1. 填空
2. 回溯法和分支限界一般用于求解什么问题

随着实例规模的增长，问题的选择次数至少呈指数增长

0-1背包问题、旅行商问题

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：深度优先遍历方式

分支限界：广度优先遍历方式

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）

求解最优化（背包，最优二叉查找树）问题 遵循最优化法则

1. 最近对问题、凸包问题（书上有）

蛮力法、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） 分治法

1. “满2-3树”高度（最矮最高）怎么算，应用在什么地方

查找、插入、删除元素

1. 选择题
2. Dijkstra算法（概念、最简单情况、最复杂情况）

贪心策略

Dijkstra算法是解决有向带权图中最短路径的算法

dijkstra算法

普通算法O(n²)

堆优化算法O（Elogn）

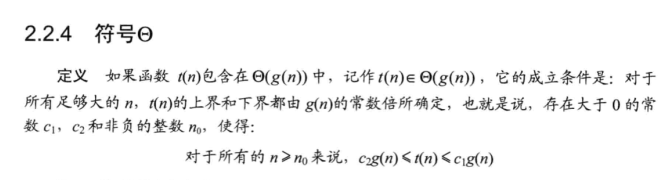
1. 动态规划是什么东西、应用场景

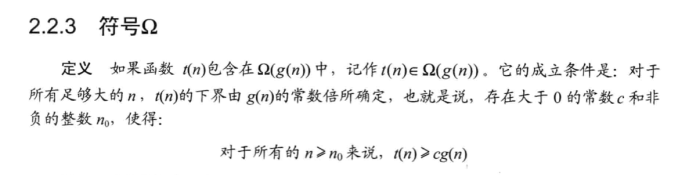
是求解决策过程最优化的过程

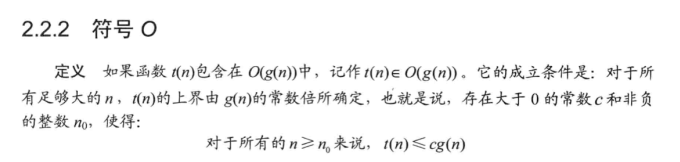
求一个问题的最优解，而且该问题能够分解成若干个子问题，并且子问题之间还有重叠的更小的子问题。

币值最大化 硬币收集 找零问题

1. 符号的性质、定义







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

贪心算法是动态规划的一种特例

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

1. 贪心算法&分支限界 各能解决什么类型问题，一个问题选哪个解决好

贪心算法：每一步找最优，不是整体最优 背包问题 均分纸牌

分支限界法：找整体最优

1. 变治算法适用于哪些问题

预排序 高斯消去法（LU分解 矩阵求逆） 平衡查找树 堆和堆排序 霍纳法则和二进制幂

最小公倍数

1. 动态规划求解的问题整理

1.硬币找零

2.字符串相似度/编辑距离

3.最长公共子序列

4.最长递增子序列

5.最长连续子序列和/积

6.矩阵链乘法

7.0-1背包问题

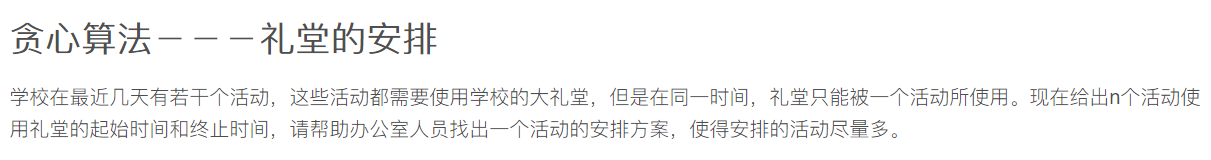
8.有代价的最短路径

9.瓷砖覆盖(状态压缩DP)

10.工作量划分

11.三路取苹果

1. 简答题
2. 相客问题（礼堂时间安排？）



**思路分析：**

首先将各个活动按照结束时间进行排序。第一个结束的活动必选。然后依次考虑剩下的各个活动，如果该活动的开始时间小于上一个活动的结束时间，则不选；  
否则，将活动方案数加一。

**解题思路：**

1.对活动进行排序，开始时间越早排在越前面，如果两个活动时间相同，则结束时间越早的排在越前面

2.始时间最早和持续时间最短的优先安排会场，并记录会场号，

3.其余活动的开始时间大于或等于已安排活动的结束时间的安排在同一会场，

4.若某活动的开始时间小于已经安排了会场的活动的结束时间，则安排在另一会场，记录会场号，

5.依次循环，直到所有活动均被安排

1. Kruskal Prime算法多角度比较两个算法的优缺点

两者都是贪心的思想，只不过考虑的角度不同：

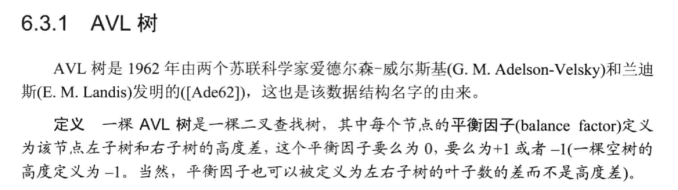
Prim算法从顶点的角度出发，每次选择距离当前节点最近的节点加入，直到所有节点都加入。Kruskal算法从边的角度出发，每次总是选择权重最小的边加入，直到加入n-1条边为止。

Kruskal算法在效率上要比Prim算法快，因为Kruskal只需要对权重边做一次排序，而Prim算法则需要做多次排序。

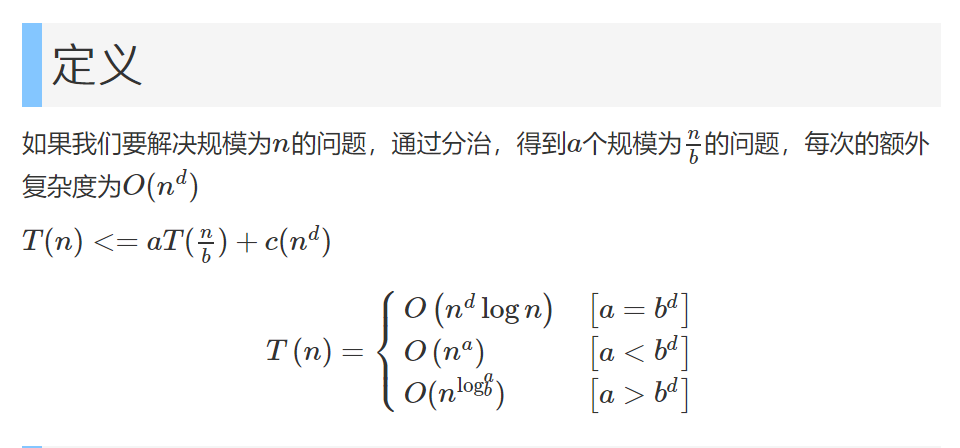
Prim算法求最小生成树时候，和边数无关，只和定点的数量相关，所以适合求稠密网的最小生成树，时间复杂度为O（n\*n）。

Kruskal算法需要对图的边进行访问，所以Kruskal算法的时间复杂度只和边又关系，可以证明其时间复杂度为O（eloge）。

1. AVL树 如何构建 什么是AVL树（给一串数字如何构建）



1. 主定理的应用 master theorem（难）



1. TSP货郎问题 给图建树、树的遍历、什么时候回溯
2. Floyd算法 给图/表如何生成目标的表

从任意节点 i 到任意节点 j 的最短路径无外乎两种可能：1、直接从 i 到 j ；2、从 i 经过若干个节点 k 到 j 。所以，我们假设 Dis（i， j）为节点 u 到节点 v 的最短路径的距离，对于每一个节点 k ，我们检查Dis（i， k） + Dis（k， j）<< Dis（i， j）是否成立，如果成立，证明从 i 到 k 再到 j 的路径比 i 直接到 j 的路径短，我们便设置DIs（i， j） = Dis（i， k） + Dis（k， j），这样一来，当我们遍历完所有节点 k ，Dis（i， j）中记录的便是 i 到 j 的最短路径的距离。

1. 问答题

[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）