Part 1:NP-Completeness定义

这一周的课程开始介绍NP-Completeness，即NP完全问题，在讨论之前先要看一下几个与之相关的定义。

首先要看的是P的定义，这里介绍P之前先要介绍Polynomial-Time Solvability的定义。

1.Polynomial-Time Solvability

定义：存在一个算法可以在$O(n^k)$时间内准确地解决一个问题，其中k为常数，n为输入的数据数量。

2.The Class P

定义：P=Polynomial-Time Solvability问题的集合，即可以在多项式时间内确定解决的问题的集合。

接着来看一下两个非常重要的概念，Reductions and Completeness。

3.Reductions（归约）

定义：现在有两个问题$\Pi 1, \Pi 2$，如果给$\Pi 2$一个多项式时间子程序，可以用它在多项式时间内来求解$\Pi 1$，那么称$\Pi 1$可以归约为$\Pi 2$。

Quiz

以下哪项陈述属实?

A）计算中位数可以归约为排序

B）检测环可以归约为深度优先搜索

C）多源最短路径都可以归约为单源最短路径

D）以上全部

A：排序完成之后已经计算出了中位数，所以A成立

B：如果在访问全部节点之前，深度优先搜索已经访问到出发点，那么必然有环

C：对每个节点使用单源最短路径算法即可求出多源最短路径

答案是D

4.Completeness

定义：C =问题的集合。对于问题Π，如果

(1)$\Pi \in C$

(2)C中的每个问题可以归约到Π

那么称问题Π 是C-complete。

可以简单理解为Π是C中最难的问题。

介绍了以上概念之后就可以介绍NP以及NP-Completeness

5.NP

定义：NP是一系列问题的集合，当一个问题满足

(1)问题关于输入数据大小总有多项式解

(2)可以在多项式时间内验证问题的解

老师这里提到NP为题的全称为Nondeterministic polynomial，即非确定多项式。

6.NP-Completeness

根据NP和Completeness，NP-Completeness可以理解为NP问题中最难的问题。

7.P VS NP

老师对P和NP做了进一步的说明

P:polynomial time solvable 可以在多项式时间内解决问题。

NP:can verify correctness of a solution in polynomial time 可以在多项式时间内验证解的正确性。

P VS NP问题是一个著名的难题，人们大部分认为P不等于NP，但是至今没有有证明了它。

总结

面对NP-Completeness问题，老师给了以下三个建议：

(1)专注于计算上易于处理的特殊情况

(2)启发式算法 - 快速的算法但并不总是正确的

(3)以指数时间求解，但比蛮力搜索更快

Part 2:NP-Complete Problems

接下来看几个具体例子。

1.The Vertex Cover Problem

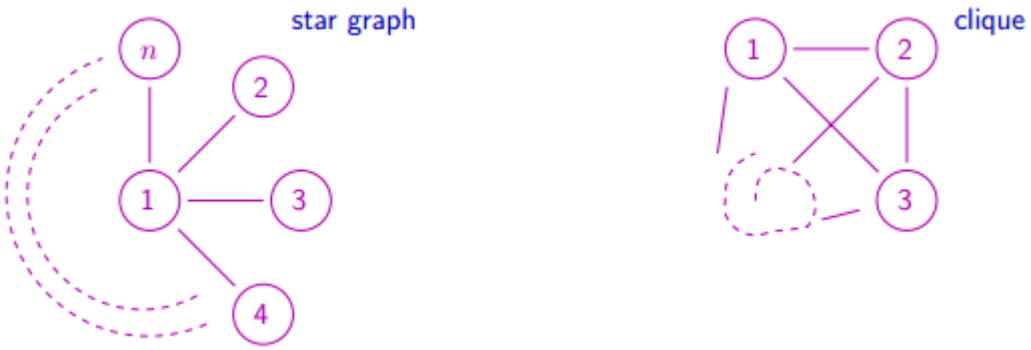
输入：无向图$G(V, E)$

输出：计算最小顶点覆盖子集，即$S \subseteq V$且S中至少包含一个G的每个边的一个端点。

先来看个具体的例子。

Quiz

对于n个点的star graph以及n个点的clique，最小顶点覆盖子集的大小分别是多少



A) 1 和 n-1 B) 1 和 n C) 2 和 n-1 D) n-1 和 n

如图，star graph只要取1这个点即可。clique的定义是任意两点之间都有边，所以任取n-1个点即可。

答案为A。

分析

这个问题直接解的话比较困难，老师这里将其换成另一个问题：

对于整数k，图中是否存在大小小于等于k的顶点覆盖。

蛮力求解的时间约等于$C_n^k = \theta(n^k)$，看起来不错，不过还可以优化，为了优化这个问题，老师先介绍了一个引理。

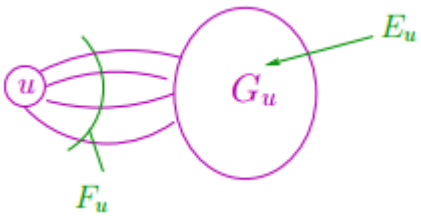
Substructure Lemma

考虑图$G(V, E)$，边$(u, v) \in G$，整数$k \geq 1$。令G_u为G删除u及其入射边被删除后的图，G_v同理。那么G有一个大小为k的顶点覆盖$\Longleftrightarrow G_u$或G_v(或两者)有一个大小为$k-1$的顶点覆盖

证明：

($\Longleftarrow$)假设G_u有一个大小为$k-1$的顶点覆盖S。将边E分解为如下形式，

$$E = E_u\left(\text{在 } G_u \text{ 内部}\right) \cup F_u\left(\text{和 } u \text{ 相邻}\right)$$



由于S有E_u中每个边的一个端点，那么$S \cup u$是一个大小为k的顶点覆盖。

($\Longrightarrow$)令S为G的一个大小为k的顶点覆盖，因为$(u,v) \in G$，那么u,v中至少有一个点（不妨设为u）属于S，因为E_u中没有一条边和u相连，所以$S-\{u\}$必然是G_u的一个大小为$k-1$的顶点覆盖。

基于以上引理，老师给出以下算法：

A Search Algorithm

[输入无向图$G(V,E)$，整数k] [忽略基本情形] （1）选取任意边$(u,v) \in G$。 （2）在G_u中递归地搜索大小为$k-1$的顶点覆盖S(G_u为G删除u及其入射边被删除后的图),如果找到了，返回$S \cup \{u\}$。 （3）在G_v中递归地搜索大小为$k-1$的顶点覆盖S,如果找到了，返回$S \cup \{v\}$。 （4）失败 [G没有大小为k的顶点覆盖集]。

算法分析

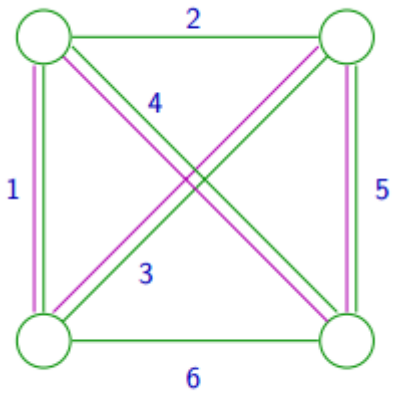
正确性：根据数学归纳法以及引理可以保证正确性。

运行时间：迭代次数为$O(2^k)$，最多两个分支，迭代深度最多k次。假设每次迭代还要花费$O(m)$的时间，那么一共需要$O(2^k m)$的时间，这个时间比我们之前介绍的$\theta(n^k)$要好。

2.The Traveling Salesman Problem(TSP)

输入：具有非负边成本的无向完全图

输出：成本最低的循环（即访问每个顶点恰好一次）



$OPT = 13$

模仿Bellman-Ford算法的思路，给出如下子问题：

子问题1：对于边的数量$i \in \{0,1,\ldots,n\}$，目的地$j \in \{1,2,\ldots,n\}$，令L_{ij}为1到j最多使用i条边的最短路径。

Quiz 1

哪些会阻止使用子问题来获得一个TSP的多项式时间算法?

A) 存在超过多项式数量级的子问题 B) 不能从较小的子问题计算更大的子问题 C) 解决所有的子问题并不能解决原始问题 D) 没有!

A:一共n^2个子问题

B:$L_{ij} = min_{k \neq j}\{L_{i-1,k} + c_{kj}\}$

C:这最后的情形为$i = j = n$，所以最后可以找到任意两点s, t之间的最短距离，但是这个最短路径不能保证经过每个点，所以不一定是TSP问题的最优解。

所以答案为C

因此我们对于子问题的定义有误，再来看个子问题的定义:

子问题2：对于边的数量$i \in \{0, 1, \ldots, n\}$，目的地$j \in \{1, 2, \ldots, n\}$，令$L_{ij}$为1到$j$准确使用$i$条边的最短路径。

Quiz 2

哪些会阻止使用子问题来获得一个TSP的多项式时间算法?

A) 存在超多项式数量级的子问题 B) 不能从较小的子问题计算更大的子问题 C) 解决所有的子问题并不能解决原始问题 D) 没有!

和上题相比，唯一变化的是C，尽管最后是n条边，但是可能访问重复顶点，所以C依旧错误。

可以看到路径长度大小以及无重复性均很关键，对子问题重新定义:

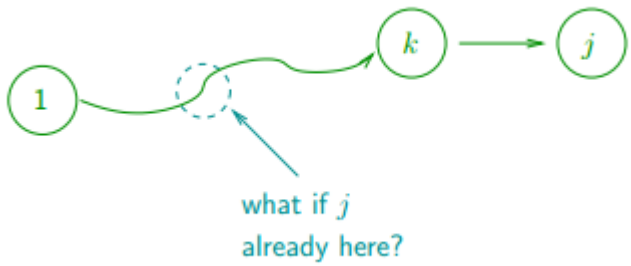
子问题3：对于边的数量$i \in \{0, 1, \ldots, n\}$，目的地$j \in \{1, 2, \ldots, n\}$，令$L_{ij}$为1到$j$准确使用$i$条边且无重复顶点的最短路径。

Quiz 3

哪些会阻止使用子问题来获得一个TSP的多项式时间算法?

A) 存在超多项式数量级的子问题 B) 不能从较小的子问题计算更大的子问题 C) 解决所有的子问题并不能解决原始问题 D) 没有!

第一眼看到这题的时候以为是D，然而答案是B，即$L_{ij} = min_{k \neq j}\{L_{i-1,k} + c_{kj}\}$不一定成立了，可以看如下图



这说明我们还需要具体已经访问了哪些节点，于是有了最终的子问题。

子问题 4：对于每个目的地$j \in \{1, 2, \ldots, n\}$，每个子集包含1和$j$的子集$S \subseteq \{1, 2, \ldots, n\}$ 令$L_{S,j}$为从1到j访问S的每个顶点一次的最短路径。

这时显然就有$L_{S,j} = min_{k \in S, k \neq j}\{L_{S-\{j\},k} + c_{kj}\}$，基于以上引理，老师给出如下算法:

A Dynamic Programming Algorithm

令A = 二维数组，索引为子集$S \subseteq \{1,2,\ldots,n\}$包含1和目的地$j \in \{1,2,\ldots,n\}$ 基本情况：

$$A[S,1] = \begin{cases} 0 & (\text{当} S = \{1\}) \\ +\infty & (\text{其余情况，因为无法避免访问顶点两次}) \end{cases}$$

对于$m = 2,3,\ldots,n$ [m为子问题的大小] 对于每个包含1的大小为m集合$S \subseteq \{1,2,\ldots,n\}$ 对于每个$j \in S, j \neq 1$ $A_{S,j} = min_{k \in S, k \neq j}\{A_{S-\{j\},k} + c_{kj}\}$ 返回$min_{j=2,\ldots,n}\{A[\{1,2,\ldots,n\},j] + c_{j1}\}$

其中$A[\{1,2,\ldots,n\},j]$为1到j访问每个节点一次的最短距离，c_{j1}为j到1的距离，因此最后返回的结果为$min_{j=2,\ldots,n}\{A[\{1,2,\ldots,n\},j] + c_{j1}\}$

算法分析

分析下运行时间，一共有2^n个子集，所以最外层循环有2^n次，内层的j要循环n次，对于$A_{S,j} = min_{k \in S, k \neq j}\{A_{S-\{j\},k} + c_{kj}\}$这个式子还要循环$n$次，所以最后的运行时间为

$$O(n \times 2^n)O(n) = O(n^2 2^n)$$

Part 3:习题

思考题

1.斯坦纳树

考虑边的大小均非负的无向图$G = （V，E）$。 现在给我们k顶点的集合$T \subseteq V$，称之为终端(terminals)。斯坦纳树是包含每对终端之间路径的边的子集。 例如，如果$T = V$，那么斯坦纳树与连接的子图相同。 事实上，斯坦纳树问题的决策版本是NP完全的。 现在为这个问题给出一个动态规划算法（例如，用于计算具有最少数量边的斯坦纳树），该算法的运行时间为$O(c^k . poly(n))$其中c是常数，poly是多项式函数。

现在考虑路径最短的斯坦纳树，用$i \in \{1,\ldots,|V|\}$表示每个点，$w_{i,j}$表示连接i,j的边的长度。设终端有k个点$S = \{x_1,\ldots,x_k\}$，对于$P \subseteq S$，注意最后要求解的是树，所以考虑根节点i，记$L[i,P]$为起点为i，树中包含P中全部元素的路径最短的树。接着构建递推关系

$$\begin{gathered} \text{关于} P \text{构建递推关系：} \\ \forall\, Q \subseteq P，\text{那么} L[i,P] \leq L[i,Q] + L[i,P-Q] \\ \text{所以} L[i,P] = min_{Q \subseteq P}\{L[i,Q] + L[i,P-Q]\} \\ \text{关于} i \text{构建递推关系：} \\ L[i,P] = min_j\{L[i,P], L[j,P] + w_{i,j}\} \end{gathered}$$

所以有以下算法：

对于$i = 1,\ldots|V|$ [i为根节点]: 对于$S = \{x_1,\ldots,x_k\}$的子集P: $L[i,P] = min_{Q \subseteq P}\{L[i,Q] + L[i,P-Q]\}$ $L[i,P] = min_j\{L[i,P], L[j,P] + w_{i,j}\}$ 最后返回$min_i L[i,S]$即可

算法分析

最外层循环$|V|$次，第二层循环考虑大小为2^i的子集P，这样有C_k^i种取法，对于该P，最内层$L[i,P] = min_{Q \subseteq P}\{L[i,Q] + L[i,P-Q]\}$操作要计算$P$的子集个数，共有$2^i$个（$P$有$i$个元素），最内层最后一步$L[i,P] = min_j\{L[i,P], L[j,P] + w_{i,j}\}$，最多需要考察每条边，所以一共需要$|E|$部操作。综上一共需要

$$|V|\sum_{i=0}^{k}C_k^i(2^i+|E|)=|V|3^k+|V||E|2^k=O(|V|3^k)$$

选择题

选择题1

鉴于目前的知识状况，下列哪些陈述不可能是真实的？ A) 没有可以在$O(n^{log\ n})$时间解决的NP完全问题，n为输入的大小。 B) 存在多项式时间可解的NP完全问题。 C) 有一个NP-complete问题可以在$O(n^{log\ n})$解决，n为输入的大小。 D) 一些NP完全问题是多项式时间可解的，并且一些NP完全问题不是多项式时间可解的。

A,B,C都是有可能成立的，如果一个NP完全问题可以在多项式时间内求解，那么所以NP完全问题都可以在多项式时间内求解，所以D不成立。

选择题2

让TSP1表示以下问题：给定一个TSP实例，其中所有边缘成本都是正整数，则计算最佳TSP巡回的值。令TSP2表示：给定一个TSP实例，其中所有边缘成本都是正整数，再给定一个正整数T，决定是否存在总长度最多为T的TSP巡回。让HAM1表示：给定一个无向图，计算哈密顿回路（恰好访问每个顶点一次的周期）的边，或确定该图没有哈密顿回路。设HAM2表示：给定一个无向图，决定该图是否包含至少一个哈密顿回路。

A) 如果TSP2是多项式时间可解，那么TSP1也是。如果HAM2是多项式时间可解的，那么HAM1也是。

B) TSP2的多项式时间可解性不一定意味着TSP1的多项式时间可解性。 HAM2的多项式时间可解性不一定意味着HAM1的多项式时间可解性。

C) 如果TSP2是多项式时间可解，那么TSP1也是。但是，HAM2的多项式时间可解性不一定意味着HAM1的多项式时间可解性。

D) TSP2的多项式时间可解性不一定意味着TSP1的多项式时间可解性。但是，如果HAM2是多项式时间可解的，那么HAM1也是。

对T=1,2...分别调用TSP1，这样就可以解决TSP2，所以TSP2可以归约到TSP1。

对于每个图，调用HAMA2，如果不存在哈密顿回路，则直接结束；如果存在哈密顿回路，那么删除一条边，如果删除后不是哈密顿回路，则删除的边必然属于原来的哈密顿回路，将这条边记录下来，然后将这条边添加回去，继续删除；如果删除后依旧是哈密顿回路，则继续删除，重复调用此操作，最后留下来的必然是哈密顿回路的边。设边的数量为m，所以这个操作需要$O(m)$的时间，因此HAM1可以归约到HAMA2。

所以这题选A

选择题3

假设 $P\neq NP$。考虑具有非负边长的无向图。以下哪个问题可以在多项式时间内解决?

提示：哈密尔顿路径问题是：给定一个n 个顶点无向图 ，决定是否有访问到每个顶点的（无循环）$n-1$条边的路径。你可以使用哈密顿路径问题是NP完全的事实。从哈密尔顿路径问题到下面4个问题中的3个有相对简单的减少。

A) 对于给定的来源 s和目的地 t，计算s到t有$n-1$条边的最短的路径 （或 $+\infty$如果不存在这样的路径）。路径不允许包含循环。

B) 在图的所有生成树中，计算叶节点数量最小的树。

C) 在图的所有生成树中，计算最大度的最小值。（回想一下顶点的度是入射边的数量。）

D) 对于给定的来源s和目的地 t，计算s到t有$n-1$条边的最短的路径（或 +∞如果不存在这样的路径）。该路径允许包含循环。

种

A:对任意两个s,t调用该算法，如果这C_n^2种组合中有经过每个点的路径，则求出哈密顿路径，否则不存在哈密顿论剑。从而哈密顿路径问题可以归约为A，所以A错误。

B:由于路径可以理解为叶节点数量为1的图，所以哈密顿路径可以归约为B，因为先对B求解叶节点数量最小的生成树，如果该树的叶节点数量为1则解决哈密顿路径问题，否则不存在哈密顿路径，所以B错误。

C:注意路径为特殊的树且路径的最大度为1，所以我们利用C求解最大度的最小值，如果该值为1，则求出了哈密顿路径，否则不存在哈密顿路径，所以C错误。

D:利用Bellman-Ford算法的思路可以求解。

所以答案为D

选择题4

选择最强的真实陈述。

A) 如果在二分图中最小顶点覆盖问题能在 $O(T(n))$时间内被解决，那么在二分图中最大独立集问题可以(bipartite graphs)中$O(T(n))$时间内解决。

B) 如果在一般图中最大独立集问题可以在 $O(T(n))$时间内被解决，那么在一般图中最小顶点覆盖问题能在 $O(T(n))$时间内被解决。

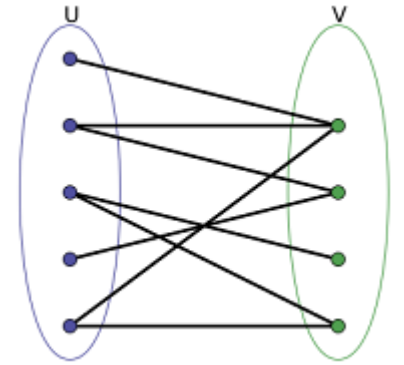
C) 所有其他三个断言都是真实的。

D) 如果在一般图中最小顶点覆盖问题能在 $O(T(n))$时间内被解决，那么在一般图中最大独立集问题可以在 $O(T(n))$时间内被解决。

首先看下几个定义

二分图：设$G=(V,E)$是一个无向图，如果顶点V可分割为两个互不相交的子集(A,B)，并且图中的每条边（i，j）所关联的两个顶点i和j分别属于这两个不同的顶点集$(i\in A,j\in B)$，则称图G为一个二分图 。

看一个图示：



独立集：图中一些两两不相邻的顶点的集合，换句话说它是一个由顶点组成的集合S，使得S中任两个顶点之间没有边。

最大独立集：一个极大独立集要么是空图中所有顶点的集合，要么是一个这样的独立集，使得添加图中任一其它顶点得到的新集合都不再是独立集。

A:在二分图中，可以看到最小顶点覆盖问题即为找出上图中的U, V，取两者中点数量较少的集合；二分图中的最大独立集问题也同样为找到上图中的U, V，所以A正确

B,D一起考虑，下面证明在一般的图中，最小顶点覆盖问题与最大独立子集等价。

先证明顶点最小顶点覆盖问题可以推出最大独立子集问题。

假设已经求出最小顶点覆盖问题的集合S，记其余点的集合为V，下面证明V为最大独立集。首先V中任意两点必然没有相连的边，因为如果有的边的话，那么这条边没有与S中的点相连，这就与S为最小顶点覆盖子集矛盾，因此V为独立子集。接着证明V为最大独立子集，给V任意加一点$u \in S$，我们将说明u与V中的点必然有边相邻。使用反证法，如果u与V中任意一点都不相邻，那么u必然与S中其他点相连，那么$S-u$必然也为顶点覆盖子集，且元素个数更小，这就与S为最小顶点覆盖子集的定义相矛盾，所以u与V必然有邻边，因此$u \bigcup S$不是独立集，由u的任意性我们知道，S为最大独立子集。

接着证明最大独立子集问题可以推出最小顶点覆盖问题。

假设已经求出最大独立子集问题的集合S，记其余点的集合为V，下面将说明V为最小覆盖问题的解。首先V必然为覆盖，使用反证法。如果V不是覆盖，那么必然有一条边的两个点均不属于集合V，所以这两个点均属于S，这与S为独立子集矛盾。其次证明V为最小覆盖，依旧使用反证法，假设存在元素更少的集合V_1为最小覆盖，那么由之前证明的结论可知$S \bigcup V - V_1$必然为最大独立子集，且这个集合的元素个数大于S的元素个数，这就与S为最大独立子集矛盾。

综上B,D均正确，所以这题选C

选择题5

下列哪项为真?

A) 考虑一个TSP实例，其中每个边的成本是该地点两点之间的欧几里德距离（就像编程任务＃5中一样）。删除一个顶点及其所有入射边缘不能增加最佳（即边长总和的最小值）总和的成本。

B) 考虑一个TSP实例，其中每个边缘成本都是负的。视频讲座中介绍的动态规划算法可能无法正确计算此实例的最优路径（即最小边长度总和）。

C) 考虑一个TSP实例，其中每个边缘成本都是负的。删除一个顶点及其所有入射边缘不能增加最佳（即最小的边长）总和的成本。

D) 考虑一个TSP实例，其中每个边的成本为1或2.然后可以在多项式时间内计算出最优旅程。

A:设删除的点为q,假设在原来问题中q之前访问的节点为p，之后访问的节点为r，那么由三角不等式（因为欧几里得距离）$d(p,r) \leq d(p,q) + d(q,r)$，所以删除之后的路径必然变短，因此A正确。

B:错误，查阅老师给的算法可以看出对边的正负没有限制。

C:画一个三角形，边分别标记为$-1,-2,-3$，那么最小值为-5，现在删除与$-2,-3$相连的点，那么删除后的结果为-1，所以结果增大了，因此C也错误。

D:根据老师的算法，边的成本为1或2并不影响算法，所以错误。

（备注：本笔记内容均来自coursera stanford算法专项课程的内容总结）