welcome

分治, 搜索

2018/10/17

Tivility/魏牧远
heu2016201621
tivility@outlook.com

- Thanks:
  - 部分灵感参考自
    - aqx老师
    - dengsiyu姐姐
    - xjtuwmg_1001老哥

tivility@outlook.com

• 分治

- 天下大势, 合久必分, 分久必合;
- 凡治众如治寡, 分数是也;

tivility@outlook.com

• 分治

- 当我们求解某些问题时，由于这些问题要处理的数据相当多，或求解过程相当复杂，使得直接求解法在时间上相当长，或者根本无法直接求出。对于这类问题，我们往往先把它分解成几个子问题，找到求出这几个子问题的解法后，再找到合适的方法，把它们组合成求整个问题的解法。如果这些子问题还较大，难以解决，可以再把它们分成几个更小的子问题，以此类推，直至可以直接求出解为止。这就是分治策略的基本思想。

• 分治

- 递归的定义是:
  参见递归的定义.

- e.g.
- int fact(int x) { return (x == 0) ? 1 : x * fact(x - 1); }
- e.g.
- int fib(int x) { return (x <= 1) ? 1 : fib(x - 1) + fib(x - 2); }

tivility@outlook.com

- # 归并排序&快速排序

  - 归并排序：每次把待排序区间一分为二，将两个子区间排序，然后将两个已经排好序的序列合并
  - 快速排序：选择一个基准，将小于基准的放在基准左边，大于基准的放在基准右边，然后对基准左右都继续执行如上操作直到全部有序。

tivility@outlook.com

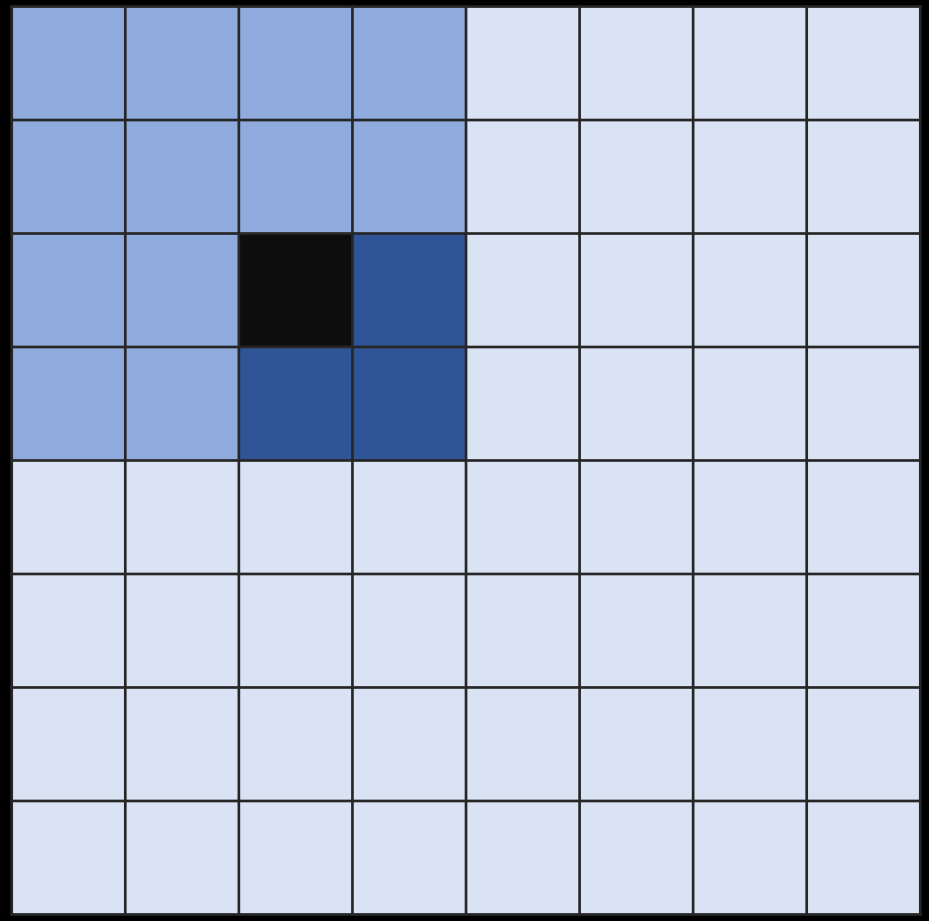
- 分治
  - 用L型骨牌覆盖2^n * 2^n的棋盘
  - 其中有一个特殊方格不能被覆盖
  - 输出任意一种覆盖方案
  - 1≤n≤10

tivility@outlook.com

• 分治

- 用L型骨牌覆盖2^n * 2^n的棋盘
- 其中有一个特殊方格不能被覆盖
- 输出任意一种覆盖方案

- 1≤n≤10



tivility@outlook.com

CF 448C PaintingFence

- 每块木板宽度均为1，高度为h[i]
- n块木板连接为宽度为n的栅栏
- 每次可以刷一横或一竖(上色)

- 最少刷多少次可以使得栅栏被全部上色

- 1≤n≤5000

tivility@outlook.com

• 搜索

- 通过不停的试探去寻找解的一种算法。
- 与其说是一种算法，不如说是一种方法。
- 基础的方法有暴力的搜索法，深搜，广搜三种。
- 更高级的有IDDFS，DBFS，A*，IDA*等等

tivility@outlook.com

- DFS

  - 用途
  - 判断图内两点是否联通

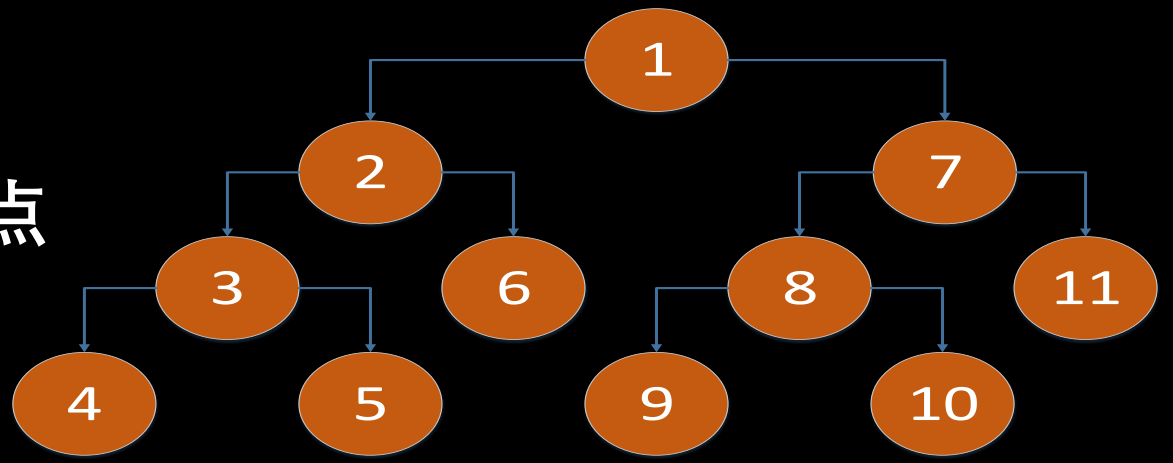
  - 方法
  - 沿着一条路尽可能的向深处搜索

  - “一条道走到黑”

tivility@outlook.com

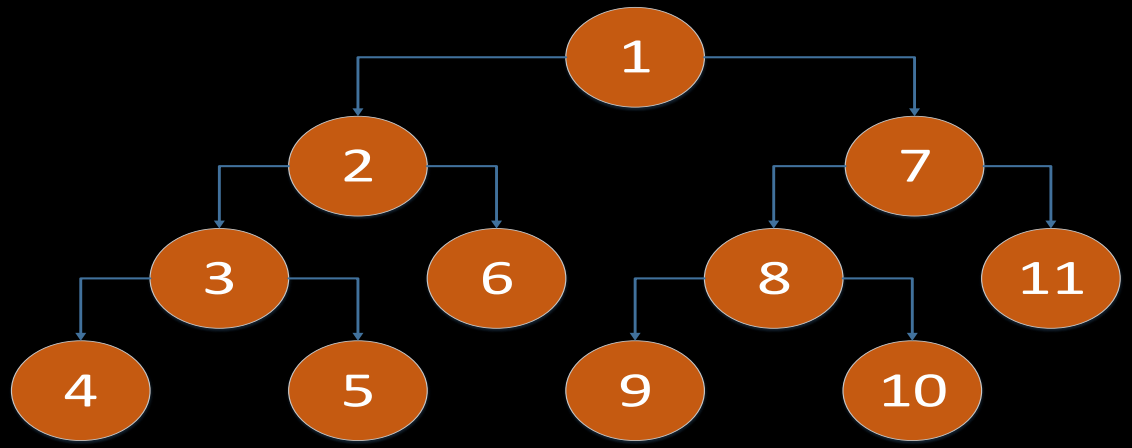
DFS

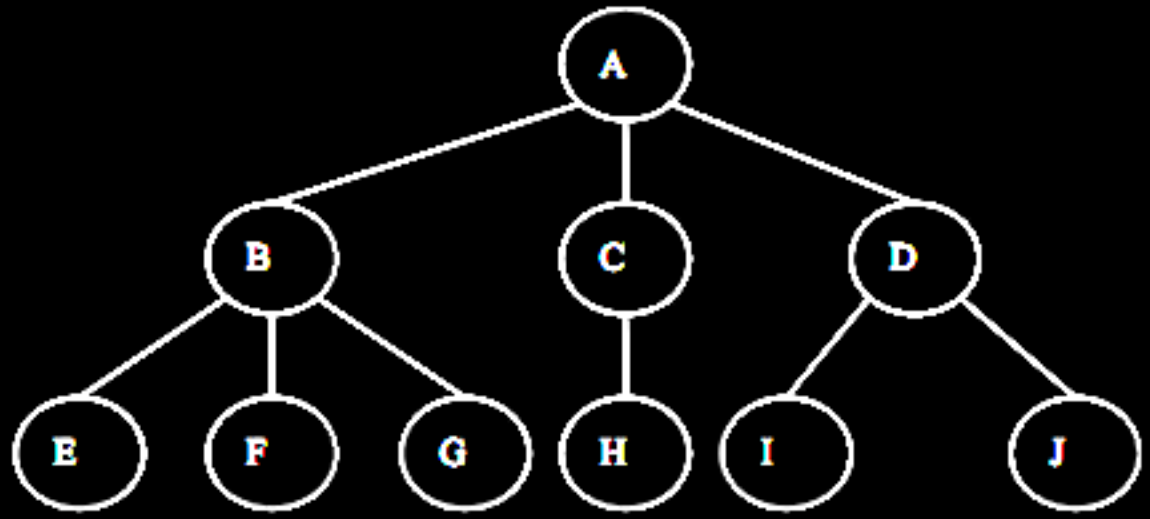
- 从当前点开始
- 如果存在下一个可以到达的点
  - 则搜索下一个点
- 如果不存在下一个点，
  - 则返回上一个点
  - 继续搜索上一个点的下一个点



tivility@outlook.com

- DFS





- ABEFGCHDIJ

tivility@outlook.com

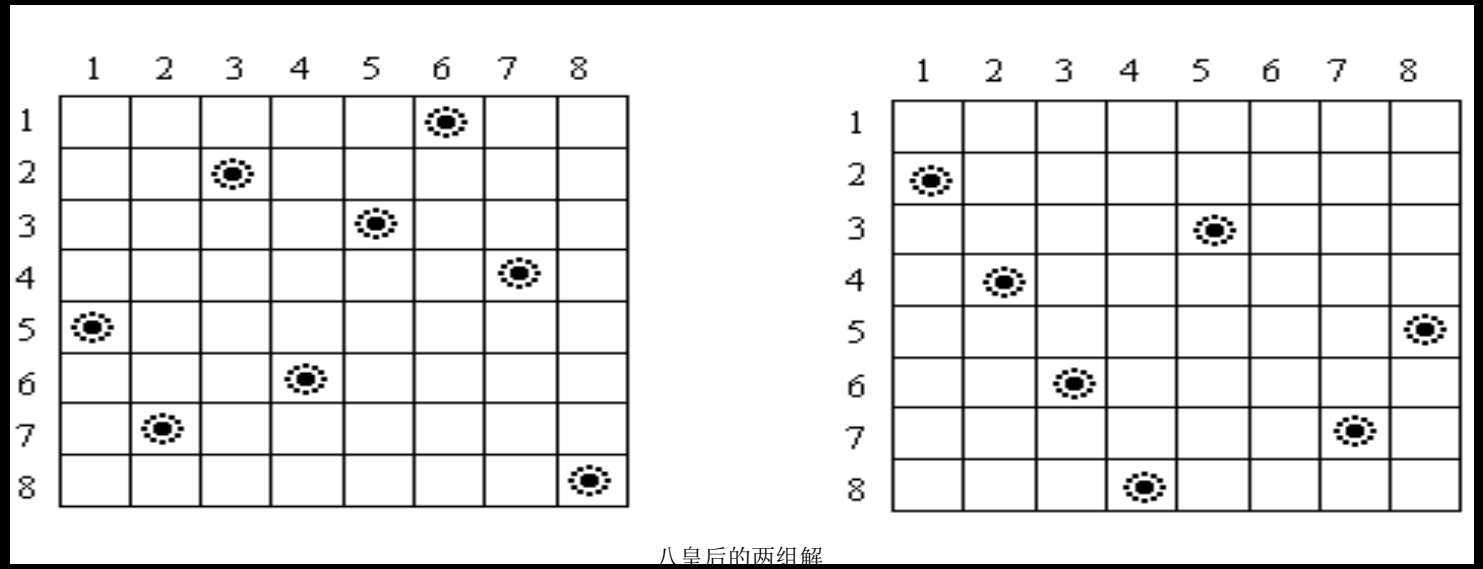
- DFS

```
1 //dfs
2 struct Point {
3     int x, y;
4 };
5 const int dir[4][2] = {1, 0, 0, 1,
6                         -1, 0, 0, -1};
7 bool used[MAX][MAX];
8 bool dfs(Point now, int deep) {
9     if (now.x == ans.x && now.y == ans.y)
10        return true;
11    if (deep > max_deep || used[now.x][now.y])
12        return false;
13    used[now.x][now.y] = true;
14    Ponit nxt;
15    for (int i = 0; i < 4; ++i) {
16        nxt.x = now.x + dir[i][0];
17        nxt.y = now.y + dir[i][1];
18        if (dfs(nxt, deep+1))
19            return true;
20    }
21    return false;
22 }
```

tivility@outlook.com

DFS

- 在N*N的棋盘上放置N个皇后而彼此不受攻击（即在棋盘的任一行，任一列和任一对角线上不能放置2个皇后），编程求解所有的摆放方法。



八皇后的两组解

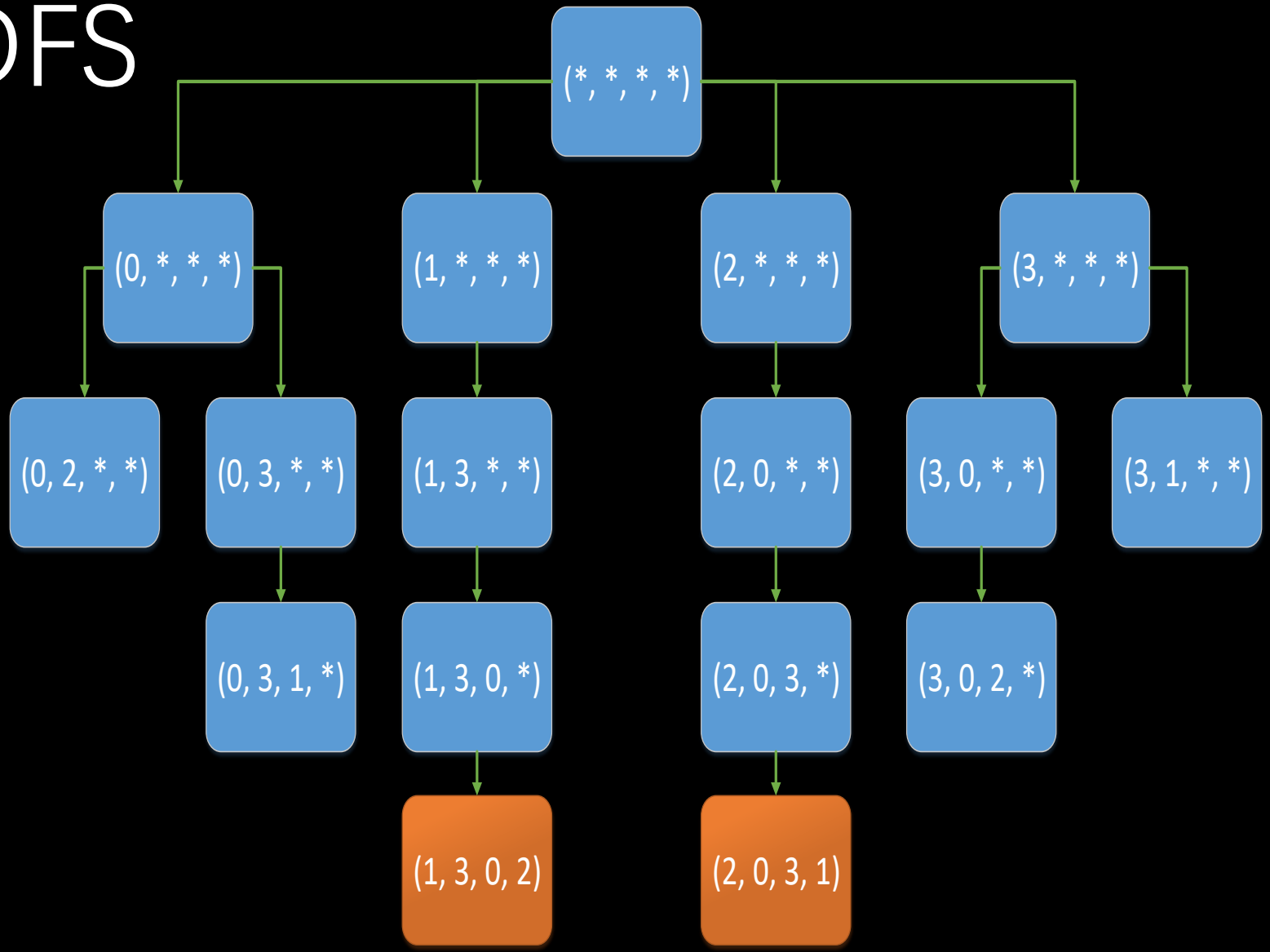
tivility@outlook.com

- DFS

  - 对于每个皇后的摆放位置都要进行试探和纠正
  - 在N个皇后未放置完成前，摆放第i个皇后和第i+1个皇后的试探方法相同
  - 从第0行开始摆放，一直摆到第n – 1行为止。
  - 以4皇后为例:

tivility@outlook.com

- DFS



tivility@outlook.com

- DFS
  - 判断行冲突?

tivility@outlook.com

- DFS
  - 判断行冲突?
  - 判断列冲突?

tivility@outlook.com

- DFS
  - 判断行冲突?
  - 判断列冲突?
  - 主对角线冲突?

tivility@outlook.com

- DFS
  - 判断行冲突?
  - 判断列冲突?
  - 主对角线冲突?
    - x-y

tivility@outlook.com

- DFS
  - 判断行冲突?
  - 判断列冲突?
  - 主对角线冲突?
    - x-y
  - 副对角线冲突?

tivility@outlook.com

- DFS
  - 判断行冲突?
  - 判断列冲突?
  - 主对角线冲突?
    - x-y
  - 副对角线冲突?
    - x+y

tivility@outlook.com

- DFS
  - 斐波那契数列?
  - n个数字的全排列?

tivility@outlook.com

- DFS
  - 状态?
  - 转移?

tivility@outlook.com

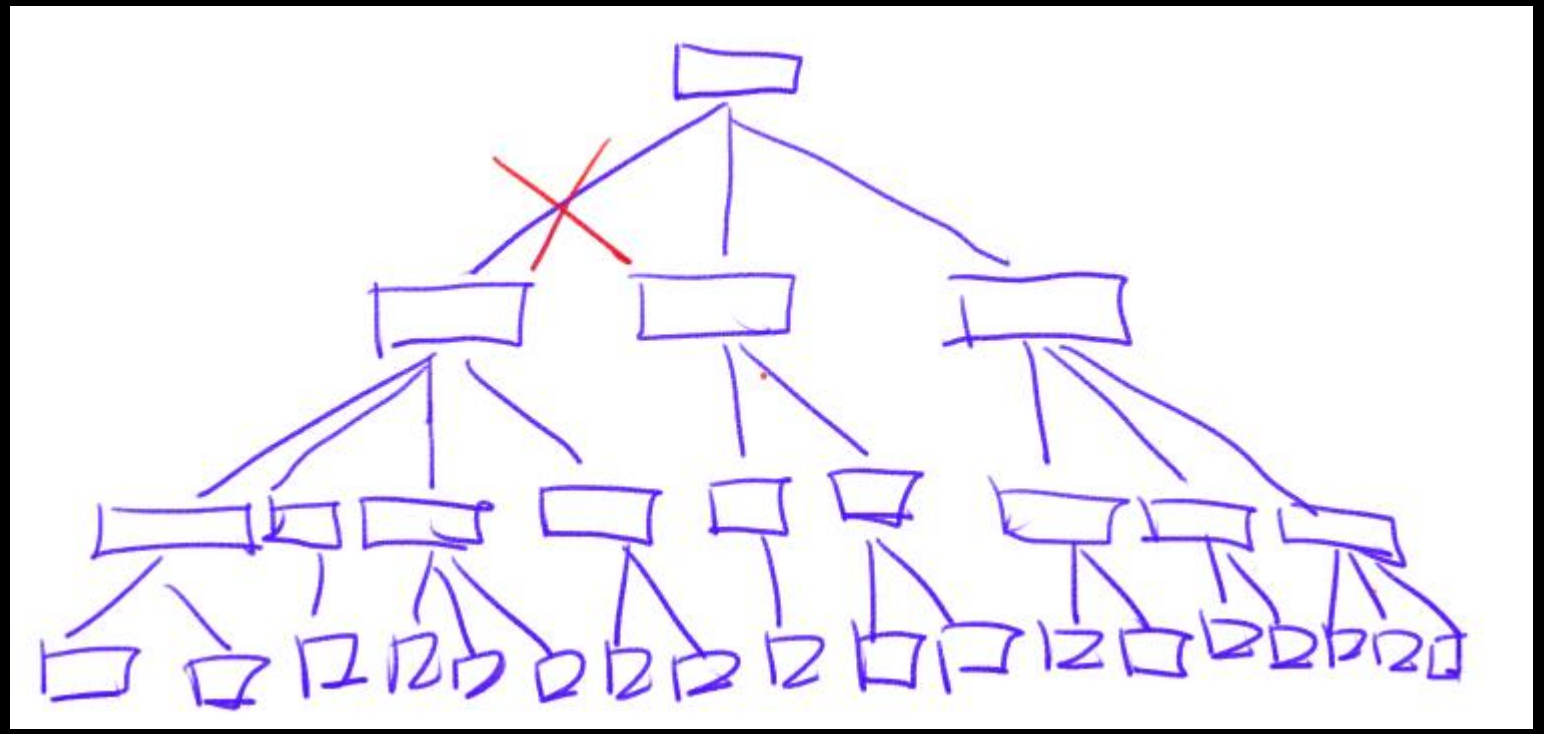
- DFS
  - 状态?
  - 转移?
  - 搜索是在解空间里寻找目标状态

tivility@outlook.com

- DFS
  - 状态?
  - 转移?
  - 搜索是在解空间里寻找目标状态
  - 状态集合太大?

tivility@outlook.com

- 剪枝



tivility@outlook.com

- 剪枝
  - 最优化剪枝
  - 可行性剪枝

tivility@outlook.com

剪枝

- N*M的迷宫中给定你起点S，和终点D，问你是否能在 T 时刻恰好到达终点D。

- S.X.
- ..X.
- ..XD

tivility@outlook.com

- 剪枝
  - deep > T

tivility@outlook.com

- 剪枝
  - deep > T
  - 曼哈顿距离 + deep > T

tivility@outlook.com

- 剪枝
  - deep > T
  - 曼哈顿距离 + deep > T
  - 奇偶可行性剪枝
    - if ((dis + d – t) & 1) return;

tivility@outlook.com

- POJ 1011
  - 乔治拿来一组等长的木棒，将它们随机地砍断，使得每一节木棍的长度都不超过50个长度单位。然后他又想把这些木棍恢复到为裁截前的状态，但忘记了初始时有多少木棒以及木棒的初始长度。请你设计一个程序，帮助乔治计算木棒的可能最小长度。每一节木棍的长度都用大于零的整数表示。
  - 小木棍的数量<=64

tivility@outlook.com

- POJ 1011
  - 朴素枚举:
  - 枚举最终小木棍的长度L, 搜索能不能拼出K根
  - (L * K = SUM)

tivility@outlook.com

- POJ 1011
  - sum = k * L

tivility@outlook.com

- POJ 1011
  - sum = k * L
  - 第i个棍子不能拼成假设的长度，则和第i个棍子相同长度的棍子也是不可能拼成.

tivility@outlook.com

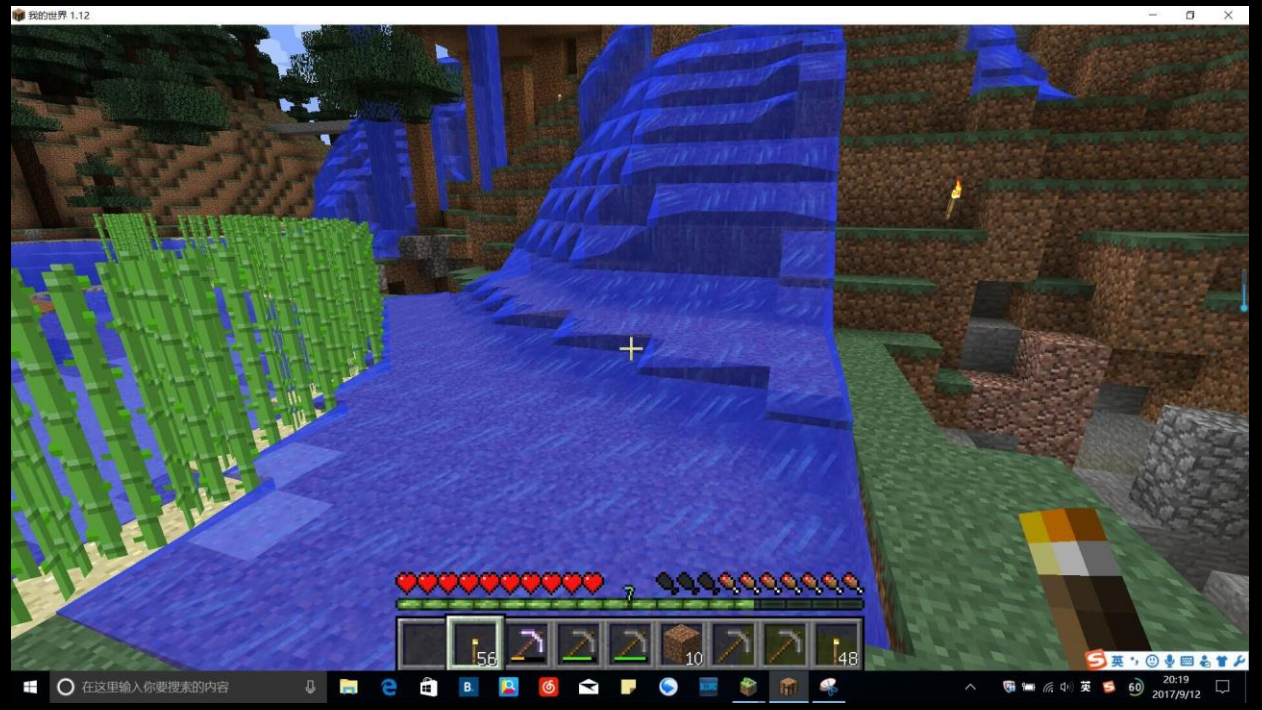
- POJ 1011
  - sum = k * L
  - 第i个棍子不能拼成假设的长度，则和第i个棍子相同长度的棍子也是不可能拼成.
  - 替换第i根棍子的第一根木棒是没用的

tivility@outlook.com

- POJ 1011
  - sum = k * L
  - 第i个棍子不能拼成假设的长度，则和第i个棍子相同长度的棍子也是不可能拼成.
  - 替换第i根棍子的第一根木棒是没用的
  - 如果某次拼接选择长度为S 的木棒，导致最终失败，则在同一位置尝试下一根木棒时，要跳过所有长度为S 的木棒。

tivility@outlook.com

- BFS
  - 流觞曲水



tivility@outlook.com

- BFS

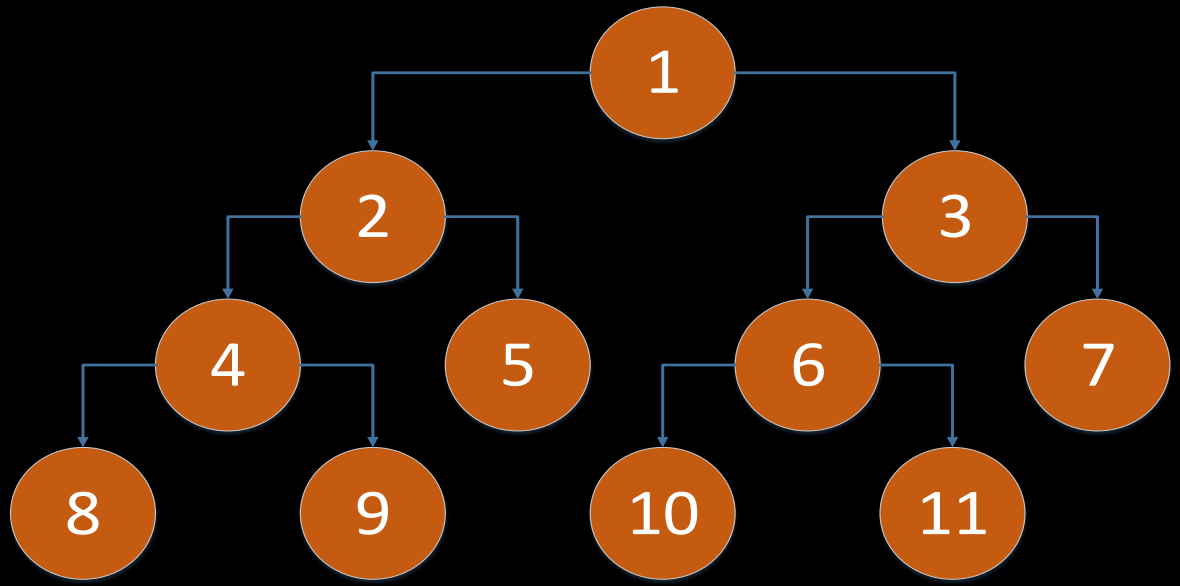
7	6	5	4	5	6	7
6	5	4	3	4	5	6
5	4	3	2	3	4	5
4	3	2	1	2	3	4
5	4	3	2	3	4	5
6	5	4	3	4	5	6
7	6	5	4	5	6	7

tivility@outlook.com

- BFS
  - 一层一层的走！
  - 离上一状态最近的状态用一个队列记录下来；
  - 如果队列不为空，取队首元素代表状态，并且以这个状态为“根节点”进行广度优先搜索。
  - 直到整个队列为空为止。

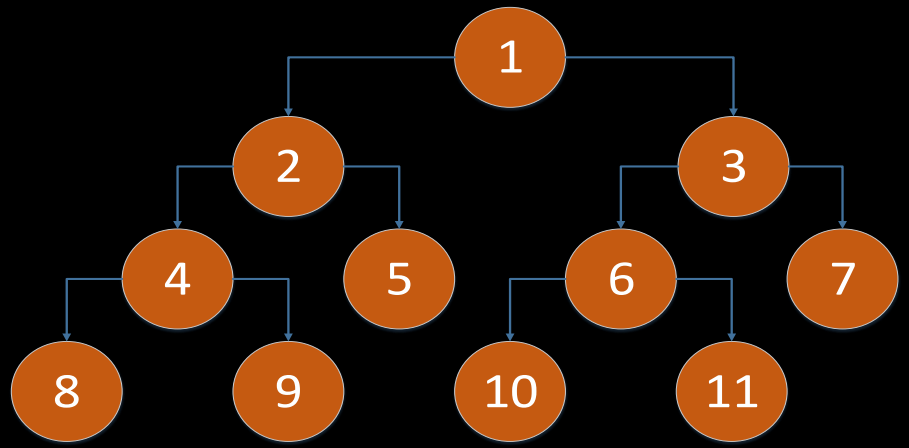
tivility@outlook.com

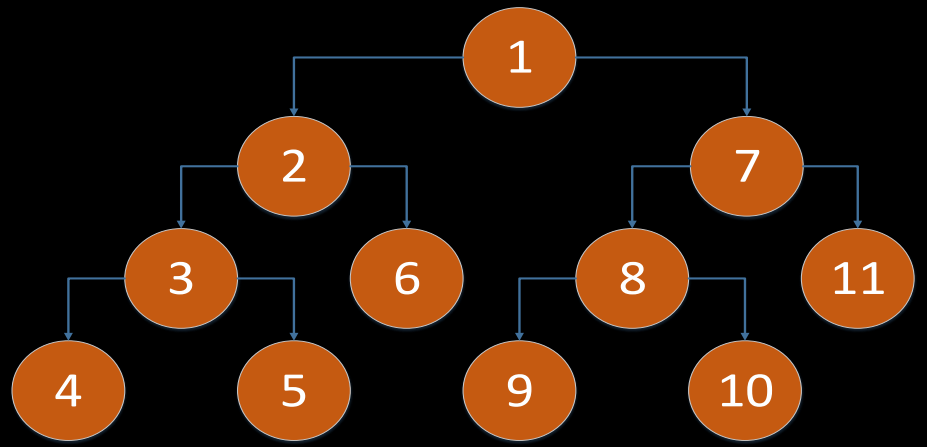
- BFS



tivility@outlook.com

- BFS





tivility@outlook.com

- BFS

```
// bfs

struct Point {
    int x, int y;
    int deep;
}p, p_tmp, p_start, p_now, p_end;
const int dir[4][2] = {1, 0, 0, 1, -1, 0, 0, -1};
queue <Point> que;
int bfs() {
    while (que.empty() == false) //init
        que.pop();
    que.push(st)
    while (que.empty() == false) {
        p_now = que.front();
        que.pop();
        if (p_now.x == p_end.x && p_now.y == p_end.y)
            return now.deep;
        p_tmp.deep = p_now.deep+1;
        for (int i = 0; i < n; ++i) {
            p_tmp.x = p_now.x + dir[i][0];
            p_tmp.y = p_now.y + dir[i][1];
            if (mp[p_tmp.x][p_tmp.y] == true &&
                    used[p_tmp.x][p_tmp.y] == false) {
                used[p_tmp.x][p_tmp.y] == true;
                que.push(p_tmp);
            }
        }
    }
    return -1;
}
```

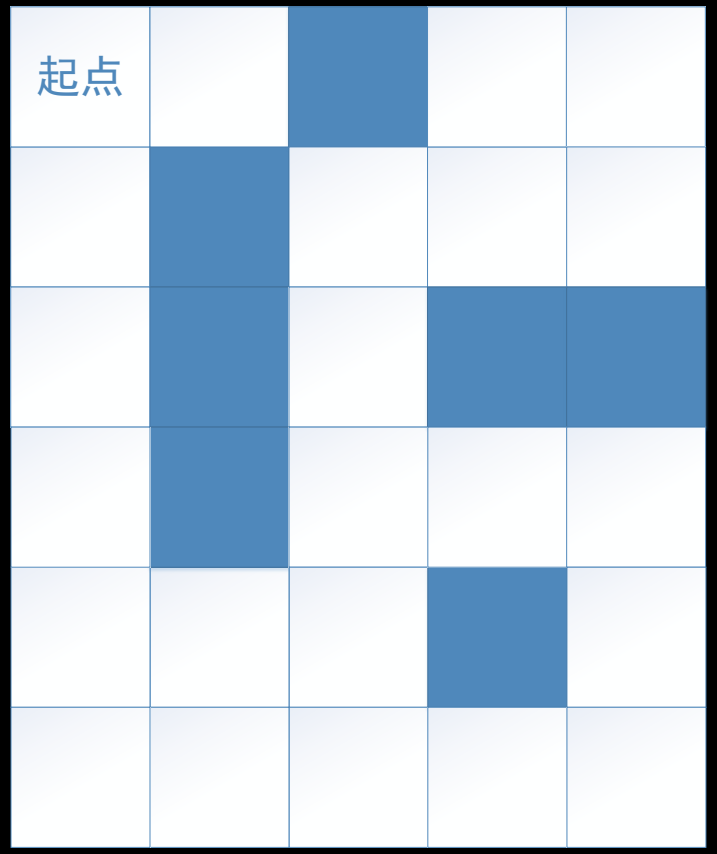
tivility@outlook.com

- 迷宫
  - n*m迷宫
  - 四联通
  - 起点恒为 (0, 0)
  - 输入终点坐标，求最短的路径

tivility@outlook.com

- 迷宫

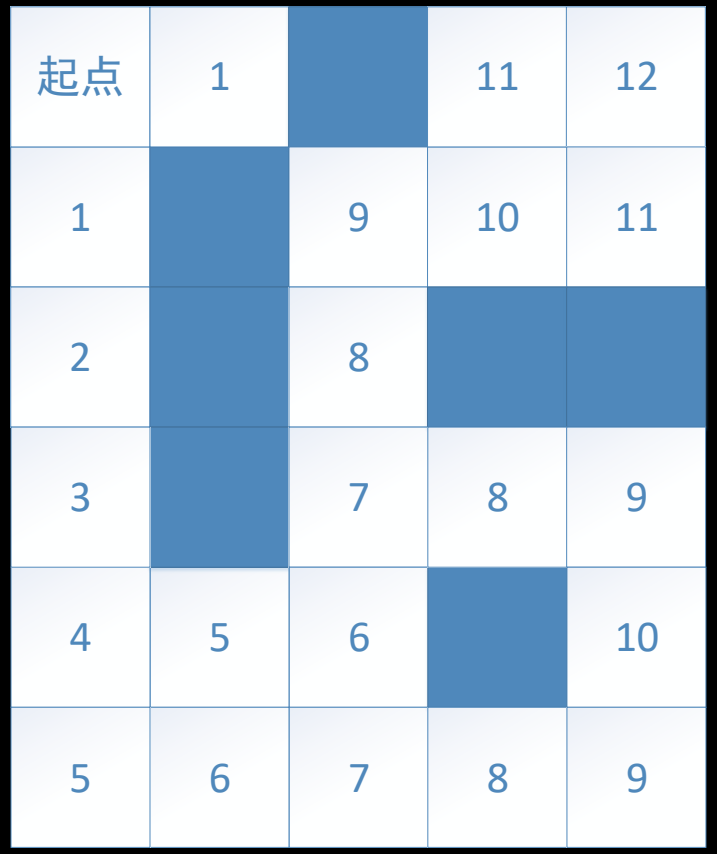
  - 6 5
  - 1 1 0 1 1
  - 1 0 1 1 1
  - 1 0 1 0 0
  - 1 0 1 1 1
  - 1 1 1 0 1
  - 1 1 1 1 1



tivility@outlook.com

- 迷宫

  - 6 5
  - 1 1 0 1 1
  - 1 0 1 1 1
  - 1 0 1 0 0
  - 1 0 1 1 1
  - 1 1 1 0 1
  - 1 1 1 1 1



tivility@outlook.com

• 八数码

• 1 2 3	• 1 2 3	• 1 2 3
• 4 5 6	• 4 5 6	• 4 X 6
• 7 8 X	• 7 X 8	• 7 5 8

tivility@outlook.com

- 康托展开
  - X=a[n]*(n-1)!+a[n-1]*(n-2)!+...+a[i]*(i-1)!+...+a[1]*0!
  - 其中a[i]为当前未出现的元素中的rank（从0开始）

tivility@outlook.com

- 康托展开

```
int Cantor(int *s,int n) { //n表示该排列有n个数
    int sum = 0;
    for(int i = 0; i < n; i++) {
        int temp = 0;
        for(int j = i + 1; j < n; j++)
          if(s[j] < s[i])
              temp++;
        sum += fac[n-i-1] * temp;//f[n]表示n的阶乘
    }
    return sum;
}
```

```
void _Cantor(int *s, int sum,int n) {
    bool v[5]={0};
    int t;
    for(int i = 0;i < n; i++) {
        t = sum / fac[n-i-1];//f[n]表示n的阶乘
        for(int j = 0; j < n; j++) {
            if(!v[j]) {
                if(t == 0) break;
                t--;
            }
        }
        s[i] = j, v[j] = 1;
        sum %= f[n - 1 - i];
    }
    return ;
}
```

tivility@outlook.com

- 优化
  - 双向bfs
  - 优先队列bfs
  - 记忆化搜索

tivility@outlook.com

- 练习
  - CCF:
    - 第7,8,9次认证的第一二题
    - 第2,3次认证的第三题

tivility@outlook.com