（装订线内不要答题）

复旦大学计算机科学与技术学院

2019 ~ 2020 学年第二学期期末考试试卷

■A 卷　□ B 卷　□ C 卷

课程名称：算法设计与分析　课程代码：COMP130011.02

开课院系：计算机科学技术学院　考试形式：线上考试（开卷）

姓名：________ 学号：________ 专业：________

我已知悉学校与考试相关的纪律以及违反纪律的后果，并将严守纪律，不作弊，不抄袭，独立答题。

学生（签名）：

题号	1	2	3	4	5	6	7	8	9	10	总分
得分											

（以下为试卷正文或课程论文题目）

一、判断下列断言的真假，如果是假的话，请简要给出理由（14%）

1. 在一个二叉搜索树中，寻找比其中任意一个元素小的最大元素需要花费 $O(1)$ 时间
2. 稳定婚姻问题的最优稳定婚姻集合总是唯一的。
3. 给定一个算法，对于相同的输入，一定会得到相同的输出。
4. 选择排序、冒泡排序、快速排序和堆排序中有三个或以上是不稳定的
5. 对于包含负权重边的图，Kruskal，Prim 和 Dijkstra 三个算法中有两个或以上会失效。
6. P 问题是 NP 问题的一个真子集
7. 如果 Ford-Fulkerson 算法在某一轮迭代中将边 (u,v) 上的流量置为 1，那么在后面的迭代中

   (u,v) 的流量至少为 1

二、请回答以下问题（16%）

1. 按照下列函数的增长次数对他们进行排序（由低到高） (4%)

   $2^{\sqrt{\log_2 n}}$ $n^{1/3}$ $n(\log_2 n)^3$ $n^{\log_2 n}$ $\log_2 n$ $n^2 \log_2 n$

2. 给定一个有向图 G=(V,E)，基于邻接链表表示，给出下面几个常用图算法的时间复杂度 (5%)

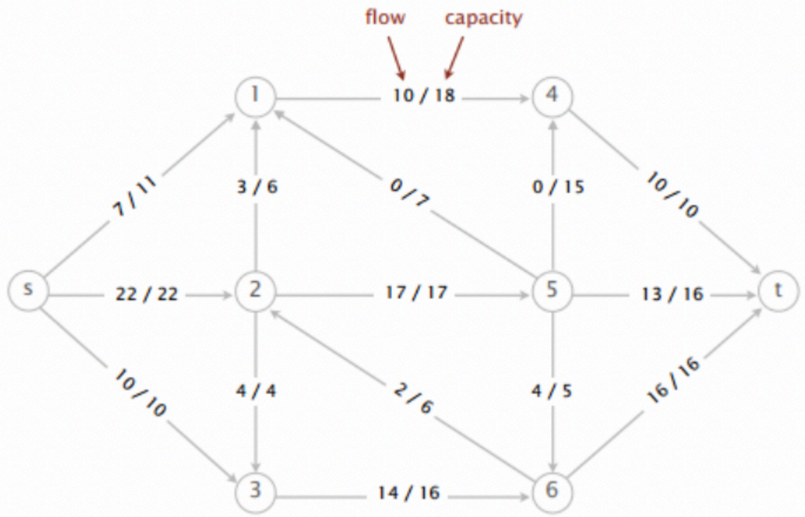
   DFS， Bellman-Ford， Dijkstra，Floyd-Warshall， Johnson

3. 给出下面几个递推关系的时间复杂度 （3%）

   a. $T(n) = 8T(n/2) + n^4$ b. $T(n) = T(n/4) + T(3n/4) + n$

4. 用 $Big-\theta$ 给出递推式 $T(n) = \log n + T(\sqrt{n})$ 的渐进时间复杂度，并简要说明理由。(4%)

三、考虑下面的流网络中的可行流 f （10%）



（1）流 f 的值为多少？

（2）运行 Ford-Fulkerson 算法一轮后，给出增广路径（augmenting path）上节点的序列。

（3）最大流的值为多少？

（4）给出最小割的具体划分？

（5）最小割的容量是多少？

四、在一个二分图中，所有顶点可以分为两个不相交的集合，每条边都连接两个集合中各一个顶点。请设计一个有效的算法来判断一个图是否是二分图，请写出伪码，并且分析算法的时间复杂度（10%）

BFS，奇数层L，偶数层R，有矛盾就不行

五、假设 $G=(V,E)$ 是一个无向连通带权图，$w_{\min}$ 和 $w_{\max}$ 代表图中最小边和最大边的权重。请注意图中边权重可能为负值，也可能不唯一。判断以下说法的正确性，并简要给出理由或反例。（10%）

（1）如果 G 有超过 $|V|-1$ 条边，并且有唯一一条边的权重是 $w_{\max}$，那么这条边不会在 G 的最小生成树中出现。

（2）任意一个权重是 $w_{\min}$ 的边肯定出现在 G 的某个最小生成树中。

(1)显然，到达孤岛唯一一条路也得选
(2)等边三角形Kruskal

（3）如果 G 有一条回路，并且 e 是这条回路中权重最小的唯一的一条边，那么 e 必须出现在 G 的所有最小生成树中。

（4）如果 e 没有出现在 G 的任何最小生成树中，那么 e 肯定是 G 中某个回路中的权重最大的边。

（5）假设边的权重非负，那么两个顶点间的最短路径肯定包含在某些最小生成树中。

六、求 n 个数中第 k 大的数。设计有效算法，写出伪代码，并分析复杂度。（10%）

七、已知哈密尔顿回路问题是 $NP-Complete$，证明哈密尔顿路径问题也是 $NP-Complete$ （10%）

来个源点s和汇点t，分别有到所有点和所有点到的有向边。（无论原图是否有向都可以规约到有向图）

八、为找零问题设计一个尽可能有效的算法：给定金额 n 以及各种面额 $d_1, d_2, \cdots, d_m$ 的数量无限的硬币，求总金额等于 n 的硬币的最少个数，或者指出该问题无解。请写出伪代码，并分析算法的时间复杂度。（10%）

九、

假设给你一个整数集合 $A=\{a_1,a_2,\cdots,a_n\}$ 和一个整数 B. 如果子集 $S\subseteq A$ 中的整数之和不超过 B，即

$$\sum_{a_i \in S} a_i \leqslant B$$

则称 S 是可行的. S 中的整数之和称做 S 的总和.

你可能希望选择 A 的一个总和尽可能大的子集 S.

例如，设 $A=\{8,2,4\}$，$B=11$，那么最优解是子集 $S=\{8,2\}$.

(a) 下面是这个问题的一个算法.

```
开始时 S = ∅
令 T=0
For  i=1,2,…,n
  If  T+a_i ≤ B  then
      S ← S∪{a_i}
      T ← T + a_i
  Endif
Endfor
```

给出一个实例使得对这个实例，该算法返回的集合 S 的总和小于 A 的另一个可行子集的总和的一半.

(b) 给出这个问题具有下述保证的多项式时间近似算法：它返回一个可行集 $S \subseteq A$，其总和不小于任何可行集 $S' \subseteq A$ 的最大总和的一半. 你的算法必须有不超过 $O(n\log n)$ 的运行时间.

（10%）

从大到小排列，设第一个未被装入的为w_i，算法得到W^\star，最优解OPT。

- OPT \le B
- w_i \le W^\star且w_i+W^\star \gt B，故2W^\star \gt B
- 故W^\star \gt OPT/2