**搜索**

**dfs**（深度优先搜索）：

深度优先搜索算法（Depth First Search，简称DFS）：一种用于遍历或搜索树或图的算法。 沿着树的深度遍历树的节点，尽可能深的搜索树的分支。当节点v的所在边都己被探寻过或者在搜寻时结点不满足条件，搜索将回溯到发现节点v的那条边的起始节点。

当然在**序列方面**dfs擅长取波动序列

如a[i] i=0~9表示i可以使用的最大次数

使用dfs可以方便的模拟出a[i]（即temp[i]<=a[i] , 用depth做i , 用for做0~a[i]）的所有可能取值情况

通过回溯可使过程返回之前的状态

恢复递归前状态：优先搜索过程中的变量,如果在递归调用前改变了它的值, 而在下一次递归调用时又要用到它原来的值,那么就需要在每一次调用返回时,恢复成它的原值.

**dfs可以模拟排列和组合**

全排列

int a[maxn]={0};

int vis[maxn]={0};

int n;

void dfs(int x){

if(x==n+1){

for(int i=1;i<=n;i++){

printf("%5d",a[i]);

}

puts("");

return;

}

**for**(int i=1;i<=n;i++){

if(!vis[i]){

**a[x]=i;vis[i]=1;**

dfs(x+1);

**vis[i]=0;a[x]=0;**

}

}

}

**dfs的常见优化**

记忆化搜索 a[i][j]

剪枝：当搜索结果已经不是最优结果时可以直接跳出

**BFS**（广度优先搜索，也可称宽度优先搜索）是连通图的一种遍历策略。因为它的基本思想是从一个顶点V0开始，辐射状地优先遍历其周围较广的区域。

广度优先搜索（BFS）类似于二叉树的层序遍历算法，它的基本思想是：首先访问起始顶点v，然后由v出发，依次访问v的各个未被访问过的邻接顶点w1,w2,w3….wn，然后再依次访问w1，w2,…,wi的所有未被访问过的邻接顶点，再从这些访问过的顶点出发，再访问它们所有未被访问过的邻接顶点….以此类推，直到途中所有的顶点都被访问过为止。类似的想法还将应用与Dijkstra

与图论联系较广

常见优化deque priority\_queue（迪杰斯卡）