复习

在线不考

线性规划

给定约束条件，求最大/最小化目标函数。

$$\min(\max)\ z=\sum_{j=1}^{n} c_j x_j \qquad \text{目标函数}$$

$$\text{s.t.}\quad \sum_{j=1}^{n} a_{ij} x_j \le (=,\ge) b_i,\quad i=1,2,\ldots,m \qquad \text{约束条件}$$

$$x_j \ge 0,\quad j \in J \subseteq \{1,2,\ldots,n\} \qquad \text{非负条件}$$

$$x_j\ \text{任意},\ j \in \{1,2,\ldots,n\}-J \qquad \text{自由变量}$$

转标准型

$$\min z=\sum_{j=1}^{n} c_j x_j$$

$$\text{s.t.}\ \sum_{j=1}^{n} a_{ij} x_j = b_i \ge 0,\quad i=1,2,\ldots,m$$

$$x_j \ge 0,\quad j=1,2,\ldots,n$$

如何从一个线性规划转化成标准型。

首先调整目标函数(max->min)。其次，对于里面的约束，如果是等号，那么判断是否非负。如果是不等号，根据不等号的方向引入松弛变量或剩余变量。最后，将自由变量变成非负。

基础概念

基、基变量、非基变量、基本解、可行解、基本可行解等

单纯形表

对偶规划

（我不知道今年考不考，但去年考了如何转对偶规划）

原始规划	对偶规划
$\max \sum_{j=1}^{n} c_j x_j$	$\min \sum_{i=1}^{m} b_i y_i$
$\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ 1 \le i \le s$	$y_i \ge 0,\ 1 \le i \le s$
$\sum_{j=1}^{n} a_{ij} x_j = b_i,\ s+1 \le i \le m$	y_i 任意, $s+1 \le i \le m$
$x_j \ge 0,\ 1 \le j \le t$	$\sum_{i=1}^{m} a_{ij} y_i \le c_j,\ 1 \le j \le t$
x_j 任意, $t+1 \le j \le n$	$\sum_{i=1}^{m} a_{ij} y_i = c_j,\ t+1 \le j \le n$

注意下等式、不等式的符号和下标，目标函数max->min

平摊分析

在一系列操作中，通过对所有操作求平均后，即使其中单一操作有较大代价，平均代价还是很小。

方法

聚集分析：求出总时间，对操作数求平均

记账法：为每一个操作分配一个代价，类似于银行存款取款

势能法：定义每个状态的势函数

例子

栈、二进制计数器、动态表

碎碎念

除了课程ppt可以看下算法导论第17章。势能法考的概率会大一些。

去年考的好像也是动态表的势能法，复习的时候首要过一下这个吧~

三、平摊分析（10）

分别给出使用下列策略的动态表的插入与删除操作的平摊复杂度以及势能函数定义。

(1) 元素满时扩大 1 倍空间，元素不足 1/3 时减少 1/2 空间。

(2) 元素满时扩大 1/3 倍空间，元素不足 4/9 时减少 1/3 空间。

就类似这个

网络流

最大流

最大流问题定义：在一个网络中，求源点到汇点的最大流量。

重要概念：

割集、割集容量
增广链，增广路径，辅助网络（残余网络），分层辅助网络
圈流、最小费用流

最大流-最小割集定理：

最大流=最小割

算法

Ford-Fulkerson算法，Dinic算法

费用流

费用：为每条边增加费用。可行流的费用是每条边流量的加权和
最小费用流：流量为 V0 的可行流中费用最小的流。

算法

消圈算法、最短路算法（不断寻找费用最小的 s-t 增广链）

写建模的时候

写清楚必要的细节：边、点、容量、流的定义，（必要时辅助画图），最后的结果是什么等

应用部分

- 带需求的流通 (顶点有需求和边有上下界)
- 运输问题 (增加超级源汇的费用流)
- 二部图最大匹配（匈牙利算法、网络流算法）
- 赋权二部图最小权最大匹配 (匈牙利算法【KM算法】、线性规划)
- 网络流实际应用：图像分割（最小割）

碎碎念

可以考前拿**网络流24题**练练手

https://loj.ac/p?keyword=%E7%BD%91%E7%BB%9C%E6%B5%81

（洛谷上好像有别人写的题解）https://www.luogu.com.cn/problem/list?keyword=%E7%BD%91%E7%BB%9C%E6%B5%8124%E9%A2%98&page=1&tag=332

算法复杂度

评价指标

正确性、时间复杂度、空间复杂度、简单性

如何寻找最优算法：首先设计一个正确的算法，得到上界W。然后分析输入规模得到下界F。之后比较上界与下界的关系，如果不同阶，就设计更优的算法或者分析出更紧的下界，直到同阶。

求问题下界

决策树 ***

对于以比较为基本运算的算法，可构造一个决策树。树深为d，树叶为t，那么$d \geq log(t)$
最坏时间复杂度：树的深度
平均时间复杂度：树的平均路径长度

构造最坏输入

操作有决定性操作和非决定性操作两种。构造最坏输入就是假设我们已经知道全部输入数据，然后分配比较的顺序，让非决定性操作尽可能多。

规约

将一个问题转化为另一个问题，然后这两个问题共享同一个下界。

	算法	最坏情况	空间
选最大	顺序比较	$n-1$	$O(1)$
选最大和最小	顺序比较	$2n-3$	$O(1)$
	算法FindMaxMin	$\lceil 3n/2 \rceil -2$	$O(1)$
选第二大	顺序比较	$2n-3$	$O(1)$
	锦标赛方法	$n+\lceil \log n \rceil -2$	$O(n)$
选中位数	排序后选择	$O(n\log n)$	$O(1)$
	算法Select	$O(n)$~$2.95n$	$O(\log n)$

算法	最坏情况	平均情况	占用空间	最优性
冒泡排序	$O(n^2)$	$O(n^2)$	原地	
快速排序	$O(n^2)$	$O(n\log n)$	$O(\log n)$	平均最优
归并排序	$O(n\log n)$	$O(n\log n)$	$O(n)$	最优
堆排序	$O(n\log n)$	$O(n\log n)$	原地	最优

NP完全

P类问题是多项式时间可解的问题。

NP类是多项式时间可验证的问题。

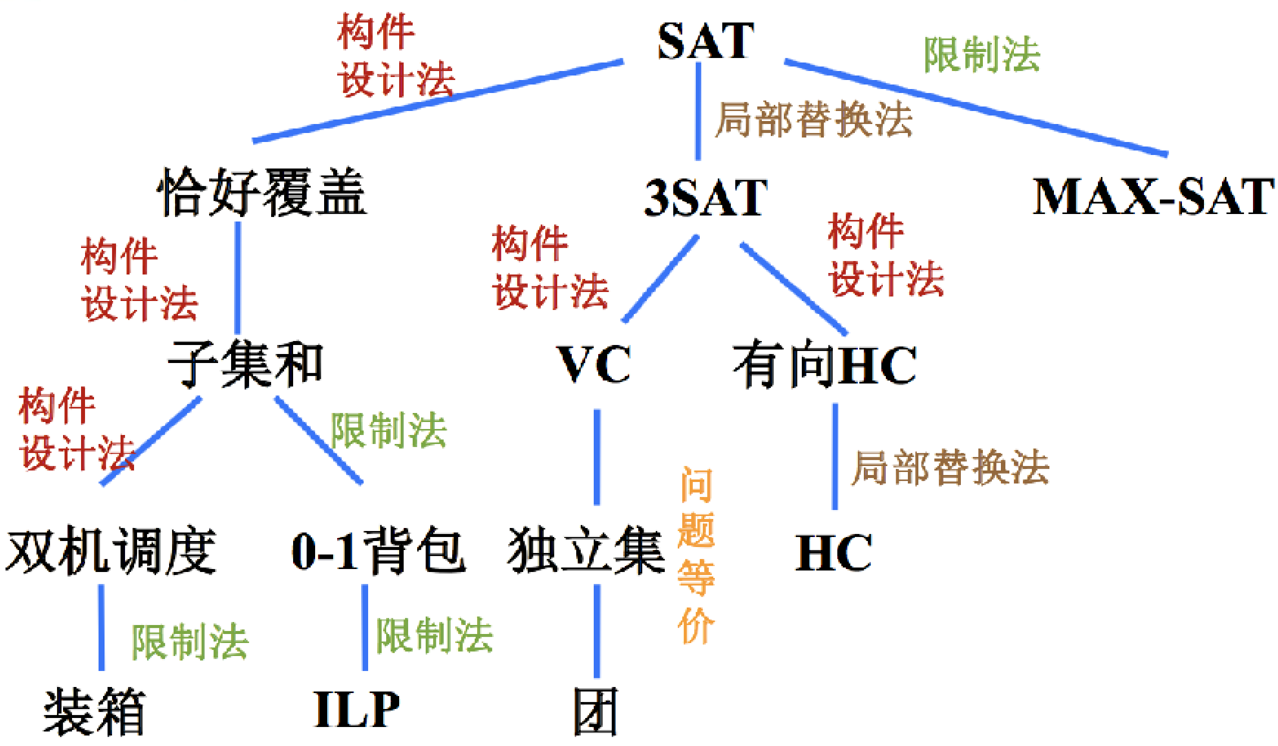
NP-hard和NPC

多项式时间变换$\leq_p$

f是多项式时间可计算的
对所有的$I \in D1$，$I \in Y1 \Leftrightarrow f(I) \in Y2$

NPC证明

(1)证明$\prod \in NP$
(2)找到一个已知的 NP 完全问题$\prod'$，并证明$\prod\prime \leq p \prod$



碎碎念

构件设计法可以多做几道题拓宽思路，防止考到

Karp’s 21 NPC problems:
https://en.wikipedia.org/wiki/Karp%27s_21_NP-complete_problems

近似算法

概念

近似算法、近似比、完全可近似、可近似、不可近似
最小化问题：$r_A(I) = \frac{A(I)}{OPT(I)}$
最大化问题：$r_A(I) = \frac{OPT(I)}{A(I)}$
$r_A(I) \le r$，近似比 r 是比值的上限

例子

多机调度、货郎问题、背包问题

随机

Las Vegas 型随机算法 （保证100%正确，时间不定）

运行时间本身是一个随机变量，期望的运行时间是输入规模的多项式且总是给出正确答案的随机算法为**有效的拉斯维加斯算法**.

随机快速排序、随机选择、随机 n 后放置

Monte Carlo型随机算法 （保证错误率和时间，允许出错）

总是在多项式时间内运行且出错率不超过1/3的随机算法为**有效的蒙特卡洛算法**

主元素测试、串相等测试、模式匹配、素数测试

碎碎念

可以看看课后题第二题里如何使用马尔科夫不等式的，

关于期中前面的碎碎念

树形dp，一些规划问题，对自己薄弱的算法设计地方加强

有一些资源网

https://www.jiumodiary.com/ 可以搜一些书

https://www.connectedpapers.com/ 顺着一篇paper漫游的时候可以看看x