研究生算法课课堂笔记

上课日期：2014 年 10 月 21 日　　第(1)节课

组长：应元翔　　组员：叶唐陟

组员：魏芳芸　　组员：张黎哲正

注意：请提交 Word 格式文档。

一．作业中的问题

第一题

第一题主要存在以下几个问题

1.没有明确说明稳定匹配中输入和输出的 preference：

输入线的 preference 是其上输出线路靠近上游的顺序

输出线的 preference 是其上输入线路靠近下游的顺序

2.没有给出正确性的说明

正确性的证明：

假设在输入 i 经过输出 o 并且还未输出，并且形成一个 junction,则存在输入 i’在输出 o 上输出，并且在 i 的上游。但是此时输出 o 会选择与输入 i 绑定因为输入 i 在下游的地方，并且输入 i 也更愿意与输出 o 绑定因为 o 在其更上游的位置。因此(i,o)会绑定在一起,从而 junction 可以消除。

3.错误的证明：

3.1 贪心法解题是可行的，但是这题需要回溯（比如说能向前滑动的活动安排问题[见课堂笔记 0929 第二节课]:每个活动有明确的 Deadline，能提前执行）

3.2 纯粹的启发式算法（输入线和输出线随机匹配，如果有冲突就调整）。但要保证解是正确的需要全局调整，难以保证复杂度是 $O(n^2)$，因为内层循环不一定能在常数时间复杂度内实现。

第一题参考答案(来自洪申达同学和叶唐陟同学的作业)

首先做如下假设：

- 对于每条输出线，它的接线盒 preference list 由下游至上游依次递减，即最下游的接线盒为 favorite
- 对于每条输入线，它的接线盒 preference list 由上游至下游依次递减，即最上游的接线盒为 favorite

假设在输入 i 经过输出 o 并且还未输出，并且形成一个 Junction,则存在输入 i’在输出 o 上输出，并且在 i 的上游。但是此时输出 o 会选择与输入 i 绑定因为输入 i 在下游的地方，并且输入 i 也更愿意与输出 o 绑定因为 o 在更上游的位置。因此(i,o)会绑定在一起,从而 junction 可以消除。

仿照稳定匹配问题的算法，如下：

```
初始所有的输入线和输出线都未配对
While 存在输入线 i 是未输出的且还没有与任何输出线相连接
    选择这样一个输出线 i
    令 o 是 i 的 preference list 中未被输出且排名最高的线
    If o 是未被输出的 then
        (i, o)配对，即 i 从 o 输出
    Else o 当前与 i’配对
        If i’的排名比 i 高 then
        i 保持未输出
        Else i 的排名比 i’高
            (i, o)配对
            i’变成未输出状态
        Endif
    Endif
Endwhile
```

第二题

第二题主要有以下几点需要注意

1. 有的同学答案存在误差，注意编程语言中数据类型的精度问题。 如 C++语言基本数据类型的精度如下表所示。

类型名	说 明	字 节	示数范围，精度
char	字符型	1	-128 ～ 127
signed char	有符号字符型	1	-128 ～ 127
unsigned char	无符号字符型	1	0 ～ 255
short [int]	短整型	2	-32768 ～ 32767
signed short [int]	有符号短整型	2	-32768 ～ 32767
unsigned short [int]	无符号短整型	2	0 ～ 65535
int	整型	4	-2147483648 ～ 2147483647
signed [int]	有符号整型	4	-2147483648 ～ 2147483647
unsigned [int]	无符号整型	4	0 ～ 4294967295
long [int]	长整型	4	-2147483648 ～ 2147483647
signed long [int]	有符号长整型	4	-2147483648 ～ 2147483647
unsigned long [int]	无符号长整型	4	0 ～ 4294967295
float	单精度浮点型	4	-3.4 × 10 38 ～ 3.4 × 10 38 ，约 6 位有效数字
double	双精度浮点型	8	-1.7 × 10 308 ～ 1.7 × 10 308 ，约 12 位有效数字
long double	长双精度浮点型	8	-3.4 × 10 4932 ～ 1.1 × 10 4932 ，约 15 位有效数字

2. d)小题可通过二分法 / 牛顿逼近解决。伪代码见第二周笔记

3.n^2 和 $100n^2$ 处理数据的能力只差 10 倍。说明算法复杂度较高，那么对其作常数级别的优化的作用就较小。计算机 CPU 的发展对指数级别的复杂度算法没有太大帮助，由于目前 CPU 运算能力是线性增长的。

第二题答案（来自张黎哲正同学的作业）

(a) n <= 6000000
(b) n <= 33019
(c) n <= 600000
(d) n <= 906316482853
(e) n <= 45
(f) n <= 5

第三题

注意：

g1(n)不是指数复杂度，不是有 N 在指数位置就是指数复杂度

$n^{loglogn}$是指数级别，但是增长很慢，因为对数本来就是一个增长很慢的函数。

扩展：几个 N 在指数位置但非指数复杂度的函数

1) $n^{\frac{1}{\log n}}$

$$n^{\frac{1}{\log n}} = 2^{\log n^{\frac{1}{\log n}}} = 2^{\log n \times \frac{1}{\log n}} = 2$$

因此 $O(n^{\frac{1}{\log n}}) = O(1)$

2) $2^{\log n}, \sqrt{2}^{\log n}, 4^{\log n}$

它们都是$k^{\log n^a}$形式

$$2^{\log k^{\log n^a}} = 2^{\log k \log n^a} = 2^{a \log n \log k} = n^{a \log k}$$

因此$O(k^{\log n^a}) = O(n^{a \log k})$

对于$2^{\log n}$中 k=2，a=1，带入得$O\left(2^{\log n}\right) = O(n)$

类似的$O\left(\sqrt{2}^{\log n}\right) = O(\sqrt{n}), O\left(4^{\log n}\right) = O(n^2)$

注：这里的log()函数的底为 2

3) $n^{\log \log n}$

$$O(n^a) < O\left(n^{\log \log n}\right) < O(a^n)$$

第三题参考答案（来自应元翔同学的作业）

g1(n) < g4(n) < g3(n) < g5(n) < g2(n) < g7(n) < g6(n)

证明过程：

$$由 \lim_{n\to\infty}\frac{\log(\log 2^{2^n})}{\log(\log 2^{n^2})}=\lim_{n\to\infty}\frac{n}{2\log n}=\infty$$

得增长率 $2^{n^2} \leq 2^{2^n}$

$$由 \lim_{n\to\infty}\frac{\log 2^{n^2}}{\log 2^n}=\lim_{n\to\infty}\frac{n^2}{n}=\infty$$

得增长率 $2^n \leq 2^{n^2}$

$$由 \lim_{n\to\infty}\frac{\log 2^n}{\log n^{\log n}}=\lim_{n\to\infty}\frac{n}{(\log n)^2}=\infty$$

得增长率 $n^{\log n} \leq 2^n$

$$由 \lim_{n\to\infty}\frac{\log n^{\log n}}{\log n^{4/3}}=\lim_{n\to\infty}\frac{\log n}{4/3}=\infty$$

得增长率 $n^{4/3} \leq n^{\log n}$

$$由 \lim_{n\to\infty}\frac{n^{4/3}}{n(\log n)^3}=\lim_{n\to\infty}\frac{n^{1/3}}{(\log n)^3}=\infty$$

得增长率 $2^{\sqrt{\log n}} \leq n^{4/3}$

$$由 \lim_{n\to\infty}\frac{\log n(\log n)^3}{\log 2^{\sqrt{\log n}}}=\lim_{n\to\infty}\frac{\log n+3\log\log n}{\log 2\cdot\sqrt{\log n}}=\infty$$

综上所述，这些函数按增长率上升排列顺序为：

$$2^{\sqrt{\log n}} \leq n(\log n)^3 \leq n^{4/3} \leq n^{\log n} \leq 2^n \leq 2^{n^2} \leq 2^{2^n}$$

第四题

O(g(n)),不一定比 f(n)大，可能小常数倍，(a)取 log 相当于压缩了常数倍的差距，仍然成立，(b)放大了让它不成立。

常见错误过程示例：

$$f(n) \leq cg(n)$$

$$2^{f(n)} \leq 2^{cg(n)}$$

$$2^{f(n)} \leq 2^c * 2^{g(n)} \rightarrow 此处出现错误$$

错误的原因在于$2^c * 2^{g(n)} = 2^{c+g(n)} \neq 2^{c*g(n)}$

第四题答案（来自张黎哲正同学的作业）

(a) $\$N_0 \hat{I}\ N, c>0,$使得$n \hat{I}\ N, n^3\ N_0, 0 £ \frac{f(n)}{g(n)} <= c$

$\frac{\log_2 f(n)}{\log_2 g(n)} <= \frac{\log_2 c * g(n)}{\log_2 g(n)} = \frac{\log_2 c}{\log_2 g(n)} + 1$ 需要找到$c_2 > 0$ 使得 $\frac{\log_2 c}{\log_2 g(n)} + 1 £ c_2$

$\log_2 f(n) Î O(\log_2 g(n))$

根据要求随 n 增加 g(n)会变得比任意常数大，所以等式左边趋向于 1。所以能找到 C_2，使得 $\log_2 f(n) Î O(\log_2 g(n))$

(b) $f(n) = 2n, g(n) = n, 2^{2n} \notin O(2^n)$

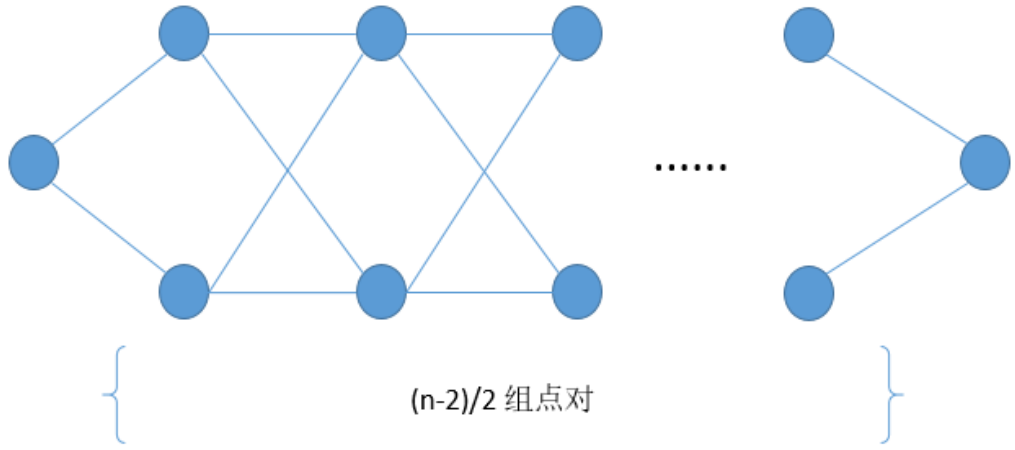
(c) $\frac{f(n)^2}{g(n)^2} = (\frac{f(n)}{g(n)})^2 <= c^2$ $f(n)^2 Î O(g(n)^2)$

第五题

常见错误：

1. 有同学通过 BFS 枚举最短路径数。这样得到的路径数一定不会超过终点的入度，而事实上最短路径数可以是节点数量的指数级，因为不同的路径可以有相同的边。

2. 第一次用 BFS 计算出最短路径长度，第二遍用 DFS 求路径数量。在 DFS 中，若遇到搜过顶点不继续则得到的结果不正确，若 DFS 枚举所有路径复杂度可能太大，比如我们可以构造 n 个点和 2n 条边的图如下图所示，使得 v-w 之间的最短路径数为2^n。



第五题答案（来自叶唐陟同学作业）

set all node's distance = infinite

```
Array count[nodes] initialed with 0
Start from node v
Count[v] = 1;
Enqueue v into queue and set v's distance = 0;
While queue is not empty
    dequeue a node n from queue
    For all n's neighbors m
        If m's distance == infinite
            Enqueue m
            Set m's distance = n's distance + 1
            Count[m] += count[n]
        Else if m's distance == n's distance + 1
            Count[m] += count[n]
        Endif
    Endfor
Endwhile
```

This algorithm is a complete bfs. The time complexity is always O(m + n)

第二次作业

第一题

中文版描述有问题

描述有问题 t_j 的可疑转账联系起来，那么：$|t_j - x_i| \leq e_j$

第五题

$A_i, A_{i+1}, \dots, A_j$，矩阵加括号方法使得运算次数最少

m[i,j]表示最优加括号的方法情况下的运算数

A_i的维度是$P_{i-1} * P_j$

A_j的维度是$P_{j-1} * P_j$

A_k的维度是$P_{k-1} * P_k$

$$m[i,j] = \min_{i \leq k < j}\{m[i,k] + m[k+1,j] + P_{i-1}P_kP_j\}$$

题意：

第一问是选择括号的位置，保证最后一次乘法的计算复杂度最低，举个反例

第二问是矩阵链中两两相乘地判断，每次找到乘法代价最小的去乘。

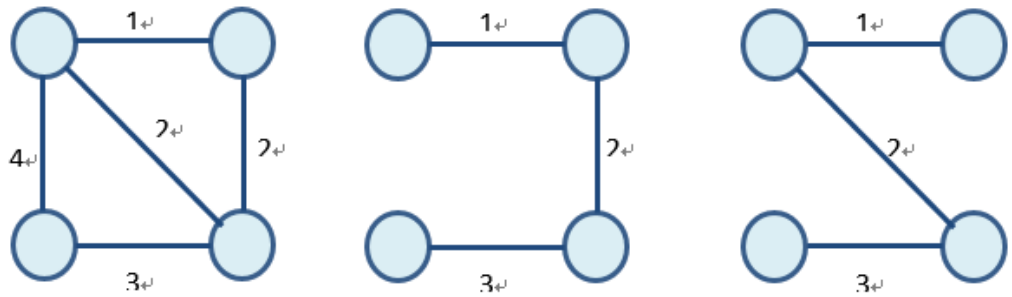
上节课遗留问题讨论

贪心算法：

对于某个割，割上最轻边一定在最小生成树(MST)上。如果不是最轻的边，也可能在 MST 上，因为有可能是另一个割的最轻边。

如果已经有一棵 MST，去掉一条边也能成为一个割。割集中不是最轻的边也可能在 MST 上， 但前提是边的权重不能是唯一的。

例子：左侧为原图，中间为某棵 MST，去掉树上权重为 1 的边，这样得到的 cut 中最轻的边是唯一的，对角线那条边的权重严格比它大，但是也可能出现在某个 MST（右图）上。



区间覆盖问题：

题目：许多区间，选择区间的子集使得任意一个未选中区间都和子集中至少一个区间有交集。

之前的算法：按结束时间排序，对结束时间最早的区间，找一个与其相交而且结束时间最晚的。再对剩下未覆盖区间重复上述过程。

结果集 R

1.先按结束时间排序。

2.选取结束时间最早的区间 A，然后选择覆盖 A 的所有区间中结束时间最晚的区间 S。

3.把 S 加入集合 R，去除 S 覆盖的所有区间，重复 1，直到去除所有区间。

反例

根据算法，取到的集合为{c,d,e,f}，而正确答案应为{a,b}

注意事项

1.第二次作业开始用新的模板。

2.记得用订书机把作业订起来。

3.15(X), X 代表助教名字缩写。W, R, NY, L, M, C 同理。分别是谁以及联系方式见第一讲课件。

4.题目的源代码或伪代码是可选项，写清算法思路、分析复杂度就可以了。