搜索

By bruteforce_

目录

一. 递归

- **递归：**是直接或间接地调用自身的算法。它的另一种定义是，用自己的简单情况，定义自己。
- 从前有座山，山里有座庙，庙里有一个老和尚和一个小和尚，老和尚对小和尚说：“从前有座山……”
- 递归代码最重要的两个特征：结束条件和自我调用。自我调用是在解决子问题，而结束条件定义了最简子问题的答案。

一. 递归

- 洛谷P1427

- 要求输入一串数字ai(长度不一定，以0结束)，并将输入的数字反过来输出一遍（除了最后的0）。

输入输出样例

输入 #1	复制	输出 #1	复制
3 65 23 5 34 1 30 0		30 1 34 5 23 65 3	

- 循环实现很简单，但是如果用递归实现呢?

一. 递归

```
void dfs(int i)
{
    if(i>n)
    {
        return;
    }
    dfs(i+1);
    cout<<a[i]<<"\n";
}
```

- 这用到了什么数据结构?
- 栈

一. 递归

- 如何理解递归与栈的关系？
- 在上个例题中，我们可以发现，递归函数实现了一个栈的功能。在函数递归的过程中，后调用的函数会被先执行完毕，而先调用的函数则会后执行完毕，这与栈的“后进先出，先进后出”的概念是相同的。
- 在递归过程中，所占用的是内存中的栈空间，而平时开数组等全局变量所占用的是内存中的堆空间，栈空间往往会比堆空间要小很多，所以在递归过程中尽量不要开大数组，避免出现内存过大等情况。

一. 递归

- 求斐波那契数列的前 n 项（n<=20）

一. 递归

- 求斐波那契数列的前 n 项（n<=20）
- f(i)=f(i-1)+f(i-2)
- 结束条件：f(0)=0，f(1)=1

```
void F(int n)
{
    if(n==1) return 1;
    else if(n==0) return 0;
    else return F(n-1)+F(n-2);
}
```

一. 递归

- 求 a, b 最大公约数(gcd)
- 辗转相除法: $\gcd(m,n)=\gcd(n,m \bmod n)$

```
int gcd(int a,int b)
{
    return !b?a:gcd(b,a%b);
}
```

二. 深度优先搜索 (DFS)

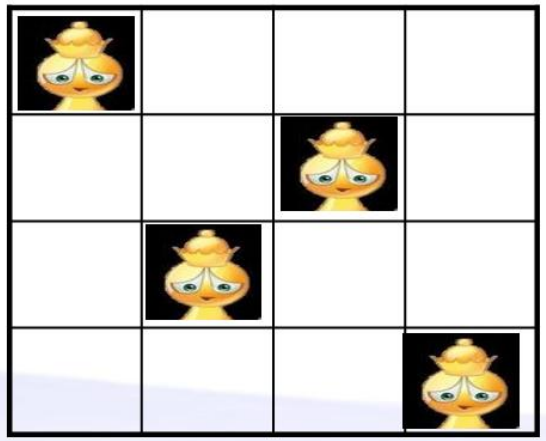
- 什么是搜索？
- 在我们遇到的一些问题当中，有些问题我们不能够确切的找出数学模型，即找不出一种直接求解的方法。
- 解决这一类问题，我们一般采用搜索的方法解决。搜索就是用问题的所有可能去试探，按照一定的顺序、规则，不断去试探，直到找到问题的解，试完了也没有找到解，那就是无解，试探时一定要试探完所有的情况（实际上就是穷举，所以搜索的复杂度最高是搜索域大小级别的（一般是指数级）。但是我们可以通过某些玄学优化将运行速度大大提高）

二. 深度优先搜索 (DFS)

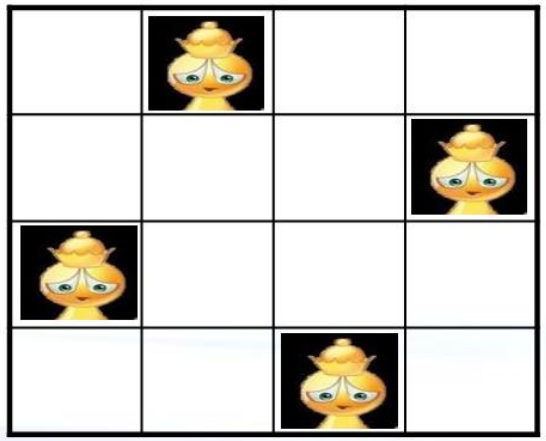
- 深度优先搜索也被称为“回溯法”
- 其过程简要来说是对每一个可能的分支路径深入到不能再深入为止。它是一种基于递归的方法。
- 刚才求解斐波那契数列的递归算法本质上就是 dfs。

二. 深度优先搜索 (DFS)

- **八皇后问题**
- 在8X8格的国际象棋的棋盘上摆放八个皇后，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?



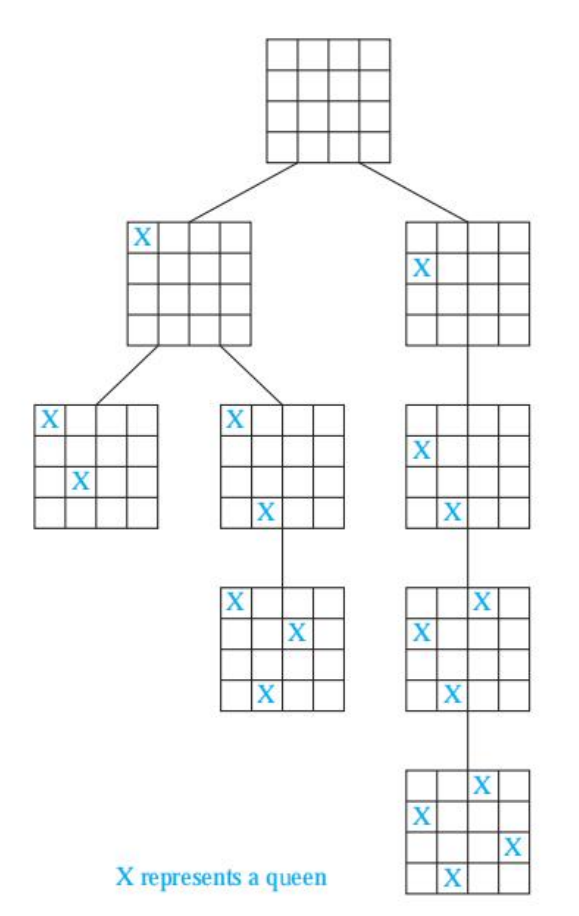
四皇后问题的无效摆放



四皇后问题的有效摆放

二. 深度优先搜索 (DFS)

- 思考：如何用回溯法摆放四皇后?
- 一行一行的放皇后，把该皇后能攻击到的位置排除掉。
- 到了没有位置可放皇后的时候判断是否放置了4个皇后，然后退回去，搜索其他情况。
- 大家根据这个思路写一下 n 皇后问题。
- [洛谷P1219]



二. 深度优先搜索 (DFS)

- CF1829D
- kls 有 n 封情书，他打算送给他那远在非洲的情人（我至今都不知道那位情人的性别）。他得知她的情人只会收恰好 m 封情书，于是 kls 打算把情书分成若干堆，使得某一堆情书恰好有 m 封。但 kls 是一个有原则的人，对于一堆情书，kls 会把它们分成两堆，其中一堆是另一堆的两倍。并且，每一封情书都是神圣的，所以 kls 不能把情书撕开，也就说每一堆情书都不能出现小数。你可以帮助 kls 分情书吗？他的性福生活就靠你们啦~

二. 深度优先搜索 (DFS)

- 简要题意：你初始有一个数 n，对于一个数t，你可以把 t 分成两个数 x, y，使得 x=2y，x+y=t。问进行若干次分数操作后，是否能使得某一个数等于 m。
- n, m<=1e7

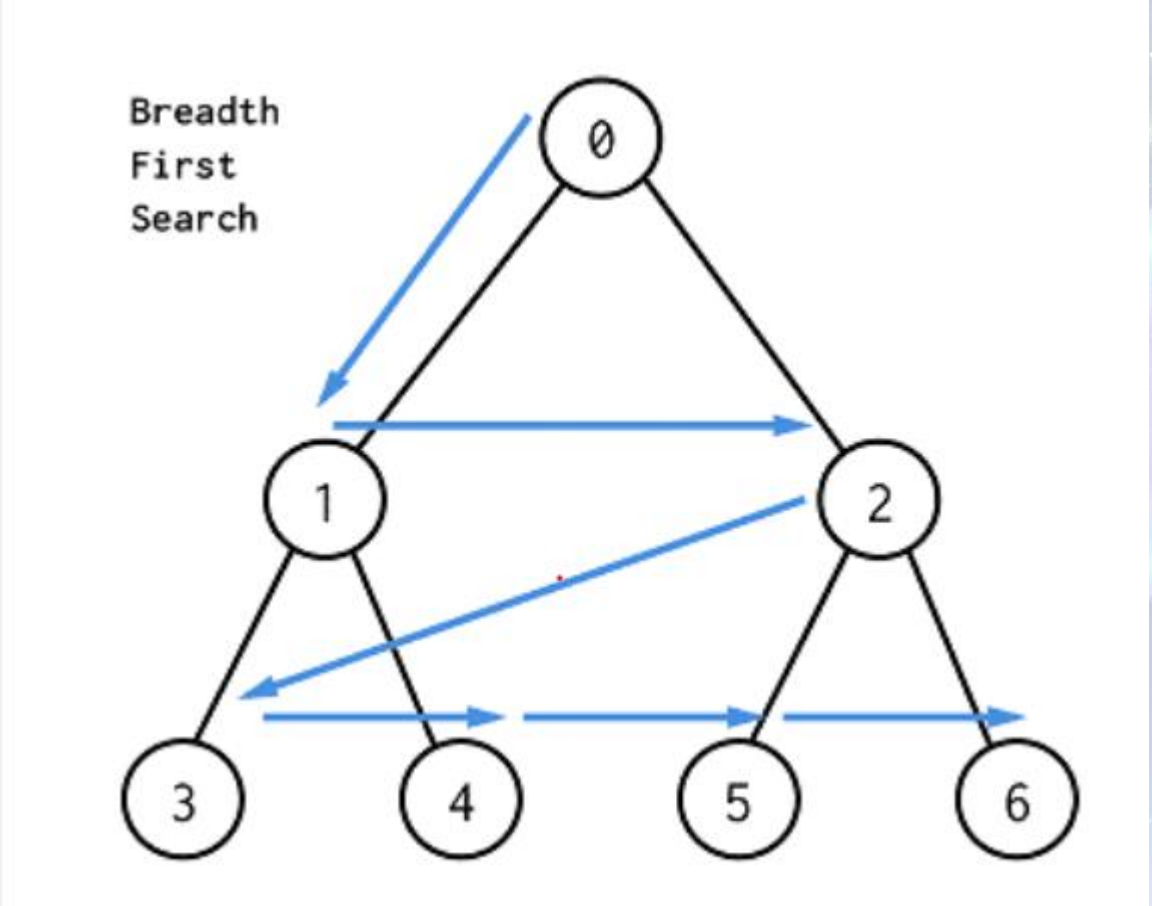
二. 深度优先搜索 (DFS)

```
bool dfs(int n,int m)
{
    if(n<m) return 0;//这一堆小于m，不可能达成目标
    if(n==m) return 1;//达成了目标
    if(n%3!=0) return 0;//分不下去了
    int p=n/3,q=p*2;//分成两堆
    return dfs(p,m)||dfs(q,m);//dfs
}
```

思考：时间复杂度是多少呢？

三. 广度优先搜索 (BFS)

- 百度优先搜索（不是）
- 所谓广度优先。就是每次都尝试访问同一层的节点。如果同一层都访问完了，再访问下一层。



三. 广度优先搜索 (BFS)

- 广度优先搜索使用队列（queue）来实现
- 1、把根节点放到队列的末尾。
- 2、每次从队列的头部取出一个元素，查看这个元素所有的下一级元素，把它们放到队列的末尾。并把这个元素记为它下一级元素的前驱。
- 3、找到所要找的元素时结束程序。
- 4、如果遍历整个树还没有找到，结束程序。

三. 广度优先搜索 (BFS)

- 洛谷B3625

题目描述

复制Markdown 展开

机器猫被困在一个矩形迷宫里。

迷宫可以视为一个 $n \times m$ 矩阵，每个位置要么是空地，要么是墙。机器猫只能从一个空地走到其上、下、左、右的空地。

机器猫初始时位于 $(1,1)$ 的位置，问能否走到 (n,m) 位置。

- n, m<=100

三. 广度优先搜索 (BFS)

```cpp
const int dx[4]={1,0,-1,0},dy[4]={0,1,0,-1};
void O_o()
{
    int n,m;
    cin>>n>>m;
    vector<string> s(n+1);
    for(int i=1; i<=n; i++)
    {
        cin>>s[i];
        s[i]=' '+s[i];
    }
    queue<array<int,2>> q;
    q.push({1,1});
    while(!q.empty())
    {
        auto [x,y]=q.front();
        q.pop();
        if(x==n&&y==m)
        {
            cout<<"Yes";
            return ;
        }
        for(int i=0; i<=3; i++)
        {
            if(s[x+dx[i]][y+dy[i]]=='.')
                q.push({x+dx[i],y+dy[i]});
        }
    }
    cout<<"No";
}
```

- 大家看看这代码有什么问题

三. 广度优先搜索 (BFS)

```
const int dx[4]={1,0,-1,0},dy[4]={0,1,0,-1};
void O_o()
{
    int n,m;
    cin>>n>>m;
    vector<string> s(n+1);
    for(int i=1; i<=n; i++)
    {
        cin>>s[i];
        s[i]=' '+s[i];
    }
    queue<array<int,2>> q;
    q.push({1,1});
    while(!q.empty())
    {
        auto [x,y]=q.front();
        q.pop();
        if(x==n&&y==m)
        {
            cout<<"Yes";
            return ;
        }
        for(int i=0; i<=3; i++)
        {
            if(x+dx[i]>0&&x+dx[i]<=n&&y+dy[i]>0&&y+dy[i]<=m&&s[x+dx[i]][y+dy[i]]=='.')//边界判断
            {
                q.push({x+dx[i],y+dy[i]});
                s[x+dx[i]][y+dy[i]]='#';//走过的路不要重复走
            }

        }
    }
    cout<<"No";
}
```

三. 广度优先搜索 (BFS)

- 大家尝试一下用 bfs 写 kls 情书那题。

四. 记忆化搜索

- 定义
- 记忆化搜索是一种通过记录已经遍历过的状态的信息，从而避免对同一状态重复遍历的搜索实现方式。
- 因为记忆化搜索确保了每个状态只访问一次，它也是一种常见的动态规划实现方式。
- 一般基于 dfs

四. 记忆化搜索

- 给定斐波那契数列的前两项，求斐波那契数列前 n 项，n<=1e7
- 洛谷P1962前60%
- 如果用之前普通的 dfs，时间复杂度是 O(2^n)，TLE。
- 开一个数组记录 f 的值，如果当前点被访问过，直接返回 f[i] 的值。因为每个点最多被访问 O(1) 次，所以时间复杂度为 O(n)

四. 记忆化搜索

```
int F(int n)
{
    if(vis[n]) return f[n];
    if(n==0) return 0;
    if(n==1) return 1;
    vis[n]=1;
    return F(n-1)+F(n-2);
}
```

四. 记忆化搜索

- 还是 kls 情书那题，尝试使用记忆化搜索

```
bool dfs(int n,int m)
{
    if(n<m) return 0;
    if(n==m) return 1;
    if(n%3!=0||vis[n]) return 0;
    vis[n]=1;
    int p=n/3,q=p*2;
    return dfs(p,m)||dfs(q,m);
}
```

Bonus: 搜索与二分的结合

- 洛谷P1902
- 给你一个 n*m 的矩阵，每个矩阵有一个权值 A[i][j], 保证 a[1][j] 和 a[n][j] 恒为 0
- 你要从第 1 行走到第 n 行，你只能上下左右的走。
- 一条路径 P 的代价为max{A[pi][pj]}
- 你需要最小化这个代价
- n, m <= 1000, A[i][j] <= 1000

Bonus: 搜索与二分的结合

- 二分代价mid
- Check中，A[i][j]<=mid可以走，A[i][j]>mid不能走
- 如果可以从第一行走到第 n 行，那么这个代价合法