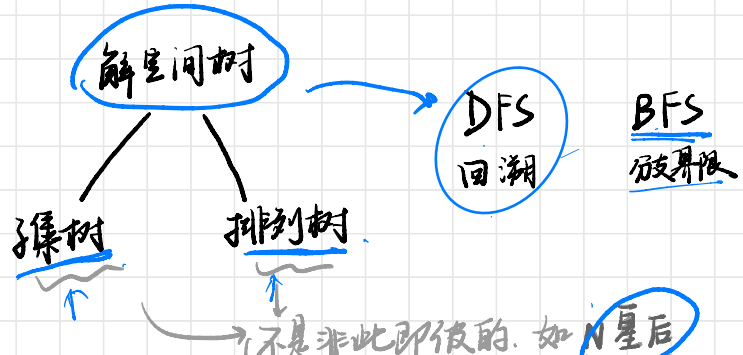


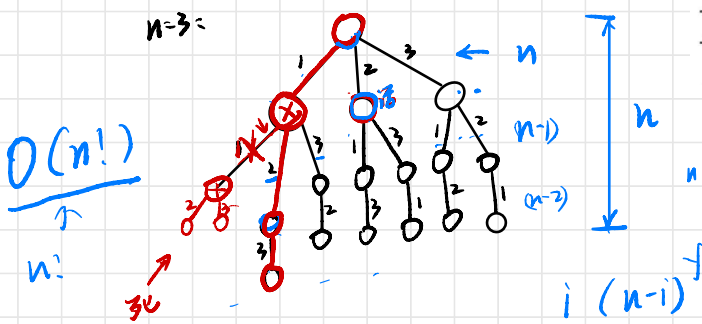
$\frac{n \cdot n!}{n!}$

根到节点的一条路径：解空间的一个元素。
子树：子问题的解空间。
活结点：该结点包含问题的解。？
死活点

$O(n \cdot n!)$

例

排列：解为 1~n 的排列：



$O(n!)$

解空间大小

$T(n) = nT(n-1) + O(1)$

用回溯法搜索排列树的算法框架可描述如下：

```
void Backtrack(int t) {
  if (t > n)
    Output(x);
  else {
    for (int i=t; i <= n; i++) {
      Swap(x[t], x[i]);
      if (Constraint(t) && Bound(t))
        Backtrack(t+1);
      Swap(x[t], x[i]);
    }
  }
}
```

前 t

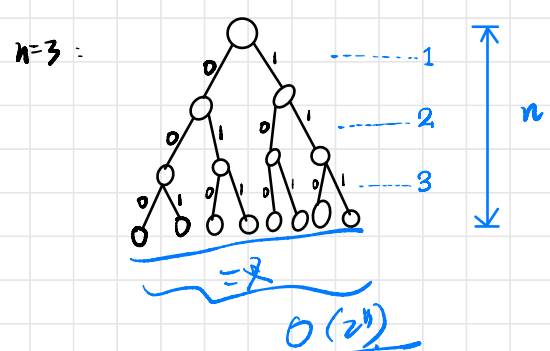
n for

$O(n!)$

$O(n \cdot n!)$

$O(n!)$

组合：解为长 n 的 0-1 串



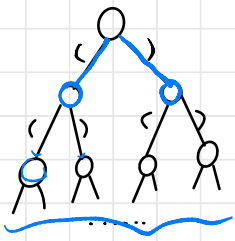
$O(2^n)$

$O(2^n)$

用回溯法搜索子集树的一般算法可描述如下：

```
void Backtrack(int t) {
  if (t>n)
    Output(x);
  else {
    for (int i=0; i <= 1; i++) {
      x[t]=i;
      if (Constraint(t) && Bound(t))
        Backtrack(t+1);
    }
  }
}
```

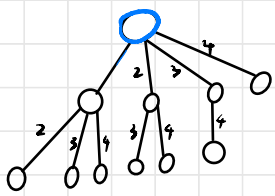
22. 括号生成：



n 对括号

共 n

131. 分割回文串：



$O(n!)$

n

$n \times (n-1)$

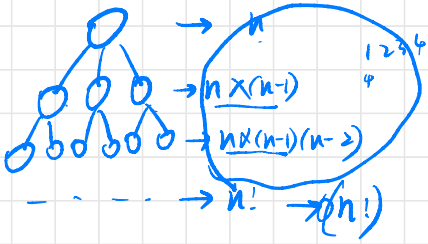
$n \times (n-1)(n-2)$

万枝没有通法

~~时间复杂度~~：

解空间大小：

n为深度



时间复杂度：

排列树：$O(n!)$

子集树：$O(2^n)$

其它：$O(m^n \cdot f(n))$

第 i 层活结点有 n-i 个扩展结点

活结点有2个扩展结点

m叉树

新结点的计算

m^n

2^n

更复杂：自己推