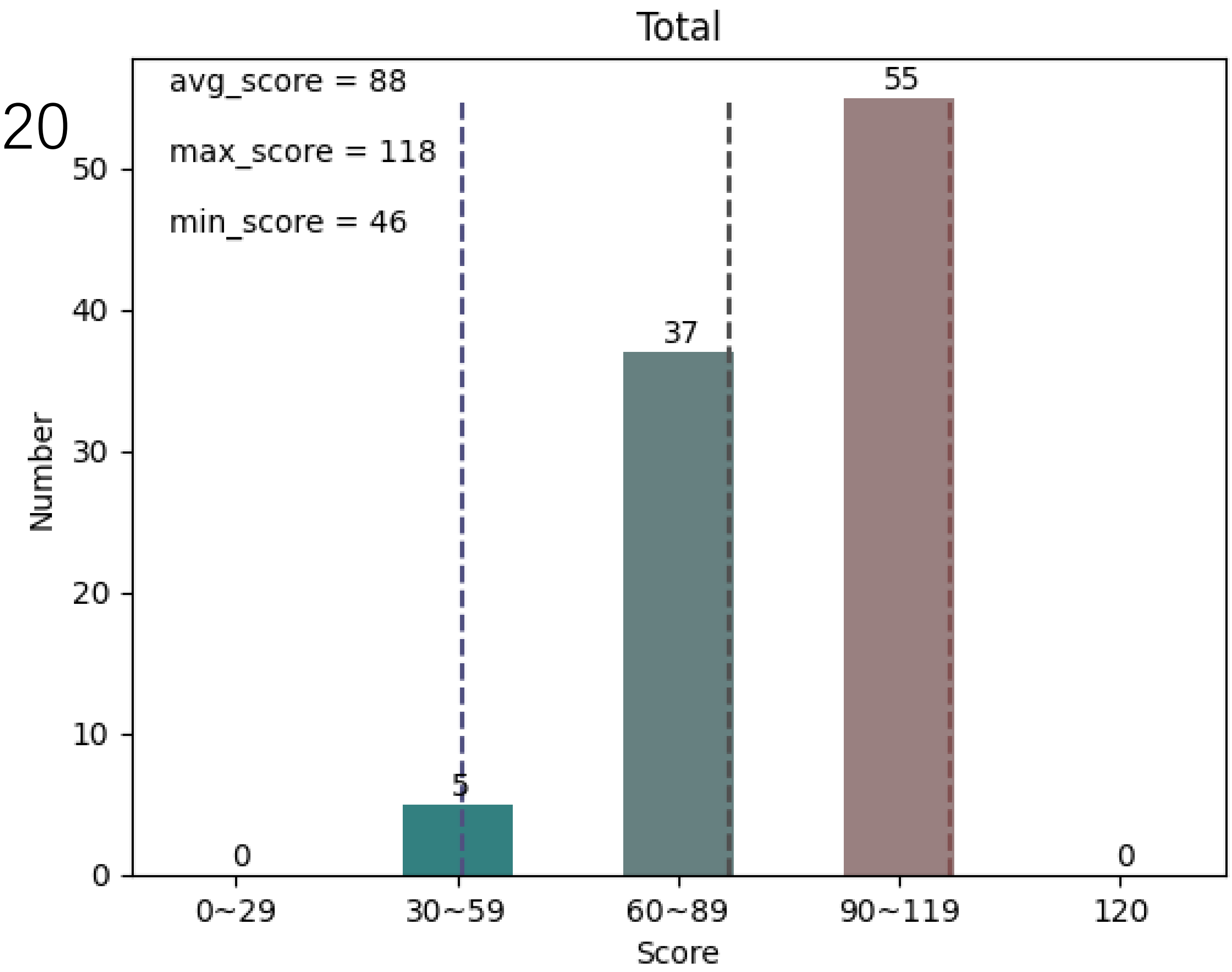
期中试卷讲解

张茂润

2022.5

基本情况

- 6道题，每道题20分，满分120
- 平均分：88，最高分：118



第1题

a) 请证明调和级数满足：$1+\frac{1}{2}+\cdots+\frac{1}{n}=\Theta(\log n)$。

- 利用高数知识，求极限：

a) 令$x_n=1+\frac{1}{2}+\cdots\frac{1}{n}$,$y_n=\log n$

解法 1：根据 Stolz 定理：

$$\lim_{n\to\infty}\frac{x_n}{y_n}=\lim_{n\to\infty}\frac{x_{n+1}-x_n}{y_{n+1}-y_n}=\lim_{n\to\infty}\frac{\frac{1}{n+1}}{\log\left(1+\frac{1}{n}\right)}=\lim_{n\to\infty}\frac{1}{\log\left(1+\frac{1}{n}\right)^{n+1}}=c\ (c<+\infty)$$

$$(\because \lim_{n\to\infty}\left(1+\frac{1}{n}\right)^n=e)$$

第1题

a) 请证明调和级数满足：$1+\frac{1}{2}+\cdots+\frac{1}{n}=\Theta(\log n)$。

- 缩放法（要注意**左右都要缩放**）—— 积分缩放法

一方面

$$1+\frac{1}{2}+\cdots+\frac{1}{n}<1+\int_{1}^{n}\frac{1}{x}dx=1+\ln n$$

另一方面

$$1+\frac{1}{2}+\cdots+\frac{1}{n}>\int_{1}^{n+1}\frac{1}{x}dx=\ln(n+1)$$

第1题

a) 请证明调和级数满足：$1+\frac{1}{2}+\cdots+\frac{1}{n}=\Theta(\log n)$。

- 缩放法（要注意**左右都要缩放**）—— 不等式缩放法

一方面: $\ln(1+x) \le x$

$$1+\frac{1}{2}+\cdots+\frac{1}{n} > \ln\left(\frac{2}{1}\right)+\ln\left(\frac{3}{2}\right)+\cdots+\ln\left(\frac{n+1}{n}\right)=\ln(n+1)$$

另一方面: $\frac{1}{x+1} < \ln(1+\frac{1}{x})$

$$1+\frac{1}{2}+\cdots+\frac{1}{n} < 1+\ln\left(\frac{2}{1}\right)+\cdots+\ln\left(\frac{n}{n-1}\right)=1+\ln n$$

第1题

b) 请证明：$\log n! = \Theta(n\log n)$。
（注：不可以使用 Stirling 公式。）

- 缩放法

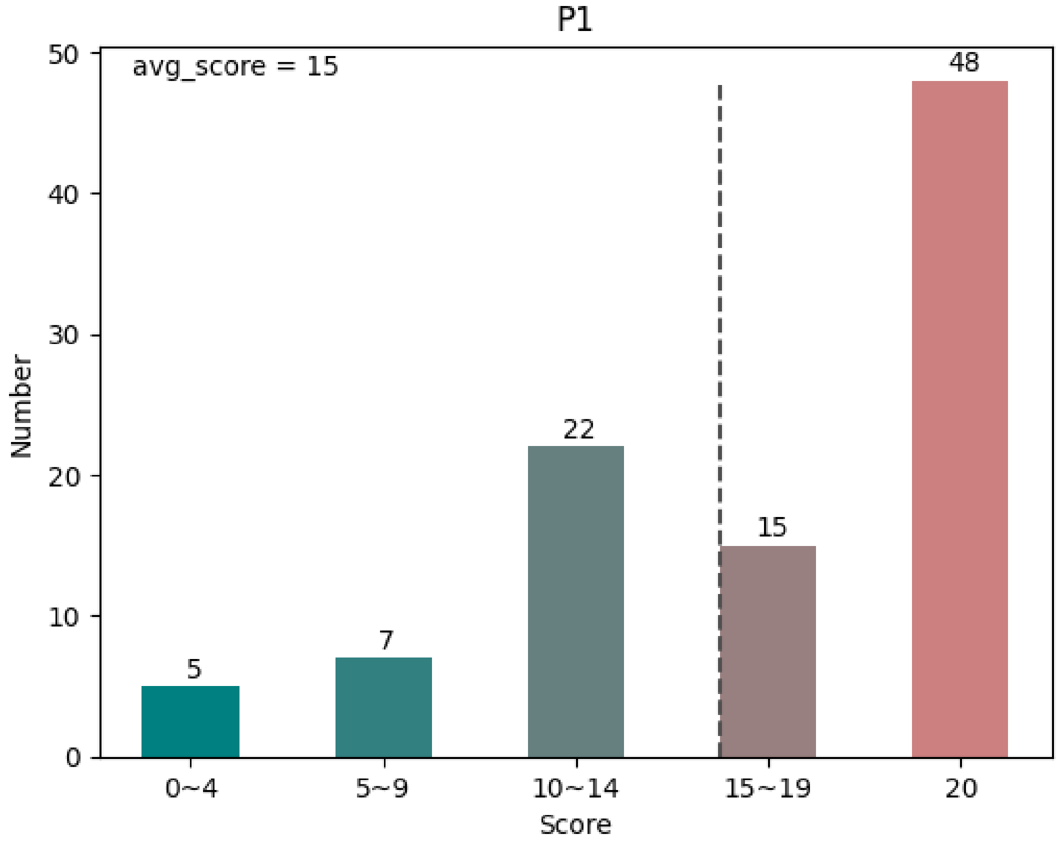
一方面: $\log n! \le \log n^n = n\log n$

另一方面:

$$\log n! = \log\left[n * (n-1) * \cdots * \left(\frac{n}{2}\right) * \cdots * 1\right] \ge \log\left[n * (n-1) * \cdots * \left(\frac{n}{2}\right)\right] \ge \log\left(\frac{n}{2}\right)^{n/2} = \frac{n}{2} * \log(\frac{n}{2})$$

第1题 主要问题

- 有些同学使用缩放法时只缩放了一边（上界/下界），就得出结论
  - 需要缩放出上界、下界，缺一不可
- 缩放出的上下界，形式不统一
  - 例如，上界缩放为 $\log n$，下界缩放为 1
- 对于级数和的形式，直接使用积分的方法求值



第2题

如果一个数组A[1..2n+1]，满足A[1] < A[2] > A[3] < A[4] > ... < A[2n] > A[2n+1]，我们称之为"蛇形"的。给定数组B[1..2n+1]，其中元素各不相同，现在需要将它变成蛇形的。你只能通过元素间的大小比较来调整数组的形态。

1) 请设计一个O(n log n)的算法。

2) 请设计一个O(n)的算法。

(注：你需要阐述算法的正确性，并分析其代价)

- 1) 思路：排序（$O(n\log n)$）+ 遍历（O(n)）
- 2) 思路：选择（O(n)）+ 遍历（O(n)）
  - 或者一趟遍历，遍历中调整元素顺序

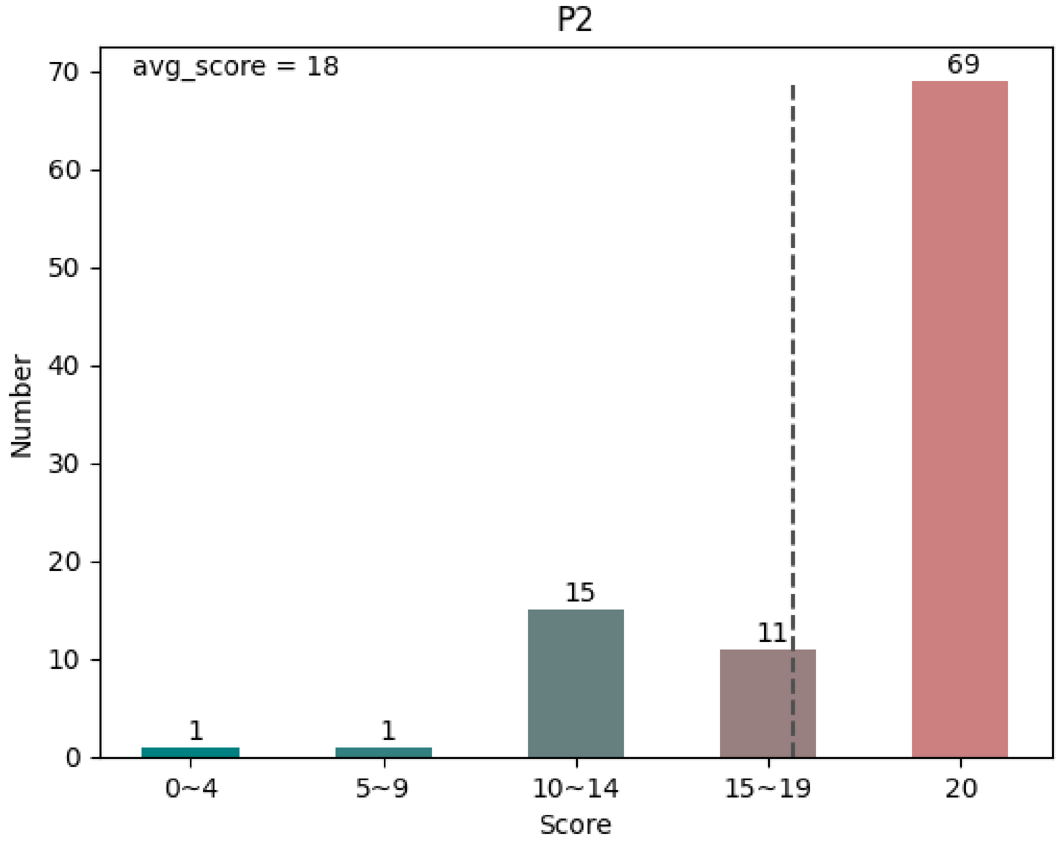
第2题

```
def select(A, p, r, k):
    if p == r:
        return A[p]
    q = Partition(A, p, r)
    x = q - p - 1
    if k == x:
        return A[q]
    elif k < x:
        return select(A, p, q-1, k)
    else:
        return select(A, q+1, r, k-x)
```

```
def snakeSort(A[1..2*n+1]):
    select(A, 1, 2*n+1, n+1)
    # 或者调用SELECT_ELINEAR
    for i in [1, n+1]:
        B[2*i-1] = A[i]
    for i in [1, n]:
        B[2*i] = A[n+1+i]
    return B
```

第2题 主要问题

- 伪代码的书写：正确 + 易读易懂
  - 因为这道题目中使用到的排序、选择的算法与课本算法功能完全一致，因此可以在答案上说明调用 xx 算法（有时间的话，手写实现最好），指出对应传入的参数内容及含义
- 阐述正确性，代价分析
  - 尤其是第2问中用遍历解决的算法，需要阐述正确性



第3题

请分析并证明：对 SELECT3 算法而言，在任何情况下总是比 m*（median-of-median）小的元素的个数不超过 $\frac{n}{3}+3$。（注：请不要忽略 n 不是 3 的倍数等边界情况。）

- 一共划分成$\lceil \frac{n}{3} \rceil$组，前$\lceil \frac{1}{2} \lceil \frac{n}{3} \rceil \rceil$组每组共享2个比m*小的数，所以不超过：

$$
\begin{aligned}
2\lceil \frac{1}{2}\lceil \frac{n}{3}\rceil\rceil &\le 2(\frac{1}{2}\lceil \frac{n}{3}\rceil + 1) \\
&= \lceil \frac{n}{3}\rceil + 2 \\
&\le (\frac{n}{3} + 1) + 2 \\
&= \frac{n}{3} + 3
\end{aligned}
$$

[这个界很宽，只需要合理证明小于$\frac{n}{3}+3$即可。]

第3题

请分析并证明算法的最坏情况时间复杂度 T(n)可以用下面的递归方程来刻画：

$$T(n) \geq T\left(\left\lceil\frac{n}{3}\right\rceil\right) + T\left(\frac{2n}{3} - 3\right) + \Omega(n)$$ 。

- 一共可以划分成$\lceil \frac{n}{3} \rceil$组对其进行递归求解m*，需要$T(\lceil \frac{n}{3} \rceil)$
- 根据a)的结果，不确定与m*关系的至少有$n - (\frac{n}{3} + 3) = \frac{2n}{3} - 3$个元素，对其递归求解，需要$T(\frac{2n}{3} - 3)$
- 还需要$\Omega(n)$的时间来分组和partition。

一共就是$T(n) \geq T(\lceil \frac{n}{3} \rceil) + T(\frac{2n}{3} - 3) + \Omega(n)$。

第3题

请用“prove by substitution”的方法证明$\mathrm{T(n)} = \Omega(n\log n)$。

- 假设$T(n) \geq cnlogn$，根据b)的结果有

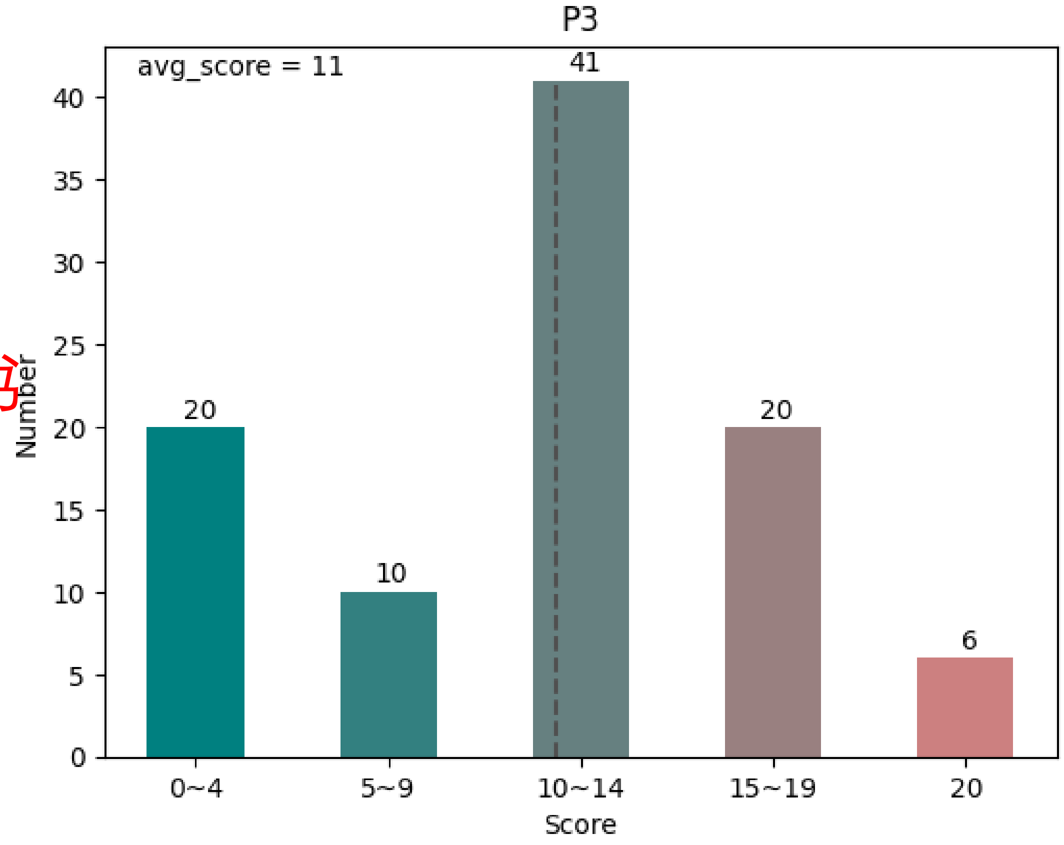
$$
\begin{aligned}
T(n) &\geq T(\lceil\frac{n}{3}\rceil) + T(\frac{2n}{3}-3) + \Omega(n) \\
&\geq c\lceil\frac{n}{3}\rceil log\lceil\frac{n}{3}\rceil + c(\frac{2n}{3}-3)log(\frac{2n}{3}-3) + c_1 n \\
&\geq c\frac{n}{3}log\frac{n}{3} + c\frac{2n}{3}log(\frac{2n}{3}-3) - 3clog(\frac{2n}{3}-3) + c_1 n \\
&= c\frac{n}{3}logn + c\frac{2n}{3}log(2n-9) - cnlog3 - 3clog(\frac{2n}{3}-3) + c_1 n \\
&\geq c\frac{n}{3}logn + c\frac{2n}{3}logn - cnlog3 - 3clog(\frac{2n}{3}-3) + c_1 n \\
&\geq cnlogn + (c_1 n - cnlog3 - 3clog(\frac{2n}{3}-3)) \\
&\geq cnlogn + (c_1 n - 2cn - 3cn) \\
&= cnlogn + (c_1 n - 5cn)
\end{aligned}
$$

此时需要$c_1 n - 5cn \geq 0$，即$c \leq \frac{c_1}{5}$，有$T(n) \geq cnlogn$，假设成立。

第3题 主要问题

- 全部空着，放弃了比较简单的1、2问
- 不等号取错，证明$T(n) = \Omega(n \log n) \Rightarrow T(n) \geq cn \log n$
- 没有替换 $\Omega(n) \geq c_1 n$
- 选定 c 的范围时，与 n 相关

- 教材中有相关的内容，复习时注意看书



第4题

给定n个各不相同的两两可比的元素$x_1, x_2, \ldots, x_n$，每个元素有正的权重值$w_1, w_2, \ldots, w_n$。记所有元素权重之和为W。定义这组元素中的1/3-median为满足下面条件的元素x_k:

$$\sum_{x_i<x_k} w_i < \frac{W}{3},\ \sum_{x_i>x_k} w_i \le \frac{2W}{3}$$

a) 请设计一个O(nlogn)的算法，找出给定元素中的1/3-median。

b) 请设计一个O(n)的算法，找出给定元素中的1/3-median。

- 作业5.10
- 思路:
  - a) 排序（$O(n\log n)$）+ 遍历（O(n)）
    - 是对 x_i 排序，而不是 w_i
  - b) 选择（O(n)）+ 遍历（O(n)）
    - 与课本的线性时间选择算法功能不完全一致，需要手写这部分伪代码

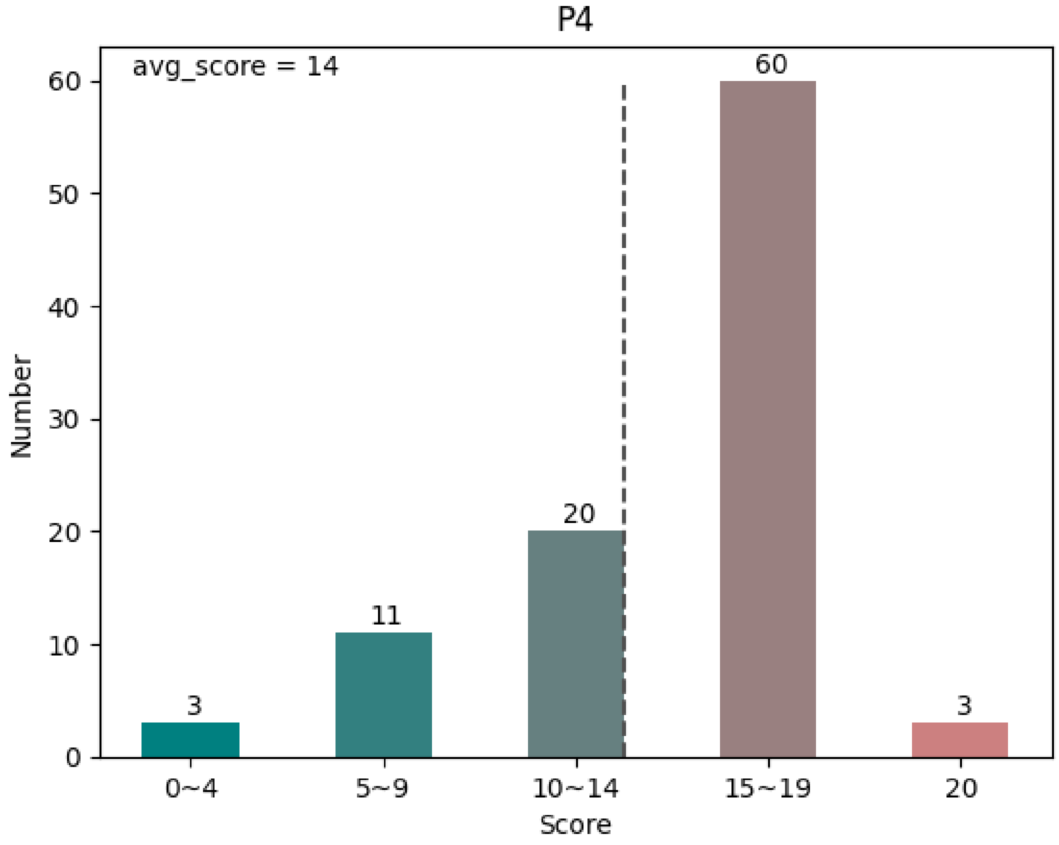
第4题

```
SOLVE(x[1..n], w[1..n], lsum, rsum, W)
if (n == 1) return x[1]
用SELECT-WLINEAR找到x[1..n]的中位数m
根据m对x[1..n]进行partition，比m小的放左边，比m大的放右边，同时wi随着xi一起移动。
l = lsum, r = rsum
for i = 1 to n/2-1 do
    l += w[i]
for i = n downto n/2+1 do
    r += w[i]
if (l < W/3 && r <= 2W/3) return m
if (l >= W/3) return SOLVE(x[1..n/2-1], w[1..n/2-1], lsum, r+w[n/2], W)
return SOLVE(x[n/2+1..n], w[n/2+1..n], l+w[n/2], rsum, W)

ALG4B(x[1..n], w[1..n])
W = 0
for i = 1 to n do
    W += w[i]
return SOLVE(x[1..n], w[1..n], 0, 0, W)
```

第4题 主要问题

- 没有写伪代码，只有思路
  - 与第2题的区别，第4题需要实现的算法功能，是与课本不完全一致的，不能简单的说明，在课本某某代码的基础上“添加 xx 内容”或“更改 xx 内容”，因此需要完整地写伪代码
- 没有时间复杂度分析
- 调用课本上已有的算法，直接调用即可
  - SELECT-WLINEAR选择中位数算法
  - partition



第5题

你在一个有 n 个代表的政治会议会场内，每个代表都隶属于一个政党，但是并不知道他们属于哪个政党。假设你直接询问一个代表，他会拒绝回答，但是你可以通过介绍两个代表认识来分辨他们是否属于同一个政党（因为同一政党的代表会礼貌地握手并给予对方微笑；不同政党的代表会怒视对方）。

a) 假设代表中的大多数（一半以上）来自同一政党（称之为主要政党）。请设计一个算法来判定每个代表是否属于这个主要政党。

b) 假设代表们来自 k 个政党，一个政党占多数当且仅当属于它的代表的数目比其他任何政党的代表都多。请设计一个算法找出一个来自占多数的政党的代表，或者返回不存在占多数的政党。

- 教材题目7.9，7.7

第5题 a)

- 思路：依次挑选一个元素，判断是不是主要政党

算法1：*朴素查找*

> ⇒ 算法思路：某个元素$A[i]$是$MainElem \iff$ 数组A中与之相等的元素个数不少于$\lfloor \frac{n}{2} \rfloor$；
>
> ⇒ 算法流程：
>
> I. 初始时令$i=1$，将$A[i]$与$A[i+1..n]$中的元素依次进行比较，并计数与$A[i]$相等的元素个数cnt；
>
> II. 若$cnt \geq \lfloor \frac{n}{2} \rfloor$，则说明$A[i]$就是$MainElem$，退出循环；
>
> III. 否则，令$i=2$，重复上述步骤直到找出$MainElem$；
>
> ⇒ 时间复杂度：易知$i \leq n - \lfloor \frac{n}{2} \rfloor = \lceil \frac{n}{2} \rceil$，否则当$i = \lceil \frac{n}{2} \rceil + 1$时，$cnt \leq n - i = \lfloor \frac{n}{2} \rfloor - 1$，$A[i..n]$都不可能是$MainElem$，故比较代价：$worstCost = \sum_{i=1}^{\lceil \frac{n}{2} \rceil}(n-i) = O(n^2)$；
>
> ⇒ 空间复杂度：易知只使用了常数额外空间，为$O(1)$；

第5题 a)

- 思路：分治法，问题的解一定至少是其中一个子问题的解

$\Rightarrow$ 算法思路：若e是数组$A[1..n]$的$MainElem$，令$A_l = A[1..\lfloor\frac{n}{2}\rfloor], A_r = A[\lfloor\frac{n}{2}\rfloor + 1..n]$，则$e$至少是$A_l$和$A_r$这两个子数组其中之一的$MainElem$，反证如下：若e既不是$A_l$也不是$A_r$的$MainElem$，则$e$在$A$中的出现次数：$cnt_e = cnt_l + cnt_r \le \lfloor\frac{n_l}{2}\rfloor + \lfloor\frac{n_r}{2}\rfloor = \lfloor\frac{\lfloor\frac{n}{2}\rfloor}{2}\rfloor + \lfloor\frac{\lceil\frac{n}{2}\rceil}{2}\rfloor \le \lfloor\frac{\lfloor\frac{n}{2}\rfloor + \lceil\frac{n}{2}\rceil}{2}\rfloor = \lfloor\frac{n}{2}\rfloor$

故e亦不是A的$MainElem$，矛盾；

$\Rightarrow$ 算法流程：

I. 递归找到子问题A_l和A_r中的$MainElem$并返回其索引，分别记为i_l和i_r(若不存在则$i = 0$)；

II. 令$cnt = 0, i = i_l$，若$i > 0$，则将$A[i]$与数组A中其他元素进行比较，并计数与$A[i]$相等的元素个数cnt，继续执行III；否则直接执行III；

III. 若$cnt \ge \lfloor\frac{n}{2}\rfloor$，则$A[i_l]$就是$A$的$MainElem$；否则令$i = i_r$再执行一遍II，若$cnt \ge \lfloor\frac{n}{2}\rfloor$，则$A[i_r]$就是$A$的$MainElem$，返回$i$；否则返回0；

第5题 a)

- 思路：一趟遍历，记录当前主要政党的“相对人数” 和 代表人物

⇒ 算法思路：假设现在所有政党代表因政见不和开始”打架“，每次打架都是1v1的单挑，同政党的代表不会打架，且不同政党的代表打架总是”两败俱伤“后被抬出会场，则对于”主要政党“，最坏情况就是所有其他政党都联合起来攻击自己的代表，但是由于自己的人数超过一半，因此最终还站在会场的一定是”主要政党“的代表（若其他政党彼此还要互相打架，只会让”主要政党“的代表所剩人数更多）；

因此，根据上述形象的思路，我们可以对数组A的遍历也模拟出一个”打架“过程：维护一个计数器cnt和索引idx，表示遍历途中，$A[idx]$所属政党目前带上了cnt个兄弟来打架，依次将$A[i]$和$A[i+1]$进行比较：

若相等则说明$A[i+1]$也是该政党的兄弟，带上一起 $\Rightarrow cnt = cnt + 1$；若不等则说明$A[i+1]$是敌对政党，则派一个人去跟他打架并抬出会场，此时人数减少$\Rightarrow cnt = cnt - 1$；

若此时$cnt = 0$，说明该政党目前带上的人已经全部牺牲，那么$A[i+2]$所属政党将”捡漏“成为目前还有人站着的政党$\Rightarrow cnt = 1, idx = i + 2$；

遍历完整个数组后，idx和cnt记录的就是：最终$A[idx]$所属政党，即”主要政党“，还剩cnt个人站着；

第5题 b)

- 思路：统计每个政党人数，取出人数最多的政党
- 如何统计？考虑到我们只能比较两个人是不是来自同一政党，因此这个问题可以抽象化为 **并查集** 的问题：计算每个并查集的大小（判断是否为主要政党）以及并查集的代表元（政党的代表人物）

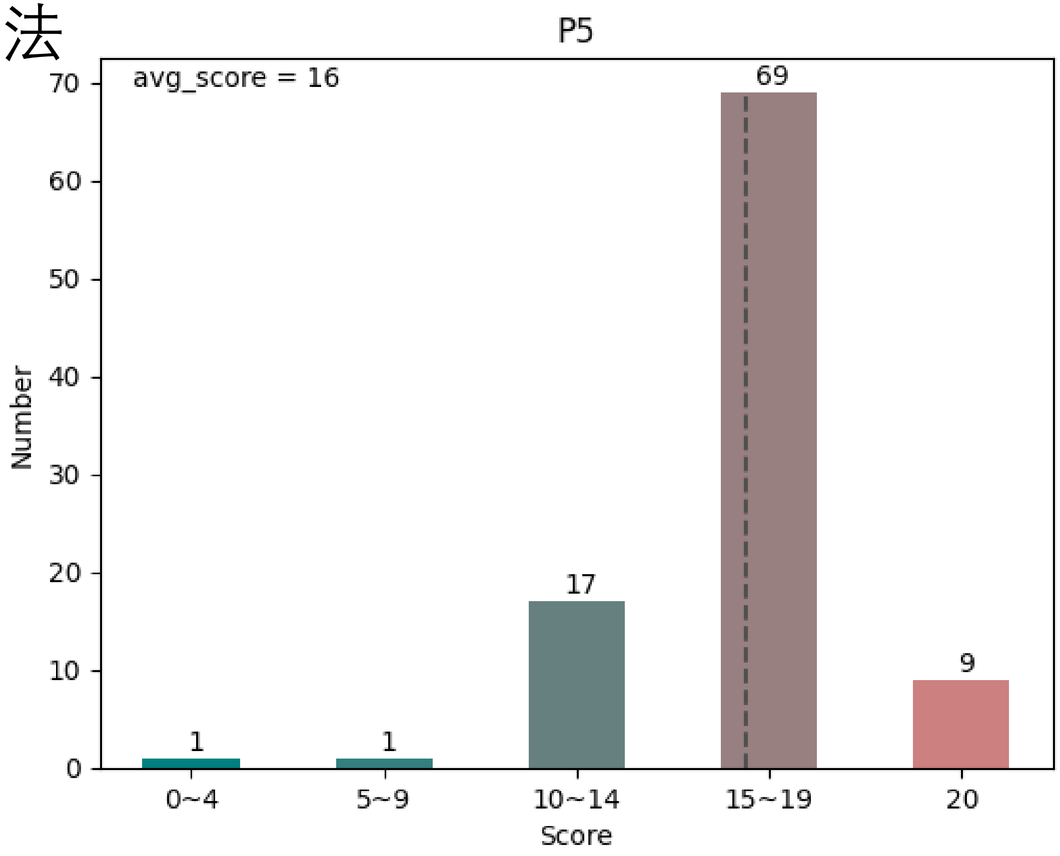
第5题 b)

- 基于数组的方式实现并查集：
  - 初始化： $E[i] = -1 \ (i \in [1, n])$
  - 含义：
    - $E[i] < 0$, 说明 i 是该并查集的代表元，其并查集元素数为 $|E[i]|$
    - $E[i] > 0$, 说明 元素 i 的代表元是 E[i]
  - 类似**筛法求素数**，遍历 $E[1] - E[n]$

```
for i := 1 to n do:
    if E[i] < 0:
        for j := i+1 to n do:
            if E[j] < 0 and party(i, j):
                E[j] = i
                E[i] -= 1
        represents.add(i, -E[i])
```

第5题 主要问题

- 因为题目中没有限制时间复杂度，约60%同学暴力求解
  - 若有时间复杂度限制，应提出符合要求的算法
  - 若无时间复杂度限制，尽量选择最优的算法
- 伪代码书写中，约20%同学纯文字



第6题

给定一个二维比特数组，它有n行k列，存放了所有可能的k比特串，仅仅有某一个k比特串被剔除，所以我们有$n=2^k-1$。例如图中k=3，n=7，唯一缺失的比特串是101。现在我们需要计算出缺失的那个k比特串，所能做的关键操作是"检查数组的某一位是0还是1"。

1) 请设计一个O(nk)的算法，找出缺失的比特串。
2) 请设计一个O(n)的算法，找出缺失的比特串。

(注：你需要阐述算法的正确性，并分析其代价)

1	1	0
1	1	1
0	0	1
0	1	0
0	0	0
0	1	1
1	0	0

1）如果是2^n个比特串全部出现，则每个位置上的0,1数目一定相等，且为2^{n-1}，因此统计每个比特位上0,1数目就可以知道，缺失的串在该比特位上是0，还是1 => 进阶做法：对该比特位上的所有元素做**异或操作**

时间复杂度：$O(nk)$

第6题

2)　　第1）问中，我们对每列的所有比特全部做了异或操作，因此时间复杂度为$O(nk)$；但实际上，我们只需要对其中一半的元素做异或操作：

　　不妨假设，我们对第 1 列所有元素做异或操作后，结果为 0，那么我们对第 2 列元素操作时，只需要对第 1 列中对应元素为 0 的那部分元素进行操作。

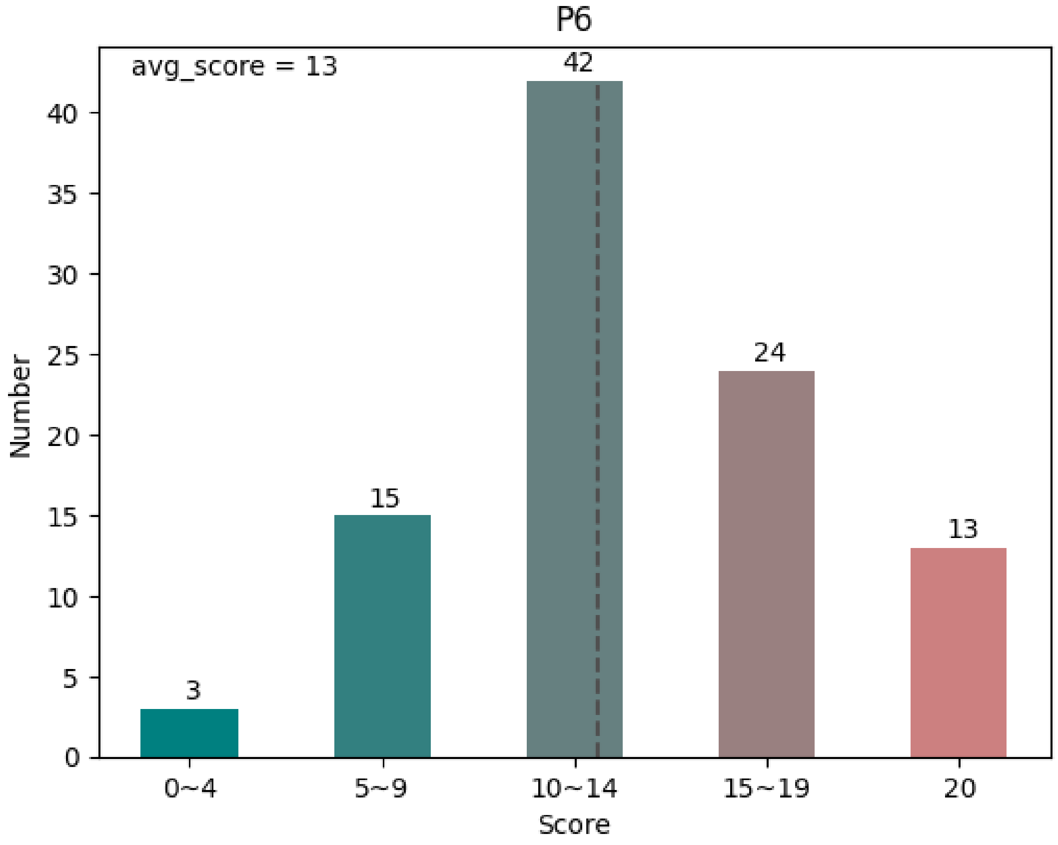
　　因此，我们每次操作的规模减半，有：

$$T(n) = T\left(\frac{n}{2}\right) + O(n)$$

$$\therefore T(n) = O(n)$$

第6题 主要问题

- 约15%同学没有做，或空了第2问
- 第二问中，部分同学仍使用比特串转数字的方式，时间复杂度不满足要求
- 伪代码书写中，约30%同学是纯文字



Q&A