计算机科学技术导论

计算系统与抽象

计算系统层面

In addition to organizational characteristics (e.g., hardware + software + data), a computing system must ( ) in order to be useful to its users.

answer: solve certain problems

构建或购买计算系统的目的是利用它并解决某些计算问题。 因此，计算系统必须为其用户提供效用。

A computing system is a dynamic entity consisting of **software, hardware, and the data they manage**.

计算系统由硬件、软件和它们管理的数据组成。

抽象 Abstraction

The TCP/IP protocol stack has four layers, namely the transport layer, the network layer, the medium-access control layer, and the physical layer. The transport layer is responsible for reliable end-to-end communication between end hosts, but it does not consider the actual paths taken by the data segments over a network. This is because network routing (i.e., assigning a path over the network to each data packet) is implemented by the network layer. This partitioning of networking functionality into different layers is a typical application of the computational thinking of ( ).

answer: **abstraction**

网络层向传输层隐藏了网络路由的复杂性。 抽象是帮助管理复杂性的心理模型。

抽象把系统划分为多个小系统进行工作。

抽象是一种计算思维，它通过隐藏不必要的细节来帮助管理复杂性。 适当的系统划分将允许不同的工程团队专注于较小的一组约束和需求，并且只要这些团队之间的“接口”（即每个团队为另一个团队实现的服务），多个工程团队将能够无缝协作 团队）是明确定义的。

do not need to understand details

TCP/IP

OSI/RM

object-oriented design(面向对象编程)

Object of one class does not need to care about the internal implementations of the other classes. **abstraction**

信息表示

信息呈现

二进制转十六进制

从左往右取四位进行转化

十进制小数转二进制

小数部分乘2取整，剩余小数继续

数字化 Digitization

When taking a digital photo or scanning a document into the computer for further processing, we essentially perform ( ) on the input data so that it can be represented with binary values.

answer: **digitization**

拍摄数码照片或扫描文档到计算机的过程本质上是一个数字化过程。

analog signals 模拟信号

Signals such as human voices are analog in nature. To process such **analog signals**（模拟信号）, the input devices of computer systems typically perform ( ) to transform the continuous-valued signals to data series with discrete magnitude levels that can easily be encoded with binary sequences.

answer: **digitization**

模拟信号需要转换为数字信号，以便计算机处理。 这个过程被称为数字化。

多媒体呈现

Vector graphics 矢量图

Raster graphics 光栅图形

audio 声音

A ( ) can be regarded as a sequence of images. The motion and/or animation effects of a video are realized by presenting the images at very short intervals to leverage the persistence of human vision.

answer: video

视频本质上是一系列以高频呈现给观众的图像。 我们的期末考试不会测试视频压缩技术。

ASCII与其呈现

Suppose that you need to compress the character string "AACCCCCGGG" using run-length encoding. Which one of the following is a possible compression result? ( )

answer: AA*C5GGG

行程编码规则：

1. 使用“*XY”表示“重复字符X Y次”；
2. 大小小于等于3的重复字符串不需要压缩。

The ASCII code for character '2' is 50 (in decimal). What is the value of the expression '2'+3? ( )

answer: '5'

'2' 是一个 ASCII 字符。 '2' + 3 将代码（'2' 或十进制的 50）移动 3 个位置到 '5'（或十进制的 53）。

计算机硬件和组织

逻辑门看看书吧

数字电路

sequential circuit 时序电路

时序电路具有记忆先前输入/状态的能力

combinational circuit 组合电路

组合电路的输出仅取决于其瞬时输入的值。

数字电路的表示方法有三种，即布尔表达式、真值表和符号图/原理图。

Boolean expressions, truth tables, and symbolic diagrams/schematics.

冯诺依曼结构

计算机内存是字节寻址的，即每8位组合在一起形成一个字节，整个内存中可以读取或写入的数据的最小单位是一个字节。

byte 字节

bits 位 8位二进制数一字节

内存层面

A memory system deploys caches between the CPU and the main memory to speed up data accesses. Suppose that every CPU read request is simultaneously issued to both the cache and the memory. Assume that it takes 200 CPU clock cycles for the memory to complete a CPU read request and that it takes 20 CPU cycles for the cache to complete the same request as long as the requested contents can be found in the cache. Therefore, a CPU read request may be completed in one of the following ways:

1) Cache hit: The CPU attempts to fetch certain data contents from both the memory and the cache at the same time, and the contents are found in the cache. In this case, the CPU will not wait for the memory read operation to complete (instead, memory read will be aborted). It will directly proceed with the data retrieved from the cache.

2) Cache miss: The CPU attempts to fetch certain data contents from both the memory and the cache simultaneously, but the contents cannot be found in the cache. In this case, the CPU will have to wait until the completion of the memory read request.

For a given program, if the ratio of cache hit is 75% and there are 10000 read operations during the program's execution, then how many CPU clock cycles are spent on data read requests? ( )

A. 10000*200 = 2000000

B. 10000*20 = 200000

C. 7500*20 + 2500*200 = 650000 ✓

D. 7500*200 + 2500*20 = 1550000

Cache hit （缓存击中，指缓存中已经有相应数据）

Cache miss

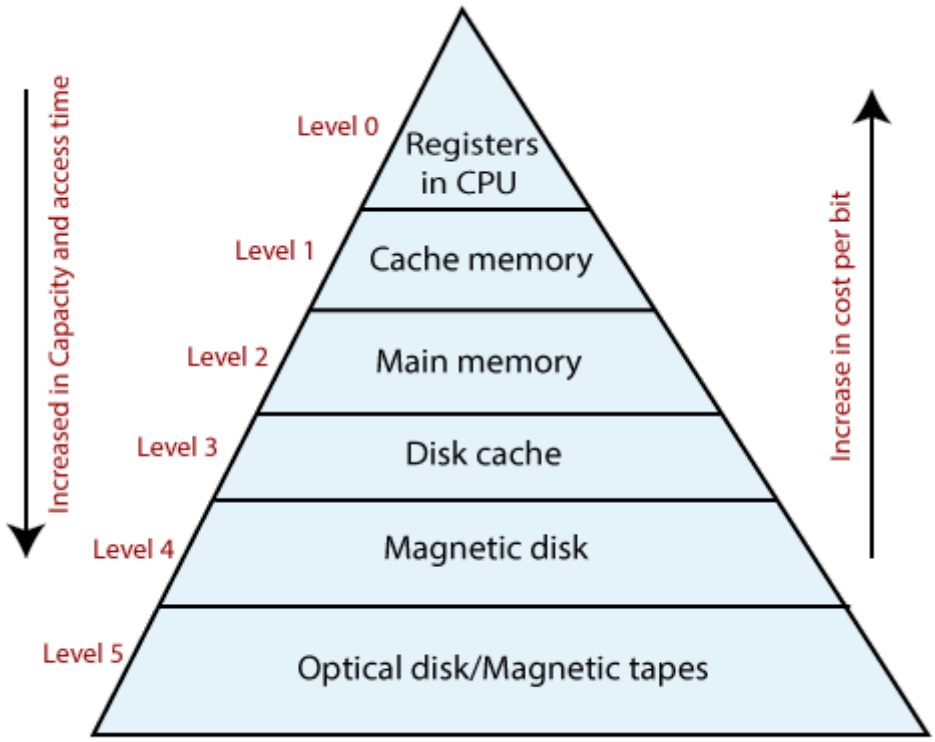
what is the acceleration ratio resulted from the deployment of cache?

部署缓存带来的加速比是多少?

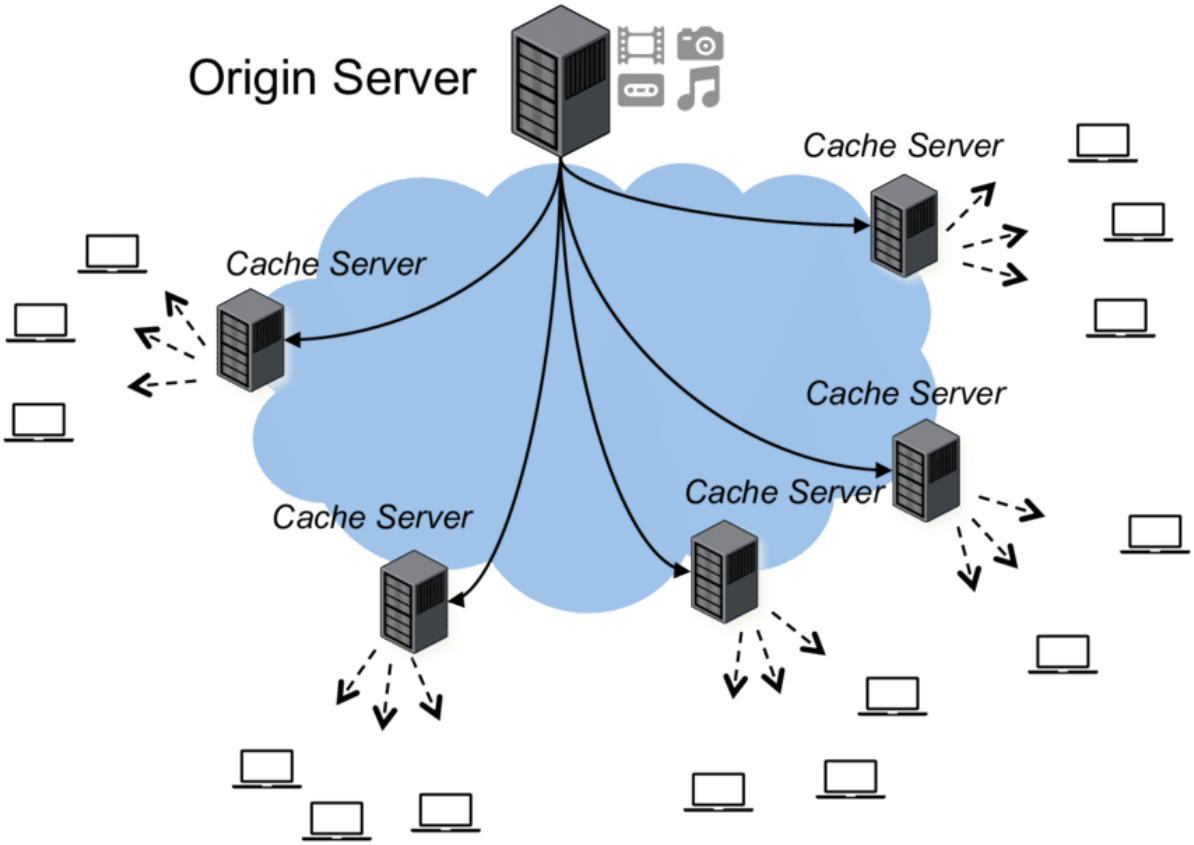
公式：[10000x200 - (7500x20 + 2500x200)] / 10000x200 = 67.5%

[(未优化时所需CPU时钟周期) - (实际所需CPU时钟周期)] / (未优化时所需CPU时钟周期)

缓存的作用



缓存的运行速度比主内存快，并存储经常访问的内存内容或极有可能很快被访问的数据。 来自 CPU 的内存请求将同时发送到缓存和内存。 如果可以在缓存中找到内容，则可以在内存访问完成之前响应请求。 同样，磁盘缓存部署在内存和磁盘之间，以缓解磁盘频繁访问时的性能瓶颈。



当全球多个用户尝试访问原始/主服务器中的相同内容时，内容分发网络部署多个“缓存”服务器以提高整体系统性能。

在缓存服务器会存储特定区域内许多用户经常访问的内容和预计会流行的内容。

操作系统层面

操作系统

An ( ) makes it easier for a user application to utlize computer system resources and compete fairly with other programs for system services.

answer: operating system

当用户应用程序需要某些硬件资源或软件服务（例如，编译器、链接器、加载器等）时，它将直接与操作系统交互，操作系统将作为代理来帮助正确管理请求的资源并协调 在多个请求之间（可能来自不同的用户应用程序）。

An ( ) offers the **virtual machine illusion** to user programs: A user program developer can safely assume that his/her program has exclusive access to all the necessary hardware resources and software services (e.g., an unlimited amount of memory, all the CPU time). The program can execute as if it is the only program that is currently running on the computer.

answer: operating system

实现虚拟机错觉是操作系统的关键作用之一。

进程调度

当进程产生的I/O请求被相关的I/O设备完成后，该进程就可以进入就绪状态，参与下一轮CPU调度。

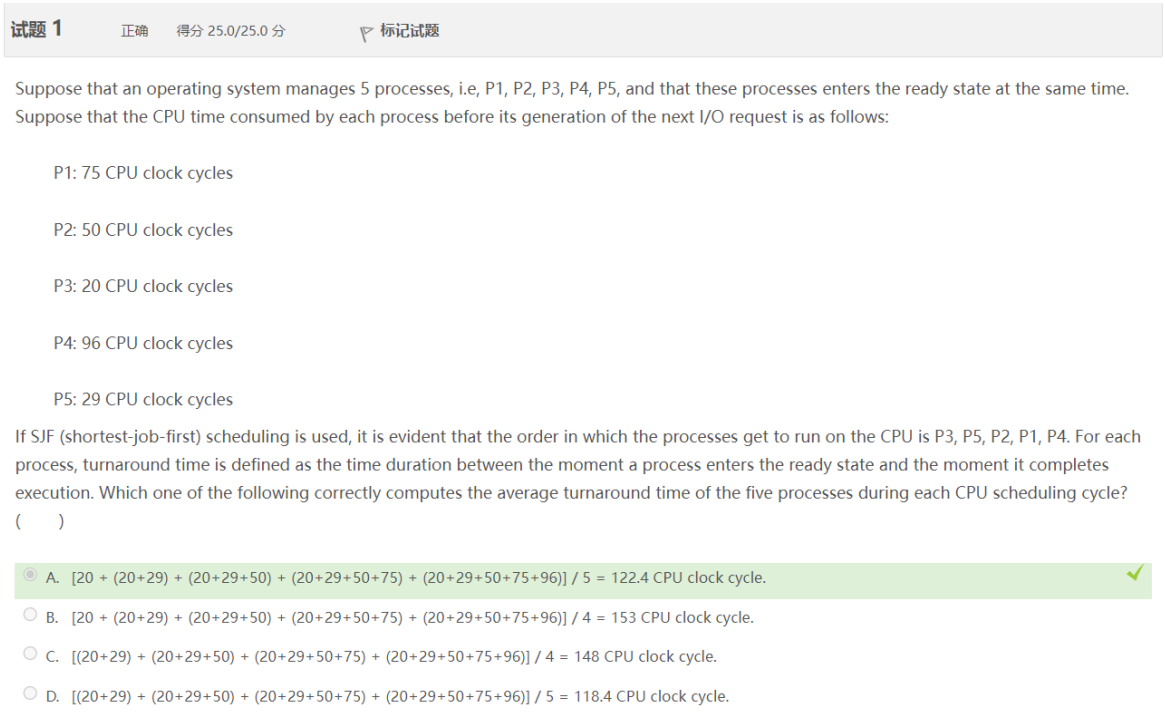
当当前运行的进程产生 I/O 请求时，操作系统会将该进程移动到等待状态，并开始新一轮的 CPU 调度，以便另一个进程有机会占用 CPU。

一个过程通常需要完成必要的计算并与用户（或其周围环境）进行交互。 计算消耗 CPU 时间，而用户交互通常通过输入/输出系统实现。 当进程发出 I/O 请求时，它必须等到 I/O 请求完成才能继续执行进一步的处理步骤。 此时应该由进程释放CPU，以便其他处于就绪状态的进程可以竞争它。

creation -> ready -> running- > waiting(maybe) -> termination

CPU调度

CPU 调度使多个进程共享同一组硬件资源成为可能：计算机硬件设备不知道请求是由多个用户程序生成的，而每个用户程序都错觉它对 电脑硬件。



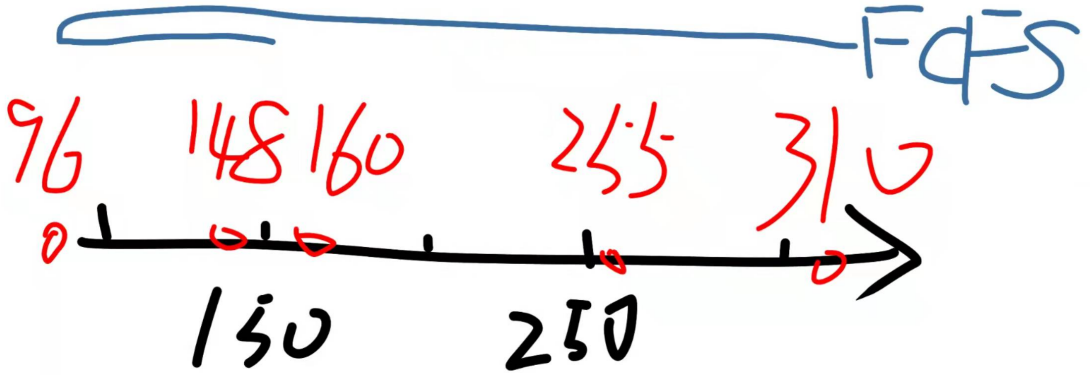
turnaround可以翻译为 进程提交给操作系统 到它真正获得一次CPU并完成在CPU上的执行
CPU调度决定多个可以执行的进程谁先占用CPU执行
占用CPU的可以先完成 **别的进程就要等**
答案中就是把各个进程需要等待的时间 也算了进去
这样才符合题目中给出的定义
所以这个不叫“把之前运行过的程序算进去” 说可以这么说 但实际上就好比你去食堂打饭
你排队半小时 打饭三分钟
你不会认为你打饭只花三分钟

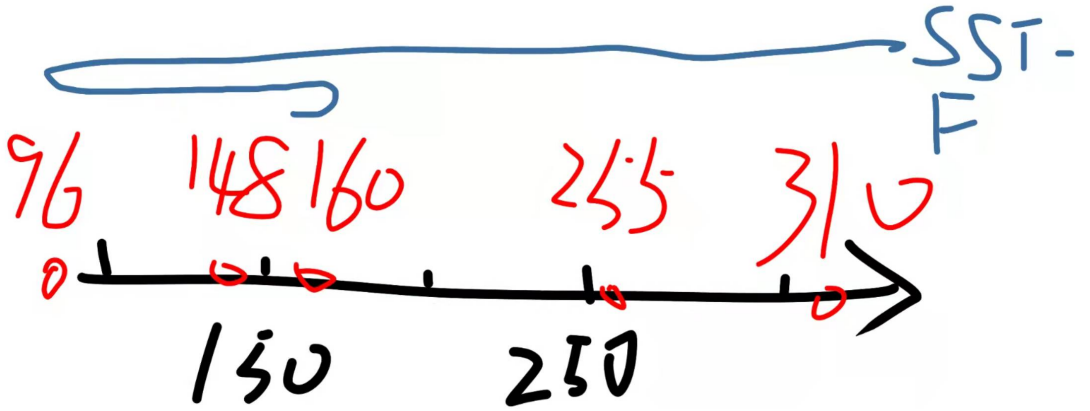
磁盘调度

Suppose that a hard disk drive has 500 tracks (numbered from 1 through 499). The current position of the read/write head is at the 150th track and it is moving toward the 499th track. Assume that a series of read/write requests at different tracks arrive in the following order: 148, 96, 160, 255, 310. Among the disk scheduling policies of FCFS, SSTF (shortest-seek-time-first), and SCAN, which one has the shortest average seek time? ( )

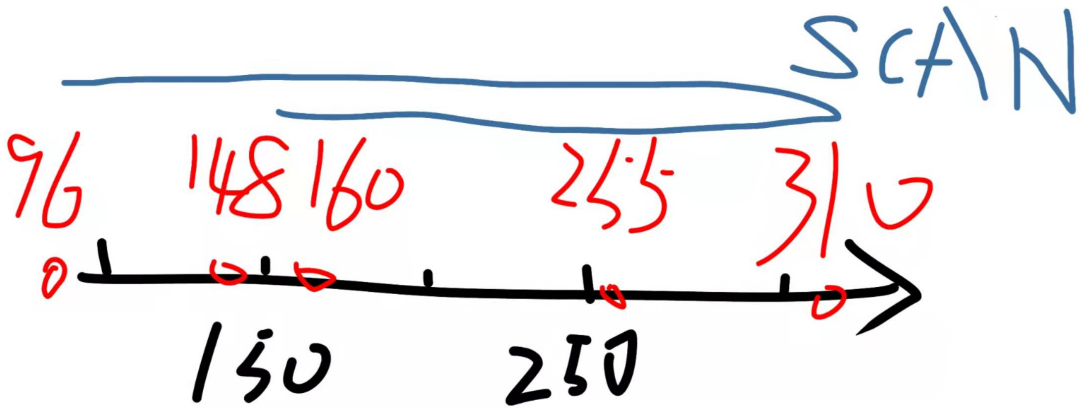
FCFS、SSTF（shortest-seek-time-first）和SCAN的磁盘调度策略中， 哪一个的平均寻道时间最短?

answer: FCFS





（勘误：先转向148再去160）



对于每个请求，所需的总时间由寻道时间、等待时间和读/写操作时间组成。 如果请求足够随机，我们可以假设不同请求的读/写头等待时间和读/写操作时间大致相同。 磁盘调度策略的平均寻道时间在某种程度上是磁盘调度性能（就延迟而言）的度量。

文件系统

relative path 相对路径

absolute path 绝对路径

Two types of files that are frequently used in the file system:

text files and binary files. 文本文件和二进制文件

File system typically realizes ( ) files as a way to represent application-specific information for quick access and/or efficient storage utilization.

answer: binary

文件系统通常将**二进制文件**作为一种表示特定于应用程序的信息以**实现快速访问和/或高效存储利用的方式。**

文件系统将**文本文件**文件作为一种快速方便的方式来编码人类可读的信息，例如字符、单词、句子、数字等。相比之下，二进制文件通常采用一些特定于应用程序的编码来有效地表示信息。

算法与机器语言

算法层面

( ) is a plain language description of the steps in an algorithm. Although structural conventions on the use of natural languages are recommended, the most important rules on the use of such algorithm description method are to make things readable and to keep a consistent style.

answer: **Pseudocode 伪代码**

伪代码由简短的自然语言短语组成，用于解释程序中的特定任务。 理想情况下，伪代码不应包含任何特定编程语言中的关键字。 伪代码应该写成一个连续短语的列表，我们甚至可以画箭头来表示赋值语句。缩进可用于以伪代码显示逻辑程序流程。

机器语言范式

Imperative programming 命令式编程

Declarative programming 声明式编程

Procedural programming 面向过程编程

Object-oriented programming 面向对象编程

声明式编程（e.g. SQL）是一类编程范式，即构建计算机程序的结构和元素的一种风格，它表达计算的逻辑而不描述其控制流。 许多应用这种风格的语言试图通过描述程序在问题域方面必须完成的事情来最小化或消除副作用，而不是将如何完成它描述为一系列编程语言原语（如何 语言的实现）。 这与命令式编程相反，命令式编程以明确的步骤实现算法。 我们课程中涵盖的两个主要声明式编程范式是函数式编程和逻辑编程。

声明式编程是以数据结构的形式来表达程序执行的逻辑。它的主要思想是告诉计算机应该做什么，但不指定具体要怎么做。

命令式编程的主要思想是关注计算机执行的步骤，即一步一步告诉计算机先做什么再做什么。

assembly language 汇编语言

machine language 机器语言

机器语言和汇编语言都是低级编程语言。

语言翻译

为了生成可执行文件，通常使用用户程序的（编译/汇编）目标文件和必要库的二进制文件调用 ( )。

answer: **Linker 链接器**

链接器（Linker）将用户程序的 object 文件与用户程序中调用的库的二进制文件结合起来。 它将生成一个可执行文件，当加载器将其加载到主内存中时，就可以执行了。

An ( ) converts a program written in assembly language to a machine code program.

( ) 将用汇编语言编写的程序转换为机器代码程序。

answer: **assembler 汇编器**

在某些开发过程中，用某种编译编程语言编写的程序可能无法直接编译成机器码。 相反，它首先被编译成汇编语言程序，然后由汇编程序进一步翻译成机器代码。

A ( ) translates a piece of high-level programming language source code into native machine code as an entirety, making it easy for the target machine to execute the program directly.

answer: **compiler 编译器**

编译器将用某种高级编程语言（实际上是它的编译实现）编写的程序转换为用本地机器代码编写的程序。 请注意，编译器通常会将完整的高级语言程序（例如，C 程序源文件）转换为完整的低级语言程序以促进执行。

In contrast to compiled programming languages such as C and C++, Python is an ( ) programming language: Each statement in a Python program is translated into a certain form suitable for execution (e.g., Python bytecode instructions) and immediately executed before the next statement is processed.

answer: **interpreted 解释的** Python是一门解释性语言

在语言的解释性实现中，源代码不是由目标机器直接运行的（即，源代码没有被翻译成本地机器代码）。 相反，另一个程序读取并执行原始源代码。 这个其他程序也称为解释器。

抽象数据结构

queue 队列

first-in first-out (FIFO)

stack 栈

last-in first-out (LIFO)

软件测试

黑盒测试

Black-box testing aims to maximize **data coverage**.

最大化数据覆盖

白盒测试

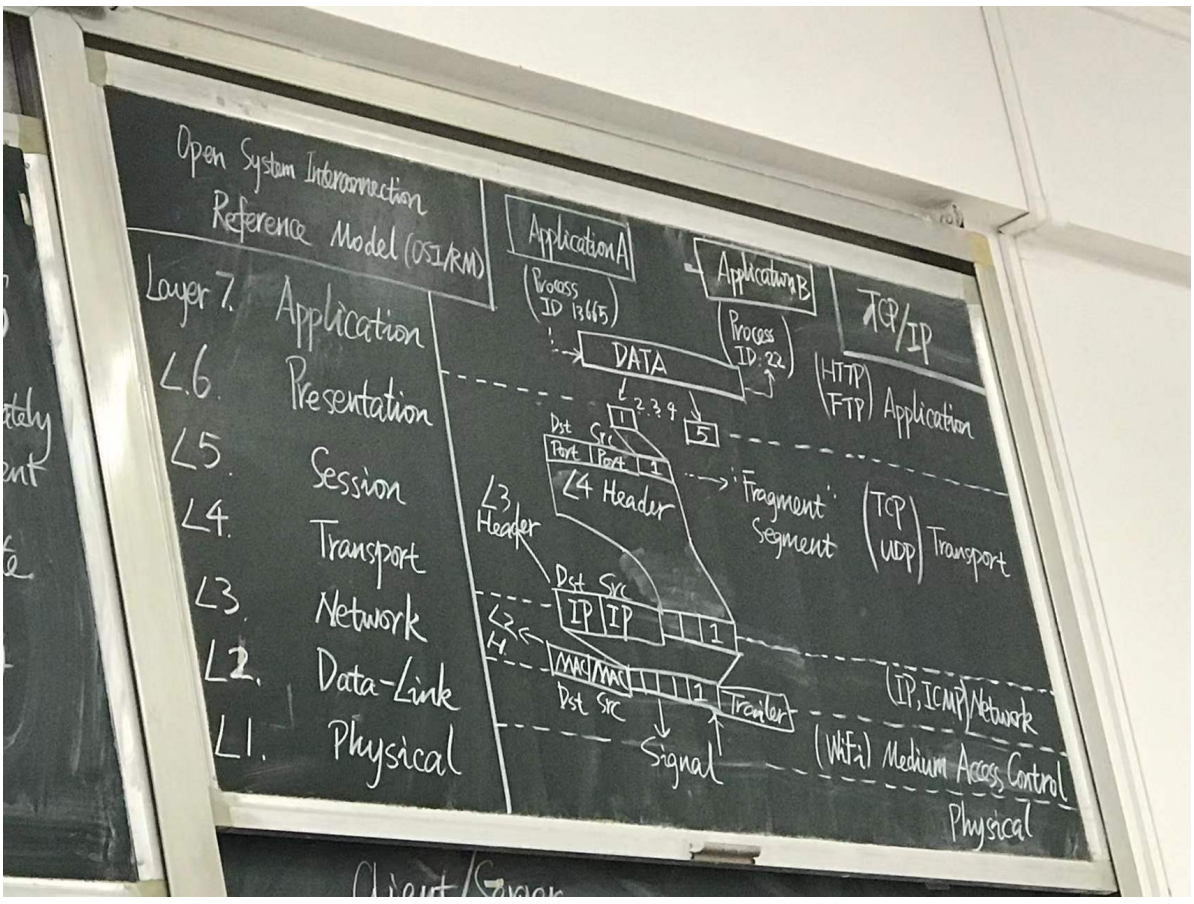
Clear-box testing (also known as **white-box testing**) should be performed to **maximize code coverage**.

最大化代码覆盖

在许多软件开发过程/模型中，测试是开发过程/模型/周期的许多阶段中必不可少的步骤，需要**定期执行。**

网络与信息安全

OSI/RM模型



The Open System Interconnection Reference Model (OSI/RM) has a ( ) layer that is responsible for the **encoding(编码), decoding(解码), encryption(加密) and decryption(解密)** of user application messages for network transmission.

answer: **presentation** 呈现层将将数据进行解压和解密以进行信息呈现，并加密和压缩进行信息传递

The ( ) layer in OSI/RM focuses on the implementation of technology-specific error detection/correction for hop-by-hop data transmission.

answer: **data-link** 数据链路层的目的是**确保可靠的传输并控制对传输介质的访问。**

( )segmentation(分割) of data streams,monitoring and controlling network congestion

answer: **transport 数据流分割，监控和控制网络拥塞**

The ( ) layer in OSI/RM is responsible for network routing, i.e., deciding on the specific path(s) that packet(s) can travel through and reach the destination.

answer: **network** 为了实现网络路由，OSI/RM 中的网络层定义了网络范围的地址（例如，IP 地址），并部署网络路由器以将数据包从源中继到目标。

网络层负责网络路由，即确定数据包可以经过并到达目的地的特定路径。

交换模型

电路交换 Circuit Switching

In order to ensure communication quality, ( ) can be leveraged to reserve network resources along the path that will be taken by data frames to be transmitted.

answer: circuit switching

电路交换的过程中，A和B两个人始终霸占着一条通信电路，他们每说一句话，都会实时被对方获取，因此数据是不用分组的。

电路交换提高通讯质量而降低了网络资源利用率

分组交换 Packet Switching

To maximize network resource utilization among multiple users, ( ) is typically implemented to allow each packet to determine their own path through the network. No resource reservation is performed.

answer：packet switching

最大限度地提高多个用户之间的网络资源利用率，允许每个数据包确定自己通过网络的路径。不执行任何资源预留（比如电话线）。

当部分网络拥塞时，可能较早发送的一些数据包较晚到达目的地。

信息安全层

CIA： confidentiality, integrity, and availability.

interoperability X 可操作性不包含在CIA里

unauthorized access 未经授权的访问 ——>保密性 confidentiality

Data **confidentiality**（保密性） is the practice of making sure that private information is kept secret.

remain intact 保持完整——>完整性 integrity

Any modification（修改） and/or deletion（删除） will result in violations（违反） of this requirement.

数据完整性是在整个生命周期内维护和保证数据准确性和一致性，是存储、处理或检索数据的任何系统的设计、实现和使用的关键方面。

available 可用的——>可用性 availability

数据可用性意味着授权用户可以访问信息。它保证了经过身份验证的用户在需要时可以访问您的系统和数据。

公钥 私钥

Public-key cryptography 公钥加密

每对由一个公钥（其他人可能知道）和一个私钥（除了所有者之外，任何人都可能不知道）组成。

Secret-key cryptography 密钥加密

密钥密码术也称为对称密码术，因为相同的密钥用于加密和解密数据。 使用密钥加密技术，通信双方 Alice 和 Bob 使用相同的密钥来加密和解密消息。 在任何加密数据可以通过网络发送之前，Alice 和 Bob 都必须拥有密钥，并且必须就他们将用于加密和解密的加密算法达成一致。

机器学习和人工智能

人工智能层面

弱人工智能，是指通过设计算法来实现人工智能的方法，这些算法可以产生与人类用户生成的结果一样好的结果。

强人工智能，是指通过精确复制人类思维的工作方式/思维方式来实现人工智能的方法。

图灵测试

如果评估员不能可靠地将机器与人类区分开来，则称机器已通过测试。
测试结果并不取决于机器对问题给出正确答案的能力，只取决于它的答案与人类给出的答案有多相似。

类似验证码的形式的测试就是图灵测试

机器学习和训练算法

Which one of the following process is **necessary** in order to adapt a generic machine learning model/algorithm to a specific problem/application? ( )

answer: **Training.**

当模型训练完成后，我们需要对其进行测试和验证。如果结果不尽如人意，应进一步改进模型。该模型经过一遍又一遍地训练和改进，直到结果令人满意。

testing往往对应新的数据

Training ----> testing ---->Training

与传统的选择排序相比（传统的排序需要我们对问题有逻辑上的深入理解），机器学习所进行的（排序）算法通常需要**大量的数据**（a relatively larger amount of **data**）进行训练推广。

机器学习范式

Unsupervised machine learning 无监督机器学习

Supervised learning 有监督的机器学习

Reinforcement learning 强化学习

Unsupervised machine learning 无监督机器学习

needs to derive proper rules from the unlabeled training data set and apply the extracted knowledge to the testing data set.

无监督机器学习是从历史数据中推断潜在的情况，在这种方法中，机器学习模型**试图自己找到数据中的任何相似之处、差异、模式和结构。**不需要事先进行人为干预（即标记）。
所给予机器的数据是没有标签的。

漫游者对火星的探索涉及就如何安全地从一个位置导航到另一个位置做出一系列决定，因此应该使用无监督学习。（X）

Supervised learning 有监督的机器学习

在**监督机器**学习中，在训练过程中提供给机器学习模型/算法。
所给予机器的数据是有标签的

监督学习是你有输入变量（x）和输出变量（Y）的地方，你使用算法来学习从输入到输出的映射函数。

Reinforcement learning 强化学习

强化学习范式**侧重于涉及一系列决策的问题**（例如，与人类对手下棋）。该范式的算法旨在**最大化累积奖励**（即，在与对手竞争时通过采取一系列步骤/决策获得的优势）。

强化学习涉及智能代理应该**如何在环境中采取行动**，以最大化累积奖励的概念。
比如漫游者在火星上的决策。

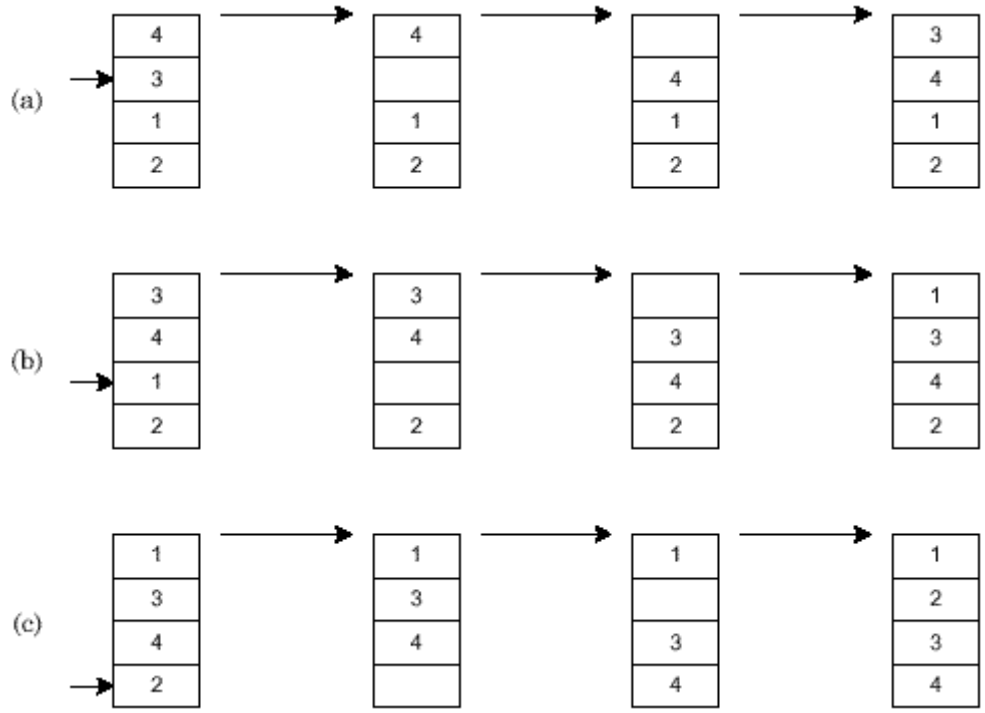
分类与回归问题

classification problem 分类问题 将数据分类

regression problem 回归问题 对未来数据进行预测（类似线性回归）

排序

插入排序



代码实现

```
int a[5]={1,3,4,5,2};
int i,p,temp;
for(i=1;i<5;i++) //从第二个元素开始
{
    temp = a[i];
    p = i-1;
    while(p>=0 && temp < a[p])//将较大的元素后挪，腾出一个空位
    {
        a[p+1] = a[p];
        p--;
    }
    a[p+1] = temp;//插入
}
```

选择排序

基本思想是：首先在未排序的数列中找到最小(or最大)元素，然后将其存放到数列的起始位置；接着，再从剩余未排序的元素中继续寻找最小(or最大)元素，然后放到已排序序列的末尾。以此类推，直到所有元素均排序完毕。

代码实现

```
int a[5]={1,3,4,5,2};
int i, j;
int min;
for(i=0;i<5;i++)
{
    min = i;
    //找"a[i+1]..a[n]"之间最小元素，并将下标赋给min
    for(j=i+1;j<5;j++)
    {
        if(a[j] < a[min])
```

```
            min = j;
    }

    //若min!=i，则交换 a[i] 和 a[min]。
    //交换后，保证了a[0]..a[i]之间元素有序。
    if(min != i)
    {
        int temp = a[i];
        a[i] = a[min];
        a[min] = temp;
    }
}
```

冒泡排序

这个都很熟了

代码实现

```
void BubbleSort(int arr[], int len) {
    int i, j, temp;
    for (j = 0; j < len - 1; j++) {
        for (i = 0; i < len - 1 - j; i++)
            if (arr[i] > arr[i + 1]) {
                temp = arr[i];
                arr[i] = arr[i + 1];
                arr[i + 1] = temp;
            }
    }
}
```

图的搜索

随便看看吧 比较难解释了

二叉树的DFS

```
void dfs(BiTree root) //root是指向树这个结构体的指针
{
    if(root==NULL)
        return;
    cout<<root->date<<" "; //和printf一个意思
    dfs(root->left);
    dfs(root->right);
}
```

图的DFS

这里用了链式前向星。。

```
bool vis[maxe];
void dfs(int u) { //深度优先
    cout << u << " ";
    vis[u] = true;
    for (int i = head[u]; i != -1; i = edge[i].next) {
        int to = edge[i].to;
        if (!vis[to]) { //阻止回头
            dfs(to);
        }
    }
}
```

——by Jerry