# 程序填空题

**全排列**

void Perm(int\* br, int k, int m)

{

if (k == m)

{

输出br数组

}

else

{

for (int j = k; j <= m; ++j)

{

Swap(br[j], br[k]);

Perm(br, k + 1, m);

Swap(br[j], br[k]);

}

}

}

**汉诺塔**

void HanoiTower(char A, char B, char C, int n)

{

if (n == 1)

{

move(A, C, n);

}

else

{

//将n-1个圆盘从A柱借助于C柱移动到B柱上

HanoiTower(A, C, B, n - 1);

//将A柱子最后一个圆盘移动到C柱上

move(A, C, n);

//将n-1个圆盘从B柱借助于A柱移动到C柱上

HanoiTower(B, A, C, n - 1);

}

}

**数塔**

getmax(1,1)

int getmax(int x,int y){

if(x==n){

return a[x][y];

}

return max(getmax(x+1,y),getmax(x+1,y+1))+a[x][y];

}

**快速排序**

# 简答题

1. **什么是复杂度？复杂度的度量指标？复杂度从低到高排列？**

复杂度是算法运行所需要的资源量的度量，通常用时间复杂度和空间复杂度来衡量。

时间复杂度是指算法运行所需要的时间资源量，

空间复杂度是指算法运行所需要的空间资源量。

复杂度从低到高排列为：O(1) < O(logn) < O(n) < O(nlogn) < O(n^2) < O(n^3) < O(2^n) < O(n!) < O(n^n)

1. **什么是蛮力法？主要缺点是什么？**

蛮力法是一种简单、直接地解决问题的方法,通常直接基于问题的描述和所涉及的概念定义。

主要缺点是设计的大多数算法的效率都不高，主要适合问题规模比较小的问题的求解。

1. **贪心算法的求解性质？**

贪心选择:指所求问题的整体最优解可以通过一系列局部最优的选择来达到。

最优子结构: 一个问题的最优解包含其子问题的最优解

1. **动态规划的求解基本要素？**

最优子结构：问题的最优解所包含的子问题的解也是最优的

重叠子问题：子问题之间是不独立的，一个子问题在下一阶段决策中可能被多次使用到。

无后效性：某阶段的状态一旦确定，就不受这个状态以后决策的影响。

1. **什么是背包问题？什么是01背包？区别是什么？**

背包问题：给定一组物品，每种物品都有自己的重量和价格，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。

01背包：给定一组物品，每种物品都有自己的重量和价格，每种物品要么选中要么不选中，在限定的总重量内，我们如何选择，才能使得物品的总价格最高。

区别：在选择物品装入背包时可以选择一部分,而不一定全部装入背包。这两类问题都具有最优子结构性质,但背包问题可以用贪心法求解,而0/1背包问题却不能用贪心法求解,因为用贪心法求解0/1背包问题可能得不到最优解。

1. **回溯法解题的四个主要步骤？**

(1）针对所给问题，定义问题的解空间;

(2）确定易于搜索的解空间结构;

(3）以深度优先的方式搜索解空间;

(4）回溯法搜遍整个解空间，最后得到最优解。

1. **回溯法分枝限界的异同？作出回溯法的表格**

**相同**：

都是在问题的状态空间树上搜索问题解的算法，都通过活结点表实现。都用约束函数剪去不含答案结点的分枝，用限界函数剪去不含最优解的分枝。

**不同**：

**1.求解目标不同**：回溯法的求解目标是找出解空间树中满足约束条件的所有可行解；而分枝限界法的求解目标是找出满足约束条件的一个可行解，或某种意义下的最优解。

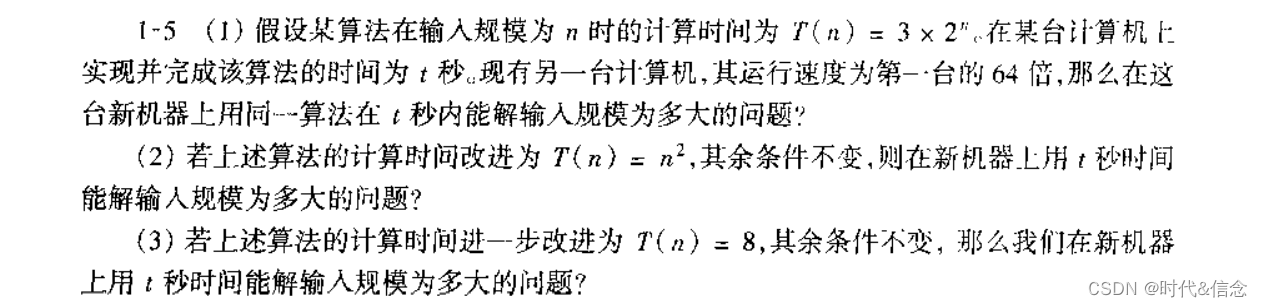
**2.搜索方式不同**：回溯法以深度优先的方式搜索解空间树，而分枝限界法以广度优先的方法搜索解空间树。

**3.对当前扩展节点的扩展方式不同**：回溯法中的每个活结点可能多次成为当前扩展节点；而分枝限界法中每个活结点只有一次机会成为扩展结点。

**4、存储结点的数据结构不同：**回溯法用栈存储，分枝限界法用队列或优先队列存储。

**5、应用不同：**回溯法用于找出满足条件的所有解，分枝限界法用于找出满足条件的一个解或者特定意义的最优解。

# 算法分析题

1. 

这里的关键在于T（n）的理解，我是这么理解的。题目中说：某算法在输入规模为n时的计算时间为T（n）=3 x 2^n ， 这个T（n）是时间关于问题规模n的函数，计算时间会随着问题规模n的增大而按这个式子增大。也就是说，这个T（n）其实代表的是这个算法的性能，是当前算法的固有属性。

（1）

我们通过时间建立等价关系。

因为规定的时间都是t,设新机器能在t时间内解问题规模为n1大的问题。

64 \* 3 x 2^n = 3 x2 ^ n1

所以n1 = n + 6

（2）

这里改变了T（n），其实相当于改变了算法的固有属性，还是一样的，根据时间t建立等价关系。

设在新机器上用t秒时间能解输入规模为n2大的问题。

64 \* (n ^ 2) = n2 ^ 2

n2 = 8n

（3）当算法改进为T（n）= 8

设新机器上用t秒时间能解输入规模为n3大的问题。

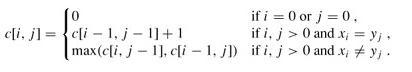
因为T(n) = 8，说明了这是一个常数级的算法，所以n3可以为任意规模。

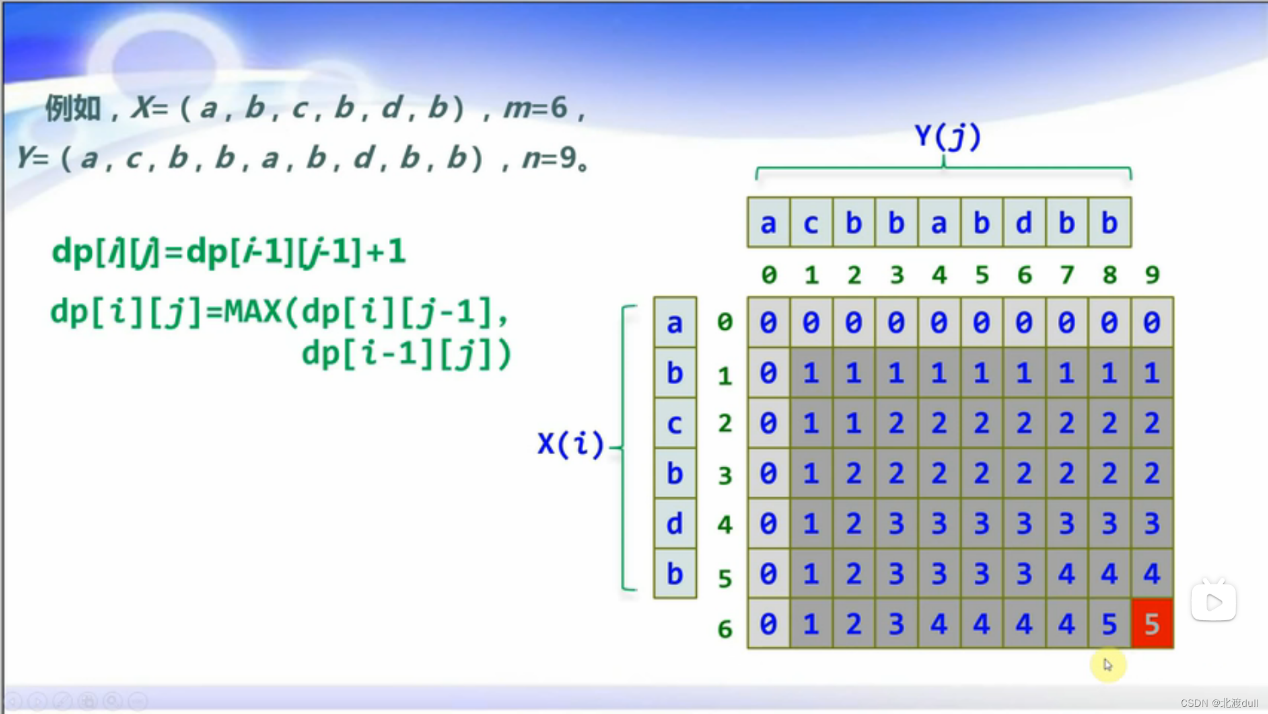
1. **排序、快排、归并排序的概念和思想、无序数排序过程、二分的写法**

快速排序的基本思想：在待排序的n个元素中任取一个元素(通常取第一个元素)作为基准,把该元素放入最终位置后,整个数据序列被基准分割成两个子序列,所有小于基准的元素放置在前子序列中,所有大于基准的元素放置在后子序列中,并把基准排在这两个子序列的中间,这个过程称为划分。然后对两个子序列分别重复上述过程,直到每个子序列内只有一个元素或空为止。

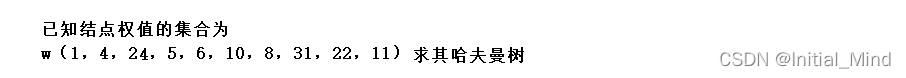
归并排序的基本思想：首先将a[0..n-1]看成n个长度为1的有序表,将相邻的k(h≥2)个有序子表成对归并,得到n/k个长度为k的有序子表;然后再将这些有序子表继续归并,得到n/k2个长度为k的有序子表,如此反复进行下去,最后得到一个长度为n的有序表。由于整个排序结果放在一个数组中,所以不需要特别地进行合并操作。

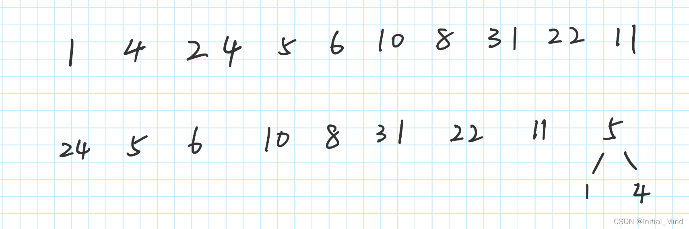
1. **动态规划：最长公共子序列，状态方程，打表结果**

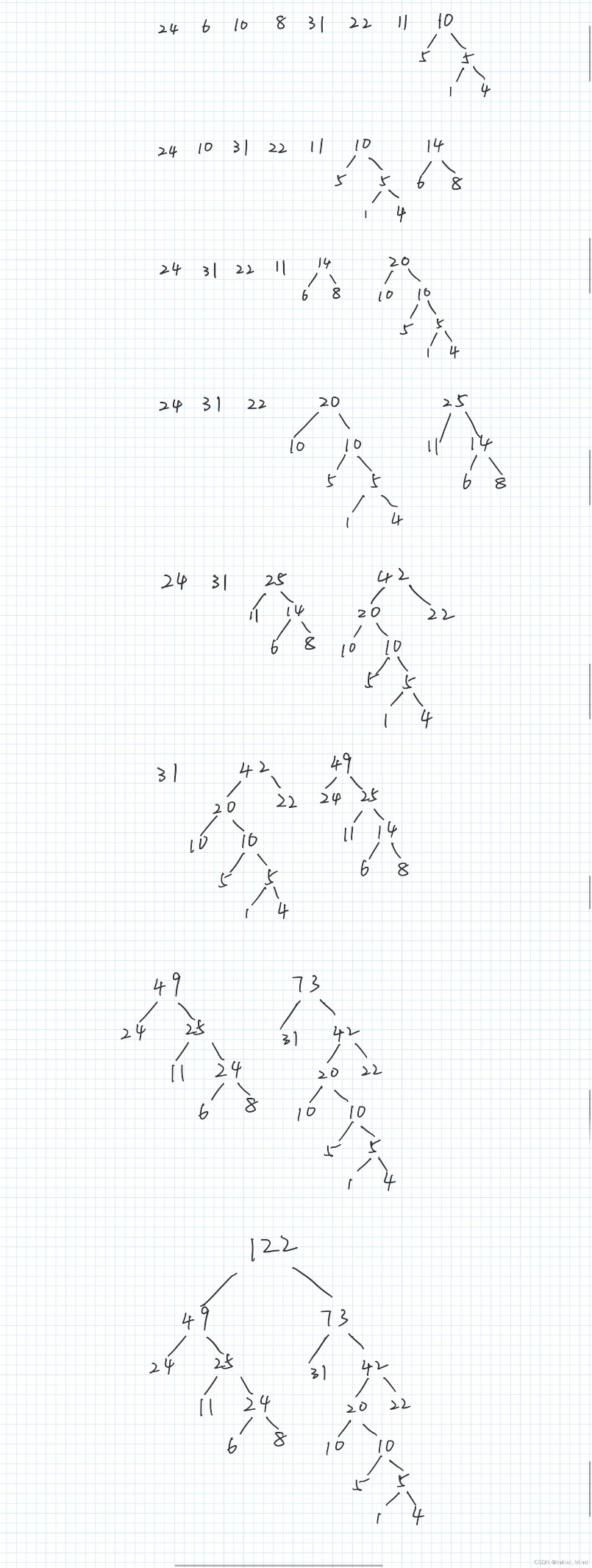


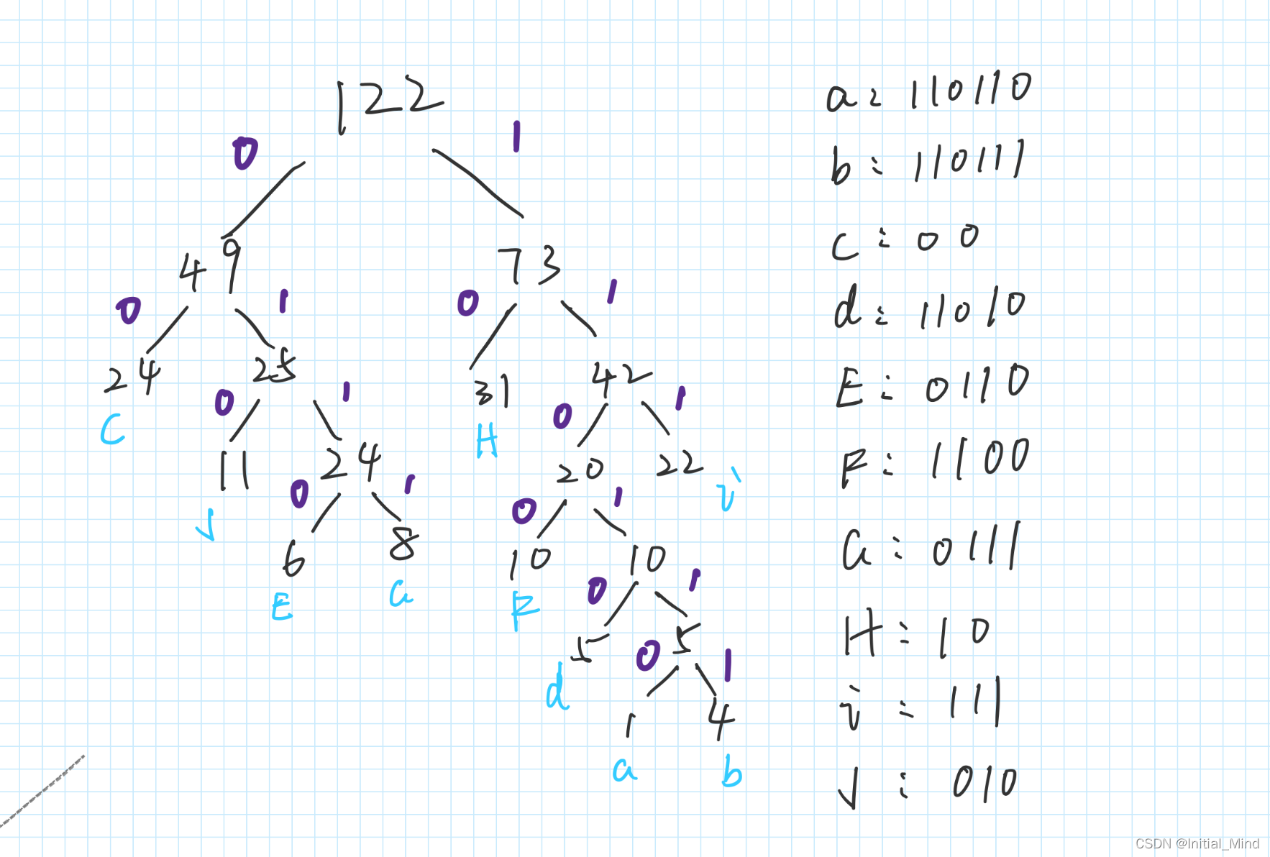


1. **哈夫曼树：画哈夫曼树、写带权路径长度之和、根据哈夫曼树写编码**









带权路径和：叶子结点\*所在层数之和

1. **回溯法：01背包，写出解空间树、写出剪枝树、写出最优解（P178）**
2. **分枝限界法：单源点求最短路径、写出求解过程、使用优先队列**
3. **旅行商问题：写出求解过程、画出状态空间树**