

第一行，两个整数N、M，分别表示城市个数及直通航线 的数量；

以下M行，每行两个整数，表示一条直通航线连接的城市编号；

0 0表示输入的结束。

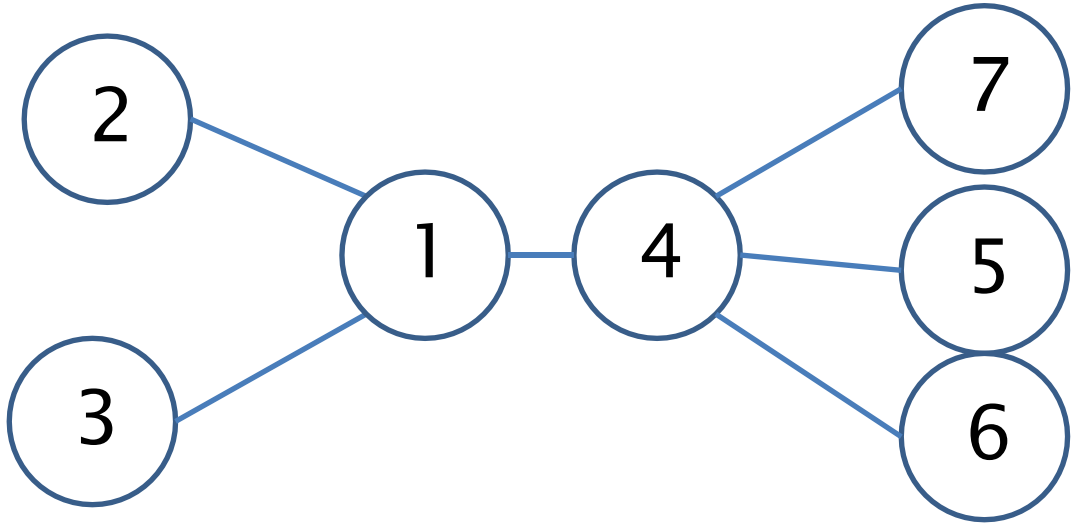
【输出格式】

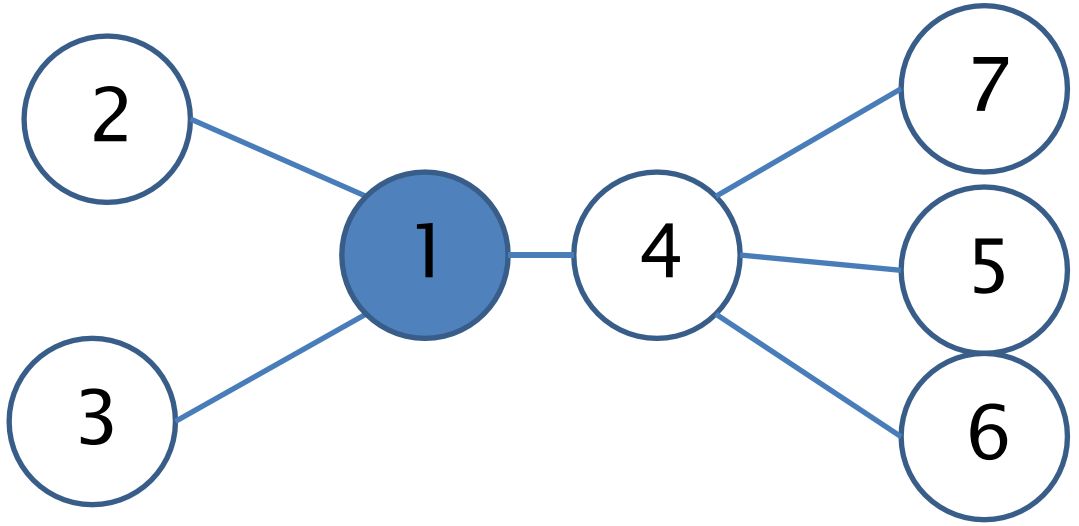
对于每组测试数据，输出一行，表示服务站的个数。

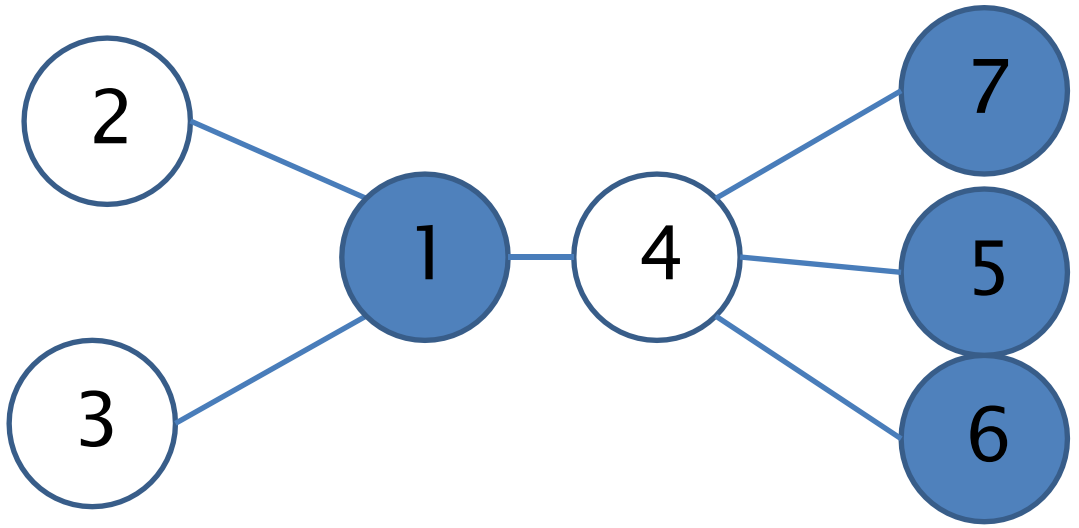
sol

- 朴素的思路，每个城市要么设为服务站，要么不设为服务站，2^N。超时
- 优化和剪枝：
- 在按城市编号依次搜索的过程中，已被覆盖的城市，不再考虑设为服务站。 实际是错误的， 反例：

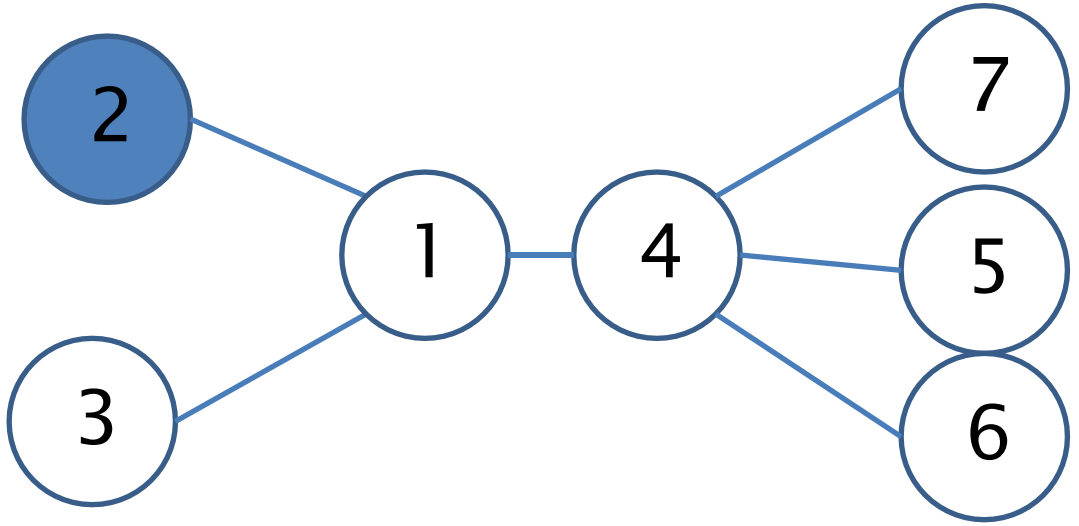
输入样例	输出样例
8 12 1 2 1 6 1 8 2 3 2 6 3 4 3 5 4 5 4 7 5 6 6 7 6 8 0 0	2

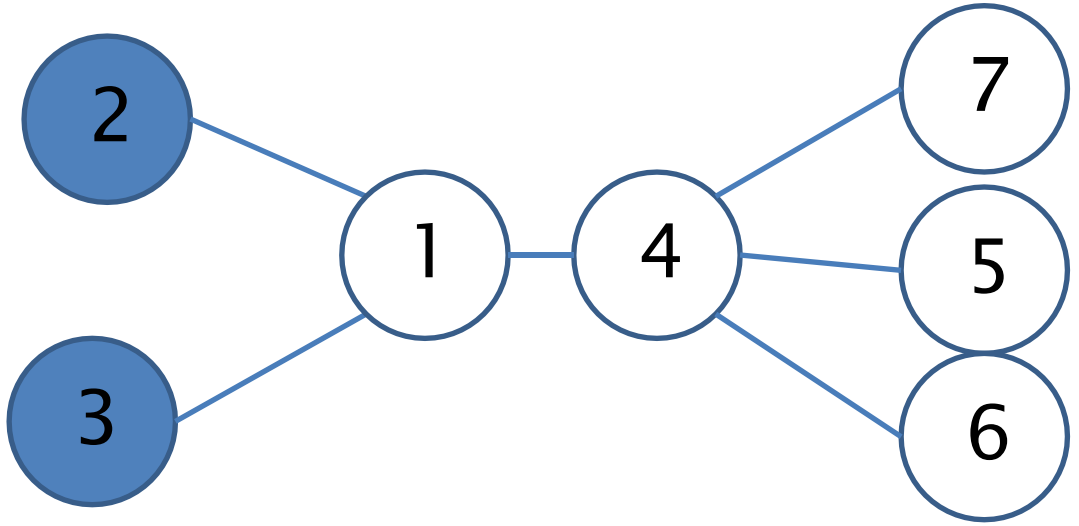

2
3
1
4
7
5
6

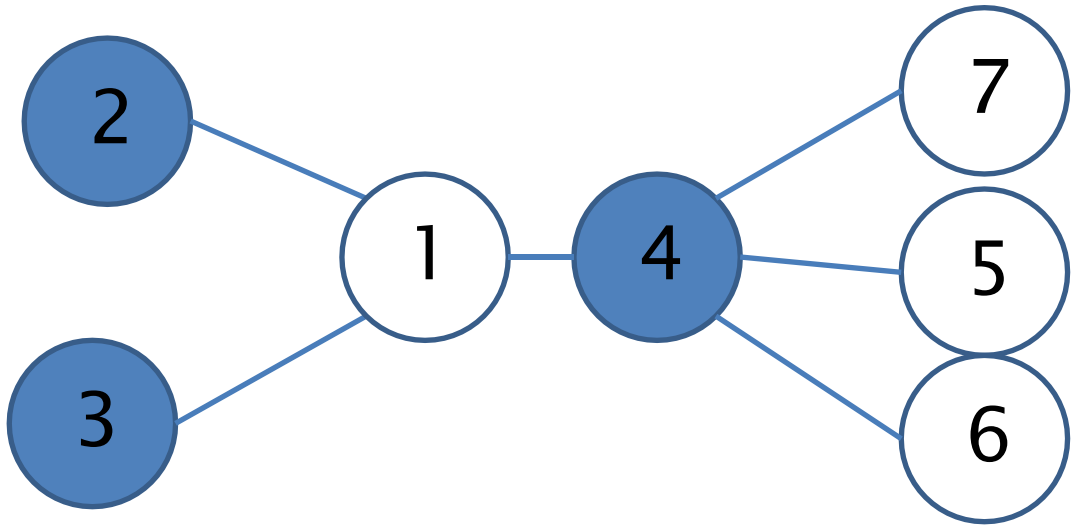

2
1
4
7
5
3
6



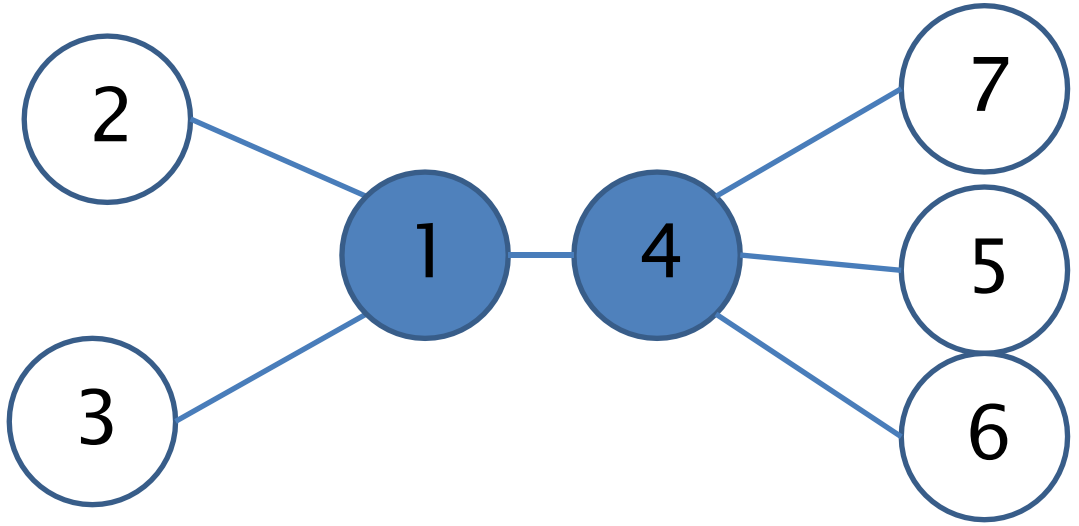
ans=4


2
1
4
7
5
3
6


2
3
1
4
7
5
6



ans=3



正确结果： ans=2

sol

- 修正:
- 如果把当前城市设为服务站，不会使得被覆盖的城市增加，那么该城市不需要设为服务站。
- 若搜索过程中设立的服务站数量已经超过当前最优值， 没必要再继续下去。
- 对于当前城市之前的某个城市，若它没有被覆盖，且它的最大邻接点编号小于当前城市，也就是说，后面再怎 么设服务站也没法覆盖这个城市，没必要再继续下去。

如何优化搜索

- 搜索的搜索树非常大，但是有很多状态是明显不会有解或者无法更新最优解的，遇到这种情况，我们显然不需要继续进行搜索。
- 让搜索树去掉一些没用的“枝条”，所以被称作“剪枝”。

剪枝

- 搜索的剪枝最基本的有可行性剪枝、最优性剪枝、记忆化剪枝。
- 可行性剪枝就是在这个不可能成为合法解的时候剪枝。
- 最优性剪枝就是在这个不可能成为最优解的时候剪枝。
- 记忆化剪枝就是记录搜索过的每一个状态，当重复搜索到相同的状态的时候直接返回结果。

例5. poj2676数独

- 九宫格问题，也有人叫数独问题。
- 把一个9行9列的网格，再细分为9个3*3的子网格，要求每行、每列、每个子网格内都只能使用一次1~9中的一个数字，即每行、每列、每个子网格内都不允许出现相同的数字。
- 给你一个9*9的网格，有些格子填了数，要你求出这个数独的解。

sol

- 爆搜每个位置填1~9，设row[i][x]表示第i行中数字x是否填了，设col[j][y]表示第j列中数字y是否填了，grid[k][z]表示第k个3*3的子网格中数字z是否填了。
- 注意搜索的时候要回溯。

sol

- 可行性剪枝
- 开row,col,gird数组是为了记录当前位置哪些数可以填，哪些数不可以填，这样就可以排除掉一些枚举的情况。
- row 和 col的标记比较好处理，关键是找出grid子网格的序号与行i列j的关系，即要知道第i行j列的数字是属于哪个子网格的。

sol

- 首先我们假设子网格的序号如下编排

0	1	2
3	4	5
6	7	8

- 由于1<=i、j<=9，我们有

i	i/3	j	j/3
0	0	0	0
1	0	1	0
2	0	2	0
3	1	3	1
4	1	4	1
5	1	5	1
6	2	6	2
7	2	7	2
8	2	8	2

sol

- 令a= i/3 ，b= j/3，根据九宫格的行列与子网格的关系，我们有

a	b	grid 序号 k
0	0	0
0	1	1
0	2	2
1	0	3
1	1	4
1	2	5
2	0	6
2	1	7
2	2	8

- 不难发现 3a+b=k，即 3*(i/3)+j/3=k
- 如果数组下标为1~9，grid编号也为1~9，上面的关系式可变形为 3*((i-1)/3)+(j-1)/3+1=k

例6. poj3009

- 在二维平面内有一个冰壶开始在起点S，要到终点T，推冰壶冰壶会一直走到出界，碰到障碍物会停下，并且障碍物会消失，求最少多少步完成。

sol

- 爆搜，每次枚举往那个方向走，然后一直走过去，注意回溯，当你从上一层递归回来的时候你要把撞掉的障碍物复原回去。

sol

- 最优性剪枝
- 当你在搜索时要最优化某个值时，你可以记录ans表示当前最优值，如果你在搜索时还没搜完答案却超过了ans，那么就退出即可。
- 这个题记录ans表示最小多少步能够完成，在搜索时判断当前步数和ans即可。

例7. 洛谷 P1731 生日蛋糕

- https://www.luogu.com.cn/problem/P1731

- 压缩半径和高度的区间
- 可行性剪枝+最优化剪枝

sol

- 问题实质就是在规定的总体积N和层高M内，分成不同的小圆柱，并堆叠，同时要求堆叠的圆柱的半径、高度逐渐递减，求这些圆柱的最小的表面积。
- 我们有几个比较基础的判断：
- 是否做到了m层
- 是否最终体积为0
- 是否当前面积最小

sol

- 优化1：可行性剪枝
- 如果当前剩下的做不到m层，肯定要跳出。
- 同理，如果剩下的太多，哪怕以最大代价做完m层都会用剩，也要跳出。
- 这些计算都是O(1)的，代价很小，但是作用很大。

sol

- 优化2：最优性剪枝
- 如果当前表面积+余下侧面积>之前的最优值，直接返回。
- 正确性非常显然。
- 每次计算这些东西也都是O(1)的，代价也很低。

例8. P3183 [HAOI2016] 食物链

- https://www.luogu.com.cn/problem/P3183
- 给一张图，求由入度为零的点到出度为零的点的路径的条数。

sol

- 记忆化搜索
- 由题意可知对于一个在图中间的有多个出度入度的点会被重复搜很多次，所以选择记忆化，记录一个点到出度为零的点的路径个数，从每个入度为零的点dfs一遍就好了。

优化的方向1

- 考虑走迷宫问题。
- 如果我们有很多方向可以走，我们可以试着尽量先走能离终点近一点的方向，这个的道理也很显然。
- 换而言之，改变搜索的顺序可以对搜索起到一定的优化作用。

优化的方向2

- 还记得素数环问题吗？
- 我们是每次用一个函数判断两个数的和是不是质数的，每次的复杂度是O(sqrt(a[i]))。
- 我们可以预处理出1~10000中哪些数是质数，这样对我们的搜索算法也起到了加速的作用，因为我每次调用判断素数的函数，它的复杂度变成了O(1)。

剪枝的注意点

- 我们剪枝的时候，会计算出一些作为条件的变量，计算这些变量会产生一定的效果，但是计算他们也会消耗一定的时间，我们称作“代价”。
- 如果一个剪枝的代价大于它所能产生的优化，那么就是不好的。

例9. 骑士巡游问题

- 一个经典问题。
- 在n×n（n<15）的国际象棋上的某一位置上放置一个马，然后采用象棋中“马走日字”的规则，要求这个马能不重复地走完n×n个格子，试用计算机解决此问题。

sol

- 最基础的做法：
- 我们考虑记录下骑士的位置和某一位置是否被访问过，直接dfs就好了。
- 那么能不能对这个算法进行优化呢？

sol

- 考虑一下马每次的移动：
- 我们将棋盘黑白染色，分为黑点和白点。
- 我们发现每次马会从一个点跳到一个异色点。

sol

- 在选择当前步的方向时去选择满足下面条件的方向，当按这个方向推进到下一位置时，这个位置所可以再选择的方向最少。也就是说在当前位置优先选一个走起来比较“艰难”的方向来推进。先遍历棋盘边缘，再到棋盘中间。
- 或者，每走一步，就判断下棋盘中是否有永远不能被走到的点。

小结

- 深搜的常用方法：
  - 剪枝
  - 改变搜索顺序
  - 预处理信息
  - 记忆化搜索
- 可行性与最优化剪枝：
  - 如果判断下不能做到了，退出。
  - 如果已经不可能再优了，退出。

迭代加深搜索

- 所谓的迭代加深，是一种dfs和bfs的融合体
- dfs之前的一大缺点就是我们不知道要搜多深。迭代加深搜索里面，每次搜索对深度有一定的限制，依次加深，从而达到目的。

迭代加深搜索

```
inline void dfs(int cur,int dep){
    if(cur==end){/*do something*/}
    else if(dep<lim){dfs(new_state,dep+1);}
    else return;
}
int main(){
    for(lim=1;lim<=n;lim++)dfs(start,0);
}
```

例10. P2324 [SCOI2005] 骑士精神

- https://www.luogu.com.cn/problem/P2324
- 一个5×5的棋盘上有12个白骑士和12个黑骑士，且有一个空位。给定一个初始的棋盘，怎样才能经过最少的移动变成如下目标棋盘。
- 答案<=15步

sol

- 启发式搜索入门经典题
- 迭代加深：本题最大次数15次，超过15次直接return。
- 估价函数：当前走的次数和与答案的偏差值，超过16直接return。

- 我们发现这个题要是每次暴力的让一个马跳到空格子中那么在搜索的时候可能就会出现环，怎么办呢？

例11. poj2811 熄灯问题

- 有一个n*m的网格，每个网格有一个权值a[i][j]等于0/1，每次你可以翻转一个格子即从0变成1或从1变成0，翻转之后上下左右相邻的格子也会翻转，问最少多少步使这个网格都变为0。
- n=5,m=6

sol

- 迭代搜索，同样是枚举翻了多少次，同时别忘了回溯的时候还原状态!

二进制搜索

- 还有一种做法，二进制搜索。

二进制搜索

- 二进制搜索听起来十分的高大上，其实就是通过枚举二进制数来进行搜索。
- 比如6的二进制表示就是110，11的二进制表示是1011。
- 考虑一个n位二进制数，当对应位是0/1时表示不同状态。

例11. poj2811 熄灯问题

- 因为有一个性质就是每个格子一定不会被翻转两次，所以只要知道每个格子是否翻转过即可。
- 我们枚举第一行的每一列是否翻转，然后我们发现在看第二行每一列是否被翻转的时候，因为只有第二行会影响到第一行，所以第二行翻转序列也是唯一确定的。

sol

- 举个例子，n=5时，二进制数枚举到11时，对应的二进制表示01011,它表达的意思就是第一行第一列是0不翻转，第二列是1翻转，第三列是0不翻转，第四列是1翻转，第五列是1翻转。
- 第一行一共有2^6种可能，选定一种，而第一行的最终状态是由第二行的开关最终决定（第i行的状态由第i+1行的开关最终决定），第i+1行开关的任务就是让第i行的灯全部关闭，所以最后检查第一行的灯的状态是不是让最后一行全部关闭就可以了。

- 二进制搜索可以不用dfs回溯，写起来方便，跑得快，缺点是二进制位每一位只能表示两种状态。

折半搜索

- meet in the middle
- 又称折半搜索，顾名思义就是先搜索前一半，再搜索后一半，然后把两半合起来求得答案，这样有时候是比搜一个整体来的优秀。

例12.

- 给你一个长为n的小写英文字符串，问你这个字符串有多少个子序列满足每个英文字符在其中出现了偶数次。
- 子序列表示的是一段坐标递增（不一定连续）的序列。
- n<=40

sol

- 我们可以用一个2^26大小的二进制数记录每个字符出现了奇/偶数次。
- 发现你直接使用二进制搜索是2^40不足以通过这个题的。

sol

- 那么我们可以把这个序列分为两段[1,n/2]和[n/2+1,n]，然后通过二进制搜索搜这两段，同时开一个大小为2^26的数组，记录的是左半段每个字符出现的奇偶性为s有多少个不同的子序列，然后在搜索右半段的时候直接统计答案即可。

小结

- 搜索是非常基础的东西，并且用处非常广泛。
- 图论的一些算法都是在某些搜索下加一些优化等产生的。
- 搜索在基础的比赛中可以用来解决问题。

Thanks