算法分析与设计
第四章

1020383827@qq.com

doubleh

搜索

- 概念：搜索的本质就是逐步试探，在试探过程中找到问题的解。即从一个初始状态开始，沿某种状态转移方式扩展状态。
- 解决问题：
  - 寻找一个满足条件的解
  - 统计满足条件的解的个数
  - 寻找满足条件的最优解
  - 图的相关问题
- 搜索方法：
  - 深度优先
  - 广度优先

什么时候用搜索

你想不到能用其他方法解决的问题

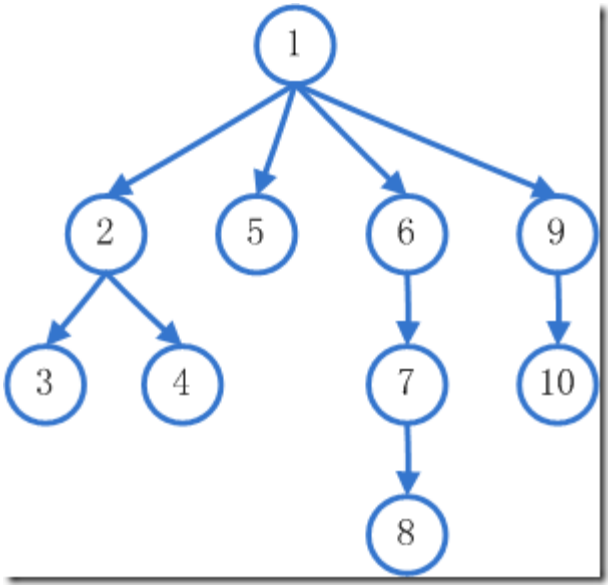
- 一般状态数目不多
- 一般时间复杂度是指数级
- 如果时间复杂度牵扯到一个N，一般N≤100

搜索的时间复杂度

- 时间复杂度
  - 一般是指数级
  - 具体是什么样的指数级？
  - N^N：有N个数，它们每个都可能是1~N
  - N!：有N个1~N的数，它们的全排列数目
  - 2^N：有N个数，每个都只能是0~1（最常见）
- 搜索算法的选择
  - 按1s运行5000w次估算一下，时间复杂度超很多就别搜了（考虑剪枝）

深度优先搜索(dfs)

- 搜索树
- 通常用递归实现
- 特点：“打破沙锅问到底”
- 优点：
  - 编写简单
  - 适用范围广
- 存在栈溢出问题

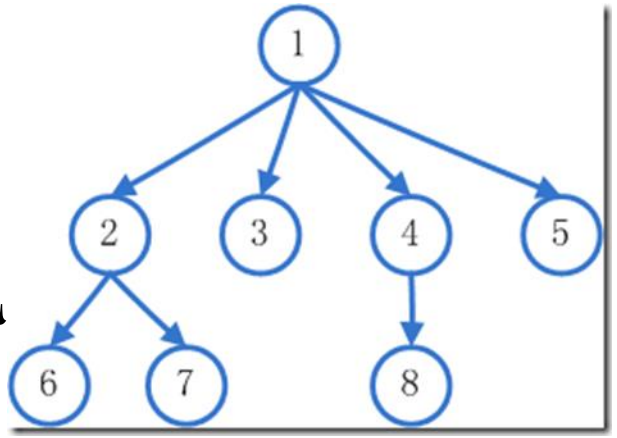


深度优先搜索(dfs)

- void dfs(…; …; …)
- {
  - if (…) {…; return;} (边界)
  - if (…) return; (剪枝)
  - ….
  - dfs(…;…;…);
  - //….回溯
- }

广度优先搜索(bfs)

- 逐层访问
- 通常使用队列(先进先出)实现
- 特点：“浅尝辄止”
- 主要用来解决“求最少步数”的问题

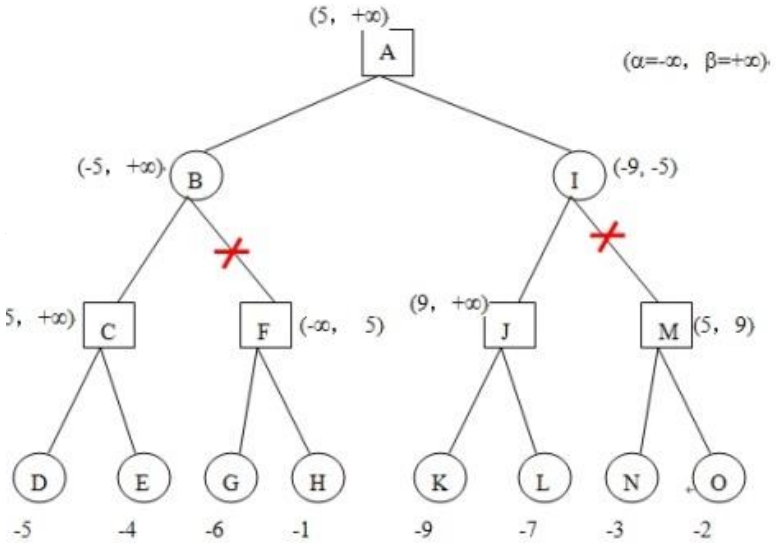


广度优先搜索(bfs)

- void bfs(…)
- {
  - queue<int> q;
  - q.push(…); (起点)
  - while (!q.empty())
  - {
    - int now = q.front(); (取队首扩展)
    - q.pop(); (弹出队首)
    - if (…) {…; break;} {边界}
    - …
    - if (…) {…; q.push(…);} (状态扩展)
  - }
- }

搜索的优化

- 搜索过程的优化：
- 改变搜索顺序
- 剪枝：避免一些不必要的遍历



- 常用方法：
  - 上下界
  - 记忆化

- 中间处理过程的优化：

- 常用方法：
  - 位运算

- 1150 1151 1515 魔板（3题算1题）
- 1317 Sudoku
- 1215脱离地牢
- 1171 The Game of Efil (?)
- 1014 Specialized Four-Dig(*)
- 1709 PropBot
- 1108 Online Selection
- 6276 Pawns
- 1471 No Left Turns
- 1182 Context-Free Clock
- 1710 Painted Calculator (?)
- 7144 Different Triangles(*)

---

*水题　?题目难理解　蓝色诗兄出品　　任意选三题写报告即可满分

Soj 1150 1151 1515 魔板

- 题意：给出魔板的起始状态，三种基本操作，步数上限和目标状态，求从起始状态到目标状态的操作序列，长度不得超过上限。
- 约束：1150限制长度为10，1151长度无限制，1515基本操作不同

其初始状态是

1 2 3 4
8 7 6 5

对魔板可进行三种基本操作：

A操作（上下行互换）：

8 7 6 5
1 2 3 4

B操作（每次以行循环右移一个）：

4 1 2 3
5 8 7 6

C操作（中间四小块顺时针转一格）：

1 7 2 4
8 6 3 5

Soj 1150 1151 1515 魔板

- 基本解法：宽度优先搜索（BFS）+去重

- 注意事项：
- 1）状态表示方法：字符串、康拓展开
- 2）去重功能：排序二叉树（set），哈希表

Soj 1150 1151 1515 魔板

- BFS：
- 优点：保证目标状态第一次出现时，是用最少的步数
- 缺点：需要内存空间巨大
- DFS：
- 优点：需要内存空间小
- 缺点：需要遍历整个搜索树
- 其他搜索算法：A*，IDA，双向BFS，…有兴趣可以自行查阅

Soj 1317 Sudoku

- 题意：
- 给出一个未完成的数独，问这个数独有多少个解

6			7	5		1		
8					3	4		
	3		9	6			2	5
			4			3		2
7								6
2		1			5			
3	1			8	9		4	
		6	5					1
		5		4	2			3

Sample Puzzle

6	2	9	7	5	4	1	3	8
8	5	7	2	1	3	4	6	9
1	3	4	9	6	8	7	2	5
5	9	8	4	7	6	3	1	2
7	4	3	8	2	1	9	5	6
2	6	1	3	9	5	8	7	4
3	1	2	6	8	9	5	4	7
4	8	6	5	3	7	2	9	1
9	7	5	1	4	2	6	8	3

Solution

Soj 1317 Sudoku

- 解法一：
- 直接DFS，枚举每个位置填什么，如果可行则继续填下一个位置

Soj 1317 Sudoku

- 解法一：
- 直接DFS，枚举每个位置填什么，如果可行则继续填下一个位置

- 需要优化程序才能过

Soj 13

- 解法二：
- 利用数独技巧

158	6	18	138	48	7	245	9	5
4	3	2	8	9	58	1	67	57
9	15	17	1	6	2	457	8	3
2	7	38	4	5	1	368	36	9
158	1458	13489	368	8	368	23678	2367	178
6	18	138	2	7	9	38	5	4
3	2	478	9	1	48	578	7	6
178	18	5	678	3	68	9	4	2
78	9	4678	5	248	468	378	1	78

Soj 1317 Sudoku

- 解法二：
- 利用数独技巧
- 好多呀：
- 1.优化枚举顺序，枚举可能填数字最少（之一）的格子
- 2.如果一行、一列、一个九宫格中某个数字能填的位置唯一，就直接填；如果没有能够填写的位置，则直接产生矛盾
- 3….

Soj 1317 Sudoku

- 解法三：
- Dancing Links
- 思想：利用四向链表优化DFS过程（数据结构加速算法的又一实例）

Soj 1215脱离地牢

- 题意：
- 有两个人Paris和Helen在一个n*m的地牢里，里面有墙壁也有熔浆
- 当Pairs向NSWE方向前进一格，那么Helen会按给定的对应方向前进一格
- 如果Helen遇到墙壁则不能前进
- 如果Helen或者Pairs到熔浆，则任务失败
- 问使两个人相遇最少需要多少步
- 如果不可能则输出Impossible
- 限制：
- 3<=n,m<=20

```
5 5
#####
#H..#
#.!.#
#.#P#
#####
WNSE
```

Soj 1215脱离地牢

- 题意：
- 输入串为WNSE
- 那么
- P往N走，H往W走；
- P往S走，H往N走；
- P往W走，H往S走；
- P往E走，H往E走；

```
5 5
#####
#H..#
#.!.#
#.#P#
#####
WNSE
```

Soj 1215脱离地牢

- 分析：
- 其实，我们只需要仔细分析，这道题其实是一种修改过的最短路，而且每一步代价相同，只需要BFS就能解决
- 那么，这道题的关键就是设计状态
- 回想原始的迷宫问题，我们用(x,y)表示当前人的位置作为状态
- 这里我们只需要把状态改成(x1,y1,x2,y2)就可以了

Soj 1215脱离地牢

- 解法：
- BFS
- 时间复杂度O(n^4

```
const int maxn = 21;
const int d[][2] = {{-1, 0}, {1, 0}, {0, -1}, {0, 1}};
const string dir = "NSWE";
int n, m;
char dir2[5];
char s[maxn][maxn];
int f[maxn][maxn][maxn][maxn];

struct Node
{
    int x1, y1, x2, y2;
    Node(int x1, int y1, int x2, int y2):
        x1(x1), y1(y1), x2(x2), y2(y2) {}
};
queue<Node> q;
```

Soj 1215脱离地牢

- 解法：

```
int bfs(int x1, int y1, int x2, int y2)
{
    memset(f, -1, sizeof(f));
    q.push(Node(x1, y1, x2, y2));
    f[x1][y1][x2][y2] = 0;
    while (!q.empty())
    {
        Node node = q.front(); q.pop();
        x1 = node.x1, y1 = node.y1;
        x2 = node.x2, y2 = node.y2;
        for (int i = 0; i < 4; i++)
        {
            int j = dir.find(dir2[i]);
            int nx1 = x1 + d[i][0], ny1 = y1 + d[i][1];
            int nx2 = x2 + d[j][0], ny2 = y2 + d[j][1];
            if (s[nx1][ny1] != '!' && s[nx1][ny1] != '#' && s[nx2][ny2] != '!')
            {
                if (s[nx2][ny2] == '#')
                    nx2 = x2, ny2 = y2;
                if (make_pair(x1, y1) == make_pair(nx2, ny2) && make_pair(x2, y2) == make_pair(nx1, ny1)
                        || make_pair(nx1, ny1) == make_pair(nx2, ny2))
                    return f[x1][y1][x2][y2] + 1;
                if (f[nx1][ny1][nx2][ny2] < 0)
                {
                    f[nx1][ny1][nx2][ny2] = f[x1][y1][x2][y2] + 1;
                    q.push(Node(nx1, ny1, nx2, ny2));
                }
            }
        }
    }
    return -1;
}
```

Soj 1171 The Game of Efil

- 题意：
- 讲述了生命游戏
- 一块m*n大小的板上，有一些细菌
- 如果一个细菌的八个方向上邻居细菌数是2或3，则在下一个回合它能保留下来，否则它会消失
- 如果有一个空格的邻居细菌数为3，则下一回合长出新的细菌
- 板的上下边是连通的，左右边是连通的 (torus)
- 给出一个板的当前状态，问上一个回合的状态有多少不同的情况
- 限制：
- 1<=m*n<=16

Soj 1171 The Game of Efil

- 分析：
- 注意，题目的限制不是1<=n,m<=16，而是1<=n*m<=16
- 那么最多只有16个格子，那么我们只需要枚举上一个状态各个格子的状态，最多枚举2^16种状态
- 对于每个状态按照规则计算出下一个状态，判断是否相同即可
- 复杂度：O(16*2^16)

Soj 1171 The Game of Efil

- 代码：

```
int n, m;
int gameOfEfil()
{
    int k;
    scanf("%d", &k);
    int target = 0;
    while (k--)
    {
        int x, y;
        scanf("%d %d", &x, &y);
        target |= 1 << (x * m + y);
    }
```

Soj 1171 The Game of Efil

- 代码：

```
int ret = 0;
for (int mask = 0; mask < 1 << (n * m); mask++)
{
    int nmask = 0;
    for (int i = 0; i < n; i++) for (int j = 0; j < m; j++)
    {
        int nbr = 0;
        for (int di = -1; di <= 1; di++) for (int dj = -1; dj <= 1; dj++)
        {
            if (!di && !dj) continue;
            int ni = (i + di + n) % n, nj = (j + dj + m) % m;
            nbr += mask >> (ni * m + nj) & 1;
        }
        int f = 0;
        if (mask >> (i * m + j) & 1)
        {
            if (nbr == 2 || nbr == 3)
                nmask |= 1 << (i * m + j), f = 1;
        }
        else
        {
            if (nbr == 3)
                nmask |= 1 << (i * m + j), f = 1;
        }
    }
    if (nmask == target)
        ret++;
}
return ret;
```

Soj 1014 Specialized Four-Dig

- 题意：
- 求出所有在十进制，十二进制，十六进制下各位数字之和相等的四位十进制数。

Soj 1014 Specialized Four-Dig

- 解法：
- 直接做！枚举所有十进制的4位数判断他们，在10进制、12进制和16进制下的数位和是否相等

Soj 1014 Specialized Four-Dig

- 代码：

```
int sum(int n, int base)
{
    int ret = 0;
    while (n)
    {
        ret += n % base;
        n /= base;
    }
    return ret;
}

bool check(int n)
{
    return sum(n, 10) == sum(n, 12) && sum(n, 10) == sum(n, 16);
}
```

soj 1709 PropBot

- 题意：一开始机器人在点(0,0)上，指向+x方向。机器人每秒有两种行动方法：向当前方向走10个单位长度，或者向右转45度。现给出目标点以及最多可以行动的时间，求出在这个过程中，机器人可以达到的离目标点最近的距离，（可以自由选择机器人的行动方法）

soj 1709 PropBot

- 题解：由于时间最多为24，而每次机器人只有两个选择，因此可以枚举出机器人所有的选择，以及每次可以达到的点，然后更新当前位置与目标位置的最小值。(by poetry)

soj 1108 Online Selection

- 题意：节目组有0到n-1共n组问题，每组问题的答案相同，并且每个问题的答案只有0,1两种。一开始参与者的状态是0，拥有0分。在状态i的参与者被给出第i组中的一个问题，如果参与者答对问题，那么分数加1。另外如果参与者回答的是0，状态i就会转移到状态i0，回答的是1，则转移到状态i1。(不论答案的对错)
- 现在给出第i组的问题的答案，以及i0和i1,并且已知这个过程持续了k轮，选手得到了m分，需求出这个过程中，选手回答0次数最多的情况。

soj 1108 Online Selection

- 题解：由于在每个状态里，只有回答0和1两个方向，因此每个状态按照这两个方向去搜索，最后得到一个最大值。
- 直接搜索会超过时间限制，可以用dp[i][j][m],其中i表示当前的状态，j表示游戏已经进行的轮数，m表示当前获得的分数。由于整个游戏状态只与这三个量有关，因此可以做记忆化搜索。(by poetry)

soj 6276 Pawns

- 题意：有1到n的格子排成一列(n<=13)，每个格子要么是空的，要么上面放黑棋或白棋
- 白棋可以向左移一格，或者跳过左边距离为1的棋子，到达那个棋子左边一格的位置（需要那个位置为空），每步移动一枚棋子
- 黑棋与白棋的移动方法类似，只是黑棋的方向往右
- 最终的状态为：黑棋全部在右边，白棋全部在左边，问从开始状态到最终状态最少需要多少步，保证题目一定有解

soj 6276 Pawns

- 题解：将棋盘的状态用一个三进制的数来表示，并用bfs来记录每个状态所需要的最小步数，每移动一枚棋子就是一次转移，最后输出最终状态所对应的答案(by poetry)

Soj 1471 No Left Turns

- 题意：一个迷宫，从起点走到终点，走一步有直走，以及直走再右转两个选择（注意不能原地转弯），问在这种情况下的最短路径是多少，起始方向可以任意选择，题目中保证这样的路径一定存在
- 题解：使用bfs，用dis[x][y][d]表示走到x,y处，朝向为d的时候的最短路是多少。遍历四个起始方向，对所有位置进行bfs，最后取dis[endx][endy][d]中最小的那个作为答案。(by poetry)

Soj 1182 Context-Free Clock

- 题意：给出一个时刻t和角度theta，求时刻ans，使得在ans时刻中，时针按顺时针到分针的角度为theta，且该时刻离t最近。得到ans后将ans按照秒截断后输出（例如答案是12:34:45’54的话，输出12:34:35）
- 题解：由于总时刻不多，所以可以预先枚举每个时刻，然后得到他们的角度（不一定是整数），之后从当前时刻开始遍历，找到第一个这样一个时刻：它的角度不大于目标，且它的下一角度大于目标。注意精度问题
- 也可以通过对角速度的分析，直接得到答案。(by poetry)

Soj 1710 Painted Calculator

- 题意：计算器中，一个数的一个译码管亮起需要消耗5毫安，如果是负数的话，那个负号需要单独消耗5毫安。已知运用这个计算器进行了一个四则运算(a ? b = c形式，且如果所做的运算是除法则b不为0)，得到每个数的功耗，问一共有多少个这样的运算满足条件，每个数绝对值都小于1000.
- 题解：由于数字很小，因此可以遍历每个数字，以及每种可能的运算，然后做出一个结果，再检查是不是符合要求，最后统计答案即可。(by poetry)

Soj 7144 Different Triangles

- 题意：给出n根棍子的长度(n<=100)，求出用其中的三根棍子可以摆出的三角形的个数。
- 题解：由于n很小，因此直接枚举三根棍子，然后看它们是不是构成三角形。(by poetry)