# 选择知识点

## 回溯法

把问题的解空间转化成了图或者树的结构表示，然后使用**深度优先搜索**策略进行遍历，遍历的过程中记录和寻找所有可行解或者最优解。

## 多项式方案

多项式级时间： n出现在底数的位置

非多项式级时间： 计算复杂度太高

一个多项式近似方案(PAS)是一个近似方案,其中每一个算法在输入实例I的规模的多项式时间内运行;

一个完全多项式近似方案(FPAS)是一个近似方案,其中每个算法在以输入实例的规模I和两者的多项式时间内运行.

若P2多项式时间转化为(polynomial transforms to) P1,则P2至少不会比P1难。P2比P1简单，所以才能用P1的表示。

多项式算法：时间复杂度是输入**数据量**的多项式表达

伪多项式算法：时间复杂度是输入**数值**的多项式表达

## NP

P问题：可以找到一个能在多项式的时间里解决它的算法。

NP问题：可以在多项式的时间里验证一个解是否正确的问题。(并不是求出解，而是猜测一个解，验证其是否正确)

NPC(完全)问题：

属于一个NP问题。

所有NP问题都可以约化成该问题。

## 算法正确性

算法正确性是对任意一个合法的输入经过有限步执行之后算法应给出正确的结果。

## 常见算法介绍

Las Vegas(拉斯维加斯)：

一种随机搜索解的算法，不一定能找到解，但找到的一定是解。

Monte Carlo 算法

一种根据概率统计设计的随机算法，是一种近似算法。一定有解，但解不一定正确。

贪心算法：

选择都是可行的、局部最优的，但不一定是全局最优。

## 近似算法

求解问题：f

近似算法解：c

最优解：

性能比： ，大于1，越接近1越准确

## 各种排序的时间复杂度

：不大于，最坏情况

：不小于，最好情况

快排： 随机快排能够避免最坏情况的出现

# 简答知识点

## 减治策略

利用了一个问题给定实例的解 和 同样问题较小实例的解 之间的关系。

1. **减一个常量**：常常是规模减1，如：插入排序
2. **减一个常因子：**每次迭代减去一个常量的倍数(小于1)，如：折半查找。
3. **减可变规模**：每次都减小，但减小的程度不是确定的，如：二叉查找树.

## 查找树

### 二叉查找树

最优情况：满二叉树，高度相同，

最差情况：单二叉树，每层只有一个节点，

插入数据有序排列时，二叉树变为一条长链。

为了防止出现最差情况，可以选择AVL树和2-3树进行查找和插入

### AVL树

AVL树是自平衡二叉查找树，任一节点对应的两棵子树的最大高度差为1，因此它也被称为高度平衡树。

查找、插入和删除在平均和最坏情况下的时间复杂度都是

### 2-3树

一种平衡树，各种情况的时间复杂度都是。

2节点：有1个数据，2个子节点

3节点：有2个数据，3个子节点

由2节点、3节点和叶节点构成的树叫做2-3树

## Monte Carlo算法转化为Las Vegas 算法

Monte Carlo算法得到结果后进行正确性验证，正确的情况下才返回解，否则不返回。

## 0-1背包问题

有n个物体，重量和价值分别为，求一个承重为W的背包中最多能装有多少价值的物体。

多项式等价的判定问题：在上述条件下，一个承重为W的背包中装有的物体价值是否？

等价原因：若判定问题存在多项式时间的解法，那么有限次内调用该解法，即可解决0-1背包问题。

# 分治

## 逆序数

写出一个求解下列问题的分治算法,推导其时间复杂性并与蛮力法相比较。 给定互不相等的n个数的一个序列若其中某两个数和的关系为: 且𝑖 < 𝑗,则称和是逆序的。要求计算该序列中的逆序个数,即具有逆序关系的元素对的总数目.

解：

总体思路：当待统计数组个数大于1时，将数组分为两部分分别统计逆序数，然后在统计这两个子数组之间的逆序数，并且在统计时对两个子数组进行归并排序。总逆序数等于这三者的和。

count(A, begin\_i, end\_i) ：递归统计数组A[begin\_i,end\_i]内的逆序数

combine\_count(A, l\_begin\_i, l\_end\_i, r\_begin\_i, r\_end\_i) :

统计数组A[l\_begin\_i,l\_end\_i]与数组A[r\_begin\_i,r\_end\_i]之间的逆序数，将l\_A和r\_A两个数据取出来，在之前的迭代中，他们是已经排好序的。现在对他们进行合并排序，并在合并的过程中记录逆序数的个数。

伪码：

count(A, begin\_i, end\_i){

if(begin\_i < end\_i) then{

temp\_i = (begin\_i + end\_i)/2 #取整

count(A, begin\_i, temp\_i)

count(A, temp\_i, end\_i)

combine\_count(A, begin\_i, temp\_i, temp\_i, end\_i)

}

}

combine\_count(A, l\_begin\_i, l\_end\_i, r\_begin\_i, r\_end\_i){

l\_A = A[begin\_i, l\_end\_i]

r\_A = A[r\_begin\_i, r\_end\_i]

i = j = 1

k = l\_begin\_i

while(i <= l\_A.length && j <= r\_A.length) do{

if l\_A[i] <= r\_A[j] then

A[k] = l\_A[i]

i ++

k ++

else

A[k] = r\_A[j]

j ++

k ++

count\_num += l\_A.length + j - i

}

}

count\_num = 0 #计数器，全局变量

main(A){

count(A, 1, A.length)

}

分析：

暴力求解法：

分治法：

每层的时间复杂度为，有层

总时间复杂度为

# 变治

## 求多数元素

A[1,…,n]为一个整数序列，A中的整数a如果在A中出现次数多余⌊n/2⌋，那么a称为多数元素。例如，在序列1,3,2,3,3,4,3中3是多数元素，因为出现了4次，大于⌊7/2⌋。求A的多数元素问题的蛮力算法复杂性如何？设计一个具有变治思想的算法，提高蛮力算法的效率，写出伪代码并分析其事件复杂性。

解法一：

方法和3相同，先归并排序，时间复杂度为，然后再遍历一遍统计元素出现次数，时间复杂度为。最终时间复杂度为

# 动态规划

有8万元的投资可以投给3个项目，每个项目在不同投资数额下（以万元为单位）的利润如下表：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 项目1 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 98 | 100 |
| 项目2 | 0 | 5 | 15 | 40 | 60 | 70 | 73 | 74 | 75 |
| 项目3 | 0 | 4 | 26 | 40 | 45 | 50 | 51 | 52 | 53 |

请安排投资计划，使总的利润最大。

写出你所设的状态变量、决策变量、状态转移方程与递推关系式，和手工求解的详细步骤和结果。

解：

假设变量：

：在前k个项目中投资x万元可以获得的最大利润,

：在第i个项目上投资j万元能够获得的利润

状态变量：前k个项目的总投资金额

决策变量：第k个项目的投资金额，

状态转移方程：

递推关系式：

k=1时：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 98 | 100 |

k=2时：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 98 | 100 |
| 2 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 120 | 140 |

k=2时：

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| k x | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 98 | 100 |
| 2 | 0 | 5 | 15 | 40 | 80 | 90 | 95 | 120 | 140 |
| 3 | 0 | 5 | 26 | 40 | 80 | 90 | 106 | 120 | 140 |

最优140，

某工厂调查了解市场情况，估计在今后4个月内，市场对其产品的需求量如下表所示：

|  |  |
| --- | --- |
| 时期(月) | 需求量(产品单位) |
| 1 | 2 |
| 2 | 3 |
| 3 | 2 |
| 4 | 4 |

已知：

对每个月来讲，生产一批产品的固定成本费为3(千元)，若不生产则为零。

每生产单位产品的成本费为1(千元)。

同时，在任何一个月内，最大生产量不超过6(千元).

每单位产品的库存费用为每月0.5(千元)

同时要求第一月初和第四个月末，均无产品库存。

问：在满足上述条件下，该厂应该如何安排各个时期的生产使所花费的总成本最低？

写出你所设的状态变量、决策变量、状态转移方程与递推关系式，和手工求解的详细步骤和结果。

解：

# 分支定界法

## AB国贸易问题

A国与B国之间尚未直接通商。与A国直接通商的有20个国家（C1, C2, …, C20）；与B国直接通商的为另外30个国家（C21, C22, …, C50）。上述50个国家之间并不是每两个国家都直接通商，任意两国之间的贸易税率由对称矩阵R给出，其中∞代表两国不能直接通商。

A国某公司与B国一公司欲通过某几个中间国家的公司完成一笔贸易，各个国家的进出口贸易通关等手续所需办理时间由向量T给出。请安排一中转贸易计划，使得该交易所产生的向各中转国缴纳的税费最低，且整个交易能够在时间t内完成。

1. 说明你是如何构造搜索树的。（要求是二叉搜索树）
2. 说明算法遍历搜索树的原则。（何时以及如何前进、分支、回溯、剪枝等等）
3. 你设计的分支定界算法的“界”是什么，他为什么是正确的和有效的？
4. 写出伪代码。

### 构造二叉树的方式

将城市作为二叉树的节点，其中根节点为城市A，记已有的贸易顺序为S，贸易顺序中最后一个国家为。

对于每个节点，都从与其连的国家中选择一个满足要求的国家，左子树代表将该国家加入到贸易顺序S中，并更新S与.

右子树代表不将该国家加入到贸易顺序中，S与维持不变。

按照先变量左子树，再遍历右子树的顺序进行遍历所有可能的解，从中选择最优的。

### 原则

在每次分支时，都需要重新计算一个下次可选国家的集合T：

1. 如果本次分支是向左分支，即将该国家加入到贸易顺序S中，则T为满足以下条件的国家的集合：
2. 与相连
3. 不在从根节点到节点的简单路径中
4. 从根节点到的时间不超过t
5. 从根节点到的税费不超过已有的最低税费
6. 如果本次分支是向右分支，即国家不加入到贸易顺序S中，则T为父节点的

前进：

当同时满足下面三个条件时可以前进：

1. 当前节点不是城市B
2. T不为空
3. 当且节点不满足剪枝条件

分支：

先遍历左分支，再遍历右分支

回溯：

当满足下述条件之一时进行回溯：

1. 当前节点是城市B，此时有解，看情况更新解
2. 为空，这时本条路径无解

在回溯时，如果当前节点是左子节点，则遍历相应的右子节点。否则，对父节点进行回溯。

剪枝：

1. 当前节点不是B，且从根节点到当前节点的简单路径所导致的税费已经不小于已求得的最小解，则进行剪枝。
2. 当前节点不是B，且从根节点到当前节点花费的时间已经不小于T，则进行剪枝。

### 界

1. 可选子节点不在从根节点到当前节点的简单路径上
2. 从根节点到当前节点的简单路径所导致的税费已经不小于已求得的最小解
3. 从根节点到当前节点花费的时间已经不小于T

第一条是防止无效的搜索，第二条是保证求得一个更优的解，第三条是为了满足限制要求。

## 地井建设问题

某部门欲建立联通分布于五个区的共50个站点的有线通讯网络，每两个站点之间的线路敷设费用由对成矩阵C给出，任意两站点之间敷设线路需建设的地井数目由对称矩阵U给出。

设计一线路敷设总费用为最小的无环网络，使得徐建设的总地井数目不超过UMAX，且需跨区敷设的线路总数目不超过DMAX(个站点所属的区由向量D给出)。

1) 说明你是如何构造搜索树的。（要求是二叉搜索树）

2) 说明算法遍历搜索树的原则。（何时以及如何前进、分支、回溯、剪枝等等）

3) 你设计的分支定界算法的“界”是什么，他为什么是正确的和有效的？

4) 写出伪代码。