深度优先搜索（DFS）算法

清华大学计算机系 岑宇阔

自我介绍

- 初中开始
- 高中经历
- 大学情况

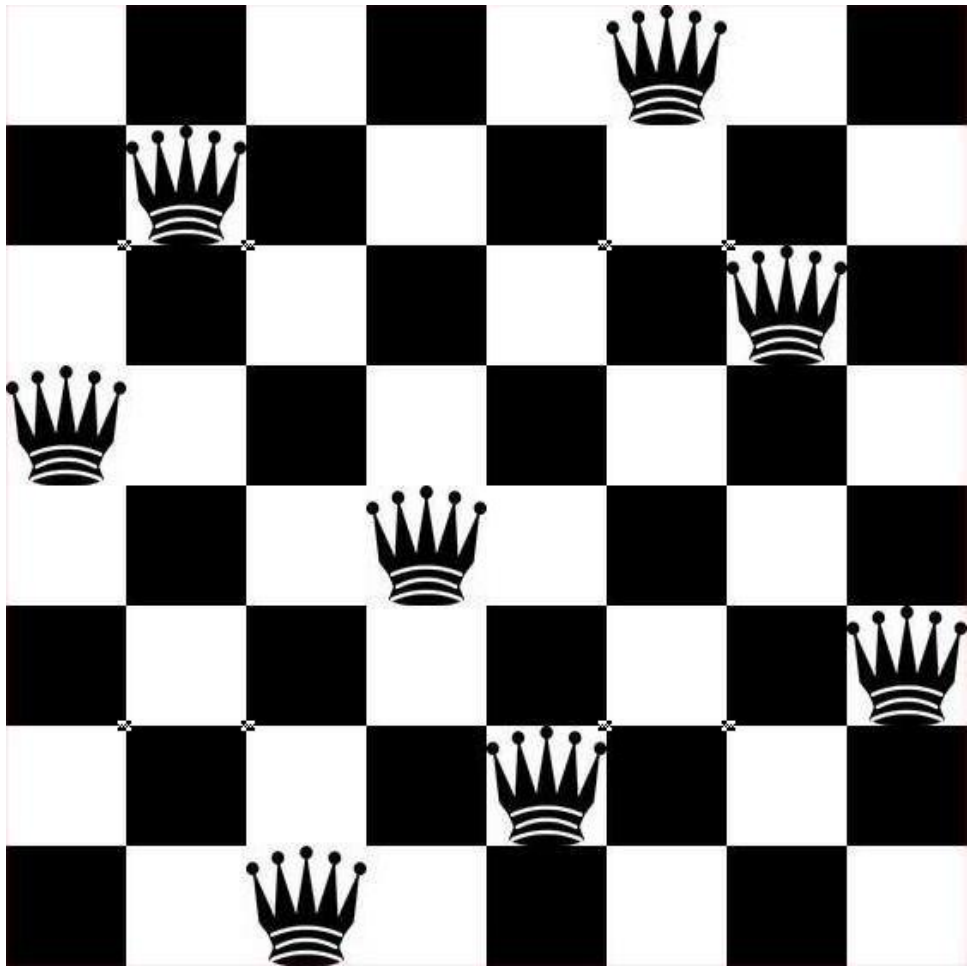
小测试

```cpp
#include <iostream>

using namespace std;

int calc(int x)
{
    if (x <= 0)
        return 0;
    return x + calc(x-1);
}

int main()
{
    cout << calc(10) << endl;
    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

int calc(int a, int b)
{
    if (a == 0)
        return b;
    return calc(b % a, a);
}

int main()
{
    cout << calc(24, 32) << endl;
    return 0;
}
```

八皇后问题

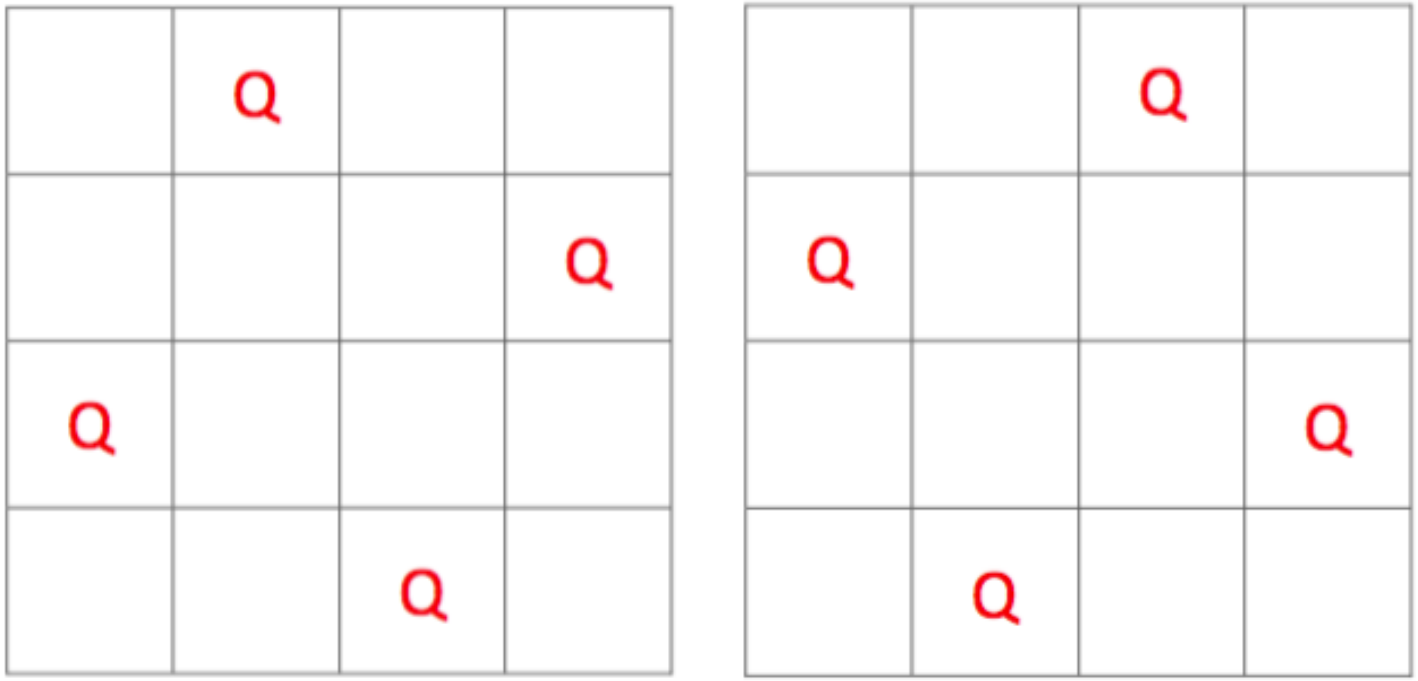
- 八皇后问题，是一个古老而著名的问题。
- 在8×8格的国际象棋上摆放八个皇后，使其不能互相攻击，即任意两个皇后都不能处于同一行、同一列或同一斜线上，问有多少种摆法。



八皇后问题

- 大家不妨试试在纸上画一画四皇后有几种摆法吧。

- [1]: (1,2) (2,4) (3,1) (4,3)
- [2]: (1,3) (2,1) (3,4) (4,2)



什么是搜索？

- 搜索是对一个问题不断地寻找可行方案，然后找到（最优的）可行方案。
- 搜索称为“通用解题法”，但是大部分情况下搜索所需的时间复杂度很高。
- 搜索一般分为：深度优先搜索（DFS）和广度优先搜索（BFS）

深度优先搜索

- 深度优先搜索(Depth First Search, DFS)是一种常见的搜索方法。

- 其中回溯法是一类适用广泛而形象的深度优先搜索算法。所谓回溯，就是“碰壁返回”。通俗的说，也就是你按照一定的顺序去产生一条路径，然后如果遇到了死路，就折回一部分然后试探下一条路径。每一次生成的路径都需要经过判断是否到达最终目标。

- 事实上，我们一般提到深度优先搜索都特指的是回溯法。以下就不再区分深度优先搜索（DFS）和回溯法。

回溯法的基本思想

- 基本思路：
- 若已有满足约束条件的部分解，不妨设为(x1,x2,x3,……xi), i < n，则添加x(i+1)，检查(x1,x2,……,xi, x(i+1))是否满足条件，满足了就继续添加x(i+2) ，若所有的x(i+1)都考察完了，就去掉xi，回溯到(xi,x2,……x(i-1))，添加那些未考察过的xi，看其是否满足约束条件，反复进行，直至得到解或证明无解。

回溯算法框架

```
int g[MAX_DEP];
void dfs(int x)
{
    if (到达终止条件，比如 x >= MAX_DEP) {
        如果搜索结果合法，则输出；
        return;
    }
    for (int i = 下界; i <= 上界; ++i) {
        g[x] = i;
        if (满足约束条件) {
            dfs(x + 1);
        }
    }
}
```

全排列问题

- 从n个不同元素中任取m（m≤n）个元素，按照一定的顺序排列起来，叫做从n个不同元素中取出m个元素的一个排列。当m=n时所有的排列情况叫全排列。

- 如1,2,3三个元素的全排列为：
- 1,2,3；1,3,2；2,1,3；2,3,1；3,1,2；3,2,1

- 输入n (1<=n<=9)，输出1..n的全排列。

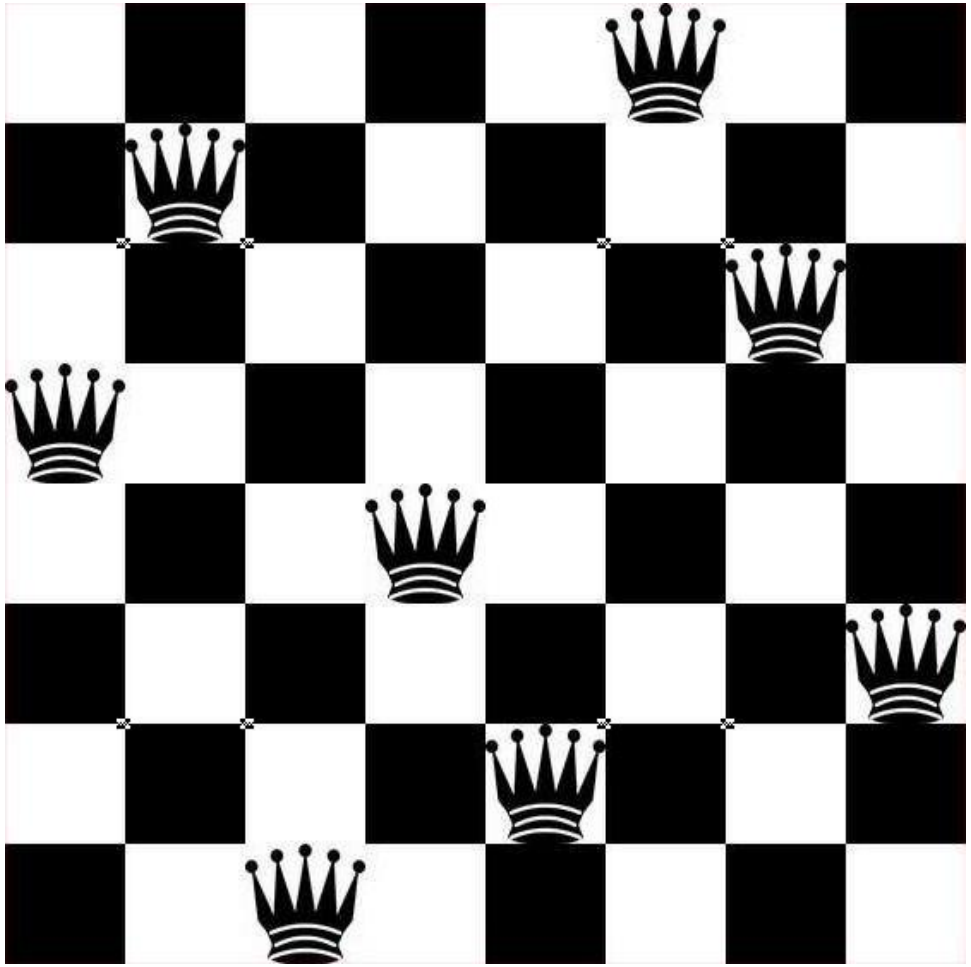
全排列代码

```
int n;
int g[15];

bool check(int x) {
    for (int i = 1; i < x; ++i)
        if (g[x] == g[i])
            return false;
    return true;
}

    int main() {
        scanf("%d", &n);
        dfs(1);
        return 0;
    }
```

```
void dfs(int x) {
    if (x > n) {
        tot++;
        printf("[%d]:", tot);
        for (int i = 1; i <= n; ++i) {
            printf(" %d", g[i]);
        }
        printf("\n");
        return;
    }
    for (int i = 1; i <= n; ++i) {
        g[x] = i;
        if (check(x)) {
            dfs(x + 1);
        }
    }
}
```

八皇后问题

- 让我们再回到八皇后问题。
- 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。



八皇后代码

```
int n;
int g[15];

bool check(int x) {
    for (int i = 1; i < x; ++i) {
        if (g[x] == g[i])
            return false;
        if (x - i == abs(g[x] - g[i]))
            return false;
    }
    return true;
}

int main() {
    scanf("%d", &n);
    dfs(1);
    return 0;
}
```

```
void dfs(int x) {
    if (x > n) {
        tot++;
        printf("[%d]:", tot);
        for (int i = 1; i <= n; ++i) {
            printf(" (%d,%d)", i, g[i]);
        }
        printf("\n");
        return;
    }
    for (int i = 1; i <= n; ++i) {
        g[x] = i;
        if (check(x)) {
            dfs(x + 1);
        }
    }
}
```

排列问题

- 从数码1-9中任选N个不同的数作不重复的排列，求出所有排列的方案。

- 输入一个数N（1<=N<=9）

- 输出所有排列的方案

- 比如输入N=2，答案是1 2; 1 3; …; 1 9; 2 1; … ; 9 7; 9 8。

排列问题

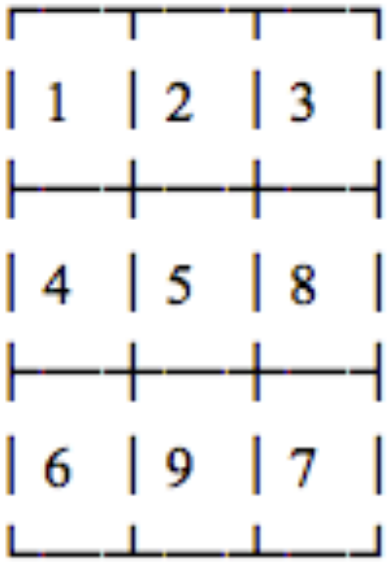
- 只需要将前面全排列问题中的一个n改成9即可。

```
void dfs(int x) {
    if (x > n) {
        tot++;
        printf("[%d]:", tot);
        for (int i = 1; i <= n; ++i) {
            printf(" %d", g[i]);
        }
        printf("\n");
        return;
    }
    for (int i = 1; i <= 9; ++i) {
        g[x] = i;
        if (check(x)) {
            dfs(x + 1);
        }
    }
}
```

一些更加有趣的题目

反幻方

- 在3*3的方格中填入1至9，使得横，竖，对角上的数字之和都不相等。下图给出的是一例。请编程找出所有可能的方案。



反幻方解答

- (1) 用一个二维数组A来存储这个3*3的矩阵。
- (2) 用x表示行，y表示列，搜索时应注意判断放入格子(x，y)的数码是否符合要求;
- （a）如果y=3，就计算这一行的数码和，如果该和已经出现过，则回溯;
- （b）如果x=3，则计算这一列的数码和，并进行判断是否需要回溯;
- （c）如果x=3，y=1还应计算从左下至右上的对角线的数码和;
- （d）如果x=3，y=3还应计算从左上至右下的对角线的数码和。

好括号列

- 括号列 S 由 N 个左括号和 N 个右括号构成，现定义好括号列如下:
- (1) 若A是好括号列，则(A)也是;
- (2) 若A和B是好括号列，则AB也是好的。

- 例如：(()(()))是好的，而(()))(()则不是。

- 现由键盘输入N，求满足条件的所有的好括号列，并打印出来。

好括号列解答

- 从好括号列的定义可知，所谓的“好括号列”就是我们在表达式里所说的正确匹配的括号列，其特点是：从任意的一个位置之前的右括号的个数不能超过左括号的个数。
- 由这个特点，可以构造一个产生好括号列的方法:用x，y记录某一状态中左右括号的个数；若左括号的个数小于N(即x<N)，则可加入一个左括号；若右括号的个数小于左括号的个数，则可加入一个右括号，如此重复操作，直至产生一个好括号列。

外星生命

地球上的科学家收到了来自外星的信号：000023*000011=002093，科学家猜想这是某个外星人的年龄。但有人指出，这些外星人好像不怎么聪明，因为 23*11=253，而非 2093。但是科学家们发现，如果把 000011 改成 00091 的话算式就成立了。他们认为这是接收信号的时候出了差错的缘故。

现在给你这样一个算式，问最少改动几个数字就能使得算式成立?（格式是??????*??????=??????，忽略最高位的进位）

外星生命解答

- 我们可以按位搜索，按照 DFS 的顺序，求出一种方案。然后本着最优原理，以这个答案为基础，把新的分支搜索进行下一位的搜索。
- 在这里我们引出剪枝的概念。在本题中，我们在第 n 步就已经比现有结果更差(需要更多的改进)，这时我们就可以不用搜下去了，我们就退出本轮搜索，这个称为剪枝。
- 在搜索中，剪枝一般都是很重要的，如果没有剪枝，我们就要做很多的无用功，而剪枝避免了很多无用的搜索，大大提高了搜索的速度。所以，剪枝在搜索中的地位是非常高的。如果学好了基础的搜索方法，再掌握一定的剪枝技巧，就能够用好搜索了。

数的划分

数的划分

数的划分：将整数n分成k份（1<=n,k<=10），且每份不能为空，任意两种分法不能相同(不考虑顺序)。例如：n=7，k=3，下面三种分法被认为是相同的：1,1,5；1,5, 1；5,1,1。问有多少种不同的分法。

数的划分

- 数的划分:很清楚,这是一道整数分解的问题。这种分解,较为直接的思路 是递归。我们在本题可以采用深度优先搜索的方法。
- 我们定义一个过程,使其反复递归穷举第 1 份、第 2 份.....第k份,然后寻 找出可行的路径,时间复杂度O(n^k)。这种方法思路十分便捷,但也许会超时。
- 我们有两个很好的剪枝: 如果剩余的够不上最小的则剪去。这是显而易见的常理,但是可以加
- 快速度。 枚举到剩余 **1** 个盘子时判断是否可行。也是加快速度的方法, 使在题目所描述的 n 和 k 的范围之中完全可行。
- 重复的只算一种,我们怎么样处理有关重复的呢?我们可以引入一个参数 min,作为至少每一个要达到 min,顺序由小而大,逐层递归。

四色图问题

四色图问题

在绘制区域地图时,要求以国家为单位着色,所有相邻国家着不同颜色。试编写一程序实现如下功能:使用至多四种不同颜色对地图进行涂色(每块涂一种 颜色),要求相邻区域的颜色互不相同,输出所有可能的涂色方案总数。国家数小于等于10。

四色图问题解答

- 本题可用深度优先搜索来解决。
- 首先,我们读入的是一个邻接矩阵。由于国家数小于等于10，所以我们最多要搜索 4^10=1048576 次，保证了不会超时。所以我们在接下来的搜索过程中，可以用回溯的方法枚举，进行相邻节点是否同色的判断即可。搜索的时候，如果当前点与相邻点的颜色相同则回溯，否则继续深入搜索，如果找出一种方案，则记录并回溯。

Antiprime数

题目：

正整数 k 是一个Antiprime数，如果这个数的约数个数超过比k小的任何数的约数个数。例如：1，2，4，6，12和24都是Antiprime数。

输入：

输入只有一个正整数n（1≤n≤2*10^9）。

输出：

输出只有一个正整数，表示不超过n的最大Antiprime数。

Antiprime数解答

- 题意即让我们求出不超过n的正整数中约数最多的数。
- 一个数的约数个数怎么求？
- 分解质因数之后，将各个指数+1后相乘即可。（乘法原理）
- 要使约数个数尽可能多，那么优先选取小的质因数。
- 2*3*5*7*11*13*17*19*23 = 223092870 < 2*10^9
- 2*3*5*7*11*13*17*19*23*29 = 6469693230 > 2*10^9
- 所以只需考虑前9个质数，即2 3 5 7 11 13 17 19 23
- 搜索k1, k2, …, k9，满足2^k1*3^k2*…*23^k9<=n，使得约数个数(1+k1)*(1+k2)*…*(1+k9)最大。