Take Home Message

- **Take Home Message 1**
  - 1.学会将问题进行简化，从最简单的case入手
  - 2.对于问题的解决方法，不要盲目去试，需要基于对于问题结构和解的观察来设计算法 -- The Art of Algorithm
- **Take Home Message 2**
  - 1.采样 - 小样本估计总体
  - 2.要深入理解代码需要每一步自己执行，观察其中规律
  - 3.对于参数的思考：由固定参数能否考虑成变量
  - 4.脱离思维定式(比如子树不一定只能选儿子)
  - 5.对于许多无结构问题，尝试引入结构(比如说二维平面求最近点)
  - 6.对于问题，先写一个笨办法，发现冗余，再尝试改进
- **Take Home Message 3**
  - 节省空间 - 以算代存
  - 以存函数代替存值
- **Take Home Message 4**
  - 动态规划省时间：去冗余
    - 只算回溯路径附近
    - OPT表中很稀疏，则只算变化处
  - 如果子问题过于粗糙，往往建立不了work的递归关系
    - 则此时做更细的子问题(例如 d(v) -> d(v, k)) - 描述解的性质
- **Take Home Message 5**
  - 1.对于有向无环图，可以线性化求解
  - 2.对于问题的定义，发现无法解决时可以考虑更细化定义问题
  - 3.Greedy算法能够解决的问题背后往往同时能够通过DP解决，虽然可能会效率低
  - 4.AI可以做一些启发式的工作，并且可以以此作为启发分析隐藏层的解释性
- **Take Home Message 6**
  - IS/SSP -- 多个选择项，立即知道必送堆
    - version 1: DP
    - version 2: Greedy
  - 算法设计的哲学
    - 找准限制，放松之，但要适度

- **Take Home Message 7**
  - 1.我们常常会观察到：对于循环等等，一次慢的操作之后会有多次较为快速的操作，此时不要再使用单次O的最坏情形估计，可以考虑均摊分析的方法(一串情况求平均)
  - 2.求和：把single op的时间表示成势能的下降(考虑到便于累加的形式)，便于求和
  - 3.抽象：将一个问题的思考转移到其等价的问题思考求解上，例如对于示例向量问题的求解方法，可以抽象到图的问题求解方法上
- **Take Home Message 8**
  - 要写整数线性规划约束目标函数的时候，先用自然语言描述，尤其是 IF THEN
- **Take Home Message 9**
  - 1.任何一个代数对象的背后往往都有几何直观
  - 2.矩阵A乘向量的理解，要抛弃仅仅是点积的理解，要有多种理解 (2.向量加权 3.线性变换)
- **Take Home Message 10**
  - A -> A+epsilon(即加上一个噪声)，噪声有时是好东西
  - 很多问题都有另外一个侧面 - > 对偶的思想
  - 把带约束的问题转化成无约束的优化函数
  - 无约束不可导 -> 找近似 -> 控制近似程度
- **Take Home Message 11**
  - 求解约束优化问题时，始终会有原目标函数大于等于拉格朗日 L(x,lamda)， 大于 g(lamda)
  - 拉格朗日乘子 就是 对偶变量 就是 边际成本
  - 拉格朗日乘子也有量纲
  - 等式的最优解是考虑拉格朗日条件；不等式考虑KKT条件
- **Take Home Message 12**
  - 1.找x时需要同时满足多个条件，怎么办？ -- IMPROVEMENT
    - 先满足其中几个，然后再迭代优化其他条件的不满足程度
  - 2.新思路哪里来？ -- 已有思路，已有方法的转化
  - 3.千方百计引入对偶变量 -- 将方法转化
- **Take Home Message 13**
  - 考虑KKT时 : 在严格约束难以直接达到的情况下，考虑分步分阶段达到，不要 one-hit
  - 不要trick，用log barrier近似
- **课程总结**
  - **如何解决问题**
    - **观察一 -- 问题的可分解性**：问题的最简单实例是什么？复杂的实例能否分解成简单的实例

- **观察二 -- 可行解的形式以及解之间的变换关系**：问题的可行解的形式是什么？可行解的总数有多少？可行解能否一步一步地逐渐构建出来？我们能否对一个可行解施加小幅扰动，将之变换成另一个可行解？
- **观察三 -- 类似的问题**：和给定问题类似的问题有哪些？解决类似问题的算法能否直接应用于解决当前的问题？如果不能，那又是什么因素造成了妨碍？能否想办法消除妨碍？
- **观察四 -- 现有算法的不足**：现有的求解算法有哪些不足？比如：是否存在冗余计算？如果存在的话，能否想办法去消除冗余计算？

- **解决问题的思想**
  - 基于对问题结构观察的算法设计
  - 从最简单的例子做起
  - 试图把大的问题分解成小问题
  - 试图从粗糙解开始逐步改进
  - 试图枚举所有的解，但是“Being Smart”
  - 难以优化的函数，用上界或下界函数代替
  - 复杂操作的潜力一定要挖尽
  - 求同时满足多个条件的解，分步满足，并利用对称性
  - 想想对偶
  - Random Sampling
  - Scaling Numbers
  - Best -> Sufficiently Good(求解最优往往耗时费力，求解次优以兼顾效率)
  - 模拟算法的执行，观察规律
  - 思考设置额外的参数能否让问题的求解变得简单
- **能够理解别人的解答，但不理解怎样能够想到这样的解答？**
  - 锻炼问题求解的思想，知其然，知其所以然