



# ICS1WG24 闭门研讨会

李知非



# 几大任务

- 课程部分

1. 梳理知识点
2. 制作 slides
3. 整理题库
4. 批改作业

- 实验部分

1. 准备习题课
2. 设计实验
3. 验收实验

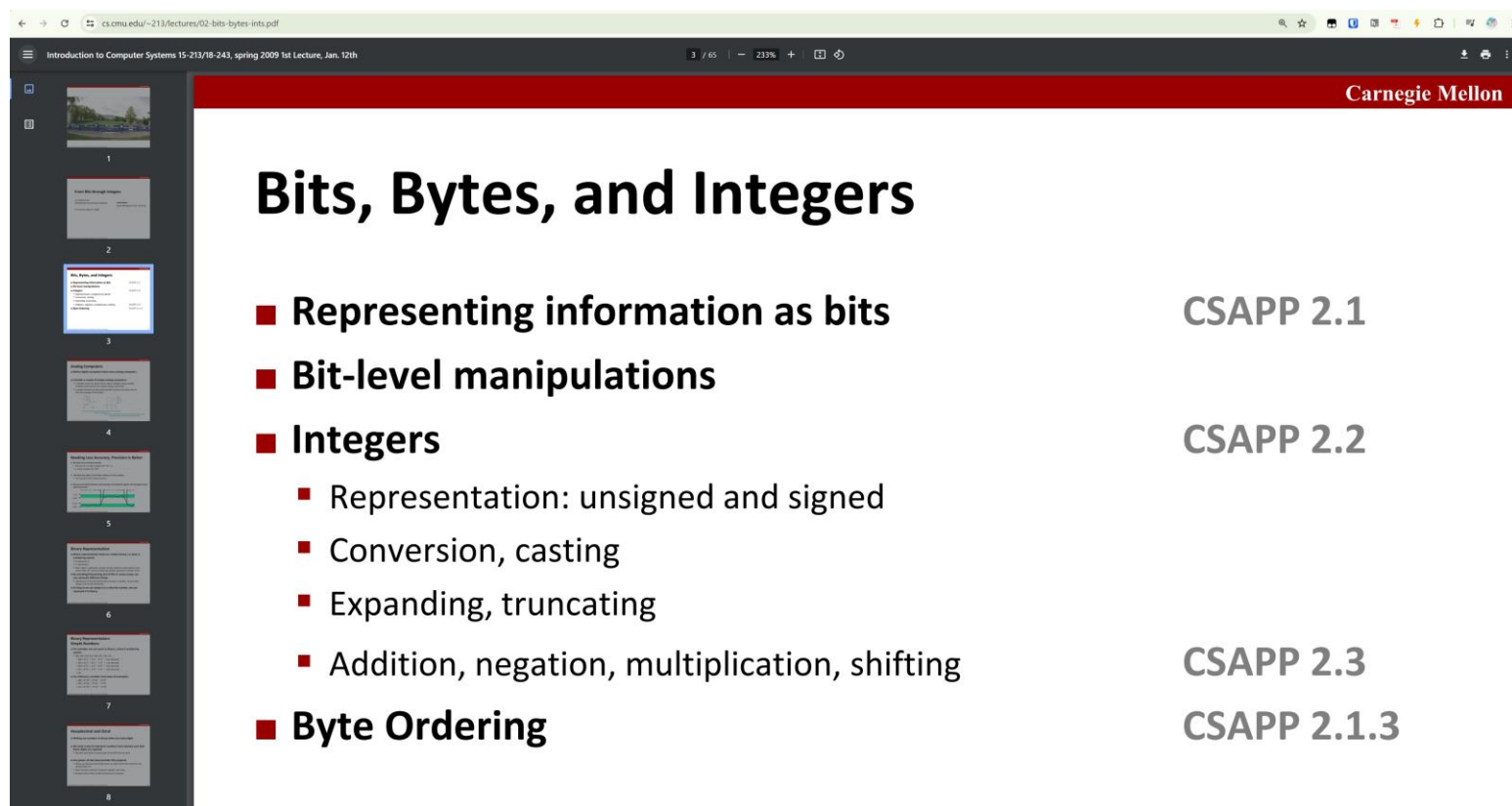
课程部分

# 梳理知识点

- 简短、精炼 10mins、10页 slides 以内
- 独立、自成体系
- 逻辑清晰，从基础到高级、从简单到复杂

# 梳理知识点

- 参考
  - [CMU的slides](#) 粗粒度



Introduction to Computer Systems 15-213/18-243, spring 2009 1st Lecture, Jan. 12th

Carnegie Mellon

## Bits, Bytes, and Integers

■ Representing information as bits	CSAPP 2.1
■ Bit-level manipulations	
■ Integers	CSAPP 2.2
■ Representation: unsigned and signed	
■ Conversion, casting	
■ Expanding, truncating	
■ Addition, negation, multiplication, shifting	CSAPP 2.3
■ Byte Ordering	CSAPP 2.1.3

# 梳理知识点

- 参考

- [CMU的slides](#) 粗粒度
- [MOOC 南大袁春风](#) 每段略长

## 第二周 数据的表示和存储

### ● 第1讲 数制和编码

- [视频](#) 1. 10进制数和2进制数 (19分钟)
- [视频](#) 2. 2/8/10/16进制数之间的转换 (20分钟)
- [文档](#) 数制和编码

### ○ 第2讲 定点数的编码表示

- [视频](#) 1. 原码和移码表示 (10分钟)
- [视频](#) 2. 模运算系统和补码表示 (17分钟)
- [视频](#) 3. 补码和真值的对应关系 (19分钟)
- [文档](#) 定点数的编码表示

### ○ 第3讲 C语言中的整数

- [视频](#) 1. 无符号整数和带符号整数 (15分钟)
- [视频](#) 2. C语言程序中整数举例 (16分钟)
- [文档](#) C语言中的整数

### ○ 第4讲 浮点数的编码表示

- [视频](#) 1. 浮点数的表示范围 (17分钟)
- [视频](#) 2. IEEE 754中规格化数的表示 (19分钟)
- [视频](#) 3. IEEE 754中特殊数的表示 (15分钟)
- [文档](#) 浮点数的编码表示

# 梳理知识点

- 参考
  - [CMU的slides](#) 粗粒度
  - [MOOC 南大袁春风](#) 每段略长
  - 陆春林 MOOC

# 制作 slides

- 先拆分知识点，没问题了再做slides
- 中文语境
  - 但术语为**统一的英文**
- 参考
  - Our slides 来自于上交对 CMU slides 的修改版
  - 南大 自成体系 独立发展 《计算机系统基础（第二版）》
  - CMU slides' update



# 整理题库

- 多收集好题
- 需要适当更新下作业题

# 批改作业

- 及时反馈

# 实验部分

# 原则

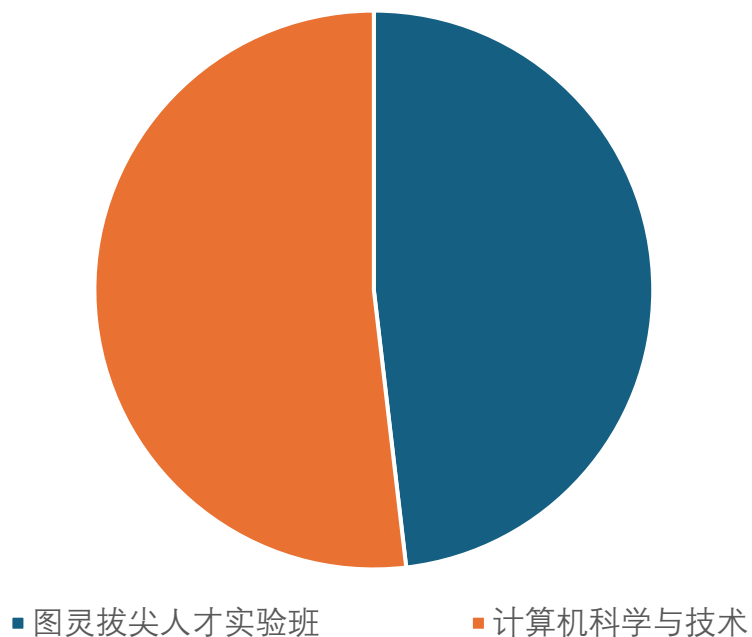
- 实践环节很重要
  - 计算机学科的工科性质
  - 纸上得来终觉浅，菜就多练
- 要让认真完成Lab的同学获得相对较高的分数
- 不一定是“难”，要区分“真难”和“真懒”
  - 前者 我们要铺台阶搭梯子
  - 后者 我们要坚决亮剑 勇于维护公平性和纯洁性

# 问卷调查

感谢 @andylizf @xsx 设计并完成调查

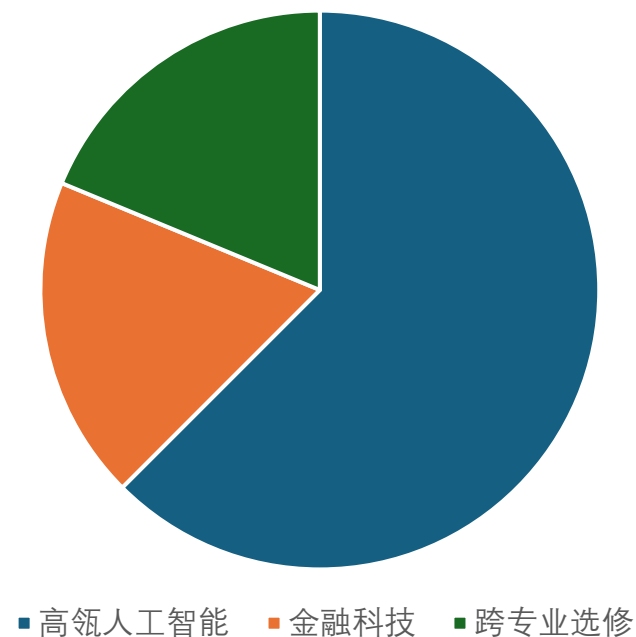
柴云鹏老师班 (样本量 27人)

选课人数



王晶老师班 (样本量 16人)

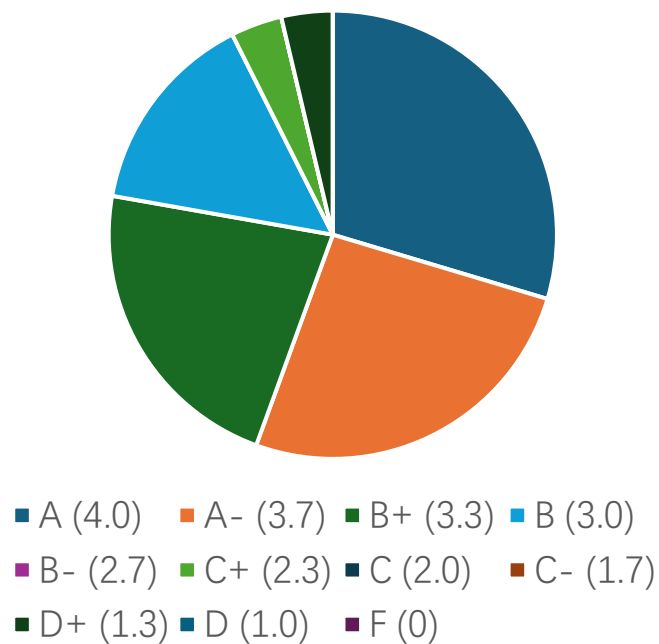
选课人数



# 问卷调查

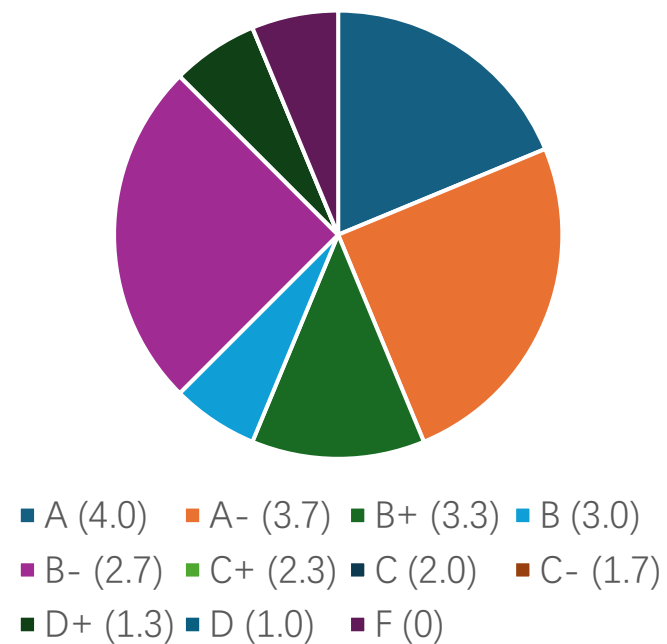
柴云鹏老师班

学分绩点



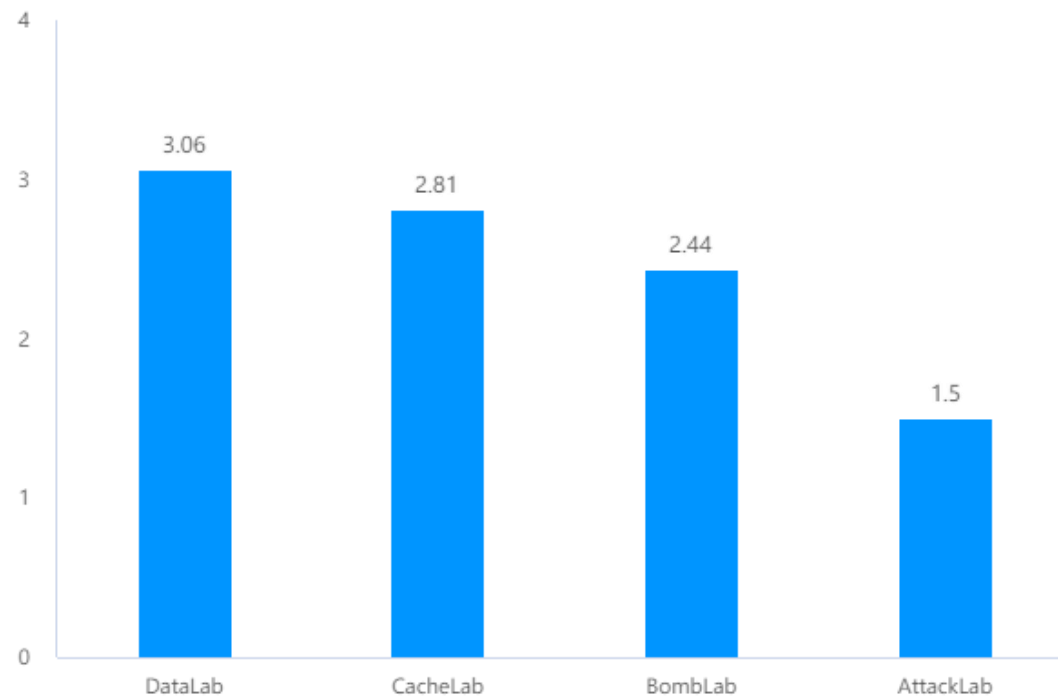
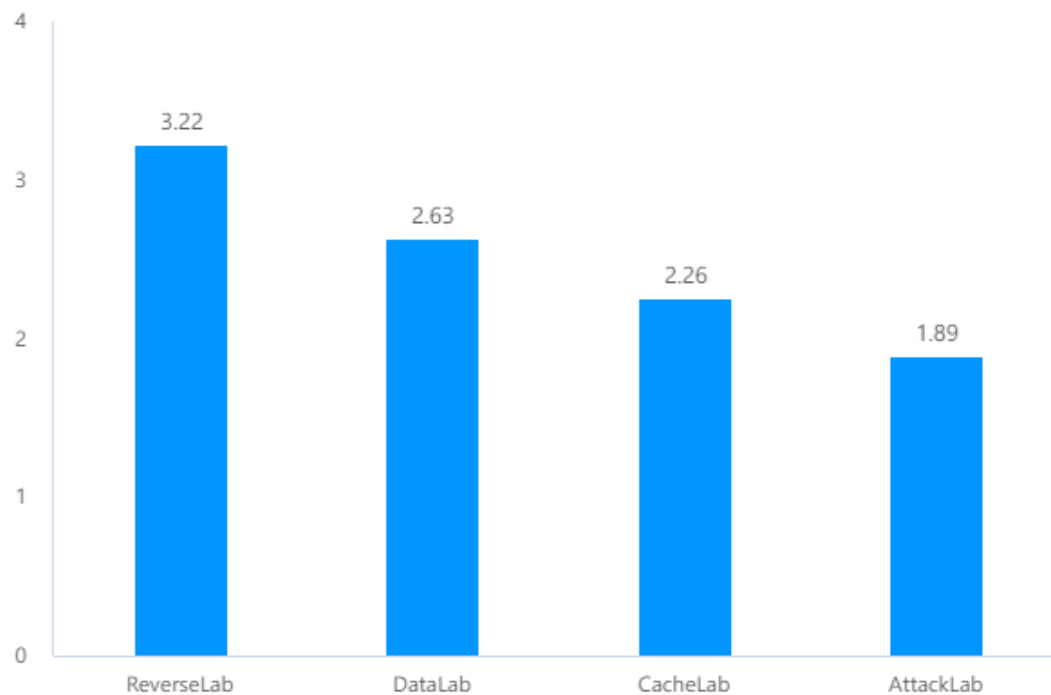
王晶老师班

学分绩点



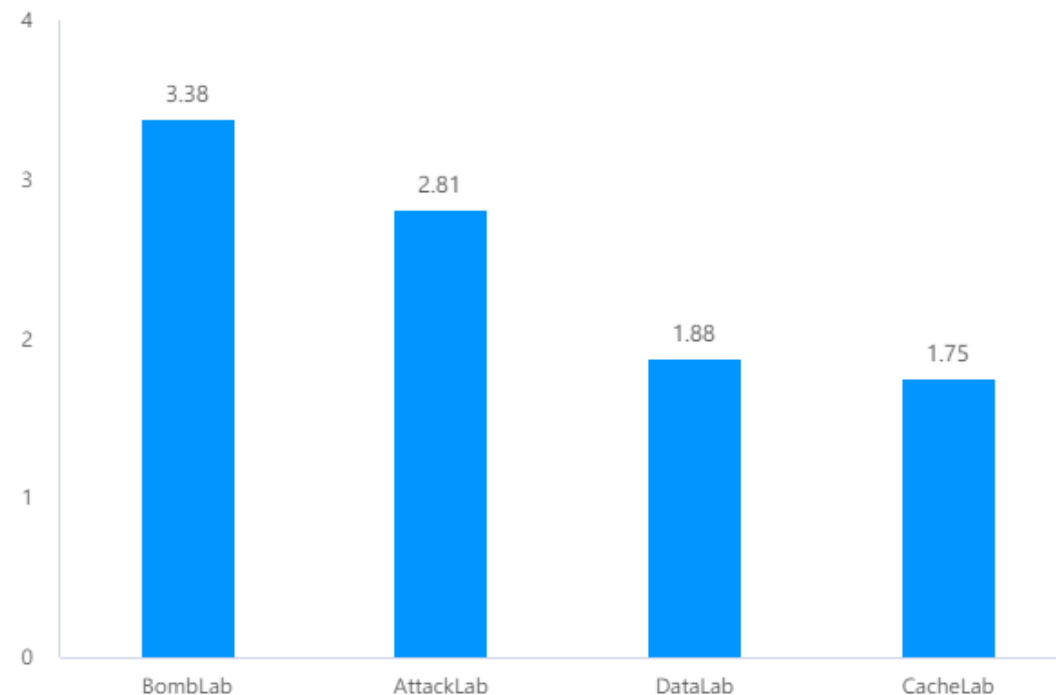
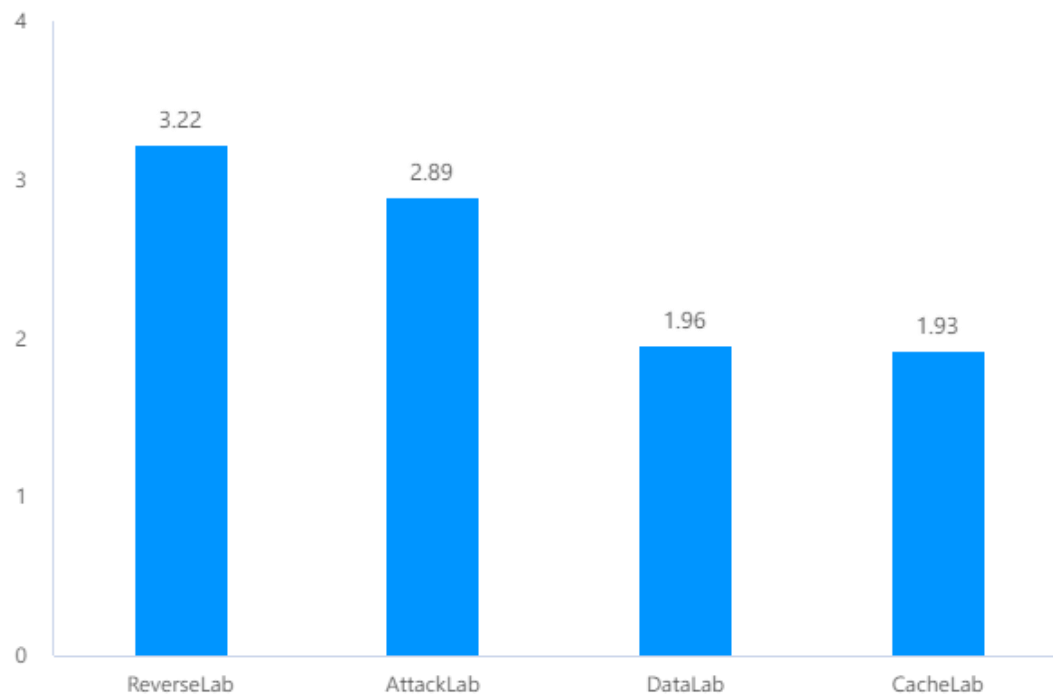
# 问卷调查

你认为这些Labs的挑战性该如何排序呢？



# 问卷调查

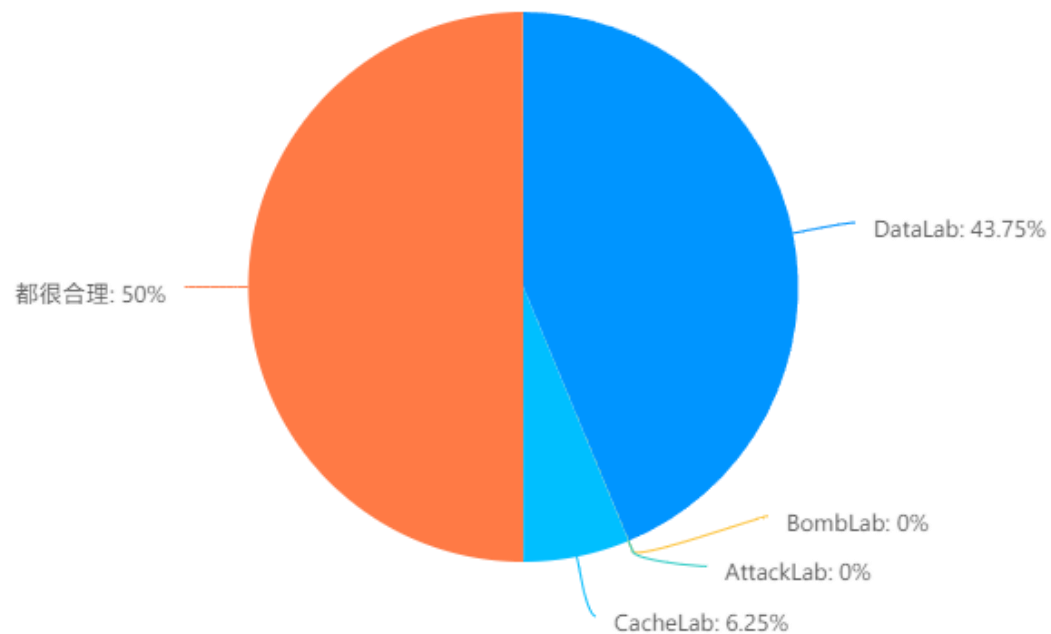
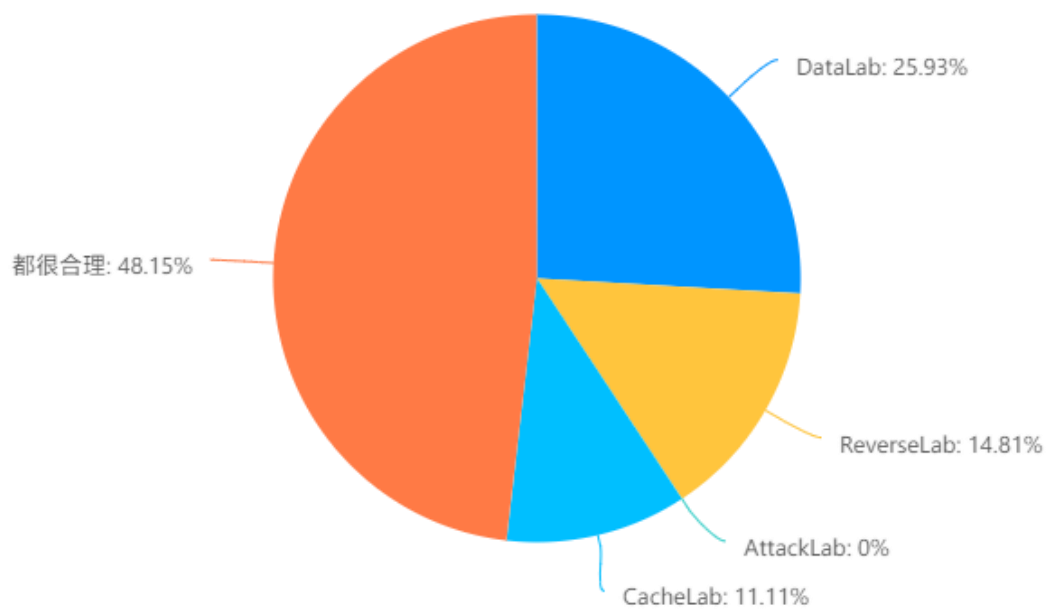
你认为这些Labs的有趣程度该如何排序呢？





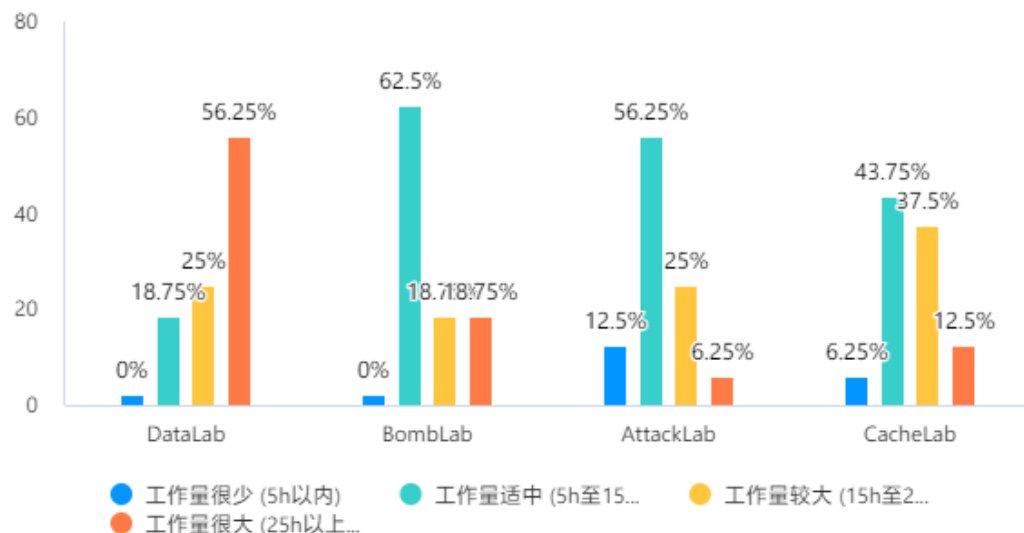
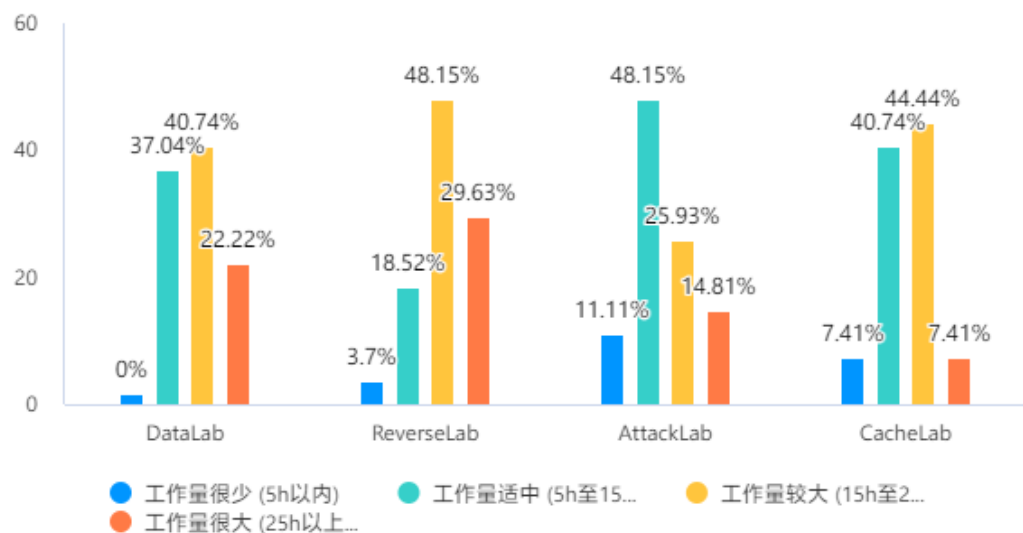
# 问卷调查

你认为哪个Lab设计得最不合理，或对你来说最无意义呢？



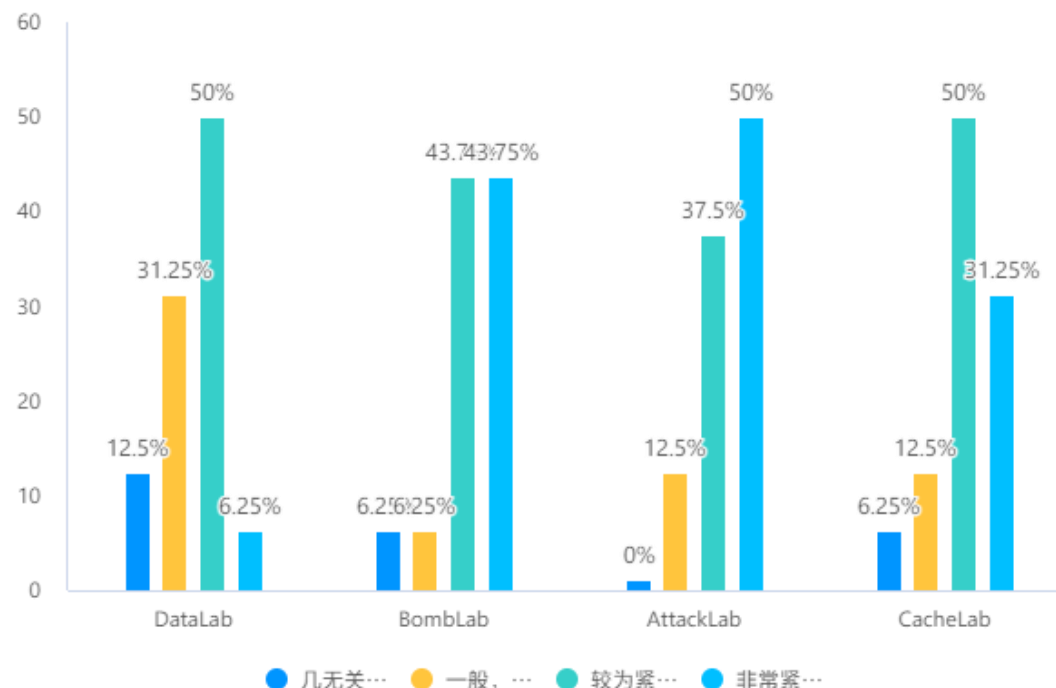
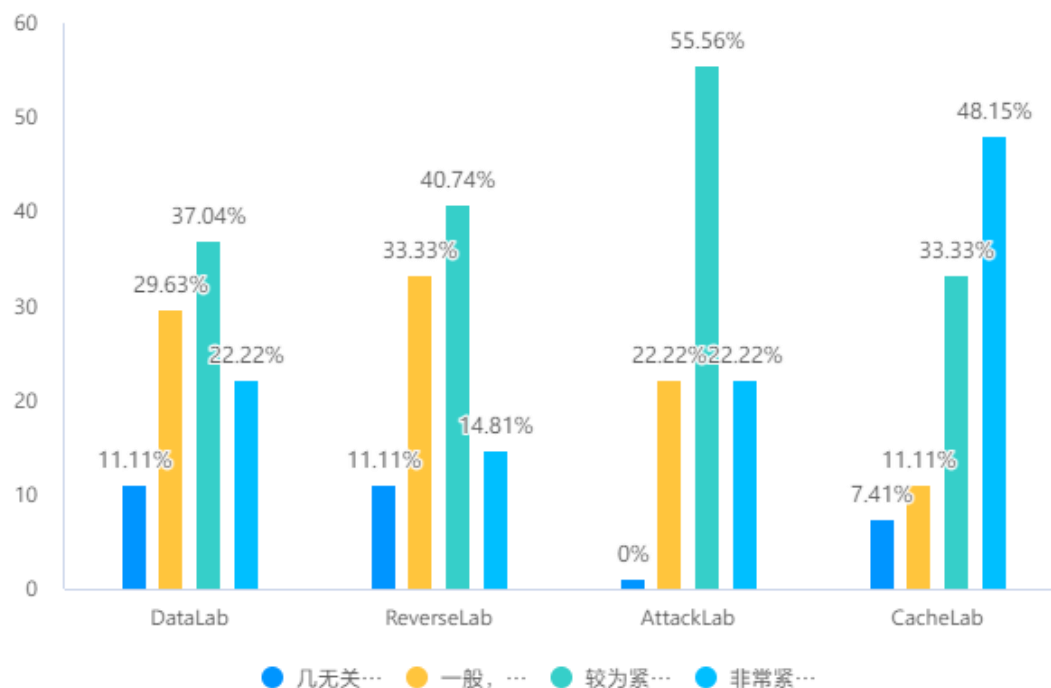
# 问卷调查

你认为各个Labs的工作量如何？



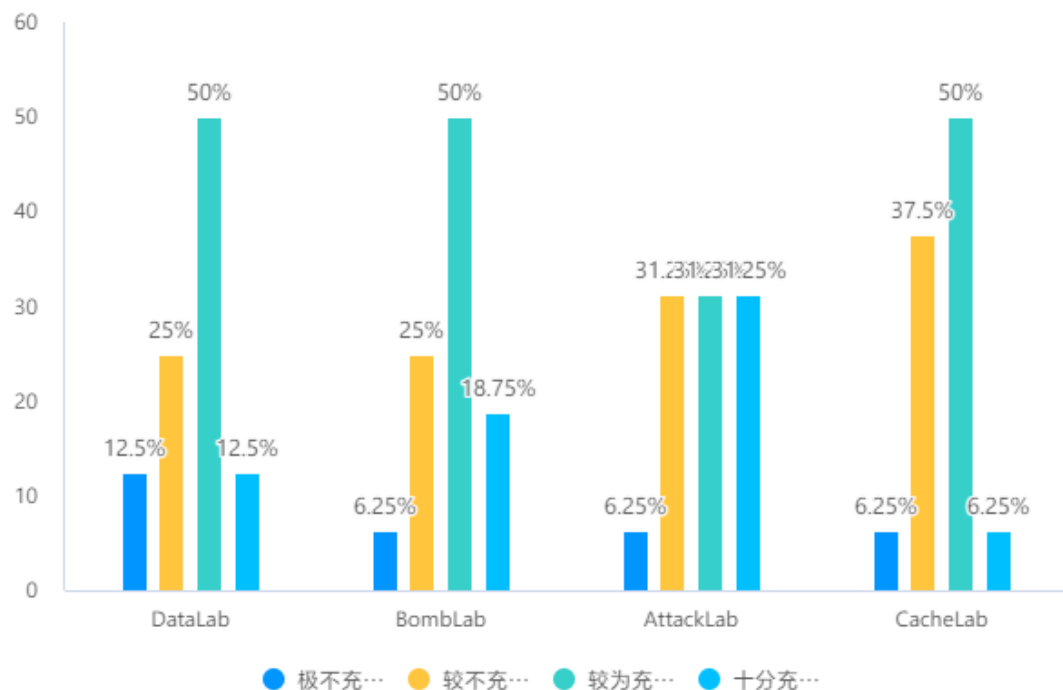
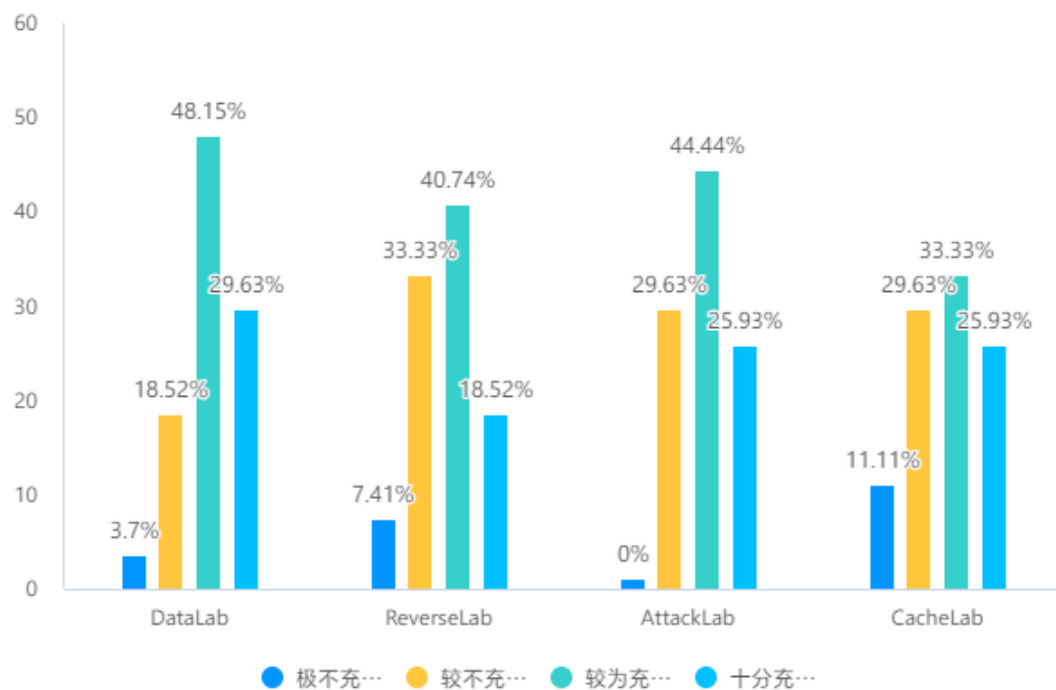
# 问卷调查

你认为各个Labs与课程内容的衔接度如何？



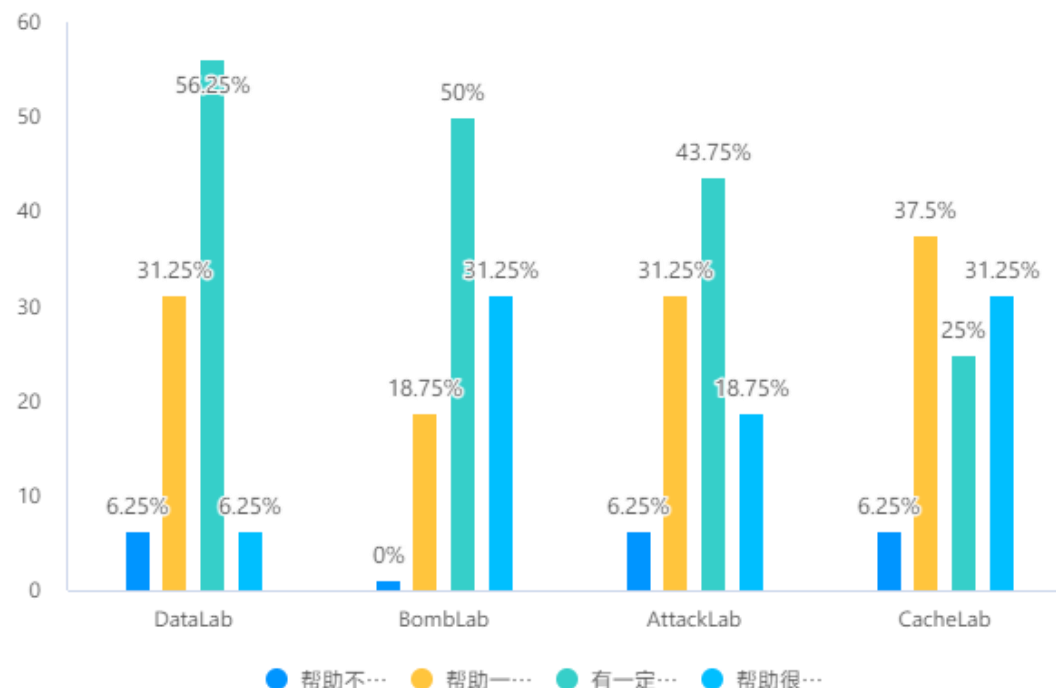
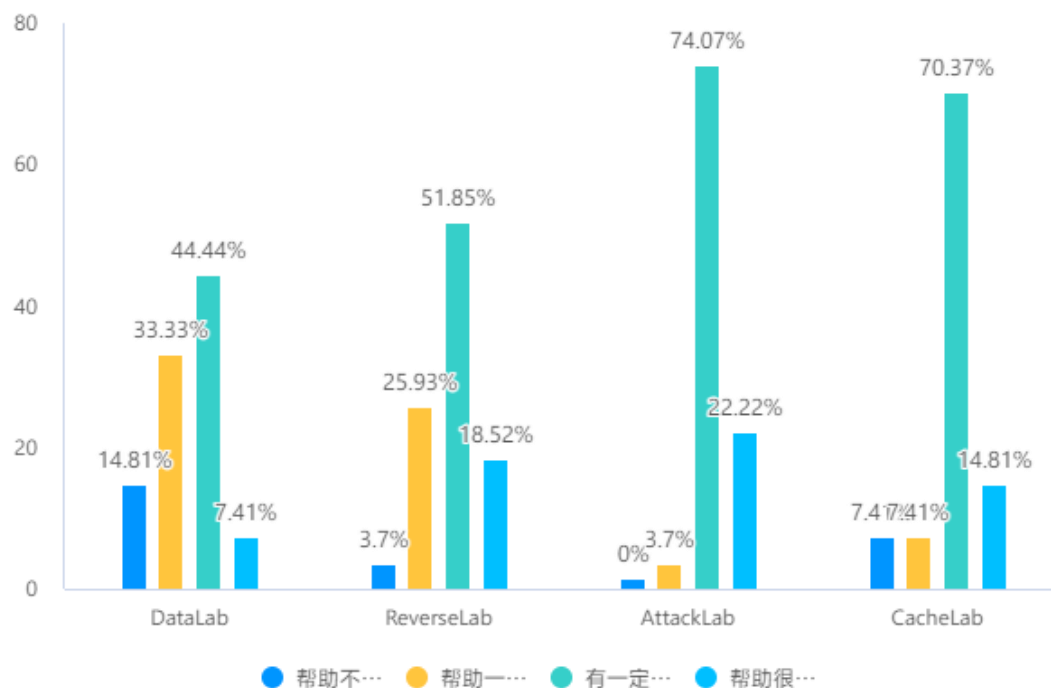
# 问卷调查

你认为各个Labs的引导是否充分？



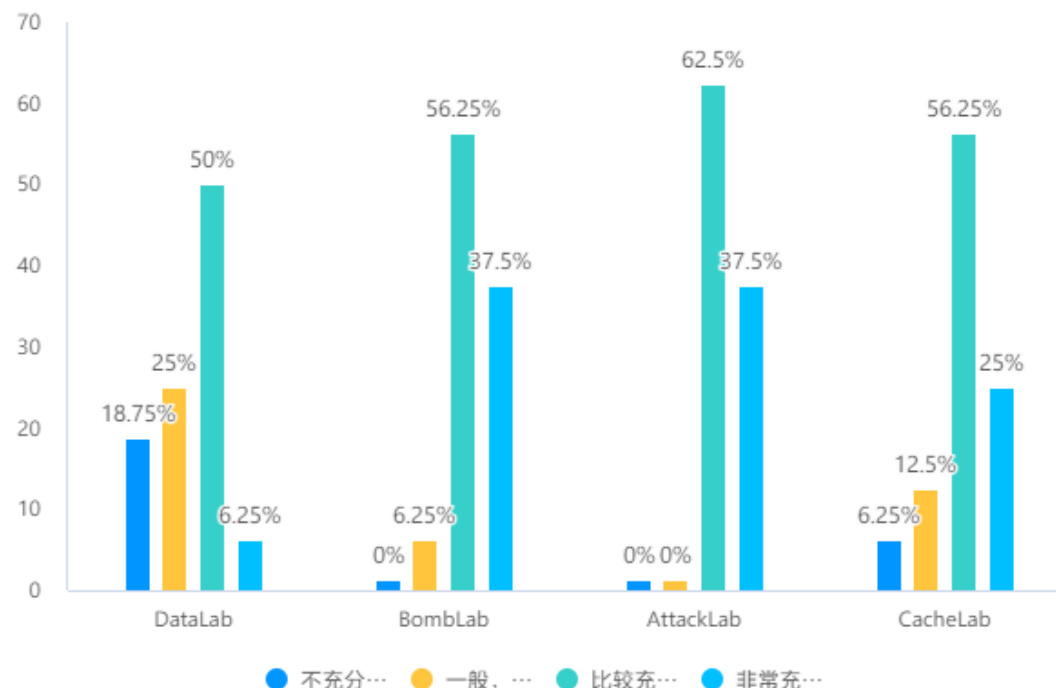
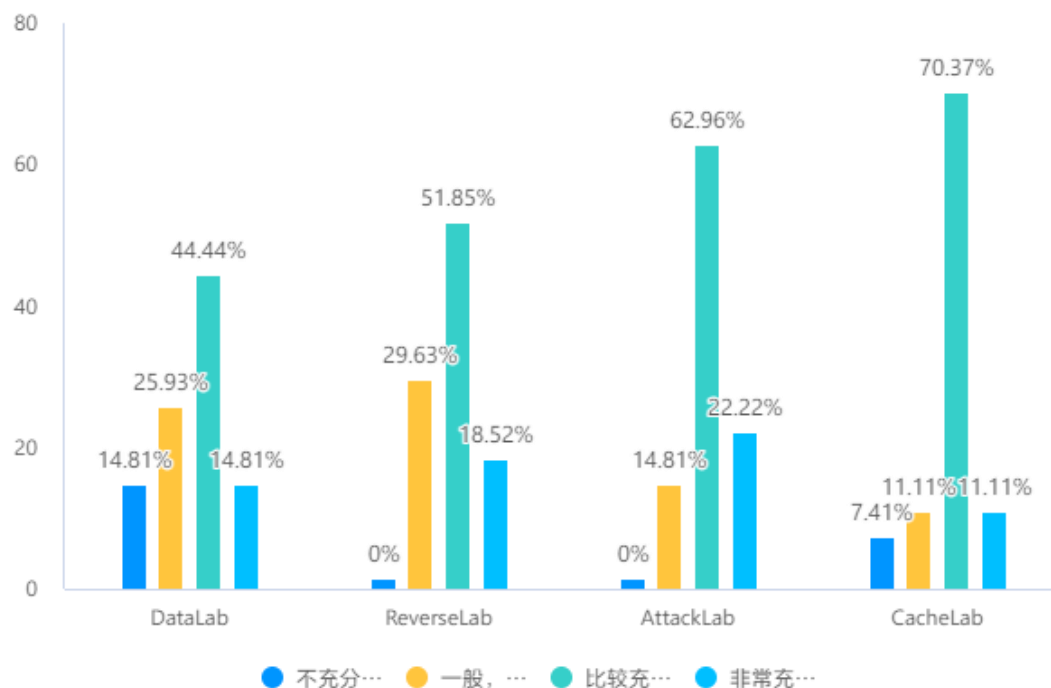
# 问卷调查

你认为各个Labs的理念给你的学术生涯带来多大帮助？



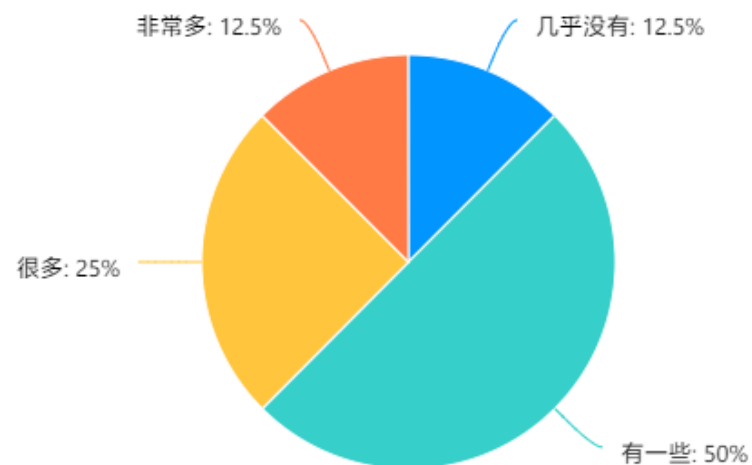
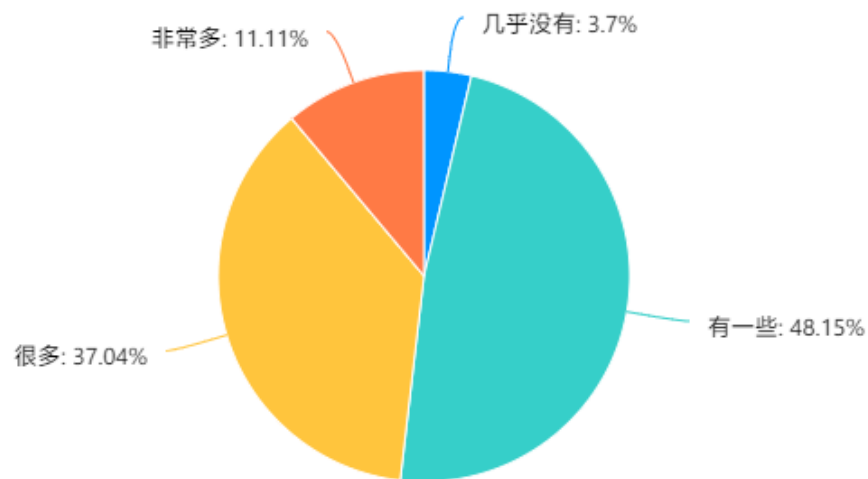
# 问卷调查

你认为各个Labs的评价方式是否充分合理？



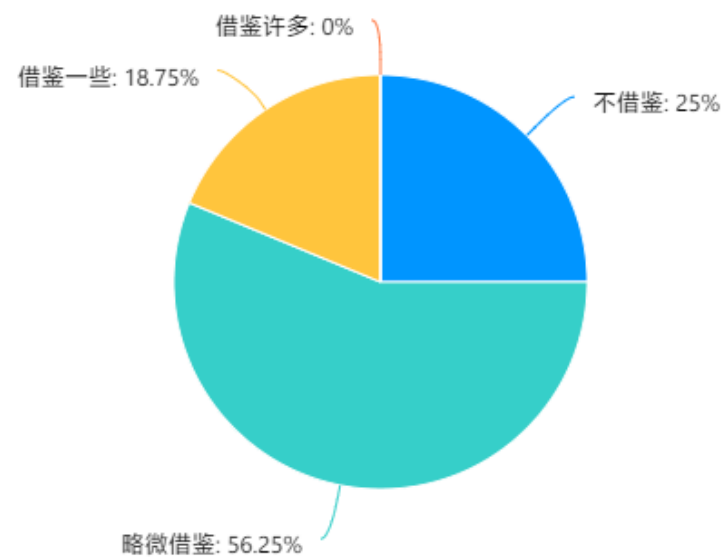
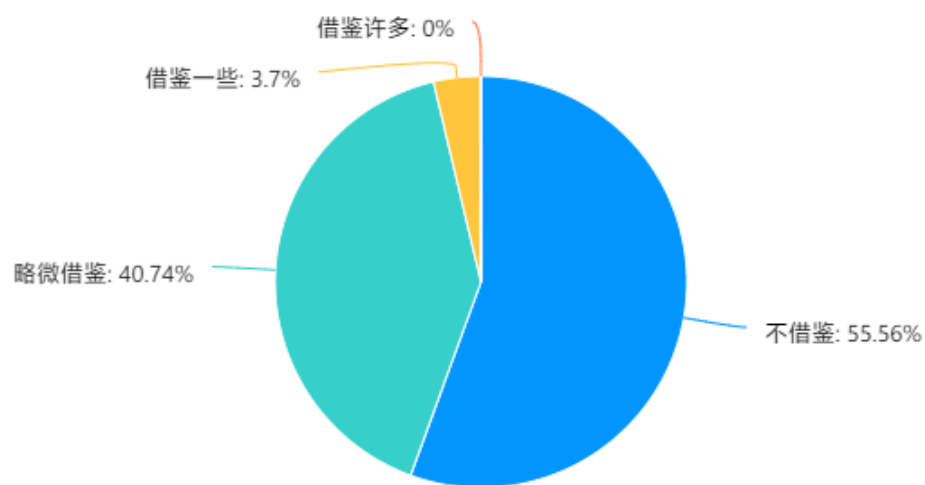
# 问卷调查

你认为身边同学抄袭赶工、直接借鉴互联网上或同学的代码的情况多吗？



# 问卷调查

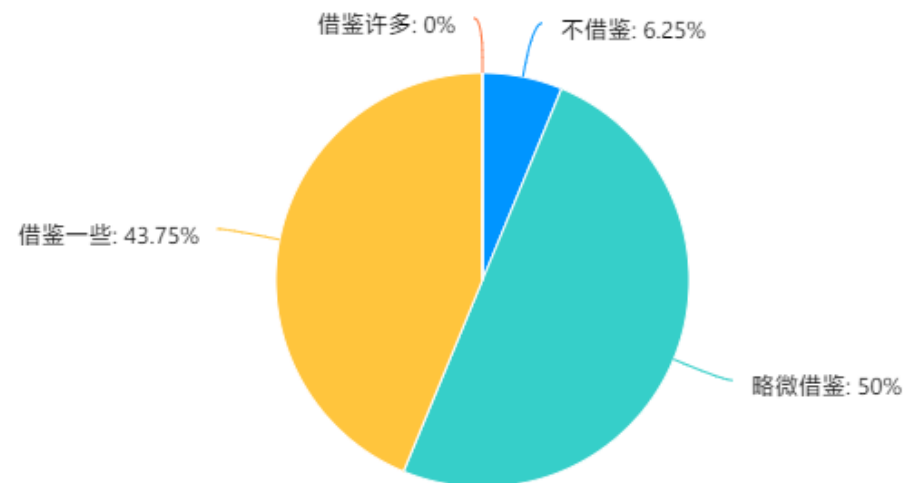
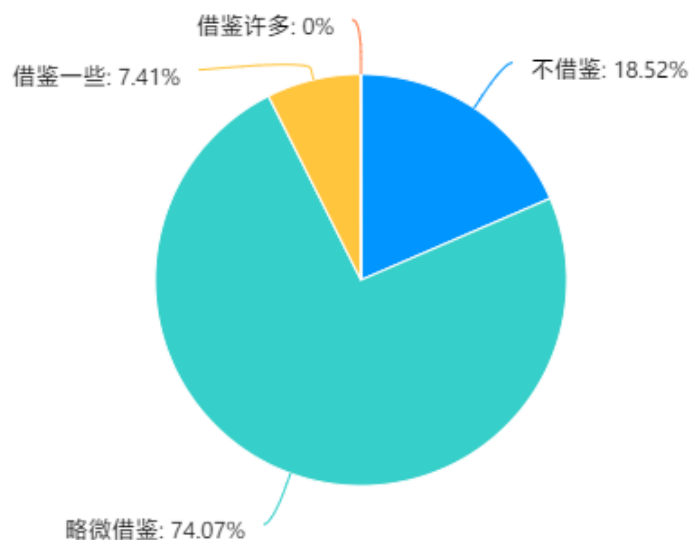
在开始做**Labs**前，你会选择直接借鉴互联网上或同学的代码吗？





# 问卷调查

在完成**Labs**的过程中，你会选择直接借鉴互联网上或同学的代码吗？



# 问卷调查

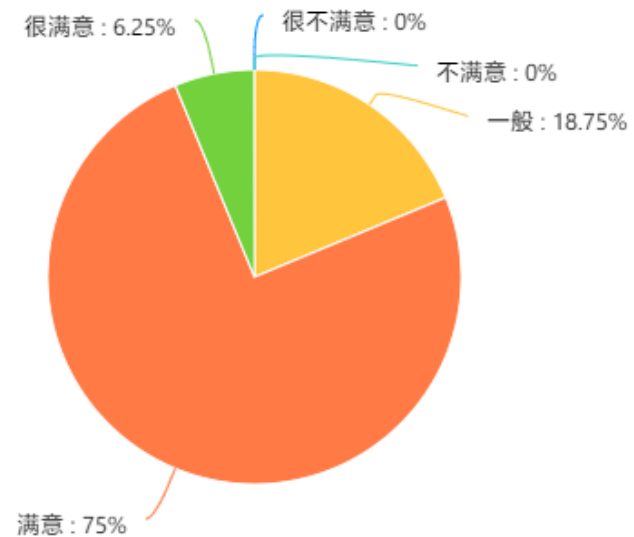
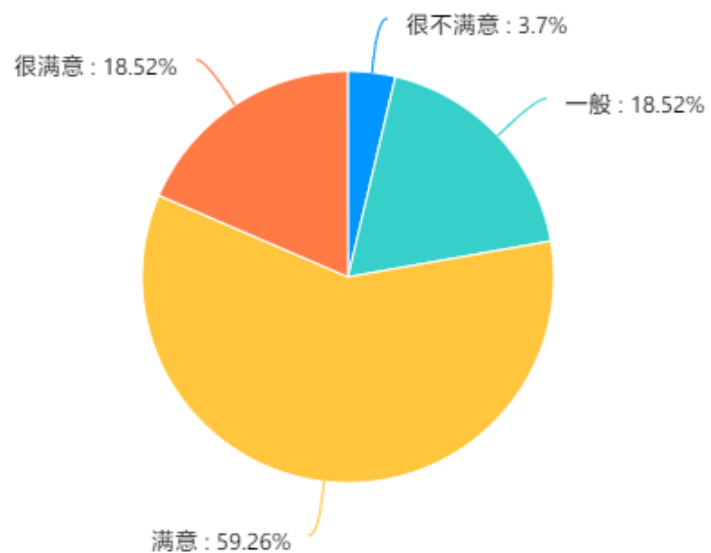
你都是因为哪些原因， 而选择直接借鉴互联网上或同学的代码呢？

选项	小计	比例
难度太高，经常没有思路	15	62.5%
我想偷懒，不想花太多时间和精力	0	0%
工期太短，没有足够的时间自己编写代码	4	16.67%
竞争太激烈，必须参考更优秀的实现	5	20.83%

选项	小计	比例
难度太高，经常没有思路	11	73.33%
我想偷懒，不想花太多时间和精力	0	0%
工期太短，没有足够的时间自己编写代码	2	13.33%
竞争太激烈，必须参考更优秀的实现	2	13.33%

# 问卷调查

你对整个实践环节的评价如何？



# 问卷调查

你在整个实践环节中学到的最印象深刻的知识、技能或理念是？

培养了对CTF的兴趣

代码优化

逆向

一定要自己动手

计算cache miss

通过cachelab理解了缓存的结构

栈缓冲区攻击印象最深，还有局部性原理，让我对系统构建有了深刻理解

栈溢出攻击

对汇编语言的了解 Cache的原理

attacklab的一些方法

“并行”的思想

时间局部性和空间局部性

汇编语言

就是一种在处理问题时候的思维，不仅仅是计算机方面的问题，这种思维往往对问题解决有根本性的帮助会往底层去想

# 问卷调查

你认为整个实践环节最大的困难是什么？期望在哪些方面改进？

课业压力大，lab耗费时间，到了期末更加繁忙时间

没困难，上课听了就好

前置知识不足，需要学大量内容（尤其是汇编语言），希望能在上机课上分几期讲一点汇编语言的入门知识，不然直接写lab太难受了

各个Labs的开头最难。

我认为FeedBack不够好。有些人的实现非常巧妙，建议助教或者老师能够收集优秀的实现方式进行实验后的讲解。

Datalab感觉像思维游戏而不是掌握位运算，希望不要刷榜

完全不会，课上内容很难转到实践中

datalab太卷了

cachelab的时间与期末考试重叠+datalab打榜太卷。attacklab或许可以删掉，提前cachelab的时间点。删掉打榜plz。

完成Lab过程中一些环节难以入手，希望能够在部分地方得到助教更加深入的帮助。

感觉助教的对lab布置时的讲解不太必要，可以等大家做一段时间再进行指导，可以降低难度，同时减少抄袭

还是基础知识薄弱，数学能力不足

# 问卷调查

你认为整个实践环节最大的困难是什么？期望在哪些方面改进？

上手。尤其是CacheLab单纯依靠实验提示基本完全无法展开工作，最明显的就是不清楚结构体可能需要哪些部分。其他Lab则是在调试时难度可能较大。

读懂实验说明文档，少一些排行榜

# 问卷调查

你认为整个实践环节最大的困难是什么？期望在哪些方面改进？

主要还是刷榜竞争问题，特别是datalab，得高分要求的操作数太苛刻，抄袭借鉴又太容易，经常出现的情况是榜上一排人同时“突破”同一个问题，单打独斗极难排到前面。如果是跟哪个地方犟上，很可能从早到晚绞尽脑汁想不出。datalab在学会基本运算之后就是类似移动火柴使等式成立的问题，这是我想了很多的比喻。在这上面过分花费时间就变成了玩智力题，虽然提升思维，但对知识的理解掌握有限，完全可以到此为止。ReverseLab个人感觉偏离课堂内容程度比较大，是最没用的，后面几道涉及花指令脱壳之类的题目如果不上网看CTF比赛的教程几乎不可能写出，但是并不是每个人都需要打CTF比赛，学了意义不大。另外就是要额外学IDA这样一个复杂的软件的各种操作，intel格式的汇编和CSAPP书上的ATT格式还是反的，学了IDA以后也基本用不到。另外Reverselab可能原意是练习看汇编，但实际上做题基本都是看反编译的C代码。Attacklab和Cachelab则实在经典，做完之后受益匪浅，得益于网上的博客资源，也不容易走弯路死钻牛角尖。综合下来，虽然后两个lab确实参考了网上思路，但是只要是自己一步一步做，比前两个lab体验更好、收获更多，只能说不愧是经典！强烈建议不要在大作业上设置太高难度，应该以练习掌握为主，做完就得高分。然后在考试当中多加与lab结合紧密的题，问lab过程中的思路和细节，是不是自己做的一考便知！

# 问卷调查

你认为该如何更全面、充分、准确地评估同学们对Lab的完成和理解情况？

比如，有同学提出了一些新颖的评估方式，代码评审、报告展示、一对一口头答辩，以及简单考察Lab相关的知识等等。你认为这些方法是否有助于巩固所学？你有无其他建议来促进深度理解？

多做实验，写代码

有必要口头答辩，让抄袭者完蛋

评价是采取这些方法整体上会有提高作用 不过对个体来说会认真做的还是会认真做 不会认真做的仍然是那样

通过实验报告评估同学对代码和知识点的理解

我认为问题不是出于ics，而是我们大一的编程实践比较缺乏，并不能支撑我们很好地完成ics课，如果有可能在大一增加一些学分和课时，增加一些对应的训练，可能会好一些。

一对一口头答辩当然最好，但可能不现实。强烈建议lab降低难度，用书上的经典lab，考试中直接考lab当中的知识点，lab是自己做的还是抄的一考就知道！

可以选择提交历史版本来证明工作量/一对一口头答辩/lab考察知识来确保知识的掌握。

有，没有



# 问卷调查

你认为该如何更全面、充分、准确地评估同学们对Lab的完成和理解情况？

我认为可以在Lab结束后，进行实验报告与代码的共享，同学可以自由的学习其他人的实现（可以借助代码托管平台进行操作，同时还能训练同学们的平台使用能力）；同时可以进行一些非成绩性质的鼓励，比如可以进行star的排行之类。如果觉得自己的实现很不错，可以主动提议将自己托管仓库链接发到群聊中。我认为在实验已经完成的情况下进行公开是合理的，但是也要注意只对本届学生开放，或者让同学们学会保护自己的代码，不要给低年级的同学进行抄袭。老师可以在同学互评结束后自己也评出一个实现不错的解决方案，让大家共同学习。

这些方法是可行的，实际自己一个人完全独立做完Lab的要求是相当高的，有时必须对于网络代码加以学习。借鉴是可以的，但也必须和简单的复制黏贴区分。而由于Lab设置的特殊性，即便是通过网络代码的学习了解了Lab要求，也可能很难写出与网络代码差别较大的代码，因而可能被认定为抄袭或借鉴过多。而答辩等方式就能确保学生对于任务和解决方案的了解，即便不是纯靠自己解决，也能得到较为深入的学习，从而巩固知识。

# 准备习题课

- 前置芝士
- 铺好台阶、搭好梯子，不能指望他们开天辟地、刀耕火种
- 第0个Lab布置前的三周，暂定内容
  - Basic Linux Commands + SSH in VS Code @hy
  - Git (commit, merge, stash, rebase) + GitHub (Issues, Review, Actions, PR) @xsx
  - VS Code LSP + debugger + C & C++ Coding Style @andylizf @huanchengfly
  - 似乎去年程云飞、符泷太等人已有PPT或指南，网上资料亦很多

# 准备习题课

- 前置芝士
- 铺好台阶、搭好梯子，不能指望他们开天辟地、刀耕火种
- 参考：
  - <https://www.cs.cmu.edu/~213/codeStyle.html>
  - <https://nju-projectn.github.io/ics-pa-gitbook/ics2022/>

# 设计实验

- Lab 内容
  - 重心?
    - DataLab 没什么意义，应该降低卷度
    - AttackLab 所涉及的知识点本就在ICS课程中处于边缘地位，且大家完成时间普遍偏少
    - BombLab 回来，至少要会用debugger，能读汇编
    - CacheLab 很有意义，因为Cache的思想在计算机各个层级各个领域都有应用。应该增加卷度
    - LinkLab 新增
      - Linking很重要，了解链接过程对部署C++项目、排查错误尤为关键，作为一个独立的章节，却没有Lab
      - 之后的课程中再也不会详细讲解Linking了
      - 纯理论性的讲解过于抽象，大部分同学在学习ICS后对Linking的理解仍较为粗浅
  - 打榜？
  - 题目创新？

# 设计实验

- Lab 内容
  - DataLab 通过性 @wyh
    - 匿名榜 一天前封榜
    - 给符号数的分数线 60, 80, 90, 100线
    - 介绍UB 要求大家不使用
      - 强制语法树检测 ( ? ) 先评估难度
  - 新题
    - PKU
    - *Hacker's Delight*

# 设计实验

- Lab 内容
  - BombLab @xSX
    - 强制用服务器 检测炸弹爆炸 扣点分、来点竞争性
    - 新题，隐藏点
  - LinkLab @andyliZf @huanchengfly
    - 参考[南大LinkLab](#) 及 南大蒋炎岩操作系统课的[可执行文件与加载](#) [动态链接与加载](#) ([课堂实录](#))
  - ~~AttackLab~~ 规模缩减 下放为作业 @wyh
  - CacheLab @hy @panjd123
    - 布置早一些，期末后ddl晚一些
    - 加难度
      - 扩展场景/改参数
      - 打榜

# 设计实验

- Lab Guiding
  - [CMU 原版 Metarials](#)
  - 通用的“实验前必读” @hy
  - 指南 per Lab
    - 又有人大自己的、又有CMU原版的，十分混乱
    - 至少整理成统一的

## Hand Out Instructions

**SITE-SPECIFIC:** Insert a paragraph here that explains how the instructor will hand out the `shlab-handout.tar` file to the students. Here are the directions we use at CMU.

# 设计实验

- Lab Guiding
  - Lab 前的习题课
    - 讲什么？念一遍 实验指南 没什么意义
    - 一些最必要的内容
      - 更易于理解 直观程度：现场演示>实验指南>RTFM
    - 具体讲的内容 每个负责的TA再研究 (TODO)
    - 仍空出来许多习题课
      - 考虑讲 Functional Programming



# 设计实验

- Submission
  - Academic Integrity
    - 作弊
      - 硬编码答案、故意骗过 Online Judge 而不实现实验要求等（南大蒋炎岩OS课[实验须知](#)）
    - 抄袭

# 设计实验

- Submission 形式
  - Liveline 和 Deadline
    - 鼓励时间管理较好的同学
    - 即 Soft DeadLine (南大蒋炎岩OS课[实验须知](#))
  - GitHub @andyfizf
    - Git/GitHub 的优势是显而易见的
      - Git 记录可以保留诚实完成作业的证据
      - 至少学会怎么版本控制 降低参与开源的门槛
    - GitHub Organization
      - 验收前Private
      - 验收后同班同学 Public
      - 助教公示好的实现供大家学习
    - AutoLab/AutoGrader by GitHub Actions

# 设计实验

- Submission 内容
  - 实验验收
    - 期末后当面验收，当场公布验收的Lab (暂定CacheLab)
    - 小朋友先花几分钟简单讲自己的实现；现场抽一个问题回答，回答得好可加分。综合衡量是否通过。
  - 代码检查
    - 查重 解决抄袭问题
    - 人工检查作弊问题
  - 实验报告，评估完成的独立性和创新度
    - 给同学们的建议 写进“实验前必读”中 @panjd123 @huanchengfly
      - 简明扼要。不宜水，水了扣分。不要讲大家都知道的事情。
      - 提供内容模板
        - 先用一段话，讲这个Lab在考察什么在做什么。
        - 叙述(0) 架构 (1) 亮点或难点 (2)收获及感悟，推荐选取其中一个小的代码片段进行说明。
        - 提供Markdown及Typst格式，须同学们导出为pdf
      - 具体措辞也可参考 南大蒋炎岩OS课[实验须知](#)

# 助教的权利与责任

# 权利

- 米米米
- 教学相长

# 责任

- 殷鉴不远

# 日程

07

周日

周一

周二

周三

周四

周五

周六

1

2

3

4

5

6

7

8

草案

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31

# 日程

08

周日

周一

周二

周三

周四

周五

周六

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

中期

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

31



# 日程

09

周日

周一

周二

周三

周四

周五

周六

1

2

3

4

5

6

7

8

9

开学

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

28

29

30

# 分工

- 责任到人