计算机保研/考研

专业课复习

目录

一、英语★★★★★

1. 请用 1min 进行个人陈述。

My name is XXX. I am from XXX. It's my great honor to attend this interview.

With my passion for computer science and technology, I have been studying hard for college courses during the undergraduate period, and my academic performance ranks first in my major. I attach great importance to the study of mathematics and professional knowledge, and actively participate in relevant competitions in college, and achieve good results. I have also accumulated some practical ability in engineering.

All in all, I deem myself a diligent, responsible, and open-minded student. I am looking forward to becoming a graduate student at your institute. Thanks for your attention.

老师们早上好，我叫 XXX。我来自 XXX。我很荣幸能参加这个夏令营。

怀着对计算机科学与技术的热爱，我在本科期间一直努力学习大学课程，成绩名列专业第一。我非常重视数学和专业知识的学习，在大学期间积极参加相关比赛，并取得了不错的成绩。我还积累了一定的工程实践能力。

总而言之，我认为自己是一个勤奋、负责、开放的学生。我的介绍到此结束。谢谢你的关注。

2. Would you like to tell some stories about your hometown?介绍你的家乡。

My hometown, XXX, is located in the southwest part of XXX Province. It is a small city with honest and kind-hearted people. The most unique characteristic of my hometown is the peach flower. When it comes to peach flower, you will feel the warm beauty and fragrance of the country. So, if you get a chance to come to XX, it's my pleasure to be your guide.

我的家乡 XXX 位于 XXX 省的西南部。这是一个有诚实和善良的人的小城市。我的家乡最独特的特色是桃花。说到桃花，你会感受到乡村的温暖美丽和芬芳。所以，如果你有机会来 XX 旅游，我很乐意做你的导游。

3. What is your favorite major and why?你最喜欢的专业课是什么，为什么？

My favorite course was data structures. On the one hand, data structure is a basic professional course, which is very important for programming. When we need to design an algorithm, choosing the right data structure increases efficiency. On the other hand, learning data structure requires abstract thinking, which helps to cultivate my thinking ability and improve my problem-solving ability.

我最喜欢的课程是数据结构。一方面，数据结构是一门专业基础课，这对程序设计非常重要。当我们需要设计算法时，选择合适的数据结构会提高效率。另一方面，学习数据结构需要运用抽象思维，这有助于培养思维能力，提升我解决问题的能力。

4. Why do you choose our university?为什么你选择我们的学校？

It has excellent teachers, rich teaching and research resources, and can provide students with a good learning environment and development opportunities.

In short, it is one of the high-level universities with important influence and reputation in China and even in the world.

There is no doubt that I will learn a lot here. It has always been my dream school.

它有着优秀的师资力量、丰富的教学科研资源，能为学生提供了良好的学习环境和发展机会。

总之，是中国乃至世界上具有重要影响力和声誉的高水平理工科学府之一。

毫无疑问，我会在这里学到很多东西。它一直是我梦想的学校。

5. Can you introduce your undergraduate school?你能介绍一下你的本科学校吗？

XXX is a high-level university in XXX Province.

It has excellent teachers, rich teaching and research resources, and can provide students with a good learning environment and development opportunities.

In short, it is a high-level university with influence and reputation in China and even in the world.

XXX 是中国的一所著名高水平大学，位于 XXX。

它有着优秀的师资力量、丰富的教学科研资源，能为学生提供了良好的学习环境和发展机会。

总之，是中国乃至世界上具有重要影响力和声誉的高水平理工科学府之一。

6. If you are admitted to our school, what do you hope to gain here?如果被录取你希望收获什么/谈谈你的研究生学习规划。

I want to improve myself in three aspects.

First of all, in terms of professional skills, I want to conduct more research under the guidance of my tutor and improve my ability of doing research.

Secondly, I want to improve my English skills by reading more English papers and writing English essays.

Thirdly, I also want to become more self-reliant, strengthen the critical thinking ability, learn to discover and solve challenges independently.

我想在三个方面提升自己。

首先，在专业技能方面，我想在导师的指导下进行更多的研究，提高自己的研究能力。

其次，我想通过阅读更多的英语论文和写英语论文来提高我的英语技能。

第三，我也想变得更加自力更生，加强批判性思维能力，学会独立发现和解决挑战。

7. What do you like to do in your spare time?你的业余爱好是什么？

My hobby is playing table tennis. Playing table tennis is good for health. It can increase reaction times, improve brain function, relieve stress, and cultivate the spirit of teamwork. Whether it is a leisure activity or a competitive sport, playing table tennis is a very rewarding hobby.

我的爱好是打乒乓球。打乒乓球对健康有益。它可以增加反应时间，改善大脑功能，缓解压力，培养团队合作精神。无论是休闲活动还是竞技运动，打乒乓球都是一种非常有益的爱好。

8. Which extracurricular book do you like best? why?你最喜欢的课外书是什么？为什么？

My favorite book is Harry Potter. It has powerful emotional resonance and touches on profound themes of friendship, family, courage and sacrifice. As a reader, I grow with them, face challenges, build relationships and overcome evil forces.

我最喜欢的书是《哈利波特》。它具有强烈的情感共鸣，触及了友谊、家庭、勇气和牺牲的深刻主题。作为一个读者，我和他们一起成长，面对挑战，建立关系，战胜邪恶势力。

9. Describe your father/ mother.描述一下你的父亲/母亲/家庭。

There are three people in my family: my father, my mother and me.

My father is a hardworking and simple farmer. He is not good at talking and always pays for me silently. He is well aware of the importance of study and has always taught me to study hard and use knowledge to change my destiny.

我的父亲是一位勤劳朴实的农民。他不善言谈，总是默默的为我付出。他深知学习的重

要性，一直教导我要好好学习，用知识改变命运。

My mother is a kind and hardworking woman who likes growing flowers and online shopping. She has always respected my ideas and never forced me to do things I don't like. Her cooking skill is very good, and I often miss her meals when I was in school.

我的母亲是一个善良且勤奋的女人，喜欢种花和网上购物。她一直非常尊重我的想法，从不强迫我去做不喜欢的事情。她的厨艺非常棒，我在学校时经常想念她做的饭菜。

10. Which city do you like best and why?最喜欢的城市是？为什么？

My favorite city is Shanghai which is an international city. It's beautiful and unique. People living in Shanghai come from all over the world, everyone can live together peacefully and find their own sense of belonging in this city. In addition, Shanghai has a developed economy and convenient transportation. There are more opportunities for development here.

我最喜欢的城市是上海，它是一座国际化的城市，它美丽而别致。生活在上海的人来自世界各地，大家都能和平共处，在这座城市中找到属于自己的归属感。另外，上海经济发达，交通便利，在这里发展可以有更多的机会。

11. What has been your greatest success in college?大学最成功的事是什么？

In my freshman year, my grades in mathematics were unsatisfactory. I know its importance well, so in the sophomore year, I worked hard and 99(ninety-nine) points in the final exam of the Discrete mathematics. This is undoubtedly the greatest motivation for me, and it also supports me to continue to study hard.

在大一时，我的数学成绩不是很理想。我深知它的重要性，在大二学年,我不断努力，在离散数学的期末考试中拿到了 99 分，这无疑是对我最大的肯定，也支撑着我继续努力学习。

12. How to view the future of your graduate direction?如何看待你研究生方向的未来？

I believe that the future of the computer industry will be bright. Computers will have a profound impact on various industries, changing work methods, improving production efficiency, and creating more business value.

At the same time, I firmly believe that people can fully develop the potential of computers and AI, guiding them towards the direction that is most beneficial to humanity.

我认为，计算机行业的未来会是前途光明的。计算机将在各行各业产生深远影响，改变工作方式、提升生产效率、创造更多的商业价值。

同时，我坚信人们能够充分发挥计算机和人工智能的潜力，将其引导朝着对人类最有益的方向发展。

二、高等数学★★

13. 解释什么是极限。

在函数中，当自变量趋近于某个特定值时，函数的取值可能会逼近某个确定的数值，这个确定的数值就被称为函数的极限。

定义：

给定一个函数 f(x)，当自变量 x 趋近于某个特定值 x0 时，如果函数的取值 f(x)随着 x 的趋近逼近一个确定的数 A，那么我们称 A 是函数 f(x)在 x 趋近于 x0 时的极限。

对于任意给定的正数ε，存在另一个正数δ，使得当 $0 < |x - x0| < \delta$时，对应的函数值满足 $|f(x) - A| < \varepsilon$。（任意ε，存在δ， $0 < |x - x0| < \delta$，$|f(x) - A| < \varepsilon$）

14. 解释一下罗尔中值定理。

如果一个函数在闭区间上连续，在开区间内可导，并且在区间的两个端点处取相同的值，那么在这个区间内，必然存在至少一个导数为零的点。

15. 解释一下拉格朗日中值定理。★★

如果一个函数在闭区间上连续，在开区间内可导，则至少存在一个点，该点的导数等于函数在区间端点的斜率。

16. 解释一下柯西中值定理。

对于函数 f(x)和 g(x)，如果它们在闭区间[a, b]上连续，并且在开区间(a, b)内可导，那么存在一个点 c∈(a, b)使得：

[f(b) - f(a)]/[g(b) - g(a)] = f'(c)/g'(c)

存在一个点 c，使得函数 f(x)和 g(x)在区间[a, b]上的平均变化率等于它们在点 c 处的瞬时变化率的比值。

17. 三个中值定理的区别、联系和应用。★★★

罗尔中值定理适用于闭区间内连续的函数。当函数在闭区间的端点上取相同的值时，罗尔中值定理保证函数在开区间内至少有一个导数为零的点。

拉格朗日中值定理适用于闭区间内连续且可导的函数。它**是罗尔中值定理的推广**，保证函数在开区间内至少有一个导数等于平均变化率的点。拉格朗日中值定理应用：在某个时间点，质点的瞬时速度等于它在某个时间段内的平均速度。

柯西中值定理适用于两个函数在闭区间内连续且可导。它**是拉格朗日中值定理的推广**，表示两个函数在开区间内的平均变化率等于它们在开区间内某个点的瞬时变化率的比值。

18. 解释一下泰勒公式。★★★

泰勒公式的初衷是用多项式函数来近似表示函数在某点周围的情况。

泰勒公式可用于将一个函数在某个点附近展开成无穷级数的形式。

泰勒公式的一般形式如下：

$f(x) = f(a) + f'(a)(x - a)/1! + f''(a)(x - a)^2/2! + f'''(a)(x - a)^3/3! + ...$

f(x) 是要近似的函数，a 是展开点。

19. 泰勒展开中皮亚诺余项和拉格朗日余项的区别。★★★

皮亚诺余项通常使用函数在展开点的高阶导数来表示，通过将剩余的高阶导数项乘以展开点到近似点的距离的幂次来估计误差大小。

拉格朗日余项使用函数在展开区间内某一点处的高阶导数来表示，并且乘以展开区间长度的适当幂次。拉格朗日余项在整个展开区间上都有较好的估计效果。

总体而言，皮亚诺余项主要用于提供粗略的近似值与截断后误差的估计，而拉格朗日余项则提供更精确的近似值与实际值之间的误差估计。

20. 一阶导和二阶导的物理意义和几何意义是什么？

一阶导数表示函数的变化率或切线的斜率，二阶导数表示一阶导数的变化率。

在物理上，一阶导数可以表示速度，二阶导数可以表示加速度。

在几何上，一阶导数可以表示切线斜率，二阶导数可以表示曲线的凸性和凹性。

21. 一元函数和多元函数可导、可微、连续和可积的关系。★★

一元函数：可导和可微等价。可导一定连续，连续不一定可导（y=|x|，在 x=0 处不可导）。

多元函数：可微一定可导，可微一定连续，偏导连续一定可微，偏导存在不一定连续。

22. 什么是方向导数和梯度★★

方向导数是一个标量。方向导数可以理解为函数**在某个点的某个方向上的**变化速率。

梯度是一个向量。它包含了函数在每个方向上变化最快的信息。对于一个多变量函数 f(x, y, z)，其梯度定义为：

$\nabla f = (\partial f/\partial x, \partial f/\partial y, \partial f/\partial z)$

梯度的方向是函数在某一点上变化最快的方向，而梯度的模长表示了函数在该点上的变化速率（或最大方向导数）。

梯度的几何意义：

函数变化增加最快的地方。**沿着梯度向量的方向，更容易找到函数的最大值。反过来说，沿着梯度向量相反的方向，更容易找到函数的最小值。**

梯度的应用：

势场：在物理场的描述中，梯度常常用来表示场的强度和方向。例如，在电场中，梯度表示电势场的变化率。

最速下降法：梯度在优化问题中发挥重要作用。它利用梯度的负方向来搜索函数的最小值。根据最速下降法，沿着梯度的负方向进行迭代更新可以逐步接近函数的最小值。

方向导数和梯度的关系：

如果我们知道了函数在某一点的梯度向量，以及一个表示方向的单位向量，就可以计算出函数在这个方向上的方向导数。

方向导数=某点的梯度与给定方向的方向余弦做内积（点乘）。

23. 什么是傅里叶级数和傅里叶变换。★★

傅里叶级数：任何周期性函数若满足狄利克雷条件，那么该函数可以用正弦函数和余弦函数构成的无穷级数来表示。

傅里叶变换：可以处理**非周期性信号**（一个信号可以看成一个周期性无穷大 T->∞的信号）。傅里叶变换将一个信号从**时域转换到频域**，得到该信号的频谱。

①傅里叶变换

$$F(\omega) = \mathcal{F}[f(t)] = \int_{-\infty}^{\infty} f(t)e^{-i\omega t}dt$$

②傅里叶逆变换

$$f(t) = \mathcal{F}^{-1}[F(\omega)] = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(\omega)e^{i\omega t}d\omega$$

应用：

通过傅里叶变换，我们可以对信号进行滤波、降噪、压缩、频谱估计等操作。这些技术在音频处理、图像处理、视频压缩、通信系统等领域都有广泛应用。

三、线性代数★★★

24. 什么是矩阵的秩？★★★

1) 基本概念

k 阶子式：在一个矩阵或行列式中取 k 行 k 列，交叉处的 k^2个元素按原顺序构成的行列式。

[1] 从子式的角度定义：矩阵的秩就是矩阵中非零子式的最高阶数。

[2] 从极大线性无关组的角度定义：矩阵的所有行向量中极大线性无关组的元素个数。

[3] 从标准型的角度定义：求一个矩阵的秩，可以先将其化为行阶梯型，非零行的个数即为矩阵的秩。

25. 什么是线性相关和线性无关？★★★

线性相关：

如果存在不全为零的系数，使得向量集合果中的某些向量的线性组合等于零向量，那么这些向量就被称为线性相关的。换句话说，如有向量 v_1、v_2、v_3、...、v_n，并且存在不全为零的标量 c_1、c_2、c_3、...、c_n，使得 $c_1v_1+ c_2v_2+ c_3v_3+ ... + c_nv_n= 0$，则这些向量就是线性相关的。

线性无关：

如果向量集合中的向量无法通过任何非平凡的线性组合（即非所有系数都为零）得到零向量，那么这些向量就是线性无关的。换句话说，如果 $c_1v_1 + c_2v_2 + c_3v_3 + ... + c_nv_n = 0$ 的唯一解是 $c_1=c_2=...=c_n=0$，则这些向量就是线性无关的。

26. 一个矩阵线性无关的等价定义有什么？

非奇异矩阵（行列式不为 0）、矩阵可逆、矩阵满秩、特征值没有 0。

27. 向量组的极大无关组和向量组的秩。

极大无关组是指在向量组中选择尽可能多的线性无关向量，使得这个子集仍然保持线性无关，并且再添加任何其他向量，就会使得它变得线性相关。

向量组的秩等于它的极大无关组合中向量的个数。

28. 向量空间（线性空间），向量空间的基与维数。

n 维向量构成的非空集合 V，对于向量加法和数乘两种运算封闭。则这个 V 是向量空间。

基：

设 V 是一向量空间，a1,a2,...,ar∈V 且满足:

a) a1,a2,...,ar 线性无关;

b) V 中向量均可由 a1,a2,..., ar 线性表示

则称 a1,a2,...,ar 为 V 的一个基。

维数

基中所含向量个数 r 称为向量空间的维数

29. 什么是矩阵的特征值与特征向量？特征值有什么应用？★★★★

定义：

给定一个 n×n 的方阵 A，若非零向量 x 满足 Ax = λx，那么λ称为 A 的**特征值**。非零向量 x 称为 A 的**特征向量**。

也就是说，特征向量 x 在经过矩阵 A 的线性变换后，只会改变长度但不会改变方向，而特征值则表示该变换的缩放比例。

计算：

|λE-A|=0 从而求得所有λ。将λi 带回(λiE-A)x=0 那么可求得方程组的基础解系，特征值为λi 的特征向量就是基础解系的线性组合。

性质：

所有特征值的积是该矩阵的行列式（行列式的本质是特征值的乘积），所有特征值的和是该矩阵的迹。

应用：

特征空间变换：特征向量可以用于将矩阵对角化，从而简化线性变换的描述。这在计算中能够提高效率。

图像处理：特征值和特征向量可以用于图像压缩、降噪和特征提取等领域。例如，主成分分析（PCA）方法就利用了特征向量来提取图像中的关键特征。

数据分析：特征值可以用于降维和数据拟合。例如，在主成分分析中，我们可以通过保留最大的特征值对应的特征向量进行数据降维，从而捕捉数据的主要变化趋势。

30. 特征值为 0 和矩阵的秩的关系。★★

如果 λ = 0，则有以下结论：

1) A 的秩小于 n

如果 A 的某个特征值为 0，那么 A 的秩必定小于 n。这是因为特征值为 0 表明 A 不是满秩矩阵，存在线性相关的列向量，导致 A 的秩小于 n。

2) A 的行向量和列向量中至少有一个线性相关

如果 λ = 0，那么对应的特征向量 x 满足 Ax = 0。由于 x ≠ 0，说明存在一个非零的向量使得 A 与它的乘积为零。这意味着 A 的行向量和列向量中至少有一个线性相关。

特征值为 0 并不意味着矩阵 A 的秩一定为 0。

31. 什么是相似矩阵和相似对角化？★★

相似矩阵定义：

两个 n×n 的矩阵 A 和 B 称为相似矩阵，如果存在一个可逆矩阵 P，使得 B = P^(-1) AP。

相似矩阵性质：

相似矩阵具有相同的特征值、秩、行列式。

相似矩阵的特征向量可以通过相似变换得到。

相似对角化定义：

相似对角化是指将一个矩阵 A 通过相似变换 P^(-1) AP 转化为对角矩阵 D 的过程。D 的对角线上的元素就是 A 的特征值，P 的列向量就是 A 的特征向量。

并不是所有的矩阵都可以相似对角化。一个矩阵可对角化的充要条件是**它具有 n 个线性无关的特征向量**。

32. 什么是对称矩阵？对称矩阵有什么性质？★★★

定义：

方阵 A 中，aij = aji，则 A 为对称矩阵，或 A^T = A。若 A 还是实矩阵，则 A 为实对称矩阵。

性质：

特征值都是实数，特征向量都是实向量。

不同特征值对应的特征向量正交。

对称矩阵可以通过**正交相似变换**对角化，即存在正交矩阵 P，使得 (P^T)AP = D，D 是对角矩阵。

（补充：正交矩阵就是满足 PP^T=E）

33. 什么是二次型？什么是正定矩阵？正定矩阵的性质是什么？★

二次型：

二次型是一个关于变量的二次多项式表达式。对于 n 维向量 $x = [x_1, x_2, ..., x_n]$，二次型可以表示为 Q(x) = (x^T)A x，其中 A 是一个对称矩阵。

正定矩阵：

给定一个 n 阶实对称矩阵，若对于任意长度 n 的非零向量 x，有(x^T)Ax>0 恒成立，则 A 是一个正定矩阵。

正定矩阵的性质：

（1）正定矩阵是实对称矩阵，即 A = A^T。这意味着矩阵的元素关于主对角线对称。

（2）所有特征值都大于 0，行列式|A| > 0，秩不是满的；是可逆的，其逆矩阵也是正定的。

（3）如果两个矩阵 A 和 B 都是正定矩阵，A+B 和 AB 也是正定的。

（4）正定矩阵的 n 次方仍然是正定矩阵。

半正定矩阵：

给定一个 n 阶实对称矩阵，若对于任意长度 n 的非零向量 x，有(x^T)Ax≥0 恒成立，则 A 是一个半正定矩阵。所有特征值都大于等于 0。

34. 什么是矩阵合同？★

若 A 和 B 是两个方阵，若存在可逆矩阵 Q，使得 B=(Q^T)AQ 成立，则 A 与 B 合同。

合同的充要条件：对 A 和行和列施以相同的初等变换变成 B。

四、概率论与数理统计★★★

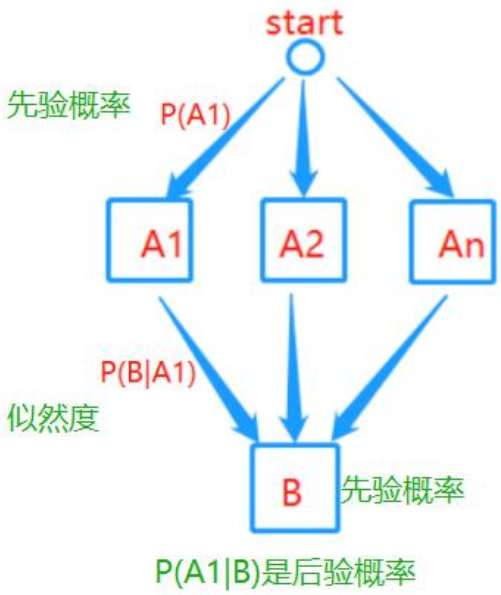
35. 什么是条件概率？

在事件 B 发生的条件下，事件 A 发生的概率记为 P(A|B)，读作“A 在 B 条件下发生的概率”。

条件概率的计算公式如下： P(A|B) = P(AB) / P(B)

P(AB)表示事件 A 和事件 B 同时发生的概率。

36. 什么是全概率公式和贝叶斯公式？★★★★★★



全概率公式：

如果有一些互斥事件 A1,A2,……,An，它们的并集是全集，则任何事件 B 发生的概率可以拆分为每一个 Ai∩B 的概率和。

P(B)=P(A1)*P(B|A1)+P(A2)*P(B|A2)+...+P(An)*P(B|An)

机器学习中的贝叶斯公式：

P(A|B) = (P(B|A) * P(A)) / P(B)

P(A|B) 是后验概率，P(B|A) 是似然度，P(A) 表示事件 A 的先验概率，P(B) 表示事件 B 的先验概率。

贝叶斯公式的实际意义：

贝叶斯公式将我们对于事件 A 发生的先验概率与新获得的证据（似然度）相结合，从而得到在给定证据的情况下事件 A 发生的后验概率。

例如：

P(咳嗽患支气管炎)=P(支气管炎有咳嗽症状)*P(支气管炎)/P(咳嗽)

P(支气管炎有咳嗽症状)是似然度，容易求得。P(支气管炎)和 P(咳嗽)是先验概率。这样

帮助医生根据症状预测患者患某种病的概率。

贝叶斯公式应用：

机器学习中贝叶斯分类器：根据已有的训练样本和特征信息，利用贝叶斯公式计算不同类别的后验概率，从而**进行分类**任务。

医学诊断与预测：通过将病人的先验概率与各种医学测试的似然度相结合，可以计算出某种疾病的后验概率，辅助医生进行诊断和预测。

37. 什么是先验概率和后验概率？★★★★★

先验概率：

事情未发生，只根据以往数据统计，分析事情发生的可能性，即先验概率。

后验概率（贝叶斯公式）：

事情已发生，已有结果，求引起这事发生的因素的可能性，由果求因，即后验概率。

后验概率和先验概率的关系：

后验概率的计算，是以先验概率为前提条件的。如果只知道事情结果，而不知道先验概率，是无法计算后验概率的。

38. 正态分布有什么性质？

（1）对称性：正态分布是概率密度函数关于其均值μ的对称分布，即在平均值两侧呈镜像对称。

（2）唯一峰值：正态分布的概率密度函数呈现单峰形状，只有一个最高峰值。

（3）分布范围无界：正态分布的取值范围是负无穷到正无穷，没有明确的上下界限。

（4）标准差决定形状：正态分布的形状由其标准差σ决定。较小的标准差会使曲线更加陡峭，较大的标准差会使曲线更加平坦。

（5）68-95-99.7 规则：在正态分布中，约有 68%的观测值落在均值的一个标准差范围内，约有 95%的观测值落在两个标准差范围内，约有 99.7%的观测值落在三个标准差范围内。

（6）中心极限定理：多个随机变量的总和（或平均值）趋向于正态分布，即使原始随机变量不满足正态分布，这是中心极限定理的重要推论。

39. 什么是二元正态分布？有什么性质？

二元正态分布是由两个连续随机变量组成的概率分布。它通常用来描述两个变量之间的关系和相互依赖性。每个变量都服从正态分布，而二元正态分布则描述了这两个变量之间的联合分布。

性质：

（1）边缘分布：在二元正态分布中，每个随机变量的边缘分布都是正态分布。也就是说，如果我们只考虑其中一个变量，将另一个变量积分或求和消除，所得到的分布将是单个变量的正态分布。

（2）条件分布：二元正态分布中的每个随机变量的条件分布也是正态分布。条件分布是指在已知另一个变量的取值后，考虑该变量的分布。换句话说，给定一个变量的取值后，另一个变量的条件分布仍然是正态分布。

（3）相关性：二元正态分布中的两个变量之间存在线性相关性。相关系数 ρ 衡量了这种相关性的强度，它的取值范围为 -1 到 1。当相关系数为 0 时，表示两个变量相互独立；当相关系数为正值时，表示两个变量呈正相关关系；当相关系数为负值时，表示两个变量呈负相关关系。

（4）协方差：二元正态分布中的两个变量之间存在协方差。协方差描述了两个变量的线性关系程度，它的值可以为正、负或零。当协方差为正值时，表示两个变量呈正相关关系；当协方差为负值时，表示两个变量呈负相关关系；当协方差为零时，表示两个变量无线性相关性。

40. 什么是协方差和相关系数？★★

协方差：

协方差是用来衡量两个随机变量之间的总体**线性关系强度和方向的指标**。对于随机变量 X 和 Y，其协方差记为 Cov(X,Y)。协方差的计算公式如下：Cov(X,Y)=E((X-EX)(Y-EY))。

相关系数（皮尔逊相关系数）：

相关系数是通过归一化协方差得到的，用来度量两个变量之间线性相关程度的指标。

ρ(X, Y) = Cov(X, Y) / (σx * σy)，σx 和 σy 分别表示 X 和 Y 的标准差。

相关系数的取值范围在 -1 到 1 之间，-1 代表完全的负相关，1 代表完全的正相关，0 代表无线性相关。相关系数绝对值越接近 1，相关程度就越强。

其他：

独立一定不相关，不相关不一定独立。

41. 什么是概率密度函数？★★

概率密度函数是用来描述连续型随机变量的概率分布的函数。对于一个连续型随机变量 X，其概率密度函数 f(x) 定义了在给定区间内，随机变量 X 取某个特定取值的概率密度。

概率密度函数 f(x) 是一个非负函数，并且满足以下两个性质：

1）非负性：对于任意实数 x，概率密度函数满足 f(x)≥0。

2）归一性：概率密度函数的总体积分等于 1，即∫f(x)dx=1。

EX=∫ xf(x)dx，DX=∫ x^2f(x)dx

42. 什么是切比雪夫不等式？★★

切比雪夫不等式给出了随机变量与其期望值之间的偏离程度的一个上界。

设随机变量 X 具有数学期望 E（X）=μ，方差 D（X）=σ^2，
则对于任意正数 ε，有下列切比雪夫不等式

$$P(|X-\mu|\geq\varepsilon)\leq\frac{\sigma^2}{\varepsilon^2}$$

43. 什么是大数定律和中心极限定理？★★★★★★

大数定律：

对于独立同分布的随机变量序列 {X1, X2, X3, ..., Xn}，随着样本容量 n 的增大，样本均值的极限将趋于随机变量的期望值。换句话说，样本均值在大样本情况下趋于稳定并接近总体均值。

三种大数定律（了解即可）：

切比雪夫大数定律：样本数量 n 充分大时，n 个是独立随机变量的平均数的离散程度很小。

伯努利大数定律：将试验进行多次，随机事件的频率接近概率。

辛钦大数定律：只要验证数学期望是否存在，就可判断其是否服从大数定律。

中心极限定理：

如果随机变量 X1, X2, ..., Xn 是独立同分布的，当样本容量 n 足够大时，样本的**均值**将近似地服从正态分布，即使原始总体并不服从正态分布。

例如：10000 人参加保险，一年内参加投保的人死亡的概率是 0.006。设一年内参加投保的死亡人数 X，X~B(10000，0.006)

Ex=60，Dx=7.72^2

根据中心极限定理，近似 X~N(60，7.72^2)。

44. 什么是最大似然估计（极大似然估计）？★★★★

最大似然估计就是一种参数估计方法。

原理：

利用已知的样本结果，反推最大概率导致这样结果的参数值。（根据结果推出参数）

方程的解θ只是一个估计值，只有在样本数趋于无限多的时候，它才会接近于真实值。

求最大似然估计量θ的一般步骤：

[1] 写出似然函数；

[2] 对似然函数取对数便于计算，并整理；

[3] 求导数；

[4] 解似然方程求出参数。

应用：

最大似然估计被用于参数估计、模型选择、假设检验等许多问题。

五、离散数学★

45. 命题逻辑和谓词逻辑有什么区别？

命题逻辑关注的是命题，即可以判断为真或假的陈述句。

谓词逻辑引入了谓词的概念，谓词是带有参数的函数。

46. 什么是幂集？

集合的全体子集构成的集合叫做幂集。

A={1,2}

P(A)={ϕ，{1}，{2}，{1,2}}

|A|=2，|P(A)|=4

47. 什么是笛卡尔积？

设有两个集合 A 与 B，以 A 的元素作为第一个分量，以 B 的元素作为第二个分量，用这种方式所组成的有序偶的全体构成一个集合。

假设集合 A={a,b}，集合 B={0,1,2}，则两个集合的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。

48. 什么是二元关系？

定义：

二元关系是指定义在两个集合上的关系，它是有序对的集合。（A 和 B 笛卡尔积的子集）

设 A 和 B 是两个集合，一个二元关系 R 定义在这两个集合上是 $A \times B$ 的子集，即 $R \subseteq A \times B$。在二元关系 R 中，如果 $(a, b) \in R$，则称 a 与 b 存在该关系，并表示为 aRb。

二元关系可能具有的性质：

自反性：对于集合 A 中的每个元素 a，都有 aRa。

对称性：如果有 aRb，那么也有 bRa。

反对称性：如果有 aRb 且有 bRa，那么必须有 a = b。

传递性：如果有 aRb 和 bRc，那么必须有 aRc。

类型：

等价关系：满足自反性、对称性、传递性的关系。

偏序关系：满足自反性、反对称性、传递性的关系。

全序关系：满足偏序关系，并且任意两个元素都是可比较的。

49. 偏序关系和等价关系的区别是什么？★★

偏序关系是一种二元关系，用于比较元素之间的相对顺序。对于集合中的任意两个元素，可以根据某种规则来判断它们的顺序关系，即元素之间可以是无序、大于等于或小于等于的

关系。

等价关系是集合上的一种二元关系，用于刻画元素之间的等价关系，即元素之间的相等性。

等价关系：满足自反性、对称性、传递性的关系。

偏序关系：满足自反性、反对称性、传递性的关系。

50. 什么是代数系统、半群、群、阿贝尔群？

代数系统：

非空集合 S 和 S 上 k 个一元或二元运算 $f_1, f_2, \ldots, f_k$ 组成的系统称为**代数系统**，简称**代数**，记作 $< S, f_1, f_2, \ldots, f_k >$.

半群：

满足**结合律**的**封闭**代数系统（结合律、封闭性）

群：

具有**单位元 e 与逆元 a-1** 的半群（结合律、封闭性、单位元、逆元）

交换群（阿贝尔群）：

群 G 中的二元运算**可交换**（交换律、结合律、封闭性、单位元、逆元）

51. 什么是哈密顿图和欧拉图？★

哈密顿图（经过所有点一次且仅有一次）：

在一个无向图中，如果存在一个哈密顿路径，即**一条经过图中每个顶点恰好一次的路径**，那么这个图就被称为哈密顿图。如果这个路径的起点和终点相连，形成一个环，那么图就是**哈密顿环**。

欧拉图（经过所有边一次且仅有一次）：

在一个连通的图中，如果存在一条路径，它**经过图中每条边恰好一次**，那么这个图就被称为欧拉图。如果这个路径的起点和终点相同，形成一个闭合回路，那么图就是**欧拉回路**。

无向图存在欧拉回路的充要条件：图所有顶点度数均为偶数，且该图是连通图

有向图存在欧拉回路的充要条件：所有顶点的入度=初度，且该图是连通图

六、数据结构与算法★★★★★

52. 介绍一下时间复杂度和空间复杂度。

时间复杂度：

时间复杂度衡量了算法运行所需的时间资源。时间复杂度分析关注的是算法在最坏情况下的时间消耗。

空间复杂度：

空间复杂度衡量了算法运行所需的存储空间。空间复杂度分析关注的是算法在最坏情况下所使用的额外空间。

53. 有哪些存储结构，以及这些存储结构的优缺点。★★

（1）顺序存储

把逻辑上相邻的元素存储在物理位置上也相邻的存储单元中，元素之间的关系由存储单元的邻接关系来体现。

优点：随机存取，每个元素占用最少的存储空间（存储密度高）；

缺点：只能使用相邻的一整块存储单元，因此可能产生较多的外部碎片。

（2）链式存储

不要求逻辑上相邻的元素在物理位置上也相邻，借助指示元素存储地址的指针来表示元

素之间的逻辑关系。

优点：是不会出现碎片现象，能充分利用所有存储单元（利用率高）；

缺点：是每个元素因存储指针而占用额外的存储空间（指针开销），且只能实现顺序存取。

（3）索引存储

在存储元素信息的同时，还建立附加的索引表。索引表中的每项称为索引项，索引项的一般形式是（关键字，地址）。

优点：是检索速度快；

缺点：是附加的索引表额外占用存储空间。另外，增加和删除数据时也要修改索引表，因而会花费较多的时间。

（4）散列存储

根据元素的关键字直接计算出该元素的存储地址，又称哈希存储。

优点：是检索、增加和删除结点的操作都很快；

缺点：是若散列函数不好，则可能出现元素存储单元的冲突，而解决冲突会增加时间和空间开销。

54. 循环比递归的效率高吗？★

循环和递归两者是可以互换的，不能决定性的说循环的效率比递归高。

递归

优点：代码简洁清晰，容易检查正确性；

缺点：当递归调用的次数较多时，要增加额外的堆栈处理，有可能产生堆栈溢出的情况，对执行效率有一定的影响。

循环

优点：结构简单，速度快；

缺点：它并不能解决全部问题，有的问题适合于用递归来解决不适合用循环。

55. 线性表包括顺序表和链表，请比较它们的区别。★★

1）存取（读写）方式

顺序表可以顺序存取，也可以随机存取。

链表只能顺序存取。

2）逻辑结构与物理结构

顺序存储：逻辑上相邻的元素，物理存储位置也相邻。

链式存储：逻辑上相邻的元素，物理存储位置不一定相邻，对应的逻辑关系是通过指针链接来表示的。

3）查找、插入和删除操作

查找：

对于按值查找，顺序表无序时，两者的时间复杂度均为 O(n); 顺序表有序时，可采用折半查找，此时的时间复杂度为 O(logn) 。

对于按序号查找，顺序表支持随机访问，时间复杂度仅为 0(1), 而链表的平均时间复杂度为 O(n)。

插入、删除：

顺序表的插入、删除操作，平均需要移动半个表长的元素。

链表的插入、删除操作，只需修改相关结点的指针域即可。由于链表的每个结点都带有指针域，故而存储密度不够大。

56. 说说栈和队列的区别。★★

栈和队列都是操作受限的线性表。

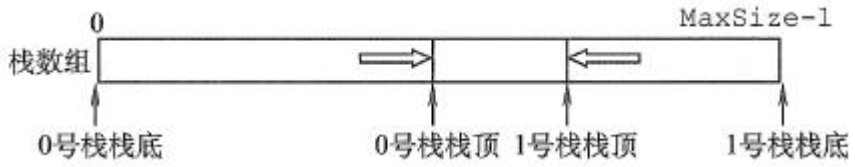
栈

对于插入到栈的元素按“后进先出”的规则处理，插入和删除操作都在栈顶进行，一般用定长数组存储栈元素。由于进栈和出栈都是在栈顶进行，因此要有一个 size 变量来记录当前栈的大小。

队列

允许在一段进行插入另一端进行删除的线性表。队列顾名思义就像排队一样，对于进入队列的元素按“先进先出”的规则处理，在表头进行删除在表尾进行插入。

57. 简要说说共享栈。★



让两个顺序栈共享一个一维数组空间，将两个栈的栈底分别设置在共享空间的两端，两个栈顶向共享空间的中间延伸。这样能够更有效的利用存储空间，只有在整个存储空间被占满时才发生溢出。

58. 如何区分循环队列是队空还是队满？★★

普通情况下，循环队列队空和队满的判定条件是一样的，都是 Q.front == Q.rear。

为了区分可采用两种方法：

方法一：牺牲一个单元来区分队空和队满，这个时候(Q.rear+1)%MaxSize == Q.front 才是队满标志。

方法二：类型中增设表示元素个数的数据成员。这样，队空的条件为 Q.size == 0；队满的条件为 Q.size == MaxSize。

59. 说说栈在括号匹配中的算法思想。★★

1）出现的凡是左括号，则进栈；

2）出现的是右括号，如果栈不空而且栈顶元素是左括号，那么相匹配，否则不匹配。

3）表达式检验结束时，如果栈空则表明表达式中匹配正确，否则表明“左括号”有余

60. 说说栈在后缀表达式求值的算法思想。★★

例如 ABCD-*+ED/-。

顺序扫描表达式的每一项，然后根据它的类型做如下相应操作：

若该项是操作数，则将其压入栈中；

若该项是操作符，则连续从栈中退出两个操作数 y 和 x，形成运算指令 XY，并将计算结果重新压入栈中。

当表达式的所有项都扫描并处理完后，栈顶存放的就是最后的计算结果。

61. 说说栈在计算机系统中的应用。★★

1）函数调用：栈在函数调用中扮演着重要角色。每当一个函数被调用时，函数的参数、局部变量和返回地址等信息都会被保存在栈中的帧中。函数执行完毕后，相应的帧会被从栈中弹出，恢复上一个函数的执行。

2）表达式求值：编译器和解释器通常使用栈来求解表达式。例如，中缀表达式转换为后缀表达式时，使用栈来调整操作符的顺序。

3）括号匹配：栈也常用于括号匹配的问题。通过遍历输入字符串并将左括号入栈，当遇到右括号时，检查栈顶元素是否与之匹配。可以利用栈的后进先出（LIFO）特性来快速判断括号是否匹配。

62. 说说队列在计算机系统中的应用。★★

1）任务调度：操作系统中的任务调度器通常使用队列数据结构来管理和调度进程或线程。

2）缓冲区管理：网络通信、磁盘 I/O 等场景中常需要使用队列来处理数据的缓冲区。

3）消息传递：消息队列是实现异步通信和解耦的重要方式。多个组件之间通过将消息放入队列来进行通信，接收者可以从队列中取出并处理消息。

63. 介绍一下 KMP 算法。★★★

KMP 算法是一种高效的字符串匹配算法，用于在一个文本串中查找一个模式串的出现位置。KMP 算法通过利用模式串自身的信息，在匹配过程中避免不必要的回溯，从而提高匹配效率。

KMP 算法的核心思想是使用一个部分匹配表，也称为 next 数组，来记录模式串中每个位置的最长公共前后缀的长度。这样，在匹配失败时，可以根据部分匹配表的信息，将模式串向右移动尽可能少的步数。

KMP 算法的时间复杂度 O(n+m)，朴素算法的时间复杂度 O(n*m)，n 和 m 是两个串的长度。

KMP 算法的具体步骤如下：

1）预处理 next 数组：对于模式串，遍历每个位置，计算该位置之前子串的最长公共前后缀的长度，并保存到 next 数组中。

2）匹配过程：从文本串的起始位置开始，用两个指针分别指向文本串和模式串的当前位置，逐个字符进行比较。

如果当前字符匹配成功，则两个指针同时向后移动一位。

如果当前字符匹配失败：

根据 next 数组中的信息，将模式串向右移动尽可能少的步数。根据当前失败位置的部分匹配值，向右移动模式串的指针。

同时，保持文本串的指针不动，继续与模式串的新位置进行比较。

3）如果模式串的指针移到末尾，则表示匹配成功，返回在文本串中的起始位置。如果文本串的指针移到末尾，则表示未找到匹配，返回-1。

64. 满二叉树和完全二叉树有什么区别？★★

满二叉树：

对于一颗高为 h 的二叉树，结点个数为 2^h-1，表现为除了最后一层叶子结点之外，根节点以及分支结点都有两个孩子，即每一层都是满的。

完全二叉树：则是在满二叉树的基础上，在最后一层从右往左依次删除一定数量的叶子结点所形成的二叉树。完全二叉树的特点是叶子结点只出现在倒数第一和第二层，且如果有分支结点仅有一个孩子，那只能是左孩子。

满二叉树和完全二叉树可以用顺序存储结构来存储。

65. 如何由遍历序列构造一棵二叉树？

1) 由二叉树的先序序列和中序序列可以唯一地确定一棵二叉树。

在先序遍历序列中，第一个结点一定是二叉树的根结点；

而在中序遍历中，根结点必然将中序序列分割成两个子序列，前一个子序列是根结点的左子树的中序序列，后一个子序列是根结点的右子树的中序序列。根据这两个子序列，在先序序列中找到对应的左子序列和右子序列。在先序序列中，左子序列的第一个结点是左子树的根结点，右子序列的第一个结点是右子树的根结点。如此递归地进行下去，便能唯一地确定这棵二叉树。

2) 由二叉树的后序序列和中序序列也可以唯一地确定一棵二叉树。

因为后序序列的最后一个结点就如同先序序列的第一个结点，可以将中序序列分割成两个子序列，然后采用类似的方法递归地进行划分，进而得到一棵二叉树。

3) 由二叉树的层序序列和中序序列也可以唯一地确定一棵二叉树。

需要注意的是，若只知道二叉树的先序序列和后序序列，则无法唯一确定一棵二叉树。

66. 简要说说线索二叉树。★

对于 n 个结点的二叉树，在二叉链存储结构中有 n+1 个空链域（n+1 是怎么来的：m=n-1，2n-m=n+1），利用这些空链域存放在某种遍历次序下该结点的前驱结点和后继结点的指针，这些指针称为线索，加上线索的二叉树称为线索二叉树。

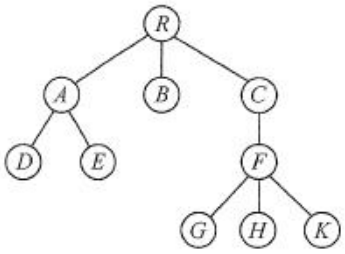
根据线索性质的不同，线索二叉树可分为前序线索二叉树、中序线索二叉树和后序线索二叉树三种。

线索二叉树解决了无法直接找到该结点在某种遍历序列中的前驱和后继结点的问题。

67. 简要说说树的存储结构。★

1）双亲表示法

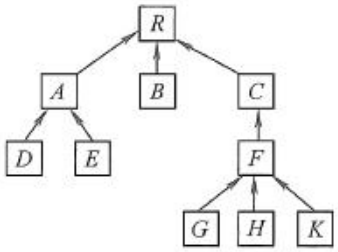
这种存储方式采用一组**连续空间来存储每个结点**，同时在每个结点中增设一个**指针**，指示其双亲结点在数组中的位置。



(a) 一棵树

	data	parent
0	R	-1
1	A	0
2	B	0
3	C	0
4	D	1
5	E	1
6	F	3
7	G	6
8	H	6
9	K	6

(b) 双亲表示

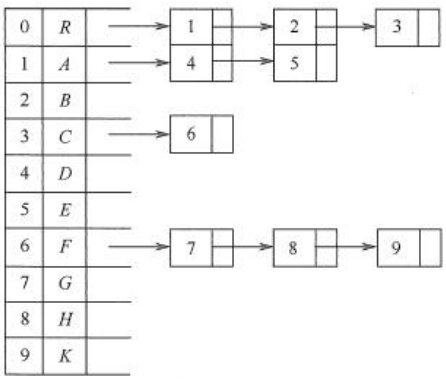


(c) 双亲指针图示

该存储结构可以很快得到每个结点的双亲结点，但求结点的孩子时需要遍历整个结构。

2）孩子表示法

孩子表示法是将**每个结点的孩子结点都用单链表链接起来**形成一个线性结构，此时 n 个结点就有 n 个孩子链表（叶子结点的孩子链表为空表）

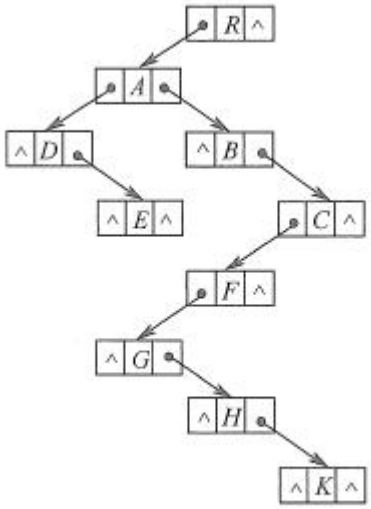


(a) 孩子表示法

这种存储方式寻找孩子容易，而寻找双亲的操作麻烦。

3）孩子兄弟表示法

孩子兄弟表示法以二叉链表作为树的存储结构。孩子兄弟表示法使每个结点包括三部分内容：结点值、指向结点第一个孩子结点的指针，及指向结点下一个兄弟结点的指针。



(b) 孩子兄弟表示法

这种存储表示法比较灵活，其最大的优点是可以方便地实现树转换为二叉树的操作。这种存储方式寻找孩子容易，而寻找双亲的操作麻烦。若为每个结点增设一个 parent 域指向其父结点，则查找结点的父结点也很方便。

4）二叉树的存储

①顺序存储：

按照二叉树层序遍历的顺序将结点存储于顺序表中，特别注意空节点也需要占有位置。若某结点下标为 i，则其左孩子下标为 2i，右孩子下标为 2i+1，父节点下下标为 i/2 向下取整。该存储结构适合存储完全二叉树；

②链式存储：

每个结点通常一个数据域与两个指针域，分别指向自己的左孩子和右孩子。而为了充分利用左右孩子指针，可以将左孩子指向自己的前序、中序或后序遍历的直接前序，右孩子指向直接后继，从而形成二叉线索树，方便查找。

68. 简要说说二叉搜索树。★★

二叉搜索树性质：非空左子树的所有键值小于其根节点的键值；非空右子树的所有键值大于其根节点的键值；左右子树都是二叉搜索树。

对二叉排序树进行中序遍历，即可以得到从小到大有序的关键码序列。它利用了二分的思想，可以快速查找到关键码，查找效率为 O（nlogn）。

缺点：如果按照从小到大插入构建 BST，会导致查找效率退回 O(n)级别。

改进：插入时要旋转保证平衡因子绝对值不大于 1，于是产生了 AVL 树。

69. 简要说说平衡二叉树。★★★

二叉平衡树是一种特殊的二叉排序树，它满足对于树上的任意一个结点，其左子树的深度与右子树的深度之差的绝对值不超过 1。平衡因子可以用于描述二叉平衡树，平衡因子是某个结点的左子树深度与右子树深度之差，对于一棵二叉平衡树，其任意结点的平衡因子只能是-1，0 或 1。

缺点：如果插入操作比查询操作多，AVL 就要花费大量开销做旋转来调整节点以保证树的平衡。

改进：为了减少旋转开销，引入了红黑树。

70. 简要说说二叉搜索树的查找、插入和删除过程。★★★

查找：

从根节点开始，如果要查找的关键码大于当前关键码，则下一个查找的结点为根节点的右子树，反之则是左子树。再以新节点为根，重复以上的查找步骤，直到查到得到匹配的关键码为止。

插入：

基于查找操作进行，查找合适的位置进行插入。该合适的位置指的是按照查找步骤进行到的叶子节点处，若欲插入的关键码大于该叶子结点，则插入为右孩子，反之为左孩子。插入的结点必须是叶子结点。若开始树空，则直接成为根节点；若欲插入的关键码已存在，则插入失败。二叉树的构造过程也是不断插入的过程。

删除：

同样是基于查找操作，首先查找到欲删除的结点。此时，删除结点通常包括三种情况

①若删除的结点是叶子结点，则可以直接将结点删去；

②若删除的结点只有左孩子或者右孩子，则用它的孩子代替它；

③若删除的结点有左右孩子，则可以寻找其中序遍历的**直接前驱**或者**直接后继**代替它，再删去该直接前驱或直接后继。

71. 简要说说红黑树。★★

红黑树中的每一个结点的颜色不是黑色就是红色。根结点和所有外部结点（NULL 节点、叶节点）的颜色是黑色。从根结点到外部结点的途中没有连续两个结点的颜色是红色。所有从根到外部结点的路径上都有相同数的黑色结点数量。对于红黑树搜索的时间复杂度为 O(logn)。

72. 简要说说 B 树。★★★★

引入 B 树的原因：

若普通二叉树作为文件系统的索引，随着数据的插入，发现树的深度会变深。而文件系统的索引在磁盘上，磁盘的数据要加载到内存中才能处理，需要反复 IO 影响查询的效率，于是引入了 B 树可以作为文件系统的索引。

B 树是多叉树，一棵 m 阶 B 树的性质：

1）每个非根节点最多 m-1 个关键字，最少 ceil(m/2)-1 个关键字；最多有 m 个分叉，最少有 ceil(m/2)个分叉。

2）根节点最多 m-1 个关键字，最少 1 个关键字；最多有 m 个分叉，最少有 2 个分叉

3）所有的失败结点（叶子节点）都位于同一层。

4）B 树每个节点可以存放键值和数据。

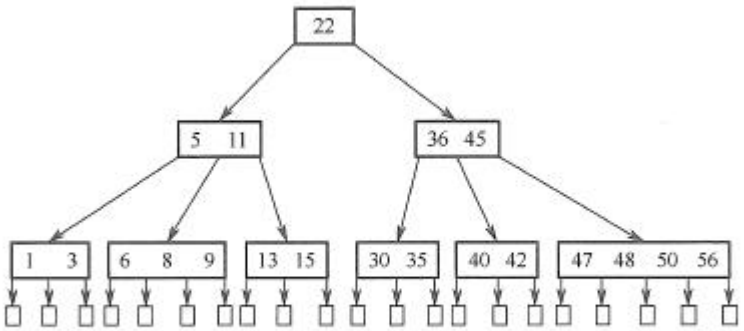


图 7.4　一棵 5 阶 B 树的实例

73. 简要说说 B+树。★★★★

引入 B+树的原因：

由于 B 树每个节点存放数据，而数据相比关键字占用的空间较大，会导致每个磁盘块存放的索引项的记录会变少。B+树的非叶子节点不存放数据，只存放指针和关键字，这样每个磁盘块就可以存放更多的记录。这样深度会减小很多，加快了 IO 速度。

一棵 m 阶的 B+树需满足下列条件：

1）每个非根节点最多 m 个关键字，最少 ceil(m/2)个关键字；最多有 m 个分叉，最少有 ceil(m/2)个分叉。

2）根节点最多 m 个关键字，最少 1 个关键字；最多有 m 个分叉，最少有 1 个分叉

3）结点的子树个数与关键字个数相等。

4）所有叶结点包含全部关键字及指向相应记录的指针，叶结点中将关键字按大小顺序排列，并且相邻叶结点按大小顺序相互链接起来。

5）所有分支结点（可视为索引的索引）中仅包含它的各个子结点中关键字的**最大值**及指向其子结点的指针。

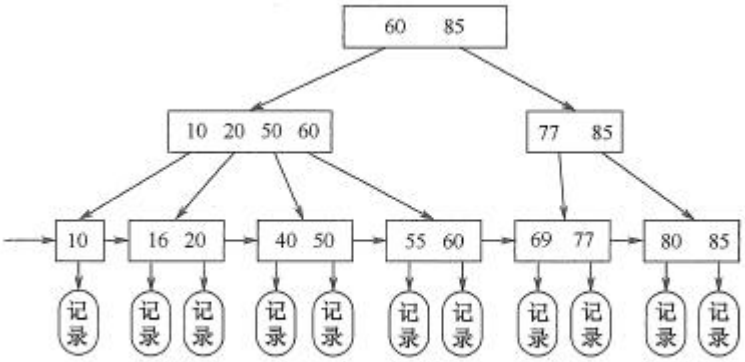


图 7.8　B+树结构示意图

B+树的应用：

在数据库中，B+树常被用作索引结构，用于快速查找和排序大量数据。如主键索引、唯一索引、辅助索引等。

在文件系统中，B+树通常用于管理磁盘上的文件块和索引节点。

74. 简要说说 B 树和 B+树的区别。★★★★★

1）m 阶 B 树和 B+树：B 树的根节点关键字个数取值 1 到 m-1，B+树的根节点关键字取值 1 到 m；B 树非根节点关键字取值 ceil(m/2)-1 到 m-1，B+树非根节点关键字取值 ceil(m/2)到 m。

关键字个数	B树	B+树
根节点	1到m-1	1到m
非根节点	ceil(m/2)-1到m-1	ceil(m/2)到m

2）B 树分叉个数等于关键字个数+1，B+树分叉个数等于关键字个数。

3）B 树的每个节点既有关键字，又有数据；B+树的数据只在叶子上，非叶子节点只有关键字。

4）B+树的叶子节点相互之间有一个链路，B 树没有。

5）当查找数据时，从根出发，B 树可能不需要查找到叶子节点就可以找到数据，而 B+

树要找到叶子节点才找到数据。

75. 什么是哈夫曼树？如何构造？哈夫曼树的应用★★★

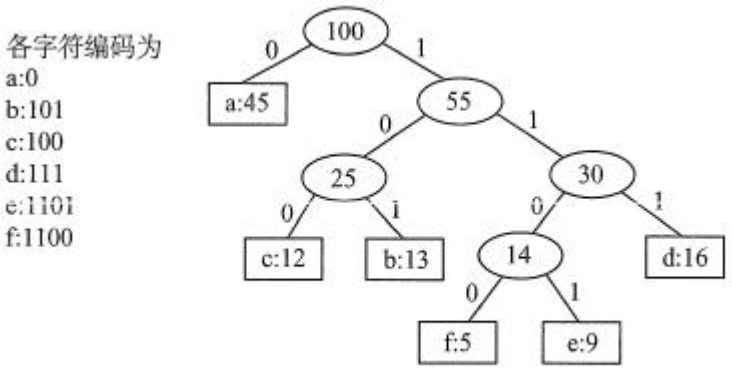
哈夫曼树又称为最优二叉树，其特点是，给定一组**带权的叶子结点**，**若构造所得到的二叉树拥有最小的带权路径长度 WPL**，则称该二叉树为一棵哈夫曼树。

构造：

将带权叶子结点并入一个集合，首先在集合中挑选出两个权值最小的叶子结点进行合并得到新的结点加入集合，再将两个被选中的结点剔除出集合。在树的构造上，将这两个结点作为叶子结点衔接到合并而成的新结点上。重复以上过程直到集合中只有一个元素，哈夫曼树则完成构造。

应用：

哈夫曼树的应用是哈夫曼编码，其特点是消除了编码前缀相同的二义性（哈夫曼编码是一种前缀编码，前缀编码就是任何一个编码都不是另外一个编码的前缀）。在哈夫曼编码中，只有哈夫曼树的叶子结点可以进行编码。



76. 简单介绍一下并查集，说说如何改进？★★★

并查集是一种用于解决集合**合并和查询**问题的数据结构。它能够高效地进行**集合的合并和判断两个元素是否属于同一集合**，并在实际应用中常用于解决图的连通性、最小生成树等问题。

在并查集中，每个元素被看作是一个节点，并以树的形式组织。每个树的根节点代表一个集合，而每个节点则指向其父节点，形成一棵树。

并查集的基本操作：

Union：用于将两个集合合并为一个集合。通过找到两个元素所属集合的根节点，将其中一个根节点的父节点指向另一个根节点，实现合并操作。

Find：查找根节点，用于确定元素所属的集合。通过迭代地或递归向上查找父节点，直到找到根节点，即可确定所属的集合。

改进措施：

路径压缩：是指在进行 Find 操作时，将节点指向根节点的路径上的所有节点直接连接到根节点，可以减小树的高度，加快查找。可以使用递归或迭代的方式进行路径压缩。

按秩合并：是指在进行 Union 操作时，将高度较低的树合并到高度较高的树上，从而保持树的平衡性。可以通过记录每个集合的秩（即树的高度或节点数量）来实现按秩合并。

77. 简单介绍一下 dfs 和 bfs，并说说它们的区别？★★★★

dfs：

使用**栈**数据结构来辅助实现深度优先搜索。

dfs 是按照一个路径一直访问到底，当前节点没有未访问的邻居节点时，然后回溯到上一个节点，不断的尝试，直到访问到目标节点或所有节点都已访问。

dfs 不保证找到最短路径，因为它一直往深处搜索。

bfs：

使用**队列**数据结构来辅助实现广度优先搜索。

bfs 是按层次访问的，先访问源点，再访问它的所有相邻节点，并且标记结点已访问，根据每个邻居结点的访问顺序，依次访问它们的邻居结点，并且标记节点已访问，重复这个过程，一直访问到目标节点或无未访问的节点为止。

bfs 能够保证找到的路径是最短路径，因为它按照距离起始节点的层级进行搜索。

使用场景：

dfs 适用于**找到一个可行解**，不需要找最短路径的情况。它的空间复杂度较低，适合在深度方向上搜索，例如拓扑排序、连通性判断、回溯等问题。

bfs 适用于**找到最短路径**。它的时间复杂度较低，适合在广度方向上搜索，例如寻找最短路径、连通性判断、社交网络中查找关系等问题。

78. 简单介绍一下最短路径算法？★★★★

最短路径算法通常有 Bfs 算法、Dijkstra 算法和 Floyd 算法。Bfs 只能处理无权图，Dijkstra 算法可以进一步解决带权图问题，Floyd 可以进一步解决带负权边图问题。

Bfs：

通过队列来实现，首先将单原点加入队列 。每次循环将队列中队头元素弹出，并且将与该元素所代表的结点相邻的结点加入队列，直到队列为空。每次循环代表距离加一，Bfs 可以找出单源点到其他结点的路径长度，取最小即为最短路。

Dijkstra：

应用了**贪心**的思想，通常解决**单源点**问题，时间复杂度为 $O(n^2)$，堆优化后是 O(nlogn)。

声明：vis[]表示这个节点是否访问，vis[u]=1 表示出圈，dis[]表示该点到原点的最短路径，u 表示圈外距离圈内最小的点，s 为源点。

①初始时，将所有点都在圈内，所有节点 vis[]=0，dis[]=inf，dis[s]=0。

②其次从圈内选择距离最小的点 u，打标记出圈 vis[u]=1。

③对 u 的所有出边进行松弛操作，v 是 u 所指向的节点，w 是 u->v 的权值，若 dis[u]+w<dis[v]，那么 dis[v]=dis[u]+w。

④重复②③直到所有节点更新完成。

```cpp
void dijkstra(int s) {
    memset( Dst: dis,  Val: inf,  Size: sizeof(dis));
    dis[s] = 0;
    for (int i = 1; i <= n; ++i) {
        int u = -1, mn = inf;
        for (int j = 1; j <= n; ++j) {
            if (!vis[j] && dis[j] < mn) {
                u = j;
                mn = dis[j];
            }
        }
        if (u == -1) break;//没有找到
        vis[u] = true;
        //遍历u的所有邻接点
        for (int e = head[u]; e; e = edges[e].next) {
            int v = edges[e].to, w = edges[e].w;
            if (!vis[v] && dis[v] > dis[u] + w) {
                dis[v] = dis[u] + w;
            }
        }
    }
}
```

Floyd：

应用了**动态规划**的思想，通常解决**多源点**问题，时间复杂度为 $O(n^3)$。

首先初始化，声明 dp 数组，dp[i][j]表示从 i 到 j 的权值，一开始数组中值均为 inf，而后更新 dp[i][i]所有点到自己的权值是 0。

输入每条边的结点和权值 u,v,w，更新 dp[u][v]=w，表示从 u 到 v 的权值是 w。

利用三层循环，然后逐步试探当前加入的点。

dp[i][j]=min(dp[i][j],dp[i][k]+dp[k][j]);如果 i 到 k 的权值+k 到 i 的权值比 i 到 j 的权值小，那么进行松弛，否则就继承原来的 dp[i][j]。

直到循环结束。

```cpp
memset( Dst: G,  Val: inf,  Size: sizeof (G));
for(int i=1;i<=n;i++){
    G[i][i]=0;
    //自己到自己距离为0
}
while (m--){
    int u,v,w;
    cin>>u>>v>>w;
    G[u][v]=min(w,G[u][v]);
    //读入每条边和权值
}
floyd();
```

```
void floyd(){
    for(int k=1;k<=n;k++){//逐个试探当前加入的点
        for(int i=1;i<=n;i++){
            for(int j=1;j<=n;j++){
                G[i][j]=min(G[i][j],G[i][k]+G[k][j]);
                //G[i][j]=G[i][j]|| (G[i][k] && G[k][j]);测试两个点是否连通，连通G[i][j]=1否则=0
            }
        }
    }

}
```

其他最短路径算法补充：

Bellman-ford 算法，单源最短路径，能处理负权边，时间复杂度为 O(VE)

SPFA 算法，求单源最短路径，能处理负权边，是 **Bellman-ford 的优化**，平均时间复杂度低于 **Bellman-ford**，最坏情况仍为 O(VE)。

A*算法，适用于在有向图中找到从起始节点到目标节点的最短路径，通过启发式函数来估计每个节点的代价。该算法综合考虑当前节点的路径长度和启发式评估值，通过优先级队列来选择下一个扩展的节点。

蚁群算法，是一种启发式的元启发算法，可以用于解决最短路径问题。蚂蚁根据信息素浓度和路径长度等因素做出选择，通过释放和更新信息素引导搜索。

79. 简单介绍一下求最小生成树的算法？★★★★

Prim：

Prim 算法是一种**贪心**算法，从一个起始节点开始逐步扩展最小生成树的边。

首先选择一个起始节点，然后将该节点标记为已访问。

通过不断选择与已访问节点相连且权重最小的边，并将相连节点标记为已访问，将这条边加入到最小生成树中。

重复上述步骤，直到所有节点都被访问过，即构建出最小生成树。

Kruskal：

Kruskal 算法也是一种**贪心**算法，通过按照边的权重从小到大的顺序逐步构建最小生成树。

将图中的所有边按照权重从小到大进行排序。

依次遍历排序后的边，如果当前边的两个端点不在同一棵树中（即不会形成环，常常借助并查集实现），则将该边加入最小生成树中，并将两个端点所在的树合并为一棵树。

重复上述步骤，直到最小生成树包含图中的所有节点。

应用场合：

Prim 算法适合在**稠密图**中进行求解，时间复杂度为 $O(v^2)$，其中 v 为节点数。

Kruskal 算法适合在边**稀疏图**中进行求解，时间复杂度为 O(eloge)，其中 e 为边数。

80. 介绍一下哈希表，如何构造哈希函数，如何解决哈希冲突？★★★★

哈希表又称为散列表，是根据**关键字**的值直接进行访问的数据结构，即它通过把关键码的值映射到表中的一个位置以加快查找速度，其中映射函数叫做哈希函数，存放记录的数组叫做哈希表。

哈希函数的构造方法：

（1）直接定址法：

取关键字的某个线性函数值作为散列地址，H(key)=a*key+b。

（2）除留余数法：

取关键字对 p 取余的值作为散列地址，即 H(key)=key%p，p 尽量选择质数，为了使散列分布更均匀。

哈希冲突的解决方法：

（1）开放地址法：

当发生冲突时，使用一定的探测序列方法，在哈希表中寻找下一个可用的空槽位，将冲突的元素插入到这个空槽位中。常见的探测序列方法有线性探测、二次探测等。

（2）链地址法

将哈希表的每个索引位置看作一个链表的头节点，当发生冲突时，将冲突的元素插入到链表中即可。这样，每个索引位置都可以存储多个元素。

（3）再哈希法

当发生冲突时，使用另一个哈希函数重新计算新的索引位置，直到找到一个空槽位来解决冲突。

81. 介绍一下各种排序算法的性能？★★★★★★

排序方法	时间复杂度（平均）	时间复杂度（最坏）	时间复杂度（最好）	空间复杂度	稳定性
插入排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
希尔排序	$O(n^{1.3})$	$O(n^2)$	$O(n)$	$O(1)$	不稳定
选择排序	$O(n^2)$	$O(n^2)$	$O(n^2)$	$O(1)$	不稳定
堆排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(1)$	不稳定
冒泡排序	$O(n^2)$	$O(n^2)$	$O(n)$	$O(1)$	稳定
快速排序	$O(nlog_2n)$	$O(n^2)$	$O(nlog_2n)$	$O(nlog_2n)$	不稳定
归并排序	$O(nlog_2n)$	$O(nlog_2n)$	$O(nlog_2n)$	$O(n)$	稳定

不稳定的排序算法有：希尔排序、选择排序、堆排序、快速排序。

82. 介绍一下插入排序？★★★★

插入排序的工作原理类似于整理扑克牌。

该算法将待排序的元素分为已排序区和未排序区，每次从未排序区中取出一个元素，插入到已排序区的适当位置，直到所有元素都被插入完毕。

插入排序的平均时间复杂度是 $O(n^2)$，最坏时间复杂度是 $O(n^2)$，空间复杂度是 $O(1)$，是一种稳定排序。

83. 介绍一下选择排序？★★★

将数组分为已排序区和未排序区。初始时，已排序区为空，而未排序区包含所有元素。

从未排序区中找到最小的元素，并记录其索引。

将最小元素与未排序区的第一个元素交换位置，将其放入已排序区的末尾。

重复步骤 2 和步骤 3，直到未排序区的元素全部交换完毕，得到最终的有序数组。

选择排序的平均时间复杂度是 $O(n^2)$，最坏时间复杂度是 $O(n^2)$，空间复杂度是 $O(1)$，是一种不稳定排序。

84. 介绍一下冒泡排序？★★★

它重复地遍历待排序数组，依次比较相邻的元素，并将较大的元素交换到右侧，从而逐步将最大的元素沉到数组的末尾。

相邻元素比较，如果前面元素比后面更大，则交换位置。第一轮把最大的元素放到末尾，第二轮把第二大的元素放到倒数第 2 个的位置，直到所有都排好序。

冒泡排序的平均时间复杂度是 $O(n^2)$，最坏时间复杂度是 $O(n^2)$，空间复杂度是 O(1)，是一种稳定排序。

85. 介绍一下快速排序？★★★★★★

快速排序采用了分治的思想。快速排序的核心思想是选择一个基准元素，通过将数组中的元素按照基准元素进行划分，使得左侧的元素都小于基准元素，右侧的元素都大于基准元素。然后对左右两个子数组分别进行递归排序，直到整个数组有序。

具体来说，选一个 pivot。例如选取最左边的元素记作 pivot。定义 i 和 j 两个指针，一开始分别指向 l 和 r，j 用来寻找比 pivot 小的元素，i 用来寻找比 pivot 大的元素，若 i 和 j 都找到而且 i<j 那么 a[i]和 a[j]交换，从而保证了左边的小于 pivot，右边的大于 pivot。若最后 i==j，那么将 pivot 移动到该位置。

快速排序的平均时间复杂度是 O(nlogn)，最坏时间复杂度是 $O(n^2)$，空间复杂度是 O(1)，是一种不稳定排序。

86. 为什么快速排序最坏情况会退化成 $O(n^2)$？★★★★★★

最坏情况发生在待排序的序列已经有序或近乎有序的情况下。在这种情况下，如果每次选择的基准元素都是当前子数组的最大或最小值，那么快速排序的分割过程将会非常不平衡，导致递归树的高度接近于 n。

在这种情况下，每次划分只能将序列分成一个空的子数组和一个包含 n-1 个元素的子数组，而不是将序列均匀地分成两个大小相等的子数组。

87. 介绍一下归并排序？★★★★★★

归并排序采用了分治的思想。归并排序的核心思想是将待排序数组逐步分割成单个元素，然后将这些单个元素合并成有序的数组。它通过不断地将两个有序的子数组合并成一个更大的有序数组，最终得到整个数组有序。

归并排序的平均时间复杂度是 O(nlogn)，最坏时间复杂度是 O(nlogn)，空间复杂度是 O(n)，是一种稳定排序。

88. 简述一下快速排序和归并排序的优缺点（从平均最坏时间复杂度、空间复杂度、稳定性的角度）。★★★★★★

（1）快速排序

优点：

①平均时间复杂度较低：快速排序的平均时间复杂度为 O(nlogn)，在大多数情况下都能够达到较好的排序效果。

②空间复杂度较低：快速排序通常只需要使用很少的额外空间，只需对原数组进行原地操作。

缺点：

①最坏情况下的性能：在最坏情况下，即待排序序列已经有序或近乎有序时，快速排序的时间复杂度会退化到 $O(n^2)$，导致性能下降。

②不稳定性：快速排序是一种不稳定的排序算法，在交换元素的过程中可能改变相同关键字元素的相对顺序。

（2）归并排序

优点：

①稳定性：归并排序是一种稳定的排序算法，它能够保持相同关键字元素的相对顺序不变。

②适用于外部排序：归并排序的特点使其非常适用于外部排序，即当排序的数据量太大无法完全加载到内存时，可以通过分阶段地读取和写入数据进行排序。

③性能稳定：归并排序的时间复杂度始终保持在 O(nlogn)，无论是最佳、最坏还是平均情况下。

缺点：

需要额外的空间：归并排序需要额外的空间来存储临时数组，因此它的空间复杂度相对较高。

89. 为什么排序需要稳定？★★★★

排序算法的稳定性意味着对于具有相同关键字的元素，排序后它们的相对顺序保持不变。在很多实际应用中，我们需要保持数据中相等元素的顺序关系。例如，在排序员工工资的数据时，如果有多名员工拥有相同的工资水平，我们可能希望按照他们的入职时间来排序，以维持他们在公司内部的先后顺序。如果使用不稳定排序，就可能打乱他们的相对顺序。

90. 归并排序的最坏时间复杂度优于快排，为什么我们还是选择快排？★★★★★★

（1）快速排序通常比归并排序更快。尽管快速排序在最坏情况下的性能可能较差，但在大多数情况下，它的平均时间复杂度要比归并排序低。

（2）快速排序是原地排序算法。原地排序算法是指排序过程中不需要额外的存储空间，只利用原始输入数组进行排序。

（3）快速排序的实现相对简单。相比于归并排序，快速排序的实现更为简洁，代码量更少。

总结：由于快速排序在平均情况下表现更好、占用更少的空间并且更易于实现。

91. 介绍一下堆排序？★★★★★★

堆排序可以分为两个主要步骤：建堆和排序。

建堆的步骤如下：

若数组有 n 个元素，从第 n/2 个元素（非叶节点）开始一直到第 1 个元素，进行堆的调整。

对于每个非叶子节点，进行一次下沉操作，将当前节点与其子节点进行比较，如果不满足堆的性质，则交换位置。

重复步骤 2，直到整个数组被构建成一个堆，即满足父节点大于等于子节点（大顶堆）或父节点小于等于子节点（小顶堆）的性质。

排序的步骤如下：

首先，将建好的堆中的根节点与最后一个元素交换位置。

然后，将堆的大小减 1，并对新的根节点进行一次下沉操作，以找到新的最大值。

重复步骤 1 和步骤 2，直到堆的大小为 1，即所有元素都排好序。

补充：

堆的插入的步骤如下：

把插入的元素放在数组的末尾，数组的长度+1。

首先，该节点将其与其父节点进行比较，如果该节点的值大于父节点的值，则交换位置。

继续将该节点与其新的父节点进行比较，重复上述步骤，直到节点上浮到正确的位置或者达到根节点。（实际上是堆的上浮）

堆的下沉（堆的调整）：

用于将一个节点下沉到合适的位置以满足堆的性质。

从待调整节点开始，将其与其左右子节点中较大的节点进行比较，如果该节点的值小于某个子节点的值，则交换位置。

继续将该节点与其新的子节点进行比较，重复上述步骤，直到节点下沉到正确的位置或者达到叶子节点。

92. 请你写出以上排序的代码。★★★★★★

冒泡排序：

```
void Bubblesort() {
    for (int i = 1; i <= n - 1; ++i) {
        for (int j = 1; j <= n - i; j++) {
            if (a[j] > a[j + 1])
                swap( &a[j], &a[j + 1]);
        }
    }
}
```

选择排序：

```
void Selectsort() {
    //选择当前序列[i+1...n]中最小的，加到有序序列[1...i]的最右边
    for (int i = 1; i <= n - 1; i++) {
        int min = i;
        for (int j = i + 1; j <= n; j++) {
            if (a[min] > a[j]) {
                min = j;
            }
        }
        if (min != i) {
            swap( &a[min], &a[i]);
        }
    }
}
```

插入排序：

```
void Insertsort() {
    for (int i = 2; i <= n; i++) {
        if (a[i] < a[i - 1]) {
            //比最大的还要小
            int temp = a[i], j;//temp用来记录当前的值
            for (j = i - 1; j > 0 && a[j] > temp; j--)
                a[j + 1] = a[j];
            a[j + 1] = temp;//a[j]是第一个小于等于temp的，所以应该把temp插到a[j+1];
        }
    }
}
```

快速排序：

```
void Quicksort(int l,int r) {
    if (l >= r) return;
    int pivot = a[l], i = l, j = r;
    while (i < j) {

        //要保证右边的比枢轴大
        while (i < j && a[j] >= pivot)
            j--;

        //要保证左边的比枢轴小
        while (i < j && a[i] <= pivot)
            i++;

        if (i < j)
            swap( &: a[i],  &: a[j]);
    }
    //此时i==j,a[l]还是pivot
    swap( &: a[l],  &: a[i]);
    Quicksort(l,  r: i - 1);
    Quicksort( l: j + 1, r);
}
```

归并排序：

```
void Mergesort(int l,int r) {
    if (l >= r) return;
    int mid = l + r >> 1;
    Mergesort(l,  r mid);
    Mergesort( l mid + 1, r);
    int i = l, j = mid + 1, k = l;
    while (i <= mid && j <= r) {
        if (a[i] < a[j]) {
            temp[k++] = a[i++];
        } else {
            temp[k++] = a[j++];
        }
    }
    while (i <= mid) {
        temp[k++] = a[i++];
    }
    while (j <= r) {
        temp[k++] = a[j++];
    }
    for (int i = l; i <= r; i++) {
        a[i] = temp[i];
    }
}
```

堆排序：

```
void HeadAdjust(int k,int len) {//下坠调整函数
    //从k开始下滤
    int temp = a[k];
    //k是当前元素，i是更大的孩子
    for (int i = k << 1; i <= len; i <<= 1) {
        if (i + 1 <= len && a[i + 1] > a[i]) {
            i++;
        }
        if (temp >= a[i]) break;//已经是大根堆了，直接退出
        else {
            a[k] = a[i];
            k = i;
        }
    }
    a[k] = temp;
}
void BuildMaxHeap(){
    for(int i=n/2;i;i--){
        HeadAdjust( k: i, len: n);
    }
}
void HeapSort() {
    BuildMaxHeap();
    for (int i = n; i >= 2; i--) {
        swap( &: a[1], &: a[i]);
        HeadAdjust( k: 1, len: i - 1);
    }
}
```

93. 如何解决 TopK 问题？★★★★

TopK 问题是指在一组元素中，找出其中最大（或最小）的 K 个元素。

（1）排序法

将所有元素进行排序，然后取出最大（或最小）的 K 个元素即可。时间复杂度为 O(nlogn)，其中 n 为元素的个数。这种方法简单直观，但对于大规模数据来说效率较低。

（2）堆

使用最大堆或最小堆来解决 TopK 问题。首先构建一个大小为 K 的堆，将前 K 个元素插入堆中，然后将剩余的元素与堆顶元素进行比较，如果大于（或小于）堆顶元素，则将其替换并进行堆调整操作。时间复杂度为 O(nlogK)，空间复杂度为 O(K)。

（3）快速选择算法

基于快速排序的思想，通过每次划分数组找到第 K 大（或第 K 小）的元素。将数组划分为两部分，左边的元素都大于（或小于）划分点，右边的元素都小于（或大于）划分点。如果划分点的下标为 K-1，则找到了第 K 大（或第 K 小）的元素。时间复杂度为 O(n)，空

间复杂度为 O(1)。

七、算法分析与设计★

94. 贪心能不能解决 0-1 背包问题？★

对于一般的 0-1 背包问题，贪心算法并不能保证得到最优解。

0-1 背包问题是一个经典的组合优化问题，其中每个物品有一个重量和一个价值，背包有一个容量限制。目标是选择一些物品放入背包中，使得它们的总重量不超过背包容量，同时总价值最大化。

贪心算法的基本思想是每次选择当前最优的物品放入背包，直到无法再添加新的物品为止。然而，对于 0-1 背包问题，贪心策略并不一定能够得到最优解。原因在于贪心算法只考虑了当前的最优选择，而没有全局地考虑所有可能的方案。

95. 简述一下 P 问题、NP 问题和 NP 完全问题。★★

（1）P 问题（通常认为比较容易解决的问题）

P 问题是指可以在多项式时间内解决的问题。也就是说，存在一个算法，对于规模为 n 的输入，能够在 O(n^k)的时间内运行完毕，其中 k 是常数。

P 问题例子：线性规划、最短路径问题、最小生成树问题等。

（2）NP 问题

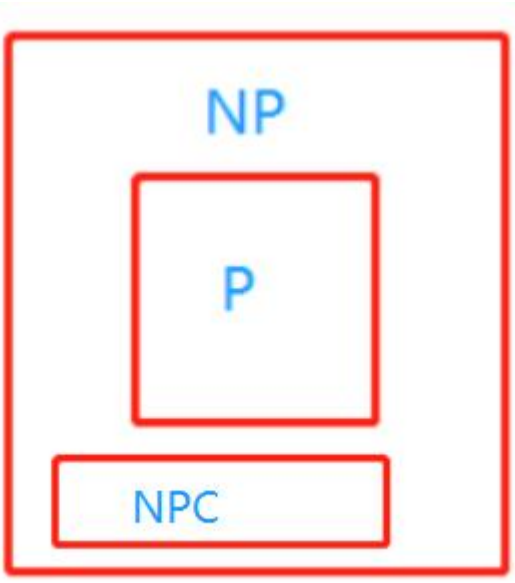
NP 问题是指可以在多项式时间内验证其解的问题。也就是说，如果有一个候选解，可以在多项式时间内验证该解的正确性。

NP 问题例子：旅行商问题（TSP）、背包问题、图的着色问题。

（3）NPC 问题

NPC 指的是 NP 完全问题。是指既属于 NP 问题，同时也是最困难的一类 NP 问题。如果我们能够在多项式时间内解决一个 NP 完全问题，那么可以在多项式时间内解决所有的 NP 问题；同样，如果一个问题是 NP 完全问题，但我们无法在多项式时间内解决它，那么可以认为所有的 NP 问题都无法在多项式时间内解决。

NP 完全问题的一个重要特征是它们之间存在着多项式时间规约关系。也就是说，如果一个问题 A 可以在多项式时间内规约到另一个问题 B，那么我们可以利用问题 B 的解法来解决问题 A。



所有 P 问题都是 NP 问题。NPC 问题一定是 NP 问题。但是 P 问题和 NPC 问题没有交集

96. 简单说说贪心、动态规划和分治的区别。★★★

贪心

就是做出在当前看来是最好的结果，它不从整体上加以考虑，也就是**局部最优解**。

贪心算法**从上往下**，从顶部一步一步最优，得到最后的结果，它**不能保证全局最优解**，与贪心策略的选择有关。

动态规划

是把问题分解成子问题，这些子问题可能有重复，**可以记录下前面子问题的结果防止重复计算**。动态规划解决子问题，前一个子问题的解对后一个子问题产生一定的影响。在求解子问题的过程中保留哪些有可能得到最优的局部解，丢弃其他局部解，直到解决最后一个问题时也就是初始问题的解。动态规划是从下到上，一步一步找到全局最优解。**（各子问题重叠）**

分治

将原问题划分成 n 个规模较小而结构与原问题相似的子问题；递归地解决这些子问题，然后再合并其结果，就得到原问题的解。**（各子问题独立）**

分治模式在每一层递归上都有三个步骤：

①分解：将原问题分解成一系列子问题；

②解决：递归地解各个子问题。若子问题足够小，则直接求解；

③合并：将子问题的结果合并成原问题的解。

分治的实例：归并排序。

八、计算机编程语言★★★★★

97. 指针数组和数组指针的区别？★★

指针数组：指针数组是指一个数组，其中每个元素都是指针类型。这意味着数组存储的是指针的地址，而指针指向的是其他数据或对象。例如，int *ptrArray[5] 定义了一个包含 5 个整型指针的数组。

数组指针：数组指针是指一个指针，它指向一个数组的起始位置。数组指针本身是一个

指针变量，存储的是数组的首地址。例如，int (*ptr)[5] 定义了一个指向包含 5 个整型元素的数组的指针。

98. 结构体和共用体的区别？★★

结构体中的成员变量分别占用独立的内存空间，并且可以同时存储不同的值。

共用体中的成员变量共享同一块内存空间，修改其中一个成员变量的值会影响到其他成员变量。

99. 指针在底层是如何实现的？★★

在底层，指针是通过内存地址来实现的。每个变量在内存中都有一个唯一的地址，指针存储了这个地址，并可以通过该地址找到对应变量的值。

100. 指针和引用的区别是什么？★★★★

指针和引用都是用于处理变量的间接访问。

（1）语法

指针：指针是一个变量，存储了另一个变量的内存地址。使用星号 (*) 来声明指针类型。

引用：引用是目标变量的别名，它是对目标变量进行直接访问。在 C++中，通过使用引用符号（&）来定义引用变量。

（2）内存管理

指针：指针可以被重新分配给其他变量，也可以指向空地址（NULL），并且可以通过 new 和 delete 操作符来动态地分配和释放内存。

引用：引用一旦被初始化，就不能再改变它所引用的变量。引用本身并不占用额外的内存，它只是目标变量的别名。

（3）空值处理

指针：指针可以存储空地址，表示未指向任何有效的内存位置。

引用：引用必须在声明时进行初始化，并且不能引用空值（NULL），它必须引用一个有效的对象。

101. C 和 C++的 malloc 和 new 有什么区别？★★★★

C 和 C++中的 malloc 和 new 都用于动态分配内存，但有一些区别：

（1）malloc 是 C 库函数，使用时需要包含头文件 stdlib.h，而 new 是 C++关键字。

（2）malloc 只负责分配一块指定大小的内存空间，返回一个指向该空间的 void*类型指针。new 更高级一些，它会在分配内存的同时调用**构造函数**初始化对象，并返回一个指向该对象的指针。

（3）malloc 返回的是 void*指针，需要进行显式的类型转换，并且对于自定义类型的对象，需要手动调用构造函数初始化。而 new 根据对象类型自动确定所需的内存空间，并执行相应的构造函数初始化操作，无需手动转换类型。

（4）new 在遇到内存分配失败时会抛出 std::bad_alloc 异常，而 malloc 则返回 NULL 指针表示分配失败。

102. 介绍一下 C++的模板？并说说模板的好处。★★

C++模板是一种通用的编程工具，它允许在编写代码时使用参数化类型和参数化函数。

（1）C++模板分为两种类型：函数模板和类模板。

①函数模板： 函数模板是一种定义函数的方式，其中某些类型或值可以作为参数进行通用化。函数模板的语法如下：

```
template <typename T>
返回类型 函数名(参数列表)
{
```

```
    // 函数体
}
```

②类模板：类模板与函数模板类似，但是用于定义通用的类。类模板的语法如下：

```
template <typename T>
class 类名
{
    // 类成员和函数定义
};
```

（2）模板的好处

通用代码：模板允许编写通用的代码，可以在不同类型上执行相同的操作。通过将类型参数化，可以实现对各种数据类型的通用处理。

泛型编程：C++模板支持泛型编程，使得可以编写与特定类型无关的代码。这样可以提高代码的重用性和可扩展性，并减少冗余代码的编写。

容器类和算法：容器类和算法可以使用模板来实现，在不同类型的数据上进行通用操作。

函数重载：通过函数模板，可以为不同类型的参数提供相同逻辑的函数实现。这避免了为每个类型编写多个重载函数的重复工作。

103. 介绍 vector 内部的数据结构是如何实现的。★★

在 C++中，std::vector 类是由一个动态分配的连续内存块实现的。它可以根据需要自动调整其大小，并提供了一组方法来方便地操作数据。

具体实现上，std::vector 通常使用一个指针（或称为迭代器）来指向内部的连续内存块。该内存块存储了容器中的元素，并且在内存中是连续分布的，这使得随机访问以及对元素的插入和删除操作变得高效。

当元素数量超过内部内存块的容量时，std::vector 会分配一个更大的内存块，并将原有元素复制到新的内存块中。这样，std::vector 能够自动调整其大小，以适应不同容量需求。

104. 深拷贝和浅拷贝的区别是什么？★★★★

深拷贝和浅拷贝的区别主要在于对于包含指针成员变量的对象：

浅拷贝仅复制指针的地址，新旧对象之间共享同一块内存。当一个对象释放内存时，会影响到其他拷贝得到的对象，容易导致悬垂指针或内存泄漏。

深拷贝将指针所指向的内存复制到新对象中。每个对象拥有独立的内存空间，在销毁对象时不会影响其他对象。

105. 如何理解构造函数和析构函数？★★

构造函数：

构造函数在**创建对象时**被调用，用于初始化对象的成员变量和执行其他必要的操作。构造函数具有以下特点：

（1）与类同名：构造函数的名称与类名相同，没有返回类型（包括 void）。

（2）自动调用：在创建对象时，编译器会自动调用相应的构造函数。

（3）可以重载：一个类可以有多个构造函数，它们可以以不同的参数列表进行重载，用于满足不同的对象创建需求。

（4）初始化对象状态：构造函数负责初始化对象的成员变量，确保对象处于有效和一致的状态。

析构函数：

析构函数在**对象销毁时**被调用，用于清理对象分配的资源、执行善后操作。析构函数具有以下特点：

（1）与类同名，前面加上波浪号（~）：析构函数的名称与类名相同，但在前面加上波

浪号（~），没有返回类型（包括 void）。

（2）自动调用：在对象销毁时，编译器会自动调用相应的析构函数。

（3）无参数：析构函数不接受任何参数，因为它只负责当前对象的清理工作。

（4）不可重载

106. 什么是多态？多态有什么好处？C++有哪些多态？ ★★★★★

多态的定义：

多态指同样的消息被不同类型的对象接收时导致不同的行为。

多态的好处：

（1）代码复用

多态允许将**通用的操作和行为**定义在父类中，子类可以继承并重写这些方法。这样可以**避免重复编写相似的代码**，提高代码的复用性和效率。

（2）可扩展性

通过添加新的子类来扩展现有代码的功能，而无需修改现有代码。这样可以使代码具有更好的可维护性和可扩展性。

（3）可读性

多态能够使代码逻辑更加简洁明了，通过使用统一的接口和抽象的父类，可以减少代码的冗余性，使代码更易于理解和维护。

（4）灵活性

多态使得对象能够以多种形态存在和被使用。通过使用基类指针来操作不同的子类对象，可以在运行时动态地选择不同的实现，从而实现代码的灵活性和动态性。

C++中的多态：

①重载多态（函数重载和运算符重载）

②强制多态（类型强制转换）

③类型参数化多态（函数模板、类模板）

④包含多态（继承及虚函数）

107. 什么是虚函数？虚函数有什么好处？ ★★★★

虚函数（Virtual Function）是在**父类中**声明的一个函数，通过在函数声明前加上关键字"**virtual**"来定义。虚函数可以在**派生类中被重写**，并且通过**基类指针或引用**调用时，根据指针或引用所指向的对象类型来动态地选择调用哪个函数的实现。

虚函数的好处：

（1）实现多态

通过基类指针或引用调用虚函数时，可以根据指向的具体对象类型来动态地决定调用哪个类的函数实现，实现不同对象的不同行为。

（2）动态绑定

通过基类指针或引用调用虚函数时，实际调用的函数实现会在运行时决定，而不是在编译时确定。这样可以根据对象的实际类型来动态地决定调用哪个函数实现，提供更大的灵活性和可扩展性。

（3）可以被覆盖

在派生类中，可以通过重写（覆盖）基类中的虚函数来改变其原有的行为，不影响其他派生类对象对该函数的调用。

108. 什么是纯虚函数？ ★★★

纯虚函数是在父类中声明的一个没有实现的虚函数，通过在函数声明后加上关键字 "= 0" 来定义。纯虚函数本身没有函数体，只用于接口定义和规范派生类的行为。

纯虚函数的特点如下：

（1）没有函数体：纯虚函数没有具体的实现代码，只有函数声明。它相当于一个接口，要求派生类必须实现该函数。

（2）强制派生类实现：这样可以确保派生类都具备该函数的功能，同时也避免了在基类中给出默认实现可能引发的歧义性。

（3）抽象类：含有纯虚函数的类被称为抽象类。抽象类不能被实例化。

使用纯虚函数的好处包括：

（1）接口规范：纯虚函数定义了一个接口，明确了基类和派生类之间的合约关系。通过纯虚函数，可以规范派生类需要实现的功能和行为。

（2）支持多态：纯虚函数在基类中作为一个可重写（覆盖）的函数存在，可以通过基类指针或引用来调用派生类的具体实现，实现多态性。

109. C/C++函数中如果 return 想返回多个值，有什么解决方案？★★★

（1）使用全局变量，从定义变量到程序结束；

（2）使用数组、结构体或类：

定义一个结构体或类，将需要返回的多个值封装在结构体或类的成员变量中，然后将该结构体或类作为函数的返回值。这样就可以一次性返回多个值。

（3）通过引用参数传递：

定义函数参数为引用类型，函数内部修改参数的值，然后在调用函数时传入外部变量作为引用参数。

（4）通过指针参数传递：

使用指针参数传递：类似于引用参数，将函数的参数定义为指向相应类型的指针，并在函数内部修改指针指向的变量的值。通过传递变量的地址作为指针参数，函数可以将多个值返回给调用者。

```
void myFunction(int* value1, float* value2, char* value3) {
    *value1 = 100;
    *value2 = 3.14f;
    *value3 = 'a';
}
int main() {
    int a;
    float b;
    char c;
    myFunction(&a, &b, &c);//返回后的 a,b,c 都会改变
    return 0;
}
```

110. 栈溢出的可能原因？★★★

局部变量过多或过大：

当函数中声明的局部变量数量过多、占用内存过大时，会占用大量的栈空间。如果栈空间不足以容纳这些局部变量，就会发生栈溢出。

动态内存分配错误：

使用动态内存分配函数（如 malloc、new）申请了大量内存但没有正确释放，导致堆积过多的内存，进而影响栈的使用。

递归调用错误：

如果在递归函数中没有正确地定义终止条件，递归可能会无限地循环调用自身，导致栈的使用超出限制。每次递归调用都会在栈上创建一个新的函数调用帧，当递归层数过多时，

栈空间被耗尽。

函数调用层级过深：

在函数内部频繁地调用其他函数，形成很深的调用层级，会占用大量的栈空间。如果递归层级过多或者函数调用层级过深，可能导致栈溢出。

111. 简述 C/C++程序编译时的内存分配情况。★★★★★

（1）栈区（编译器自动分配释放、内存分配连续、存放函数参数和局部变量）

在函数的执行过程中，每次函数调用时会在栈上创建一个新的帧，用于存储函数的局部变量和返回地址。当函数执行完毕时，这个帧会被销毁。栈的内存分配和释放是自动进行的，由编译器负责管理。

（2）堆区（程序员分配释放、内存分配不连续、要通过指针链接）

堆的内存分配和释放需要显式地进行管理。在 C 中，使用 malloc 和 free 函数进行内存的分配和释放；而在 C++中，可以使用 new 和 delete 运算符进行内存的分配和释放。

一般由程序员分配释放， 若程序员不释放，程序结束时可能由操作系统回收。类似于链表，在内存中的分布不是连续的，它们是不同区域的内存块通过指针链接起来的。

（3）静态存储区（全局内存区）

全局内存区是用于存储全局变量、静态变量和静态常量的一块内存区域。这块内存在程序启动时被分配，并在整个程序的生命周期中一直存在，直到程序结束才被释放。

（4）常量区

常量区用于存储常量字符串和其他常量数据。这些数据在程序运行期间是只读的。

（5）程序代码区

代码区存储程序的指令，也就是可执行代码（编译后的机器代码）。

112. C 和 C++的区别是什么？★

C++是 C 的超集，也就是说，C++包括了 C 的所有基础特性，并且还增加了一些新的特性。下面列举一些 C 和 C++之间的主要区别：

面向对象编程

C++ 是一种面向对象的编程语言，而 C 是面向过程语言。因此，C++ 支持类、继承、封装、多态等一系列面向对象的概念和特性。

标准库

C++ 标准库比 C 标准库更加完善和强大。C++ 标准库包括了很多容器类，如 vector、map、set 等，以及输入输出流、字符串处理等常用功能。这些库函数可以在许多情况下提高开发效率。

命名空间

C++ 引入了命名空间的概念，可以避免函数命名相同的冲突。使用命名空间可以将代码按照逻辑分组，并更好地组织代码。

异常处理

C++ 支持异常处理机制，这个机制可以增强程序的容错性和可靠性。当程序发生异常时，可以抛出异常并在可控范围内进行处理，避免程序崩溃。而 C 不支持异常处理机制。

运算符重载

C++ 允许对运算符进行重载，可以使得运算符在处理特定类型的数据时更具有描述性。而 C 不支持运算符重载。

113. C++和 Python 在底层上的区别？★★

（1）编译与解释

C++是一种编译型语言，源代码通过编译器转换为可执行的机器码才能运行。

Python 是一种解释型语言，代码逐行解释执行，无需编译，相对而言速度较慢。然而，Python 具有丰富的库和模块，可以通过集成高性能库（如 NumPy 和 Pandas）来提高执行效率。

（2）静态类型和动态类型

C++是一种静态类型语言，变量的类型需要在编译时确定，并且不能随意更改。

Python 是一种动态类型语言，变量的类型在运行时确定，并且可以随意更改。

（3）内存管理

C++中开发人员需要**手动管理内存**，包括分配和释放内存。这可能导致内存泄漏和野指针等问题。

Python 使用**垃圾回收机制，自动处理内存管理**，开发人员无需手动分配和释放内存。

114. C++和 Java 的区别？★★

（1）**语法风格**：C++是一种面向过程和面向对象的语言，语法相对较为灵活，可以直接操作内存和指针。而 Java 是一种纯面向对象的语言，语法较为严谨，所有的代码都必须定义在类中。

（2）**平台依赖性**：C++编译生成的程序可以在不同平台上运行，包括 Windows、Linux 和 macOS 等。而 Java 程序需要在 Java 虚拟机（JVM）上运行，因此具有更高的平台依赖性。

（3）**内存管理**：C++允许手动管理内存，开发人员需要自行分配和释放内存。而 Java 使用自动内存管理机制（垃圾回收），开发人员无需显式地处理内存分配和释放。

（4）**异常处理**：C++使用 try-catch 语句来捕获和处理异常，开发人员可以自定义异常类型。而 Java 使用 try-catch-finally 块来处理异常，并且提供了统一的异常类层次结构。

（5）**执行速度**：由于 C++更接近底层语言，其执行速度通常比 Java 更快。而 Java 通过 JVM 的即时编译器进行优化，因此在某些情况下可能具有更好的性能。

（6）**应用领域**：C++主要用于系统级开发、游戏开发和性能敏感的应用程序。Java 主要用于企业级应用程序、移动应用开发和大型跨平台项目。

115. C++和 Java 的面向对象多态性的区别？★★

在 C++中，多态性是通过虚函数和继承实现的，Java 中的多态性是通过继承和方法重写实现的。

总结一下 C++和 Java 的面向对象多态性区别：

（1）C++中需要使用关键字 virtual 显式声明虚函数，而 Java 中所有非静态方法默认都是虚函数，不需特别声明。

（2）C++中使用指针或引用调用虚函数时，需要通过基类的指针或引用来实现多态性；而 Java 中任何对象引用都可以实现多态性。

（3）C++中通过纯虚函数和抽象类来定义接口，派生类必须实现这些纯虚函数；而 Java 中没有纯虚函数的概念，可以通过接口（interface）来定义规范。接口中的方法必须由实现类提供具体实现。

116. Python 和 Java 的区别？

（1）语法风格：Python 具有简洁、优雅的语法风格，注重可读性，使用缩进来表示代码块。而 Java 语法较为严谨，使用大括号来表示代码块。

（2）应用领域：Python 适用于广泛的领域，如数据分析、机器学习、科学计算、Web 开发等。Java 主要用于企业级应用程序开发、Android 应用开发和大型系统开发等。

（3）执行方式：Python 是解释型语言，在运行时由解释器逐行解释执行。而 Java 是编译型语言，源代码首先被编译成字节码，然后在 Java 虚拟机（JVM）上执行。

（4）平台依赖性：Python 程序可以在多个平台上运行，包括 Windows、Linux 和 macOS 等。而 Java 程序需要依赖 JVM 来实现跨平台性。

（5）内存管理：Python 使用自动内存管理机制（垃圾回收），开发人员无需手动管理内存。而 Java 也提供了垃圾回收机制，但开发人员可以通过手动操作对象的引用来进行内存管理。

（6）编程风格：Python 强调代码的简洁和可读性，鼓励使用简单明了的语句和表达式。而 Java 更注重面向对象的编程范式，需要明确定义类、方法和接口等。

（7）生态系统：Python 拥有强大的第三方库和工具支持，如 NumPy、Pandas、Django 等，可以快速开发各种应用。Java 也有丰富的库和框架，如 Spring、Hibernate 等，适用于大型企业级项目。

117. 面向对象编程和面向过程编程的区别，面向对象编程的性质是什么？

面向对象编程的特点使得代码更易于理解、扩展和维护，提高了代码的重用性和可靠性。它强调的是将问题中的实体和行为映射到代码中的对象和方法，更贴近实际问题的描述和思考方式。

而面向过程编程则关注解决问题的步骤和流程，更加直接和线性。

面向对象编程性质：

继承性：一个类可以继承另一个已存在的类的属性和方法，减少了代码重复，并支持代码的扩展和修改。

封装性：将相关的数据和方法封装在对象中，隐藏内部实现细节并提供公共接口。

多态性：相同的消息可以被不同的对象类型处理，提供了灵活性和可扩展性。

抽象性：通过接口和抽象类定义通用规范，屏蔽细节，便于代码设计和维护。

九、计算机网络★★★★

118. 介绍一下 OSI 七层模型各层之间的功能与对应的协议？★★★★★

OSI 七层模型自顶向下分别为：应用层、表示层、会话层、传输层、网络层、数据链路层、物理层。

（1）应用层：为应用程序提供服务。主要的协议有：FTP、SMTP、HTTP、DNS。

（2）表示层：表示层负责数据的格式转换、加密、解密和压缩等。常见协议有 JPEG、ASCII 等。

（3）会话层：建立或解除与其他节点的联系

（4）传输层：提供端到端的可靠数据传输服务，还提供可靠传输、差错检测、流量控制和拥塞控制等功能。主要的协议有：TCP、UDP 协议等。

（5）网络层：负责进行数据包的路由选择和分组转发，确保数据可以从源主机传输到目标主机。向上提供简单的、无连接的、尽最大努力的交付的服务。主要的协议有：IP、ICMP、IGMP、RIP、OSPF 协议等。

（6）数据链路层：提供点到点的可靠数据传输服务，将物理层提供的原始比特流封装

成帧。它还提供了错误检测和纠正机制。常见协议有以太网（Ethernet）、Wi-Fi（802.11）、PPP、CSMA/CD 协议等。

（7）物理层：屏蔽传输媒体以及通信手段的差异，只关注如何在物理介质上传输二进制数据，主要协议有 IEEE 802 等。

119. 计算机网络为什么要分层？分层的优点是什么？★★

由于计算机网络是一个复杂而且大的系统，分层主要是利用分而治之的思想，将大的问题分解为若干个更小的子问题，从而将复杂问题简单化。

分层的主要优点有：

（1）各层之间相互独立：每层只需要用合适的技术解决独立的问题，并不关心相邻层之间的实现，只是接受下层提供的服务，并为上层提供服务，降低问题复杂度；

（2）灵活性好：当任何一层发生变化时，只要保持各层接口不变，则对某一层的修改将不影响相邻层间的功能实现；

（3）结构上可分割：每层都可以采用最合适的技术来实现该层的功能；

（4）易于实现和维护；

（5）促进标准化工作：每层之间有明确的功能与服务。

120. 协议的三要素是什么？★★

（1）语法：规定传输数据的格式。

（2）语义：规定所要完成的功能。

（3）同步：规定各种操作的顺序。

121. 电路交换、报文交换、分组交换的技术特点。★★★

（1）电路交换：

定义：传输前，必须建立一条专用的物理通信路径，一直被独占，结束后才释放。

优点：通信时延小，实时性好；有序传输；没有冲突；控制简单；适用范围广。

缺点：建立连接时间长；线路使用效率低；灵活性差；控制简单。

（2）报文交换：

定义：数据交换的单位是报文，采用存储转发方式。

优点：无须建立连接；动态分配线路；提高线路可靠性；利用率高；提供多目标服务。

缺点：存储转发时延打；较大的缓存空间。

（3）分组交换：

定义：采用存储转发，限制每次传送数据块大小上限，加上一些控制信息，构成分组

优点：没有建立时延；线路利用率高；简化了存储管理；并行加速传输；减少出错几率和重发数据量。

缺点：存在传输时延；需要传输额外的信息量；采用数据报服务，出现失序、丢失、重复分组。

122. MAC 地址、IP 地址、域名分别在哪一层，有什么作用，用哪些协议建立这些地址的映射。★★★

MAC 地址在数据链路层，IP 地址在网络层，域名在应用层。

作用：

MAC 地址作用：区分网卡，这个地址不能改变。

IP 地址作用：区分用户，为每个用户对应一个 IP 地址。

域名：便于记住网站。

映射：

IP 转 MAC 协议：ARP

MAC 转 IP 协议：RARP

域名转换成 IP：DNS

123. 局域网中数据怎么传的？和 WiFi 无线网中传播的区别。★★

局域网中的数据采用 CSMA/CD 协议。

WIFI 无线网中采用 CSMA/CA 协议。

124. 简述一下 CSMA/CD 协议。★★★★★

CSMA/CD（载波监听多点接入/碰撞检测）协议的执行流程如下：

（1）在发送数据前需要监听信道。如果信道忙，则持续等待；如果信道空闲，则在等待一段细微时间后将开始传输数据；（载波监听，碰撞检测）

（2）在传输数据的过程中，一边传输一边监听信道是否发生碰撞。如果在争用期之内没有检测到碰撞，则说明已经独占信道，可以持续至完成发送；若在争用期内发生碰撞，则将停止数据发送，并发送一个拥塞信号；（边听边发，冲突停发）

（3）检测到发生碰撞后，适配器将执行二进制退避算法随机等待一段时间后，在进行数据发送。（截断二进制指数规避算法，随机重发）

125. 介绍一下 Http1.0~Http3.0★★★★★

HTTP1.0：（非持续性连接）

HTTP1.1：（持续性连接）（默认流水线）先来先服务（FCFS）

HTTP2：划分成帧，调度帧以减轻阻塞，根据优先级来传输数据而不是 FCFS

HTTP2 之前还是 TCP 服务，TCP 连接没有安全性；

HTTP3 是基于 QUIC 协议，出现了 UDP 服务，增加了安全性

126. 简述一下与自治系统 AS 相关的内部网关协议和外部网关协议？★★★★★

（1）内部网关协议 IGP

指自治系统内部所使用的路由选择协议，一般有路由信息协议 RIP 和开放最短路径优先协议 OSPF。

RIP：

RIP 协议是**基于距离向量**的路由选择协议。一条 RIP 路径最多包含 15 个路由器，当距离为 16 时表示不可达。该协议中路由器只和相邻路由器进行信息交换，交换的信息是路由表。它适用于小型网络，但在大型网络中存在收敛慢、更新频繁等缺点。

OSPF：

OSPF 是一种**基于链路状态**的路由选择协议。协议中路由器发送的信息是相邻路由器的链路状态，并且使用洪泛法进行发送。最终，每个路由器都将建立一个链路状态数据库，即整个网路的拓扑结构。

（2）外部网关协议 EGP

是指在不同自治系统之间交换信息的协议。例如 BGP 是一种 EGP。通常每个自治系统需要选择一个路由器作为 BGP 发言人，并且建立使用 TCP 连接，利用 BGP 会话交换路由信息。

127. 为什么无线局域网不能使用 CSMA/CD？★★★★

（1）无线信道特性：无线信道具有广播性质和随机性质，无法实现完全的碰撞检测。由于无线信道中存在传播延迟、多径效应、信号衰落等因素，无法准确检测到信号的冲突。与有线媒体不同，无线信道上的碰撞无法通过监听到碰撞信号来进行检测和处理。

（2）信号干扰：在无线局域网中，多个设备共享同一无线信道进行通信。由于无线信号的传播范围和干扰范围较大，相邻的无线设备发送信号时容易相互干扰，导致更多的碰撞发生。

（3）隐藏终端问题：无线局域网中存在隐藏终端问题，指的是即使在发送前进行了碰撞检测，但由于某些设备之间的距离较远或者受到障碍物的干扰，无法感知到对方的存在而

导致碰撞。

128. 简述流量控制、拥塞控制和可靠传输机制的原理。★★★★★

（1）流量控制

流量控制用于确保发送方与接收方之间的数据传输速率相匹配，以防止接收方无法及时处理大量的数据。

流量控制通过使用**滑动窗口协议**来实现。

发送方会根据接收方的接收能力和缓冲区的剩余空间来动态调整发送数据的速率。当接收方的缓冲区快满时，它会发送一个流量控制信号给发送方，告知发送方降低发送速率，以避免数据丢失或溢出。

（2）拥塞控制

拥塞控制用于在网络中防止过多的数据流量导致网络拥塞。拥塞控制是一个端到端的过程，涉及到发送方、接收方以及中间设备（如路由器）。

拥塞控制通过监测网络中的拥塞状态来调整发送速率。当网络中发生拥塞时，路由器会发送拥塞通知给发送方，告知其减少数据发送速率。发送方接收到拥塞通知后，会减少发送数据的速率以减轻网络拥塞。通过不断检测和调整发送速率，拥塞控制可以维持网络中的稳定性和公平性。

（3）可靠传输

可靠传输是一种确保数据在通信过程中不丢失、不损坏且按正确顺序到达目标地的机制。实现可靠传输通常需要以下几个关键技术：

序号与确认、超时重传、滑动窗口、错误检测与纠正。

129. 什么是 ARP 协议？说说 ARP 的工作流程以及四种典型情况。★★★

ARP 用于将 IP 地址解析为物理 MAC 地址。每个网络设备都有一个唯一的 MAC 地址，而 IP 地址则用于在互联网上进行通信。

ARP 的工作流程：

当一个设备需要发送数据到另一个设备时，它会首先检查自己的 ARP 缓存表。ARP 缓存表存储了 IP 地址和对应的 MAC 地址。

如果目标设备的 MAC 地址不在 ARP 缓存表中，发送设备将**广播一个 ARP 请求**消息到局域网上的所有设备，询问该目标 IP 地址对应的 MAC 地址。

接收到 ARP 请求的设备会检查自己的 IP 地址是否与请求中的目标 IP 地址匹配，如果匹配，则会回复一个 **ARP 应答消息**，将自己的 MAC 地址提供给发送设备。

发送设备接收到 ARP 应答后，将目标 IP 地址和对应的 MAC 地址**添加到 ARP 缓存表中**，并使用这个 MAC 地址发送数据包。

四种典型情况：

（1）发送方是主机，接收方是本网络的主机：用 ARP 找到目的主机的 MAC 地址。

（2）发送方是主机，接收方是另一个网络的主机：用 ARP 找到本网络路由器的硬件地址，剩下工作交给路由器完成。

（3）发送方是路由器，接收方是本网络的主机：用 ARP 找到目的主机的 MAC 地址。

（4）发送方是路由器，接收方是另一个网络的主机：用 ARP 找到本网络路由器的硬件地址，剩下工作交给路由器完成

130. 什么是滑动窗口协议和停止等待协议，有什么区别，滑动窗口协议有什么优点。★★★

滑动窗口协议：

发送方和接收方都维护了一个固定大小的窗口，用于控制发送和接收的数据包数量。

停止等待协议：

发送方发送一个数据包后，就会停止发送并等待接收方的确认，只有在收到接收方的确

认后，才会发送下一个数据包。

与停止等待协议相比，滑动窗口协议具有以下区别和优点：

（1）发送窗口和接收窗口：滑动窗口协议中，发送方和接收方都维护了一个窗口。发送窗口控制着允许发送的数据包数量，而接收窗口控制着允许接收的数据包数量。发送方可以连续发送多个数据包，而不需要等待每个数据包的确认。

（2）流水线传输：滑动窗口协议支持流水线传输，即发送方可以一次发送多个数据包，而无需等待每个数据包的确认。**这具有更高的传输效率和带宽利用率。**

（3）窗口大小动态调整：滑动窗口协议允许动态调整发送窗口和接收窗口的大小，根据网络状况进行自适应。通过调整窗口大小，可以更好地适应网络的延迟、丢包率等变化，提高传输效率和性能。**这具有更好的动态适应性。**

131. 为什么出现 IPv6，和 IPv4 相比有什么优点？★★★★

IPv6 的可以解决 IPv4 地址短缺的问题。IPv6 还增加了更多的功能和特性，改进首部的格式，快速处理/转发数据报，提供更高效的路由和更好的网络配置灵活性，同时还支持更多的安全特性。

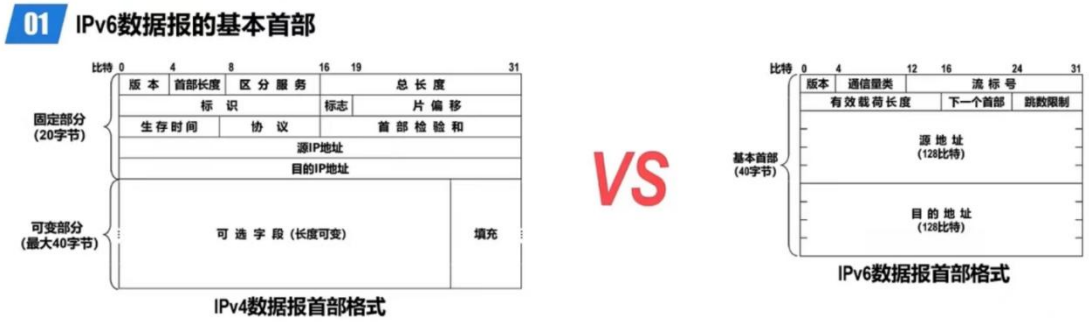
优点：

（1）大规模地址空间：IPv6 采用 128 位地址，相较于 IPv4 的 32 位地址，IPv6 的地址空间巨大到几乎无限。

（2）改进的性能和效率：IPv6 在头部格式和地址分配方式上进行了优化，减少了路由器和主机处理数据包的开销，提高了网络性能和传输效率。

（3）增强网络的安全性：IPv6 在设计时考虑了一些安全机制的加强，例如 IPSec（IP Security）协议的原生支持，提供了对数据传输的认证、加密和完整性校验，增强了网络的安全性。

132. IPv6 和 IPv4 相比，报文首部的变化？★★



（1）IPv6 取消了首部长度字段，因为 IPv6 首部长度固定 40 字节。

（2）IPv6 取消了区分服务字段，IPv6 数据报首部中的通信量类和流表号字段实现了区分服务字段的功能。

（3）IPv6 取消了总长度字段，改为有效载荷长度字段。这是因为 IPv6 数据报的首部长度是固定的 40B，只有后面的有效载荷长度是不变的。

（4）IPv6 取消了标识、标志、片偏移字段，因为这些功能已经包含在 IPv6 数据报的分片扩展首部中。

（5）IPv6 把生存时间 TTL 字段改成跳数限制字段，这样名称与左右更加一致。

（6）IPv6 取消了协议字段，改用下一个首部字段。

（7）IPv6 取消了首部检验和字段，加快路由器处理 IPv6 数据报的速度。

（8）IPv6 取消了选项字段，改用扩展首部来实现选项功能。

133. 简述一下 SDN 的工作原理。★★★★

SDN 通过将网络的控制平面与数据平面分离，并将网络控制逻辑集中到一个中心化的控制器中来实现网络的灵活性和可编程性。

传统的网络架构中，网络设备（如路由器）负责同时处理数据转发和网络控制功能。而在 SDN 中，网络设备仅负责数据转发，而控制逻辑则由集中式的控制器进行管理。控制器作为网络控制平面的中心，通过与网络设备间的 OpenFlow 等协议进行交互，向网络设备下发指令来控制和管理整个网络。

SDN 的工作原理可以简述如下：

（1）分离控制平面和数据平面：SDN 将网络设备的控制功能从设备本身分离出来，在控制器上进行集中管理和配置。这样可以实现控制逻辑的灵活编程和网络策略的集中管理。

（2）集中式控制器：SDN 网络中的控制器作为中心节点，负责网络的全局控制和管理。控制器与网络设备之间通过标准化的协议进行通信，如 OpenFlow。控制器接收来自网络设备的状态信息，来指导数据包的转发行为。

（3）网络编程和应用：在 SDN 中，控制器可以通过编程接口提供给应用程序，让应用能够根据需要控制和管理网络。

134. 简述一下 TCP 和 UDP 的区别。★★★★★

（1）连接性

TCP 是面向连接的协议，它在通信之前需要建立一个可靠的**双向连接**，然后再进行数据传输。

UDP 是无连接的协议，它不需要在通信之前建立连接，直接将数据包发送出去。

（2）可靠性

TCP 提供可靠的数据传输，通过确认机制、重传机制和数据校验等方式来确保数据的完整性和正确性。如果数据包丢失或损坏，TCP 会自动重新发送。

UDP 不提供可靠性保证，它只是简单地将数据包发送出去，不进行确认、重传或校验。如果数据包丢失或损坏，UDP 也不会重新发送，因此可能会导致丢失部分数据。

（3）速度和效率

TCP 对数据传输进行了较多的控制和管理，包括流量控制、拥塞控制和差错检测等。这些额外的机制会增加延迟和网络开销，使得 TCP 的传输速度相对较慢。

UDP 没有这些额外的控制机制，使得它的传输速度更快、延迟更低。它适合用于实时性要求较高的应用，如音视频流媒体。

（4）数据完整性

TCP 保证数据的可靠性和完整性，它会对数据进行序号标记、校验和重传等操作，确保数据包按正确的顺序到达目的地。

UDP 不提供数据的完整性保证，数据包可能会在传输过程中丢失、重复或乱序。因此，如果应用程序需要保证数据的完整性，需要自行实现相关机制。

（5）报文格式

TCP 的首部开销大，首部最小 20 字节，最大 60 字节。

UDP 的首部开销小，只有 8 字节。

135. TCP 的机制是什么？★★★★

（1）三次握手：在建立连接时，客户端和服务器之间进行三次握手来确认彼此的可靠性和同步序列号。握手过程中包含 SYN（同步）和 ACK（确认）标志。

（2）可靠数据传输：TCP 使用序列号对发送的数据进行标记，并通过确认应答机制来确保数据的正确传输。如果发送方没有收到确认应答或者发生超时，会重新发送数据。

（3）流量控制：TCP 通过滑动窗口机制来控制发送方的数据发送速率，防止接收方被淹没。接收方通过通告窗口大小来告知发送方自己的可接收能力。

（4）拥塞控制：TCP 通过拥塞窗口和拥塞避免算法来检测网络的拥塞程度，并动态调整发送速率，以避免网络拥塞的发生。

（5）四次挥手：在关闭连接时，客户端和服务器之间进行四次挥手来正常关闭连接，释放资源。挥手过程中包含 FIN（结束）和 ACK（确认）标志。

136. 详细说说 TCP 的拥塞控制？★★★

TCP 阻塞控制的基本原理是通过监测网络的拥塞情况和调整发送速率来实现。当网络出现拥塞时，数据包的丢失率和延迟会增加，为了避免进一步加重网络负载，TCP 会降低发送数据的速率。

以下是 TCP 阻塞控制的主要机制：

（1）慢启动：在 TCP 连接建立后，发送方初始将拥塞窗口设置为一个较小的值（通常为 1 个 MSS），然后每经过一个往返时间（RTT），拥塞窗口就会加倍。这种机制使得发送方在开始时以较慢的速度逐渐增加发送的数据量，以便观察网络状况。

（2）拥塞避免：慢启动阶段过后，发送方会进入拥塞避免阶段。此时，拥塞窗口的增长速率变为线性增加，即每经过一个 RTT，拥塞窗口只增加一个 MSS。通过缓慢增加发送速率，TCP 可以更好地探测到网络的容量，并避免过度拥塞。

（3）快速重传：当发送方连续收到 3 个重复的确认应答时，它会认为某个数据包丢失，并立即进行重传，而不必等待超时。这可以提前恢复丢失的数据包，减少重传带来的延迟。

（4）快速恢复：在快速重传后，发送方将进入快速恢复状态。此时，发送方会将拥塞窗口减半，并逐渐增加拥塞窗口的大小。这样做是为了降低发送速率，减少网络负载，同时也允许接收方继续接收数据。

137. 建立 TCP 连接为什么要三次握手，而不是不两次握手？★★

TCP 客户进程发出一个 TCP 连接报文段，但该报文段在某些网络节点长时间滞留，必然引起该报文段的超时重传。假设重传的报文段被 TCP 服务器进程正常接收，TCP 服务器进程给 TCP 客户进程发送一个 TCP 请求确认报文段，进入已连接状态；进行数据传输和释放连接后再过一段时间，之前滞留在网络中失效的 TCP 连接请求报文段到达了 TCP 服务器进程，TCP 服务器进程又发起一个 TCP 连接请求，而 TCP 客户进程处于关闭状态，因此不理会该报文段，而 TCP 服务器进程一直等待主机的资源。

138. 流量控制引发的死锁？怎么避免死锁的发生？★★

当发送者收到了一个窗口为 0 的应答，发送者便停止发送，等待接收者的下一个应答。但是如果这个窗口不为 0 的应答在传输过程丢失，发送者一直等待下去，而接收者以为发送者已经收到该应答，等待接收新数据，这样双方就相互等待，从而产生死锁。

为了避免流量控制引发的死锁，TCP 使用了持续计时器。每当发送者收到一个零窗口的应答后就启动该计时器。时间一到便主动发送报文询问接收者的窗口大小。若接收者仍然返回零窗口，则重置该计时器继续等待；若窗口不为 0，则表示应答报文丢失了，此时重置发送窗口后开始发送，这样就避免了死锁的产生

139. DNS 域名解析的过程？★★★

DNS 域名解析是将域名转换 IP。

（1）本地计算机的缓存查询：当用户在浏览器中输入一个域名时，首先会在本地计算机的缓存中进行查询。如果之前已经查询过该域名且缓存未过期，就可以直接获取到 IP 地址，跳过后续步骤。

（2）本地域名服务器查询： 如果本地缓存中没有相关的域名记录，本地计算机会向配置的本地域名服务器发送查询请求。

（3）进行迭代或递归查询： 如果本地域名服务器也没有该域名的缓存记录，它将充当客户端，向根域名服务器发出递归查询请求，询问根域名服务器该域名对应的顶级域名服务

器的 IP 地址。进行迭代或递归查询，根域名服务器->顶级域名服务器->权限域名服务器。

140. DHCP 的作用和工作原理。★★★

作用：

DHCP 是一种基于 UDP 的网络协议，用于动态分配 IP 地址和其他网络配置信息给计算机设备。

工作原理：

DHCP 服务器发现：当计算机设备加入网络或重新启动时，它会广播一个 DHCP 发现消息，以寻找可用的 DHCP 服务器。

DHCP 服务器提供：当有 DHCP 服务器收到 DHCP 发现消息后，它会回复一个 DHCP 提供消息，其中包含了可用的 IP 地址、子网掩码、网关、DNS 服务器等网络配置信息。

DHCP 请求：计算机设备收到 DHCP 提供消息后，会选择一份提供中的 IP 地址配置，并向相应的 DHCP 服务器发送 DHCP 请求消息，请求获取该 IP 地址。

DHCP 确认：DHCP 服务器接收到 DHCP 请求消息后，会向设备发送 DHCP 确认消息，确认分配给设备的 IP 地址和其他网络配置信息。

IP 地址续约：设备在租用 IP 地址的过程中，会周期性地向 DHCP 服务器发送续约请求，以延长 IP 地址的有效期。

IP 地址回收：当设备不再需要 IP 地址或断开与网络的连接时，它会发送 DHCP 释放消息，通知 DHCP 服务器回收该 IP 地址。

141. 为什么有 MAC 地址还需要 IP 地址？★★

（1）各式各样的网络使用不同的硬件地址，IP 地址的作用是屏蔽异构网路的差异，使得不同网络之间的设备可以进行信息交互。

（2）IP 地---址的本质是终点地址，它在跳过路由器的时候不会改变，而 MAC 地址则是下一跳的地址，每一次经过一个路由器都会改变。

（3）分别用 MAC 地址和 IP 地址表示物理地址和逻辑地址方便网络分层，使得网络层和数据链路层的协议可以更灵活地替换。

142. 在 TCP 拥塞控制中，什么是慢开始，拥塞避免，快重传和快恢复算法？★★★

慢开始和拥塞避免：

发送方会维持一个拥塞窗口，刚开始拥塞窗口和发送窗口大小相同，初值为 1。每次收到一个确认，就让拥塞窗口大小变为原来的两倍，并以此类推，形成指数增大。而当窗口值等于慢开始门限值时，就会执行拥塞避免，窗口值每次加 1，形成加法增大。若此时出现网络拥塞，则将拥塞窗口值重新设置为 1，并且修改门限值为发生网络拥塞时的拥塞窗口值的一半；

快重传和快恢复：

当发送方收到三个连续确认时，发送方执行乘法减小策略，将拥塞窗口设置为当前的一半，同时对接收方请求的帧执行快重传算法立刻进行重传而不是等待超时，并且直接执行快恢复算法，即进入拥塞避免状态，采取加法增大每次将拥塞窗口大小+1。

143. 什么叫做 HTTP 无状态，为什么要无状态，如何让它有状态？★★★

HTTP 被称为无状态协议，是因为它本身不会保存任何关于客户端之前请求的状态信息。

无状态的特点意味着每个 HTTP 请求都是独立的，服务器不会记住之前的请求信息。每次请求都需要携带完整的请求信息，服务器接收请求后进行处理并返回响应，然后与客户端断开连接。下次请求时，服务器不会知道这是同一个客户端的请求，也不会知道之前的请求的状态。

HTTP 设计为无状态的主要原因：

可扩展性：无状态使得服务器不需要为每个客户端维护状态信息，可以处理大量的并发

请求。如果有状态可能会增加服务器负担。

简化：无状态使得请求和响应之间的交互相对简单，提高了协议的灵活性。

虽然 HTTP 默认是无状态的，但有时候需要保持会话状态。常见的实现方式有：

使用 Cookie：服务器可以在响应中设置一个包含会话标识符的 Cookie，客户端收到后保存起来，并在后续的请求中将 Cookie 发送给服务器，以此来识别和保持会话状态。

144. 简述 HTTP 和 HTTPS 的区别。★★★★

HTTPS 是基于 HTTP 的加密传输协议，使用 SSL 或 TLS 协议对 HTTP 进行加密，使得传输过程更加安全可靠。过使用公钥加密和私钥解密的方式，对客户端和服务器之间的通信进行加密保护，使得第三方无法轻易获取和篡改传输的数据。

主要区别如下：

（1）安全性：HTTP 不进行数据加密，数据以明文形式传输，容易被拦截、窃取和篡改；而 HTTPS 通过加密技术对数据进行加密，通信过程更加安全。

（2）协议端口：HTTP 默认使用 80 端口进行通信，而 HTTPS 默认使用 443 端口进行通信。

（3）证书认证：HTTPS 使用数字证书来验证网站的身份，确保通信的安全性和可信度，防止中间人攻击等安全问题；而 HTTP 不需要进行证书认证。

（4）网络性能：由于 HTTPS 需要进行数据加密和解密操作，相比 HTTP 会增加一定的计算和传输开销，因此可能会稍微降低网络性能。

应用场景：

HTTPS 通过加密技术保护了通信过程中的数据安全，适用于对数据安全性要求较高的场景，例如**在线支付、用户登录**等；

而 HTTP 适用于一些对数据安全性要求不高的场景，例如**普通的网页浏览**。

145. 在浏览器里输入一个网址，会发生什么？★★★★

（1）对输入的网址进行域名解析

（2）通过三次握手的方式建立 TCP 连接

（3）建立 TCP 链接之后发起 HTTP 请求；

（4）服务器接收并且响应 HTTP 请求；

（5）浏览器解析 HTML 代码，并且请求 HTML 代码中的其他资源；

（6）通过四次挥手断开 TCP 连接；

（7）浏览器对页面进行渲染并且呈现给用户。

十、操作系统★★★★★

146. 操作系统的特点和功能是什么？★★★★

特点：

并发、共享、虚拟、异步。其中，并发和共享是操作系统主要的特点。

功能：

（1）进程管理：操作系统负责管理和调度各个进程（程序的执行实例），包括进程的创建、终止、调度、同步和通信等，以确保它们能够按照一定的顺序和优先级运行。

（2）内存管理：操作系统负责管理计算机的内存资源，包括内存的分配、回收和地址转换等，以确保不同程序和数据可以正确地加载到内存中，并且彼此之间不会互相干扰。

（3）文件系统管理：操作系统提供了对计算机存储设备上文件的组织和访问方式，包括文件的创建、读写、删除、权限控制等，使用户能够方便地管理和利用存储空间。

（4）设备驱动和输入输出管理：操作系统提供了对计算机输入输出设备的管理和控制，

包括设备驱动程序的加载、设备的初始化、读写数据等，使用户能够通过输入输出设备与计算机进行交互。

（5）提供用户界面：操作系统为用户提供了与计算机系统交互的接口，例如命令行界面或图形用户界面（GUI），使用户能够通过输入指令或点击图标来操作计算机。

147. 请你解释一下什么是并发和并行。★★

并发是指多个任务在相同时间段内交替执行的能力。当系统中有多个任务同时存在时，它们可以通过时间片轮转、线程切换等方式交替执行，给用户一种同时执行的感觉。在并发模型中，各个任务之间可能会进行频繁的切换，以达到多任务的同时进行。

并行是指多个任务在同一时刻同时执行的能力。在具备多个处理器或多个核心的系统中，不同的任务可以同时执行，每个任务在独立的处理器上执行，互不干扰。并行计算能够加速任务的执行速度，提高整体的计算能力。

148. 中断与系统调用的区别是什么？★★

中断是 CPU 对例如 I/O 设备发送的中断信号的响应，此时 CPU 应该检测当前是否有正在处理的程序，如果有，则需要保存 CPU 上下文环境，从用户态切换至内核态对中断请求进行处理，完成处理后再返回原处理程序的断点位置继续执行。

中断通常分为硬中断和软中断两种。硬中断是指从硬件发出的中断信号，例如 I/O 请求。而软中断是指由指令执行过程中发出的中断。

系统调用就是一种软中断，它是指通过应用程序通过操作系统间接地调用 I/O 过程。由于应用程序只能执行于用户态，而 I/O 操作需要进入内核态，因此通过中断请求，操作系统进行用户态向内核态的转换完成 I/O 操作。

149. 进程和线程的概念和区别是什么？★★★★★

概念：

进程：进程是计算机中正在运行的程序的实例。每个进程都有自己独立的地址空间，包括**代码、数据和堆栈**等。进程之间是独立的，彼此不会直接共享内存，通信需要通过特定的机制来实现。

线程：线程是进程中的一个执行单元。进程可以包含多个线程，这些线程共享相同的地址空间，即它们可以直接访问相同的内存区域和资源。**线程之间可以通过共享变量来进行通信和同步。**

区别：

（1）资源分配与调度

引入线程前：进程是资源分配、调度的基本单位；

引入线程后：进程是资源分配的基本单位、线程是调度的基本单位；

（2）并发性

引入线程前：只能进程间并发

引入线程后：进程间并发，同一进程的不同线程可以并发，不同进程的不同线程也可以并发

（3）系统开销

引入线程前：进程间并发，需要切换进程的运行环境，系统开销大

引入线程后：线程间并发，如果是同一进程内的线程切换，不需要切换进程环境，系统开销小；线程之间的切换也比进程之间的切换快。

150. 进程有哪几种状态？他们之间如何进行转换？★★★★

进程的基本转态有：就绪、执行、阻塞。

就绪状态：进程已经分配到除了处理机之外所有的执行所需要的资源，一旦获得处理器就可以立刻执行；

执行状态：进程获得处理机并且在处理器上执行；

阻塞状态：当执行中的程序由于等待某个事件发生而无法执行时，放弃处理机从而进入阻塞状态，例如等待 I/O 完成。

三种状态之间的相互转换：

执行状态的程序遇到 I/O 请求等将会转入阻塞状态。

I/O 请求完成时会进入就绪状态。

获得处理机之后将重新进入执行状态。

执行状态的程序当时间片用完时，将会重新回到就绪状态。

151. 进程的调度策略有哪些？★★★★★

（1）先来先服务

在作业调度中，该算法每次会从后备队列中选择最先进入该队列的作业调入内存，为其分配资源并且创建进程，并放入就绪队列；在进程调度中，算法将从就绪队列中选择最先进入队列的进程，分配处理机执行。（对短作业不友好）

（2）短作业优先

在进程调度之前，操作系统将获取后备队列中作业的执行时间或者就绪队列中进程的执行时间，并且选择时间最短的优先进行调度。该算法会导致长作业或者长进程的饥饿现象。（对长作业不友好）

（3）优先权调度

为了照顾一些紧迫类型的作业，为每个作业按照紧急程度赋予了优先级。在进行作业调度时，系统将从后备队列中选择拥有高优先权的作业装入内存。该算法可以分为抢占式和非抢占式，在抢占式方式下，一旦在程序执行期间有比当前执行程序优先权更高的进程进入就绪队列，调度程序将立即停止当前执行程序，转而将处理机分配给高优先权的进程；而在非抢占方式下，执行的程序会一直执行直至完成或者主动放弃处理机，调度程序才进行调度。

（4）高响应比优先调度

为了解决短作业优先算法中长作业的饥饿问题，该算法为每个作业引入一个动态优先权，该优先权等于（等待时间 + 要求服务时间）/ 要求服务时间，该算法在利用短作业优先的思想之下，同时照顾等待时间长的作业（周转时间/要求服务时间）；

（5）轮转算法

系统将所有就绪进程排成一个队列，在每次调度时，处理机将分配给队列的起一个进程，并且令其执行一个时间片长度。当时间片执行完时，由计时器发出一个时钟请求，此时调度程序停止该程序并将其移动至队尾；

（6）多级反馈队列调度

该算法不需要知道各种进程的执行时间，同时可以满足各种不同类型进程的需求，例如短进程会很快完成，长进程不会饥饿。算法的执行过程为：

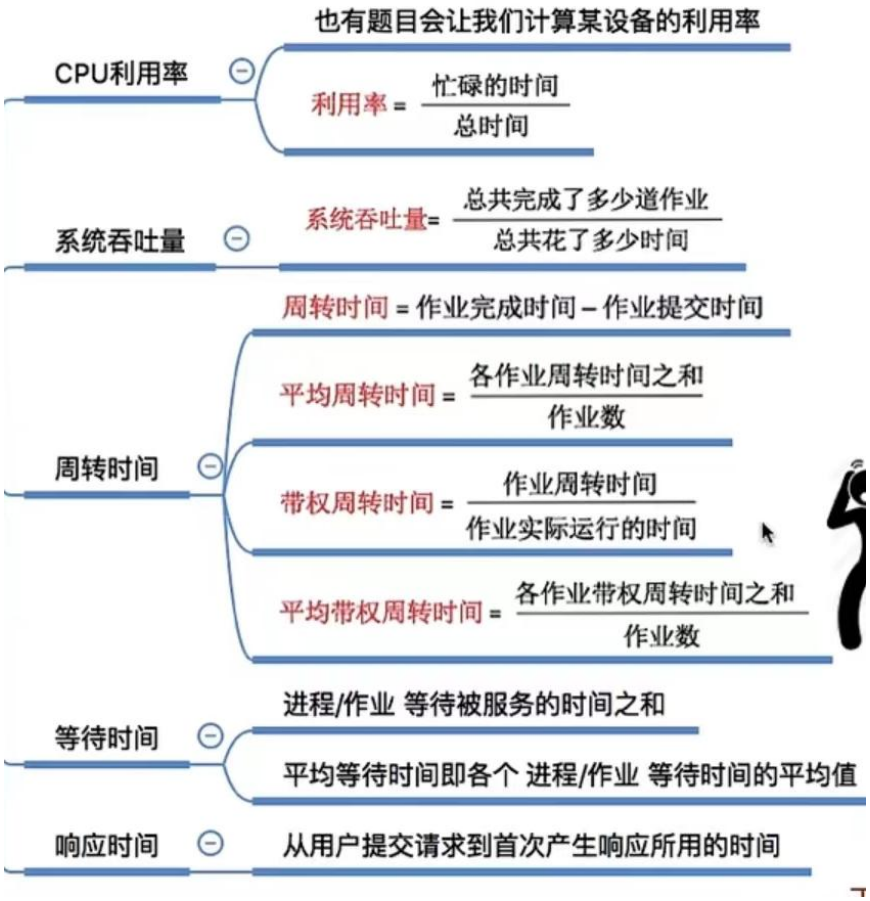
①设置多个就绪队列，每个队列的优先级不同。第一个队列优先级最高并依次递减，优先级越低的队列时间片越长，一般按照倍数增长；

②当一个新的进程进入内存后，其首先被放入第一队列的队尾，并且按照先来先服务的原则等待调度。若其在第一个时间片可以完成执行，则将撤出系统；若不能，则其在时间片执行完毕之后会被移动至下一个就绪队列的队尾，如此反复；

③上一级队列为空的时候，下一级队列才开始进行调度。

152. 进程的调度性能指标有哪些？★★★★

CPU 利用率、系统吞吐量、周转时间、平均周转时间、等待时间、带权周转时间=作业周转时间/作业实际运行的时间（带权周转时间越大，说明相对等待得越久）、平均带权周转时间。



153. 进程之间的通信方式有哪些？★★★

（1）共享存储器系统

相互通信的进程共享某些数据结构或者共享存储区，并且借助这些空间进行通信，主要有基于共享数据结构的通信方式（例如队列实现的缓冲区）与基于共享存储区的通信方式（例如内存中划分出来的共享区）。

（2）管道通信系统

管道是连接一个读进程与一个写进程之间通信的一个共享文件，又称为 pipe 文件，它必须被互斥地访问，即运作于半双工方式。同时必须进行同步，即一方写满后另一方才读，或者一方读空后另一方才写。

（3）消息传递系统

在该机制中，以格式化的消息为单位，将通信的数据封装在消息中，并利用操作系统提供的一组通信命令（原语）在进程之间进行数据交换。

154. 什么是死锁？产生死锁的原因？产生死锁的必要条件是什么？★★★★★

死锁的定义：

死锁是指多个进程因为竞争资源而造成相互等待的情况，若无外力作用，这些进程都无法向前推进。

产生死锁的原因：

①竞争不可抢占型资源

②竞争可消耗型资源（硬件中断、信号等）

③进程推进顺序不当。

产生死锁的四个必要条件：

①互斥条件：进程对所获得的资源必须互斥访问，即同一时刻只能有一个进程访问；

②请求和保持：进程在至少获得一个资源的情况下，又提出新的资源需求；

③不可抢占：进程所获得的资源在其完成之前不可被其他进程抢占；

④循环等待链：当发生死锁时，必然存在一个进程——资源循环等待链。

155. 解决死锁有什么办法？★★★★★

（1）死锁的预防：破坏死锁产生的四个条件之一即可。

（2）死锁的避免：银行家算法。先看能不能给他，再假设给他，看是否存在一条安全性序列，不存在的话就不安全。（不安全的状态有可能会导致死锁，安全状态不会的）。

（3）死锁的检测：构造资源分配表，看是否能够化简完全（即看是否存在环路）。

（4）死锁的解除：一般是撤销一部分进程使得死锁解除。

156. 如何预防死锁？★★★★★

预防死锁可以通过破坏产生死锁的四个必要条件中的一个实现。

（1）破坏互斥条件：如 Spooling 技术。使用 Spooling 技术，可以将所有的 I/O 请求都先缓冲起来，然后按顺序依次处理。

（2）破坏请求保持条件：资源一次性分配，则一次性分配进程运行所需要资源，或分配进行运行初期需要的资源，在进程运行过程中逐步释放资源并且请求新资源

（3）破坏不可抢占条件：当进程获得部分资源，但是得不到其他资源时，它将释放已经占有的资源

（4）破坏循环等待链：资源有序分配，即系统为没类资源赋予一个编号，每一个进程按照编号递增的顺序请求资源。

157. 什么是银行家算法？★★★

银行家算法用于解决操作系统中的资源分配问题和预防死锁。

银行家算法的核心思想是通过动态地检查系统资源的状态，来判断是否分配资源会导致死锁，并只允许安全的分配。它基于资源的最大需求量和当前已分配资源的情况，对进程提出资源请求进行评估。

银行家算法的实现过程为：

设置四种数据结构：

Avaliable（系统中所有可用资源数量）；

Max（系统中 n 个进程对 m 类资源的最大需求量）；

Allocation（每类资源已经分配给进程的资源数）；

Need（每个进程尚需的各类进程数）。

申请过程中，Requires 需要小于 Need 与 Avaliable，若满足，则系统试探性的进行分配，并执行安全性算法，假设将可用资源分配给某一进程，进程完成后释放资源。如果所有进程可以完成则说明系统仍处于安全状态，则此次分配成功。否则，不进行分配并且等待。

158. 哲学家进餐有哪些实现方式？★★★

（1）最多只允许 n-1 个哲学家拿起筷子就餐

（2）奇数号哲学家拿先拿左边的筷子，偶数号哲学家先拿右边的筷子

（3）当一个哲学家拿起一只筷子时，如果另外一只筷子无法拿起，则放下刚刚拿起的筷子

159. 简单操作系统内存管理的功能。★★★

（1）内存的分配与回收

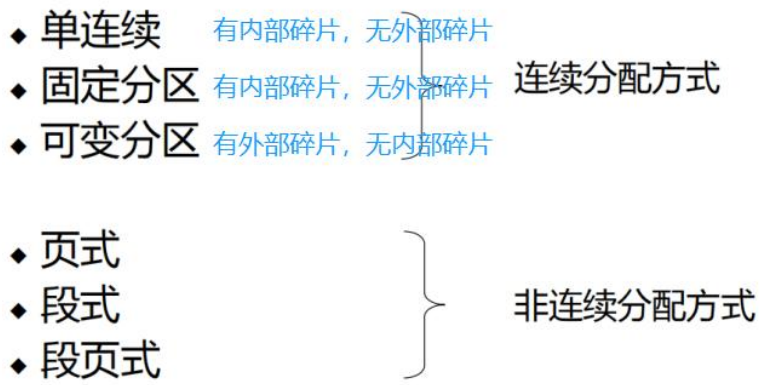
（2）地址转换（逻辑地址->物理地址）

（3）内存保护（防止越界，非法访问）

（4）内存共享（多个进程对应一个程序）

（5）内存扩充（覆盖、交换、虚拟存储器）

160. 简单叙述一下内存空间的分配。★★★★★



连续分配管理方式，用户进程分配的必须是一个连续的内存空间。连续分配方式包括：

单一连续分配：整个内存空间只分配给一个进程，其他进程被限制在外部存储器上。

固定分区分配：将内存划分为若干固定大小的区域（可以分区大小相等，也可以分区大小不等），每个区域只能分配给一个进程使用。

动态分区分配：将内存划分为不同大小的区域，根据进程的大小动态地分配合适大小的空闲内存块。

非连续分配管理方式：为用户进程分配的不一定是连续的内存。非连续分配方式包括：

分页方式：将物理内存和进程的逻辑地址空间分成固定大小的页，从而实现将进程的页映射到物理内存的任意位置。

分段方式：将进程的逻辑地址空间划分为多个逻辑段，每个逻辑段映射到物理内存的任意位置。

段页式方式：结合了分段和分页的特点，将进程的逻辑地址空间划分为多个段，每个段再被分成固定大小的页。

161. 简单叙述一下动态分区分配算法。★★★★★

（1）首次适应算法

从空闲内存块列表中找到第一个满足进程需求的空闲块，分配给进程，并将剩余的空闲块插入到空闲块列表中合适的位置。这种算法执行效率较高，但可能会导致内存碎片问题。

（2）最佳适应算法

在空闲内存块列表中找到最小且能满足进程需求的空闲块，分配给进程。这种算法能够更好地利用内存，减少内存碎片，但执行效率较低。

（3）最差适应算法

在空闲内存块列表中找到最大的空闲块，然后将其分配给进程。这种算法可以产生较大的空闲块，但也容易导致外部碎片问题。

162. 简单叙述一下动态分区回收算法。★★★

上邻空闲区。与上空闲区合并，修改其长度。

下邻空闲区。与下空闲区合并，修改其长度和首址。

上、下邻空闲区。与上、下空闲区合并，修改其长度和首址，从空闲区表中或空闲链表中删除下接的空闲区表项。

上下都不相邻空闲区时，在空闲区表中或空闲链表中添加新项，添上被回收分区长度和首址。

163. 非连续内存分配中，分页的好处是什么，分页和分段有什么区别？★★★★★★

分页的好处：

（1）实现了虚拟内存：分页将进程的逻辑地址空间划分为固定大小的页，并将物理内存也划分为相同大小的页框。这样，每个进程只需要将所需的页加载到物理内存中，而不需要一次性加载整个进程。

（2）内存利用率高：分页可以更灵活地分配内存，当进程的大小不是固定的且不规则时，分页可以更好地利用内存空间，减少内存碎片的产生。

（3）页表简单：由于页的大小是固定的，页表可以更加简单有效地实现。

分页和分段的区别：

（1）单位不同：分页以固定大小的页作为单位进行管理，而分段以逻辑段作为单位进行管理。分页的大小是固定的，由操作系统指定；而分段的大小是可变的，由程序员决定。

（2）地址映射方式不同：在分页中，逻辑地址被划分为页号和页内偏移量，通过页表进行映射。而在分段中，逻辑地址是由段号和段内偏移量组成，使用段表进行映射。

（3）内存碎片情况不同：由于分页使用固定大小的页，会产生内部碎片（页内的未利用空间），而分段可以更好地适应变长的逻辑段，避免内部碎片。

（4）管理灵活性不同：分页更适合处理不规则且动态变化的进程需求，而分段更适合处理结构化的程序，如编译器、数据库等。

164. 什么是虚拟内存？什么是共享内存？★★★

（1）虚拟内存

虚拟内存是一种计算机系统的内存管理技术，它扩展了物理内存的容量，并允许进程能够使用比实际可用物理内存更大的地址空间。当进程需要访问一个不在物理内存中的页时，操作系统会将该页从外部存储器中加载到物理内存，这个过程被称为页的置换。虚拟内存的主要优势包括：提供了比物理内存更大的地址空间、允许多个进程并发执行、简化了内存管理等。

（2）共享内存

共享内存是一种进程间通信的机制，它允许多个进程访问同一块物理内存区域，从而实现数据的共享和交换。共享内存适用于需要高效地进行大量数据传输的场景，比如多个进程之间需要共享大型数据结构、缓存或其他共享资源时。

（3）总结

虚拟内存和共享内存是两个不同的概念和技术，虚拟内存是一种扩展物理内存容量的技术，而共享内存是进程间通信的一种机制，用于实现数据的共享和交换。

165. 有哪些页面置换算法？请简述。★★★★★

（1）最佳置换算法（OPT）

每次选择淘汰的页面是以后最长时间内不再被访问的页面。这种算法是最理想的算法，目前无法实现，但可以作为衡量一个置换算法好坏的标杆。

（2）先进先出置换算法（FIFO）

每次选择淘汰最先进入内存的页面，这种置换算法保证了公平性，但是会出现 belady 现象，即当内存空间增大时，页命中率下降。

（3）最近最久未使用算法（LRU）

每次淘汰的页面是最近最久未被使用的页面，该算法利用了栈的特性，因此不会出现 belady 现象。

（4）时钟置换算法（Clock）

为每个页面设置一个访问位，页面被访问时，将其访问位设置为 1，每次系统选择淘汰访问位为 0 的页面。每次进行淘汰时，系统循环地检查每个页面访问位，若为 1，则设置为 0；若为 0，则淘汰页面。Clock 算法给了页面留存内存的第二次机会。

（5）改进时钟算法

在访问位的基础上添加一个修改位，两个位组合一共有四种组合：00,01,10,11。扫描到第一个 00，则将页面淘汰；若第一轮失败，则第二轮扫描第一个 01 淘汰，并且将 10 设置为 00，将 11 设置为 01；若前两轮失败，则重复前两轮。因为此时 10 与 11 已经被更新为 00 和 01，因此一定可以找到用于替换的页面。

166. 有哪些磁盘调度算法？请简述。★★★★★

（1）先来先服务（FCFS）

这是一种公平调度算法，其根据请求访问磁盘的先后顺序进行调度；

（2）最短寻道时间优先（SSTF）

该算法考虑当前磁头距离哪个被请求的磁道，则哪个磁道将被优先访问。但是该算法可能产生饥饿现象；

（3）扫描算法（SCAN）：

该算法是先满足一个方向所有的请求，再满足反方向的所有请求，如此循环往复。该算法如果在一个方向上已经没有请求，则可以提前掉头，无需扫描到末端；

（4）循环扫描算法（CSCAN）：

该算法在 SCAN 算法的基础上，规定磁头只能单向移动，即只满足同一个方向上的所有请求；

167. 磁盘存储管理有哪些方法？★★★★

（1）连续分配：要求每一个文件分配一组相邻接的盘块，访问速度快，但是必须事先知道文件的大小；

（2）链接分配：分为隐式链接和显示链接。隐式链接是指在每个目录项中都含有指向链接文件第一盘块和最后一个盘块的指针，每个盘块都有指向下一个盘块的指针，显示链接是指把用于链接文件各物理块的指针都存放在内存中的一张链接表中。它不支持随机访问，但是利于文件的动态增长；

（3）索引分配：分为单级索引分配以及多级索引分配。它为每个文件分配一个索引表，把分配给该文件的所有盘号都记录在该索引块中。它不能支持高效的直接存取。

168. 文件系统中文件是如何组织的？★★★★

（1）逻辑结构

文件的逻辑组织结构通常分为有结构文件和无结构文件。其中，有结构文件即在逻辑上被视为一组连续记录的集合的文件，分为定长记录文件和不定长记录文件两种；无结构文件则是内部不在划分记录，而是由一组相关信息组成的有序字符流。

（2）物理结构

文件的物理结构通常分为**顺序文件、索引文件和索引顺序文件**。顺序文件是指由一系列记录按照某种顺序排列所形成的文件，索引文件是指为变长记录建立一张索引表，为每个记录设置一个表项，从而加速对记录的检索。索引顺序文件是为一组记录的第一个记录建立索引表。

169. 一段代码是怎么装入内存的★★

（1）编译

首先，开发者使用编译器将源代码翻译为目标文件。编译过程包括词法分析、语法分析、语义分析、代码生成等步骤。

（2）目标文件生成

编译器生成的目标文件包含了已编译的机器指令、数据和符号表等信息。目标文件的格式可能是可执行文件格式（如 ELF、Mach-O、PE 等）或者是对象文件格式（如.o、.obj 等）。

（3）链接

如果代码依赖于其他模块或库文件中的函数或变量，链接器会将这些模块或库文件与目

标文件进行链接，解决符号引用和定义之间的关联。链接器会创建一个最终可执行文件或共享库文件。

（4）装入

操作系统负责将生成的可执行文件或共享库文件从存储介质（如硬盘）加载到内存中。加载过程包括分配足够的内存空间来存放代码段、数据段以及其他必要的信息。

（5）重定位

在加载过程中，操作系统可能需要进行地址重定位。因为程序在内存中的实际地址与编译时或链接时使用的虚拟地址可能不一致。重定位的目的是确保程序在内存中的地址能够正确地访问和跳转。

170. 说一说链接的三种方式。★★★

链接是将编译后的多个模块或库文件合并为一个**可执行文件**的过程。

（1）静态链接

在程序运行之前先将各目标模块及它们所需的库函数连接成一个完整的可执行文件（装入块），之后不再拆开。

（2）装入时动态链接

将各目标模块装入内存时，边装入边链接的链接方式。

（3）运行时动态链接

在程序执行中需要该目标模块时，才对它进行链接。其优点是便于修改和更新，便于实现对目标模块的共享。

十一、计算机组成原理★★★

171. 什么是摩尔定律？★

当价格不变时，集成电路上可容纳的元器件的数目，约每隔 18~24 个月便会增加一倍，性能也将提升一倍。

172. 什么是冯诺依曼机？冯诺依曼结构的特点是什么？什么是存储程序？★★★

冯诺依曼机

“存储程序”的思想奠定了现代计算机的基本结构，以此概念为基础的各类计算机通称为冯·诺依曼机。

特点：

（1）计算机硬件系统由运算器、存储器、控制器、输入设备和输出设备 5 大部件组成。

（2）指令和数据以同等地位存储在存储器中，并可按地址寻访。

（3）指令和数据均用二进制代码表示。

（4）指令由操作码和地址码组成，操作码用来表示操作的性质，地址码用来表示操作数在存储器中的位置。

（5）指令在存储器内按顺序存放。通常，指令是顺序执行的，在特定条件下可根据运算结果或根据设定的条件改变执行顺序。

（6）早期的冯诺依曼机以运算器为中心，输入 / 输出设备通过运算器与存储器传送数据。现代计算机以存储器为中心。

存储程序

“存储程序”的概念是指将指令以代码的形式事先输入计算机的主存储器，然后按其在存储器中的首地址执行程序的第一条指令，以后就按该程序的规定顺序执行其他指令，直至程序执行结束。

173. 简述一下计算机的主要性能指标。★★

（1）机器字长

机器字长是指计算机进行一次整数运算（即定点整数运算）所能处理的二进制数据的位数。

（2）数据通路带宽

数据通路带宽是指数据总线一次所能并行传送信息的位数。

（3）主存容量

（4）运算速度

①吞吐量和响应时间

②主频和 CPU 时钟周期。

174. 为什么 Dram 需要刷新？★★

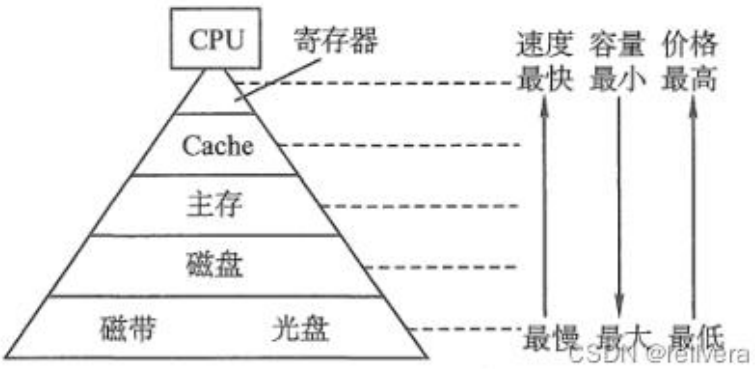
DRAM 是一种易失性存储器，它通过电容来存储数据。每个存储单元由一个电容和一个访问晶体管组成，电容的充放电状态表示存储的数据值。然而，电容存储的电荷会逐渐漏失，导致存储的数据值逐渐衰减。为了保持数据的正确性，DRAM 需要周期性地进行刷新操作。

175. 简述多级存储系统。★★★

为了解决存储系统大容量、高速度和低成本 3 个相互制约的矛盾，在计算机系统中，通常采用多级存储器结构，在图中由上至下，位价越来越低，速度越来越慢，容量越来越大，CPU 访问的频度也越来越低。

实际上，存储系统层次结构主要体现在“Cache－主存”层次和“主存－辅存”层次。

前者主要解决 CPU 和主存速度不匹配的问题，后者主要解决存储系统的容量问题。



176. 什么是 Cache？Cache 有什么作用？★★★★

Cache

高速缓冲存储器，是位于 CPU 和主存储器 DRAM 之间，规模较小，但速度很高的存储器，通常由 SRAM 组成。

作用

是提高 CPU 数据输入输出的速率（解决 CPU 和主存速度不匹配的问题）。Cache 容量小但速度快，内存速度较低但容量大。

177. Cache 三种映射方式的优缺点是什么？★★★★

Cache 的三种映射方式分别是直接映射、全相联映射和组相联映射。

直接映射

优点：直接映射是最简单和最直接的映射方式，在硬件设计上较为简单，存储器结构相对较小，成本低廉。

缺点：容易产生冲突，当不同的主存块被映射到相同的缓存块时，就会发生冲突，导致性能损失。

全相联映射

优点：全相联映射可以避免直接映射中的冲突问题，因为任何一个主存块都可以映射到任何一个缓存块中。

缺点：存储器结构复杂，需要额外的硬件支持，包括比较电路和选择电路，因此成本较高。另外，由于所有块都可映射到任意位置，查找所需的时间长，不利于快速访问。

组相联映射

优点：组相联映射综合了直接映射和全相联映射的特点，在硬件结构设计上既可以减小成本，又能够一定程度上解决冲突问题。

缺点：虽然减小了全相联映射的比较电路和选择电路的复杂度，但仍然需要进行块在组内的比较。此外，如果组的数量设置不合理，仍然可能导致冲突，性能下降。

178. 介绍一下多级 Cache。★★

多级缓存是一种层次化的缓存结构，通常包括 L1、L2、L3 等多个级别的缓存。它的设计目的是提供更高的访问速度和更好的性能。

多级缓存的主要思想是根据访问速度和容量的权衡，将缓存划分为多个级别，每个级别的缓存大小和访问速度逐级降低。通常，L1 缓存是距离处理器（CPU）最近的、最小但速度最快的缓存，而 L2 和 L3 缓存则相对较大但速度较慢。这种层次化的缓存结构可以有效地利用不同类型的缓存来提高缓存命中率，并减少对主存的访问次数，从而加快数据访问速度。

优点：

（1）更快的访问速度：由于 L1 缓存离处理器最近且速度最快，能够更快地响应 CPU 的数据访问请求，减少访问延迟。

（2）更高的命中率：多级缓存可以利用不同的缓存级别来提高命中率，减少对主存的访问次数，提高数据的局部性。

（3）更大的容量：通过增加多个级别的缓存，整个缓存系统的容量也得到扩展，能够存储更多的数据，提供更高的数据吞吐量。

（4）降低总线和内存带宽的负载：多级缓存可以减少对总线和主存的访问次数，从而减轻了内存带宽的压力，提高系统整体的效率。

179. 介绍一下写更新策略。★★

缓存更新算法是用于决定何时在缓存中更新数据的策略。以下是几种常见的缓存更新算法：

（1）全写法（写命中时）

直写策略在每次数据写入操作时，会立即更新缓存和主存中的对应数据。

优点：保持缓存和主存的数据一致性，可以避免数据丢失。

缺点：写操作的延迟较高，因为需要等待数据写入到主存后才能继续执行。

（2）写回法（写命中时）

回写策略将数据修改操作仅限于缓存中进行，而不是立即更新主存。

当被修改的数据被替换出缓存时，才将其写回主存。也可以通过显式写回指令控制写回时机。

优点：减少了写操作的延迟，提高了缓存访问速度。

缺点：可能导致缓存中的数据和主存数据不一致，需要额外的机制来维护缓存和主存之间的一致性。

（3）写分配法（写不命中时）

写分配策略在发生写操作时，将数据加载到缓存中，并在缓存中进行修改。

这样可以利用缓存来加速写操作，避免频繁地访问主存。

优点：提高了写操作的性能，减少了对主存的访问次数。

缺点：会消耗更多的缓存空间，因为要将数据加载到缓存中。

（4）非写分配法（写不命中时）

不写分配策略在发生写操作时，直接更新主存，而不将数据加载到缓存中。

只有当相应的数据已经在缓存中时，才进行写操作。

优点：减少了不必要的数据加载和缓存内部的写操作。

缺点：可能增加主存访问次数，降低了写操作的性能。

180. 简述一下指令执行的过程。★★

下面以取数指令（即将指令地址码指示的存储单元中的操作数取出后送至运算器的 ACC 中）为例进行说明。

（1）取指令：PC->MAR->M->MDR->IR

根据 PC 取指令到 IR，将 PC 的内容送 MAR，MAR 中的内容直接送地址线，同时控制器将读信号送读 / 写信号线，主存根据地址线上的地址和读信号，从指定存储单元读出指令，送到数据线上，MDR 从数据线接收指令信息，并传送到 IR 中。

（2）分析指令：OP(IR)->CU

指令译码并送出控制信号。控制器根据 IR 中指令的操作码，生成相应的控制信号，送到不同的执行部件。在本例中，**IR 中是取数指令**，因此读控制信号被送到总线的控制线上。

（3）执行指令：Ad(IR)->MAR->M->MDR->ACC

取数操作。将 IR 中指令的地址码送 MAR，MAR 中的内容送地址线，同时控制器将读信号送读 / 写信号线从主存指定存储单元读出操作数，并通过数据线送至 MDR，再传送到 ACC 中。

181. 说说 CPU 的功能。★★

CPU 由运算器和控制器组成。其中，控制器的功能是负责协调并控制计算机各部件执行程序的指令序列，包括取指令、分析指令和执行指令；运算器的功能是对数据进行加工。

CPU 的具体功能包括：

（1）指令控制。完成取指令、分析指令和执行指令的操作，即程序的顺序控制。

（2）操作控制。一条指令的功能往往由若干操作信号的组合来实现。CPU 管理并产生由内存取出每条指令的操作信号，把各种操作信号送往相应的部件，从而控制这些部件按指令的要求进行动作。

（3）时间控制。对各种操作加以时间上的控制。时间控制要为每条指令按时间顺序提供应有的控制信号。

（4）数据加工。对数据进行算术和逻辑运算。

（5）中断处理。对计算机运行过程中出现的异常情况和特殊请求进行处理。

182. 影响流水线性能的因素。★★★

（1）结构相关

是当多条指令同一时刻争用同一资源形成冲突。

解决方案：

①暂停一个时钟周期

②单独设置数据存储器和指令存储器

（2）数据相关

是指令在流水线中重叠执行时，当后继指令需要用到前面指令的执行结果时发生的。

解决方案：

①暂停一个时钟周期

②数据旁路：把前一条指令的 ALU 计算结果直接输入到下一条指令

（3）**控制相关**

是当流水线遇到分支指令和其他改变 PC 值的指令时引起的。

解决方案：

①延迟转移技术。将转移指令与其前面的与转移指令无关的一条或几条指令对换位置，让成功转移总是在紧跟的指令被执行之后发生，从而使预取的指令不作废。

②转移预测技术。

183. 说说 CISC 和 RISC 的区别。★★★

类别 / 对比项目	CISC	RISC
指令系统	复杂，庞大	简单，精简
指令数目	一般大于 200 条	一般小于 100 条
指令字长	不固定	定长
可访存指令	不加限制	只有 Load/Store 指令
各种指令执行时间	相差较大	绝大多数在一个周期内完成
各种指令使用频度	相差很大	都比较常用
通用寄存器数量	较少	多
目标代码	难以用优化编译生成高效的目标代码程序	采用优化的编译程序，生成代码较为高效
控制方式	绝大多数为微程序控制	绝大多数为组合逻辑控制
指令流水线	可以通过一定方式实现	必须实现

184. 解释一下指令周期和时钟周期。★★★

指令周期是指完成一条指令的基本操作所需要的时间，也称为执行周期或机器周期。在经典的冯·诺伊曼计算机结构中，一个指令周期通常包括以下几个阶段：取指、译码、执行、存储器访问、写回。

时钟周期是计算机内部时钟震荡器发出一个完整的周期所需的时间。

指令周期是由若干个时钟周期组成的，每个阶段在不同的时钟周期中执行。每个时钟周期都是固定且稳定的，它们共同构成了整个指令周期。

185. 介绍一下 IO 方式。★★★★

（1）程序查询方式

程序查询方式的核心问题是每时每刻需要不断查询 I/O 设备是否准备好，浪费了 CPU 大量的时间。

（2）程序中断方式

是在一条指令执行完成后检查是否有中断发生，若出现中断则处理，否则继续执行下一条指令。

（3）DMA 方式

DMA 方式是指外部设备不通过 CPU 而直接与主存进行数据交换的方式。DMA 控制器发出请求，CPU 让出系统总线由 DMA 接管，让外设可以直接访问内存，这样外设的读写就不需要 CPU 参与，降低了 CPU 的占用率。

（4）通道方式

通道方式是构建一个通道，主存和 I/O 设备之间的数据交换通过通道来完成，通道方式是对 DMA 的发展，由一个数据块的读写发展成为一组数据块的处理。

186. 介绍一下 DMA 技术。★★★★

DMA 是 Direct Memory Access（直接内存访问）的缩写。通过使用 DMA，设备可以直接读取和写入系统内存，而无需通过 CPU 进行中介。这种方式可以提高数据传输的效率和系统的性能。

在传统的 I/O 操作中，CPU 负责处理数据的传输。当设备需要读取或写入数据时，它会

发出中断请求，使 CPU 中断并处理相应的 I/O 操作。这种方式效率较低，因为每次传输都需要占用 CPU 的时间和资源。

而使用 DMA 技术，设备可以直接与系统内存进行通信，而无需通过 CPU 的干预。DMA 控制器负责管理数据传输的过程。它通过在总线上请求总线的控制权，将设备与系统内存连接起来，并直接在两者之间完成数据的传输。这样可以减轻 CPU 的负担，提高数据传输的速度和效率。

DMA 广泛应用于各种设备，如硬盘驱动器、网络适配器、声卡、显卡等。它可以提高存储设备的读写速度，加快网络传输速度，改善音频和视频的播放效果等。同时，DMA 也使得 CPU 能够更好地专注于执行其他任务，提高系统的整体性能。

总之，DMA 是一种通过直接访问内存来实现设备之间高速数据传输的技术，可以提高系统性能和数据传输效率。

187. 什么是总线？总线有什么好处？总线有什么功能？★★★

总线：

总线是指计算机系统中连接各个硬件组件的物理通道或逻辑通道。总线可以传输数据、地址和控制信号，允许不同的设备之间进行通信和交换信息。

好处：

（1）简化硬件设计：总线提供了一种标准化的接口和通信方式，使得硬件设计变得更加简单和灵活。设备可以通过总线连接到系统，而不需要与每个设备直接连接。

（2）提高可扩展性：总线允许在计算机系统中添加或移除设备，以满足不同的需求。通过总线，可以轻松地连接新设备，而无需对整个系统进行重大改动。

（3）实现设备间的通信：总线提供了设备之间进行数据和信号交换的通道。不同的设备可以通过总线发送和接收数据，实现彼此之间的通信和协作。

（4）节省成本和资源：使用总线可以减少硬件的复杂性和成本。它可以共享信号线和通信资源，减少了系统所需的电路板空间和连接线数量。

功能：

（1）数据传输：总线用于在不同的设备之间传输数据。它提供了数据线路，使得设备能够发送和接收数据。

（2）地址传输：总线可以传输地址信息，指示数据在内存中的位置或设备的寻址方式。

（3）控制信号传输：总线传递控制信号，用于启动、停止、同步和调度数据传输。

（4）中断传输：总线还可以传输中断信号，以通知 CPU 需要处理的事件或请求。

188. 介绍一下磁盘阵列。★★★

磁盘阵列，也称为 RAID，是将多个独立的硬盘驱动器组合在一起，形成一个单独的存储单元。磁盘阵列通过在多个硬盘驱动器上分布和复制数据，提供了数据冗余、容错性、高性能和扩展性。

常见的磁盘阵列级别：

RAID 0：RAID 0 没有冗余功能，一块磁盘损坏会导致所有数据丢失。

RAID 1：镜像（Mirroring），所有数据同时写入两个磁盘，提供数据冗余和高可靠性。当一个磁盘发生故障时，另一个磁盘仍然可以正常工作。

RAID 5：分布式奇偶校验，将数据和校验信息分散存储在多个磁盘上，提供数据冗余和较高的读取性能。当一个磁盘出现故障时，可以通过校验信息和其他磁盘上的数据进行恢复。

RAID 6：双分布式奇偶校验，与 RAID 5 类似，但使用两个奇偶校验块提供更高的容错能力。即使同时发生两个磁盘故障，数据仍然可以恢复。

189. 在计算机所有组成部分中，哪些部分的性能会影响大批量图像输入的效率？★★★

（1）CPU（中央处理器）：CPU 的性能对于图像处理是至关重要的。高性能的多核 CPU 可以提供更快的图像处理速度和并行计算能力。

（2）GPU（图形处理器）：在图像处理中，GPU 可以加速大规模并行计算，例如图像的特征提取、滤波和深度学习等任务。具备高性能 GPU 的计算机能够快速处理大量的图像数据。

（3）内存：大批量图像输入通常需要占用大量内存空间。有足够的 RAM 可以避免频繁的磁盘读取和写入操作，提高图像处理的效率。

（4）硬盘/存储：存储速度对于读取和写入大批量图像数据非常重要。使用高速硬盘（如 SSD）或者专门的存储系统可以减少 I/O 延迟，提高数据的读写速度。

（5）网络带宽：如果大批量图像输入涉及到网络传输，那么网络带宽将成为一个关键因素。具有高速、稳定的网络连接可以提高图像传输的速度和效率。

（6）图像处理算法和软件优化：除了硬件性能外，优化图像处理算法和软件实现也是提高效率的关键。使用高效的算法和进行针对性的软件优化可以更好地利用硬件资源，提高图像处理的速度和效果。

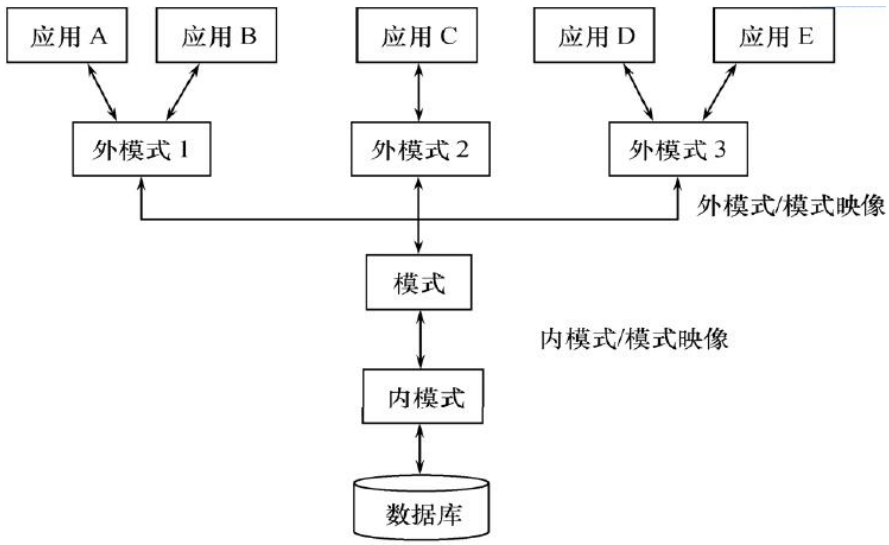
十二、数据库★★★★

190. 数据库系统 DBS 和数据库管理系统 DBMS 的区别是什么？★★★

数据库系统是指由数据库、数据库管理系统和应用程序组成的数据管理系统。它是为了有效地组织、存储、管理和检索大量数据而设计的软件系统。

数据库管理系统是一种软件系统，用于管理数据库中的数据。它提供了一组操作数据库的工具和接口，允许用户定义、创建、修改和查询数据库中的数据。

191. 数据库系统的三级模式结构是什么？★★★



（1）外模式

也称子模式或用户模式，它是数据库用户能够看见和使用的局部数据的逻辑结构和特征的描述，是数据库用户的数据视图，是与某一应用有关的数据的逻辑表示。一个数据库可以有多个外模式。数据库管理系统提供外模式数据定义语言（外模式 DDL）来严格定义外模式。

（2）模式

也称逻辑模式，是数据库的全局逻辑结构和组织方式的描述。模式定义了数据库中所有

数据的完整视图，包括实体、关系、属性以及它们之间的联系。模式层面上描述了数据库的总体逻辑结构，是所有用户和应用程序共享的数据库抽象模型。一个数据库只有一种模式。

（3）内部模式

也称为存储模式或物理模式，是数据库在物理存储介质上的实际存储方式和组织结构的描述。内部模式定义了数据在磁盘或其他存储介质上的存储格式、索引方式、数据分布方式等信息。内部模式侧重于数据库的物理实现细节，供数据库管理系统使用。

192. 简述数据库系统的特点。★★

（1）数据结构化；

（2）数据的共享性高，冗余度低，易扩充；

（3）数据独立性高；

（4）数据由 DBM 统一管理和控制。

193. 数据库管理系统的主要功能有哪些？★★

（1）数据的安全性保护

（2）数据的完整性保护

（3）并发控制

（4）数据库恢复

194. 什么叫数据与程序的物理独立性？什么叫数据与程序的逻辑独立性？为什么数据库系统具有数据与程序的独立性？★★★

物理独立性：是指用户的应用程序与存储在磁盘上的数据库中数据是相互独立的。

逻辑独立性：是指用户的应用程序与数据的逻辑结构是相互独立的，也就是说，数据的逻辑结构改变了，用户程序也可以不变。

为什么：数据独立性是由 DBMS 的二级映像功能来保证的（外模式/模式映像，模式/内模式映像），这两层映像机制保证了数据库系统中数据的逻辑独立性和物理独立性。

195. 笛卡尔积和自然连接的区别。★★★

笛卡尔积

笛卡尔积是一个基本的集合操作，它将两个或多个表中的所有行组合在一起，生成一个包含所有可能组合的结果集。在笛卡尔积中，每个表的每一行都与其他表的每一行进行组合，并生成新的行。结果集的行数等于每个表的行数乘积。

自然连接

自然连接是基于两个或多个表中相同列名的列进行连接操作。它将这些相同列名的列作为连接条件，自动找到两个表中相应列值相等的行，并将它们组合在一起形成结果集。自然连接省略了连接条件的指定，只要对应列名相同，就会自动匹配。自然连接可以更方便地进行表之间的关联操作，无需显式指定连接条件，但需要特别注意列名的一致性。

196. 什么是视图？视图有什么优点？视图什么时候是不可更新的？★★★

视图

是从一个或多个基本表（也可能包括其他视图）中检索出的虚拟表，它本身不存储数据。视图可以看作是基于基本表的查询结果集的临时表。通过定义视图，我们可以以更方便、更简洁的方式获取特定的数据子集，而无需每次都编写复杂的查询语句。视图可以进行查询操作，但不能直接进行数据修改。视图的定义存储在数据库中。

优点：

（1）视图能够简化用户的操作

（2）视图使用户能以多种角度看待同一数据

（3）视图对重构数据库提供了一定程度的逻辑独立性

（4）视图能够对机密数据提供安全保护

（5）适当的利用视图可以更清晰的表达查询

不可更新的视图：

（1）若视图是由两个以上基表导出的，此视图不允许更新

（2）若视图的字段来自字段表达式或常数，则不允许对此视图执行 INSERT 和 UPDATES 作，但允许 DELETE 操作

（3）若视图的字段来自聚集函数，则此视图不允许更新

（4）若视图定义中含有 GROUP BY 子句，则此视图不允许更新

（5）若视图定义中含有 DISTINCT 短语，则此视图不允许更新

（6）若视图定义中有嵌套查询，并且内层查询的 FROM 子句中涉及的表也是导出该视图的基本表，则此视图不允许更新

（7）一个不允许更新的视图上定义的视图也不允许更新

197. 关系数据库的完整性规则有哪几类？★★★

（1）实体完整性

实体完整性要求每个表中的主键必须具有唯一性和非空性。也就是说，表中的每个实体必须能够被唯一地标识，并且主键列不能包含空值。

（2）参照完整性

参照完整性是指在两个表之间建立外键关系时，要求引用表中的外键必须与被引用表中的主键值相匹配。也就是说，如果一个表中的外键引用了另一个表中的某个主键值，那么被引用的主键值必须存在于被引用表中。

（3）用户定义的完整性

用户定义的完整性规则是根据具体业务需求而定义的规则。这些规则可通过触发器、自定义约束或存储过程等方式实现。例如，限制某个字段的取值范围、检查复杂的业务逻辑规则等。

198. 什么是触发器？触发器什么时候会触发？触发器的作用是什么？★★★★

触发器：

触发器是与表有关的数据库对象，在满足预定义的条件时会被触发，从而执行触发器中定义的语句集合。

触发条件：

在数据库进行数据变更的时候，触发器可以被触发。即对数据表进行插入、删除、修改数据的时候，可以执行触发器。触发器定义在表上，也依附于表存在。触发器可以在数据表变更进行前执行，也可能在变更进行后执行。触发器对于每一行都会被执行。

触发器作用：

（1）强制数据完整性：触发器可以用于强制实施数据完整性规则。例如，当插入或修改表中的数据时，触发器可以检查数据的有效性并拒绝不符合规定的操作。

（2）数据验证和转换：触发器可以对即将插入、更新或删除的数据进行验证和转换。例如，在插入数据之前，触发器可以验证是否满足某些条件，或者在插入数据之后，触发器可以对数据进行后处理操作。

（3）日志记录和审计：触发器可以用于记录数据操作的日志或进行审计跟踪。例如，在数据更新之后，触发器可以将旧值和新值记录到审计表中，以便后续分析和审计目的。

（4）自动化业务逻辑：触发器可以用于执行复杂的业务逻辑。例如，在某个表上进行插入操作时，触发器可以自动在其他相关表上进行相应的更新操作，以保持数据的一致性。

199. 简述一下第一范式、第二范式、第三范式、BC 范式、第四范式。★★★★★

范式是数据库设计中用于消除数据冗余和提高数据存储效率的一种规范化方法。

（1）第一范式（1NF）

属性是不可分割的最小单元，即不会有重复的列，体现原子性。

（2）第二范式（2NF）

满足 1NF 前提下，存在一个候选码，非主属性全部依赖该候选码，即存在主键，体现唯一性。总之，就是消除部分函数依赖。（ps：完全函数依赖指的是缺一不可，部分函数依赖可以去掉左边的某个属性仍然可以推出右边的属性）

（3）第三范式（3NF）

满足 2NF 前提下，非主属性必须互不依赖，消除传递依赖。

（4）BC 范式

在满足第三范式的基础上，确保表中的每个非主属性都完全依赖于每个候选码。总之，就是每一个决定属性因素都包含码。

（5）第四范式（4NF）

在满足 BC 范式的基础上，通过进一步拆分表来处理多值依赖问题。多值依赖指的是当一个非主键字段的值依赖于其他非主键字段的多个值时，应将其拆分成独立的表。

200. 什么是事务？事务的 ACID 特性是什么？如何保证 ACID 特性？★★★★★★

事务是用户定义的一个数据库操作序列，这些操作要么全做要么全不做，是一个不可分割的工作单位。

事务的 ACID 性质

（1）原子性（Atomicity）：事务是一个原子操作单元，要么全部执行成功，要么全部回滚到事务开始之前的状态。如果在事务执行过程中发生错误，所有已执行的操作都会被撤销，数据库会恢复到事务开始之前的状态。

（2）一致性（Consistency）：事务在执行前后，数据库必须满足所有预定义的一致性规则。这意味着事务在执行过程中，不能违反数据库的完整性约束和业务规则。

（3）隔离性（Isolation）：事务的隔离性规定多个事务并发执行时，每个事务都必须与其他事务相互隔离，互不干扰。这样可以确保并发执行的事务不会相互影响，避免了数据的不一致和并发访问引起的问题。

（4）持久性（Durability）：一旦事务被提交，其所做的修改将永久保存到数据库中，并且对于后续的数据库操作和系统故障都是持久的。即使在系统崩溃或断电后，数据库也应该能够恢复到事务提交的状态。

如何保证 ACID 性质

（1）通过实现事务管理器来处理事务的执行、提交和回滚，以保证原子性和一致性。

（2）使用锁机制和并发控制算法来控制事务之间的相互干扰，以确保隔离性。

（3）将事务操作记录在事务日志中，并使用恢复和重做机制来保证持久性。

（4）设计合理的数据库模式和应用程序逻辑来确保一致性。

201. 什么是封锁？基本的封锁类型有几种？★★★

在数据库中，锁是用于控制对数据库对象（如表、行、页等）进行并发访问的机制。通过使用锁，可以确保数据库操作的正确性和一致性。

类型：

共享锁：也称为读锁，多个事务可以同时持有共享锁，并且可以并发地读取被锁定对象的数据，但不能修改或写入数据。共享锁之间不互斥，不会阻塞其他共享锁的获取。

排他锁：也称为写锁，只能由一个事务持有，它阻塞其他事务对被锁定对象进行读取或修改操作。排他锁可以保证事务对被锁定对象的独占性访问，以便进行数据的修改。

202. 在数据库中为什么要并发控制？并发控制技术可以保证事务的哪些特征？★★★

数据库是共享资源，通常有许多个事务同时在运行。当多个事务并发地存取数据库时就会产生同时读取或修改同一数据的情况。 若对并发操作不加控制就可能会存取和存储不正

确的数据，破坏数据库的一致性。

并发控制技术能够保证事务的 ACID 特性。

203. 简述并发操作带来的三类数据不一致性。★★★★

（1）丢失修改

当多个事务同时对同一个数据项进行修改时，由于并发执行的无序性，可能导致某些事务的更新被**覆盖或丢失**。例如，事务 A 和事务 B 同时读取数据 X，然后分别进行修改，并提交更新。如果事务 B 的修改操作覆盖了事务 A 的修改结果，那么事务 A 的更新就会丢失。

（2）不可重复读

当一个事务在相同的查询中多次读取同一行数据，但在此期间，其他事务对该行进行了修改或删除，导致每次读取的结果不一致。例如，事务 A 读取数据 X 的值，然后事务 B 修改了数据 X 并提交，接着事务 A 再次读取数据 X，得到的结果与之前不一样。

（3）读脏数据

当一个事务读取到另一个未提交事务的数据时，称为读脏数据。例如，事务 A 修改了数据 X，但还没有提交，此时事务 B 读取了数据 X，得到的是事务 A 未提交的修改结果。如果事务 A 最终回滚（即不提交），那么事务 B 读取到的数据就是无效的，造成了数据不一致的情况。

204. 简述两段封锁协议的内容。★★★

指所有事务必须分两个阶段对数据项加锁和解锁。

（1）在对任何数据进行读、写操作之前，事务首先要获得对该数据的封锁

（2）在释放一个封锁之后，事务不再申请和获得任何其他封锁

205. 什么叫做数据库的恢复？数据库恢复的基本技术有哪些？ ★★

数据库恢复是指通过技术手段，将保存在数据库中上丢失的电子数据进行抢救和恢复的技术。

技术：

（1）数据转储

转储是指 DBA 将整个数据库复制到磁带或另一个磁盘上保存起来的过程，备用的数据称为后备副本或后援副本

（2）登录日志文件

206. 简述数据库系统中可能发生的故障类型，以及数据恢复方法。★★★

（1）事务内部的故障

指的是单个事务执行过程中发生的错误或异常。例如，事务逻辑错误、约束冲突、死锁等。

恢复方法：由恢复子系统应利用日志文件撤消（UNDO）此事务已对数据库进行的修改。

（2）系统故障

称为软故障，是指造成系统停止运转的任何事件，使得系统要重新启动。

①发生系统故障时，事务未提交

恢复方法：强行撤消（UNDO）所有未完成事务

②发生系统故障时，事务已提交，但缓冲区中的信息尚未完全写回到磁盘上。

恢复方法：重做（REDO）所有已提交的事务

（3）介质故障

称为硬故障，指的是存储介质（如硬盘）出现故障导致数据丢失或损坏。

恢复方法：装入数据库发生介质故障前某个时刻的数据副本；重做自此时始的所有成功事务，将这些事务已提交的结果重新记入数据库

（4）计算机病毒

指的是恶意软件对数据库系统的破坏。

恢复方法：包括使用杀毒软件清除病毒，并通过备份恢复受感染的数据。

207. 什么是索引？请简述索引的几种类型或分类。★★★★

索引

索引是对数据库表中一列或多列的值进行排序的数据结构，用于快速访问数据库表中的特定信息。

分类

（1）从物理结构上可以分为聚簇索引和非聚集索引两类

聚簇索引指索引的键值的逻辑顺序与表中相应行的物理顺序一致，即每张表只能有一个聚簇索引，也就是我们常说的主键索引；

非聚簇索引的逻辑顺序则与数据行的物理顺序不一致。

（2）从应用上可以划分为一下几类：

普通索引：MySQL 中的基本索引类型，没有什么限制，允许在定义索引的列中插入重复值和空值，纯粹为了提高查询效率。

唯一索引：索引列中的值必须是唯一的，但是允许为空值。

主键索引：特殊的唯一索引，也成聚簇索引，不允许有空值，并由数据库帮我们自动创建。

组合索引：组合表中多个字段创建的索引，遵守最左前缀匹配规则。

全文索引：用于支持全文搜索的一种技术。它允许对文本内容进行快速、高效的搜索，并返回与搜索条件匹配的结果。

208. 索引常用的两种数据结构是什么？有什么区别？★★★★

MySQL 中常用的是 Hash 和 B+树索引。

区别：

（1）实现方式

Hash 索引底层就是 Hash 表，进行查询时调用 Hash 函数获取到相应的键值（对应地址），然后回表查询获得实际数据。

B+树索引底层实现原理是多路平衡查找树，对于每一次的查询都是从根节点出发，查询到叶子节点方可以获得所查键值，最后查询判断是否需要回表查询。

（2）使用场景

Hash 索引适合于等值查询，即根据索引列的具体值进行查询。对于范围查询或排序操作，Hash 索引的效果不好。

B+树索引适用于范围查询、排序和模糊匹配等操作。B+树索引的有序性使得范围查询更加高效，而且能够支持一些特殊的查询需求，如前缀匹配等。

（3）性能特点

Hash 索引的读取性能通常比 B+树索引高，因为通过散列值可以直接定位到存储索引项的位置。但如果存在哈希冲突，可能需要进行链式查找，会影响性能。

B+树索引的读取性能在范围查询和排序操作上通常更好。B+树索引具有较高的数据聚集性，利于顺序访问，且支持部分加载等优化策略。

Hash 索引对于插入和更新操作性能较好，因为哈希表的插入和查找操作复杂度为 O(1)。而 B+树索引由于需要维护有序性，插入和更新操作相对较慢。

209. 为什么 B+树比 B 树更适合应用于数据库索引？★★★★

（1）B+树减少了 IO 次数

由于索引文件很大因此索引文件存储在磁盘上，B+树的非叶子结点只存关键字不存数据，因而单个页可以存储更多的关键字，即一次性读入内存的需要查找的关键字也就越多，

磁盘的随机 I/O 读取次数相对就减少了。

（2）B+树查询效率更稳定

由于数据只存在在叶子结点上，所以查找效率固定为 O(log n)，所以 B+树的查询效率相比 B 树更加稳定。

（3）B+树更加适合范围查找

B+树叶子结点之间用链表有序连接，所以扫描全部数据只需扫描一遍叶子结点，利于扫库和范围查询；B 树由于非叶子结点也存数据，所以只能通过中序遍历按序来扫。也就是说，对于范围查询和有序遍历而言，B+树的效率更高。

十三、编译原理★

210. 什么是编译？★

编译是将高级程序语言（源代码）转换为计算机可执行的低级机器语言（目标代码）的过程。编译器是执行这一过程的软件工具。

编译过程一般包括以下几个步骤：

（1）词法分析：将源代码分解为一个个词法单元（Token），如关键字、标识符、运算符和常量等。词法分析器根据事先定义好的词法规则进行匹配和划分。

（2）语法分析：根据语法规则，将词法单元构建成语法树（Parse Tree）或抽象语法树（Abstract Syntax Tree）。语法分析器检查语法的正确性，若发现错误则报告语法错误。

（3）语义分析：对语法树进行遍历和分析，进行类型检查、语义检查以及生成中间表示等。语义分析器会检查变量是否被声明、函数调用是否正确以及其他语义规则是否满足。

（4）中间代码生成：将语法树或抽象语法树转换为一种中间表示形式，如三地址码（Three-Address Code）、虚拟机代码（Virtual Machine Code）等。中间代码更加接近于机器语言，但仍是与具体机器无关的形式。

（5）代码优化：对中间代码进行优化，以提高程序的执行效率、减少资源消耗或改进代码质量。优化可能包括删除冗余代码、重新排序指令、利用寄存器等。

（6）目标代码生成：将优化后的中间代码转换为目标机器语言，即计算机能够直接执行的机器指令。这些指令可以是特定 CPU 架构的汇编语言或者是二进制表示的机器码。

211. 编译和解释有什么区别？★

编译和解释是两种不同的程序执行方式。

（1）编译

是将整个源代码文件一次性转换为目标机器语言的过程。编译器会对源代码进行词法分析、语法分析、语义分析、中间代码生成、代码优化和目标代码生成等步骤，生成一个可执行文件。

编译的优点：

执行速度快：编译器将整个源代码转换为机器码，执行效率更高。

独立性强：生成的目标代码与具体机器无关，可以在不同平台上执行。

一次编译，多次执行：编译后的可执行文件可以重复使用，无需再次进行翻译。

编译的缺点：

开发周期长：需要在编译阶段进行多个步骤的处理，产生可执行文件前的过程较为繁琐。

调试困难：由于编译生成的是目标代码，调试时难以直接定位到源代码的具体位置。

（2）解释

是逐行解释执行源代码的过程。解释器会逐行读入源代码，并逐行解释执行，将每一行翻译成目标机器语言并立即执行。解释器不会事先将整个源代码文件转换成目标代码，而是

边解释边执行。每次运行程序时，都需要重新解释源代码。

解释的优点：

快速开发和测试：解释器无需额外的编译过程，可以直接执行源代码，加快了开发和调试的速度。

跨平台性：解释器本身可以在不同平台上运行，解释执行源代码时也可以保持跨平台的特性。

解释的缺点：

执行速度相对较慢：解释器需要逐行解释执行源代码，每次都需要进行翻译，执行效率相对较低。

依赖解释器环境：每次运行程序都需要有对应的解释器环境。

综上所述，编译适用于需要长时间运行、对执行效率要求较高的场景，而解释适用于快速开发、跨平台性要求较高的场景。实际选择编译还是解释取决于具体的应用需求和执行环境。

212. 自上而下分析法和自下而上分析的思想是什么？★

（1）自上而下分析

自上而下分析的思想是从文法的起始符号开始，逐步向下展开产生式，直到推导出输入串。它的基本过程是匹配输入串中的符号，并根据产生式选择合适的扩展方式，最终得到输入串。自上而下分析是一种预测性的分析方法，通过预测下一个可能的语法结构来构建语法树。常见的自上而下分析方法有递归下降分析和 LL(k)分析。

（2）自下而上分析

自下而上分析的思想是从输入串的底部开始，逐步将终结符号转化为非终结符号，直到达到文法的起始符号。它的基本过程是将输入串进行规约操作，即根据产生式将较长的符号串替换为更短的符号串，最终得到文法的起始符号。自下而上分析的重点在于识别并构造出输入串中的局部语法结构，然后将其合并成更大的语法结构，直到得到整个语法树。常见的自下而上分析方法有 LR(k)和 LALR(k)分析。

213. NFA 和 DFA 的区别。★

NFA：在当前状态和输入符号的组合下，可以有多个可能的下一个状态。换句话说，NFA 允许非确定性转移，即可以通过空转移或者多个转移路径到达下一个状态。

DFA：在当前状态和输入符号的组合下，只能存在唯一的下一个状态。DFA 不允许非确定性转移，每个输入都有且只有一个对应的转移路径。

214. 简述文法的分类。★★

文法是乔姆斯基（Chomsky）提出的一种文法分类体系。

0 型文法（无限制文法）：

没有限制产生式规则的形式，可以包括任意的字符串。

1 型文法（上下文有关文法）：

1 型文法限制了产生式规则的形式，要求产生式的左部长度不小于右部长度。也就是说，产生式的左部必须至少包含一个非终结符号，并且非终结符号的数量不能超过右部非终结符号的数量。

2 型文法（上下文无关文法）：

产生式的左部只能是一个非终结符号，而右部可以是任意字符串。

3 型文法（正规文法）：

3 型文法是最受限制的文法类型。它的产生式规则形式非常简单，要求每个产生式的右部只能有以下三种情况：

(1) A -> ε（A 可以直接推导出空串）；

(2) A -> a（A 可以直接推导出一个终结符号 a）；

(3) A -> aB（A 可以推导出一个终结符号 a 和一个非终结符号 B）。

3 型文法对应着正则语言，可以通过有限状态自动机进行识别和处理。

十四、软件工程★

215. 简述软件的生命周期。★★

软件生命周期又称为软件生存周期或系统开发生命周期，是软件的产生直到报废的生命周期。

软件生存周期包括：

（1）问题定义：弄清"用户需要计算机解决什么样的问题”，提出"系统目标和范围的说明“，提交用户审查和确认。

（2）可行性研究：可行性研究的目的就是用最小的代价在尽可能短的时间内确定问题是否能够解决。并从经济、技术、法律等多个方面进行可行性分析。

（3）需求分析：弄清用户对软件系统的全部需求，编写需求规格说明书和初步的用户手册，提交评审。

（4）开发阶段：总体设计、详细设计、编码、测试

（5）维护：改正性维护（由于开发测试的不彻底、不完全），适应性维护（适应环境变化），完善性维护（使用过程中提出的一些建设性意见），预防性维护（改善软件系统的可维护性和可靠性）。

216. 什么是软件危机？产生原因是什么？软件危机有哪些表现？★★★

（1）软件危机

软件危机是指在计算机开发和维护过程中所遇到的一系列严重的问题。

主要包含以下两个问题：如何开发软件，以满足对软件日益增长的需求；如何维护数量不断膨胀的已有软件

（2）产生原因

一方面与软件本身的特点有关，另一方面也与软件开发与维护的方法不正确。

①软件规模越来越大，结构越来越复杂

②软件开发管理困难而复杂

③软件包开发费用不断增加

④软件开发技术落后

⑤生产方式落后，仍采用手工方式

⑥开发工具落后，生产率提高缓慢

（3）典型表现

①对软件开发成本和进度的估计常常不准确

②用户对已完成的软件系统不满意的现象经常发生

③软件产品的质量往往靠不住

④软件常常是不可维护的

⑤软件通常没有适当的文档资料

⑥软件成本在计算机系统总成本中所占的比例逐年上升

⑦软件开发生产的速率，跟不上计算机应用迅速普及深入的趋势

217. 软件开发中有哪几种过程模型？★★★

软件开发中常见的软件过程模型有瀑布模型、原型模型、螺旋模型、喷泉模型、统一软件过程、智能模型、演化模型等。

其中喷泉模型、统一软件过程适用于面向对象的软件开发。

218. 什么是瀑布模型？瀑布模型有什么特点？瀑布模型的优点和缺点是什么？★★★★

（1）瀑布模型概念

瀑布模型是软件开发中最早也是最经典的一种开发模型，它是一种线性顺序的软件开发过程模型。瀑布模型将软件开发过程划分为一系列严格的阶段，并且每个阶段的输出作为下个阶段的输入，按照顺序逐步推进。

（2）瀑布模型的特点

线性顺序：每个阶段都是按照固定的顺序进行，前一阶段的输出作为后一阶段的输入。

阶段划分明确：每个阶段有明确定义的输入、输出和可交付物。

文档驱动：瀑布模型非常注重文档的书写和管理，以记录需求、设计和测试等信息。

可迭代性较低：在该模型中，通常只能在前一个阶段完成后才能进入下一个阶段。

（3）瀑布模型的优点和缺点

优点：结构清晰，易于管理和控制项目进度

缺点：对需求变化的适应性较差，一旦某个阶段出现问题，可能需要重新回到上一个阶段进行修改，增加了开发成本和时间。

219. 什么是敏捷开发过程与极限编程？★★

敏捷开发

为了使软件开发团队具有高效工作和快速响应的能力，软件专家起草了敏捷软件开发宣言，由下述 4 个简单的价值观声明组成：

（1）个体和交互胜过过程和工具

（2）可以工作的软件胜过面面俱到的文档

（3）客户合作胜过合同谈判

（4）响应变化胜过遵循计划

根据上述价值观提出的软件过程称为敏捷开发过程。在敏捷开发中，软件项目在构建初期被切分成多个子项目，各个子项目的成果都经过测试，都具备集成和可运行的特征。

极限编程

极限编程是敏捷开发过程中最富盛名的一个，“极限”是指把好的开发实践运用到极致。极限编程广泛应用于需求模糊且经常改变的场合。

220. 模块的独立程度如何度量？★★

模块的独立程度可以通过两个标准度量，它们分别是耦合度和内聚度。

耦合度：耦合度衡量了模块之间的相互依赖程度。耦合度越低，表示模块之间的联系越少，相互独立性越高。

内聚度：内聚度评估了一个模块内部各个元素之间联系的紧密程度。内聚度越高，则表示模块内的元素彼此关联紧密，模块功能单一且清晰。

通过衡量耦合度和内聚度可以评估模块的独立程度。理想情况下，我们希望高内聚和低耦合。这样可以提高模块的独立性，使得系统更容易理解、维护和扩展。

221. 什么是黑盒测试和白盒测试，黑盒测试和白盒测试有哪些常用方法？★★★★

黑盒测试和白盒测试是软件测试中常用的两种测试方法。

（1）黑盒测试法

黑盒测试是一种基于软件外部行为和功能的测试方法，测试者不考虑软件内部结构和实现细节，仅通过输入和输出来验证软件的功能是否符合预期。

黑盒测试又称为功能测试，包含等价类划分法、边界值法、错误推测法、因果图法等。

（2）白盒测试法

白盒测试是一种基于软件内部结构和实现细节的测试方法，测试者具有对被测试软件的

代码和逻辑的详细了解，针对代码覆盖率和程序路径进行测试。

常见的白盒测试方法包括语句覆盖、分支覆盖、路径覆盖、条件覆盖、循环覆盖等。

（3）黑盒测试和白盒测试联系

黑盒测试和白盒测试并不是相互排斥的，实际测试中常常结合使用。综合使用这两种测试方法可以提高测试的全面性和有效性，帮助发现更多的错误和潜在问题。

222. UML 是什么？有什么特点？★★★

UML，是统一建模语言的缩写，也就是说，UML 是一种语言，是用来对面向对象软件编程进行建模所使用的。另外，对关系数据库建模时也可以用 UML。

特点：

统一了面向对象方法的有关概念和描述方法。

表达能力强，能对各种并发分布式系统进行建模，且 UML 还提供了扩展机制。

UML 专注于一种标准的建模语言，而不是一个开发过程。

223. UML 图里面有哪些图？★★

静态建模机制

用例图：描述系统功能。

类图：描述所定义的类。

对象图：描述各个类在某一时间点上的实例。

动态建模机制

活动图：描述一个操作的执行过程中所完成的工作。

状态图：描述一个特定对象的所有可能的状态及其引起状态改变的事件。

顺序图：模拟对象之间的交互。

协作图：描述对象协作关系。

UML 物理框架机制

构件图：描述各种软件构件之间的依赖关系。

部署图：描述了一个系统运行时的硬件结点。

224. 谈一谈模块化有什么好处？★★

软件工程中的模块化是一种将软件系统划分为独立、可组合且相互依赖的模块的方法。它是一种软件设计和组织的原则，旨在提高软件的可维护性、可重用性和可扩展性。

优点：

分而治之：模块化将复杂的软件系统拆分为多个相对独立的模块，每个模块负责特定的功能或任务。这种拆分可以降低开发难度，使得开发人员能够将注意力集中在一个小而可管理的范围内进行设计、实现和测试。

接口定义：模块化要求模块之间通过明确定义的接口进行通信和交互。接口定义了每个模块对外暴露的功能和规约，使得模块之间能够独立开发、测试和替换。良好定义的接口可以提高模块的可组合性和可替换性。

高内聚低耦合：模块化的设计追求高内聚性和低耦合性。高内聚性意味着模块内部的元素彼此关联紧密，功能相近，达到单一职责的设计原则。低耦合性表示模块之间的依赖关系尽量减少，模块之间的修改不会影响到其他模块。

可重用性：模块化的设计使得模块可以独立于系统进行开发和测试，并且可重复使用于不同的应用场景。通过模块的复用，可以提高软件开发效率、降低维护成本，并且改进软件系统的质量。

易于维护：模块化使得软件系统的维护更加容易。当需要修改或扩展一个功能时，只需关注与之相关的模块，而无需了解整个系统的内部实现细节。这减少了错误的引入和维护的复杂性。