1. 填空（20分）
2. 回溯法和分支限界一般用于求解什么问题

0-1背包问题、旅行商问题

随着实例规模的增长，问题的选择次数至少呈指数增长。

1. Kruskal和Prime算法基于什么策略（分支、动态规划、分治）

贪心策略

1. 二分算法复杂度（最好最坏平均）[二分搜索]

最好：O（1）最坏：O（logn）平均：O（logn）

1. 分治法实现快速排序（最好最坏平均复杂度）

最好：O（nlogn）最坏：O（n^2）平均：O（nlogn）

1. 回溯和分支限界对树的遍历方式

回溯：深度优先 分支：广度优先

1. 0-1背包问题有哪些策略可以求解

蛮力法、递归与分治策略、动态规划、贪心算法、回溯法、分支限界法

1. 动态规划一般用于求解什么问题，遵循什么条件（法则）

求解背包问题、最优二叉查找树 遵循最优化法则

1. 最近对问题、凸包问题使用什么策略（书上有）

蛮力算法、分治法

1. 递归算法的定义

通过重复将问题分解为同类的子问题而解决问题的方法

1. 快包法的时间复杂度（最好最坏）一般用什么策略

最好：O（nlogn） 最坏：O（n^2） 策略：分治法

1. “满2-3树”高度（最矮最高）怎么算，应用在什么地方

最矮：log3（n+1）最高：log2（n+1）-1 应用：有效的查找、添加和删除元素

1. 选择题（10分）
2. Dijkstra算法（概念、最简单情况、最复杂情况）

概念：是典型的单源最短路径算法，用于计算一个节点到其他所有节点的最短路径。

最简单情况：O(|V|\*|V|)

最复杂情况：O(|E|log|v|)

1. 动态规划是什么东西、应用场景

动态规划：是求解[决策过程](https://baike.baidu.com/item/%E5%86%B3%E7%AD%96%E8%BF%87%E7%A8%8B/6714639" \t "https://baike.baidu.com/item/%E5%8A%A8%E6%80%81%E8%A7%84%E5%88%92/_blank)最优化的过程

应用场景：求一个问题的最优解，而且该问题能够分解成若干个子问题，并且子问题之间还有重叠的更小的子问题。0-1背包问题、公共子序列

1. “⚪”“⚪中间又H”两个符号的性质、定义

O：对于所有的n≥n0来说，t(n)≤cg(n)

Ω：对于所有的n≥n0来说，t(n)≥cg(n)

⚪中间有H：对于所有的n≥n0来说，c2g(n)≤t(n)≤c1g(n)

1. 贪心算法&动态规划算法是否可以解决相似问题，可以解什么类型的问题

可以解决相似问题

动态规划法和贪心法的区别  
共同点：  
贪心算法和动态规划算法都要求问题具有最优子结构性质。  
不同点：  
动态规划算法中，每步所做的选择往往依赖于相关子问题的解，因而只有在解出相关子问题时才能做出选择。  
而贪心算法，仅在当前状态下做出最好选择，即局部最优选择，然后再去解做出这个选择后产生的相应的子问题。  
贪心无法解决动态规划的问题，但是动态规划能解决贪心的问题。虽然能够应用贪心算法一定能够应用动态规划法，但是一般来说，贪心算法的效率高于动态规划法，因而还是应用贪心算法。  
动态规划算法通常以自底向上的方式解各子问题，而贪心算法则通常以自顶向下的方式进行，以迭代的方式作出相继的贪心选择，每做一次贪心选择就将所求问题简化为规模更小的子问题。

1. 贪心算法&分治限界 各能解决什么类型问题，一个问题选哪个解决好
2. 变治算法适用于哪些问题
3. 动态规划求解的问题整理
4. 简答题（30分）
5. 相客问题（礼堂时间安排？）
6. Kruskal Prime算法多角度比较两个算法的优缺点

Prim算法和Kruskal算法都是从连通图中找出最小生成树的经典算法。

从策略上来说，Prim算法是直接查找，多次寻找邻边的权重最小值，而Kruskal是需要先对权重排序后查找的

Prim算法：稠密图 Kruskal算法：稀疏图

从时间复杂度来说：Prim算法：O(|V|logV) Kruskal算法：O(ElogV)

1. AVL树 如何构建 什么是AVL树

AVL树：是一棵二叉查找树

1. 主定理的应用 master theorem（难）
2. TSP货郎问题 给图建树、树的遍历、什么时候回溯
3. Floyd算法 给图/表如何生成目标的表
4. 问答题（40分）

[标明“伪代码”的题写，没标不写]

数塔问题（最常规问题）解决方案：动态规划

一个队列中多数元素 分治策略

二分查找 分治策略（伪代码）

回溯策略 回文

找中位数 分治策略

0-1背包问题 动态规划（变形问题）

子集合问题 回溯策略

求最长递增字符串长度 动态规划

（递增递减，形式变换要认出）