山东大学软件学院

2021级本科生学业朋辈导师辅导答疑会（第六期）

2023年5月31日

Test 6 (Week 16)

Operating System

1. [I/O管理]I/O方式中，简述轮询、中断、DMA方式的机制和各自优缺点，并给出一个外部空闲空间分配和回收的方案。（10分）[2022]
    - 轮询：计算机从外部设备读取的每个字，CPU需要对外设状态进行循环检查，直到确定该字已经在I/O控制器的数据寄存器中。在程序直接控制方式中，由于CPU的高速性和I/O设备的低速性，致使CPU的绝大部分时间都处于等待I/O设备完成数据I/O的循环测试中，造成了CPU资源的极大浪费。在该方式中，CPU之所以要不断地测试I/O设备的状态，就是因为在CPU中未采用中断机构，使I/O设备无法向CPU报告它已完成了一个字符的输入操作。程序直接控制方式虽然简单且易于实现，但其缺点也显而易见，由于CPU和I/O设备只能串行工作，导致CPU的利用率相当低。
    - 中断：中断驱动方式的思想是，允许I/O设备主动打断CPU的运行并请求服务，从而“解放”CPU,使得其向I/O控制器发送读命令后可以继续做其他有用的工作。我们从I/O控制器和CPU两个角度分别来看中断驱动方式的工作过程。从I/O控制器的角度来看，I/O控制器从CPU接收一个读命令，然后从外部设备读数据。一旦数据读入I/O控制器的数据寄存器，便通过控制线给CPU发出中断信号，表示数据已准备好，然后等待CPU请求该数据。I/O控制器收到CPU发出的取数据请求后，将数据放到数据总线上，传到CPU的寄存器中。至此，本次I/O操作完成，I/O控制器又可开始下一次I/O操作。从CPU的角度来看，CPU发出读命令，然后保存当前运行程序的上下文（现场，包括程序计数器及处理机寄存器），转去执行其他程序。在每个指令周期的末尾，CPU检查中断。当有来自I/O控制器的中断时，CPU保存当前正在运行程序的上下文，转去执行中断处理程序以处理该中断。这时，CPU从I/O控制器读一个字的数据传送到寄存器，并存入主存。接着，CPU恢复发出I/O命令的程序（或其他程序）的上下文，然后继续运行。中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在存储器与I/O控制器之间的传输都必须经过CPU,这就导致了中断驱动方式仍然会消耗较多的CPU时间。
    - DMA：在中断驱动方式中，I/O设备与内存之间的数据交换必须要经过CPU中的寄存器，所以速度还是受限，而DMA（直接存储器存取）方式的基本思想是在I/O设备和内存之间开辟直接的数据交换通路，彻底“解放”CPU。DMA方式与中断方式的主要区别是，中断方式在每个数据需要传输时中断CPU,而DMA方式则是在所要求传送的一批数据全部传送结束时才中断CPU;此外，中断方式的数据传送是在中断处理时由CPU控制完成的，而DMA方式则是在DMA控制器的控制下完成的。
    - 分配和回收的方案见题目3.

2. [I/O管理]非阻塞和阻塞I/O是什么，主要有什么不同，分别用在哪里.

   操作系统的I/O接口还涉及两种模式：阻塞和非阻塞。

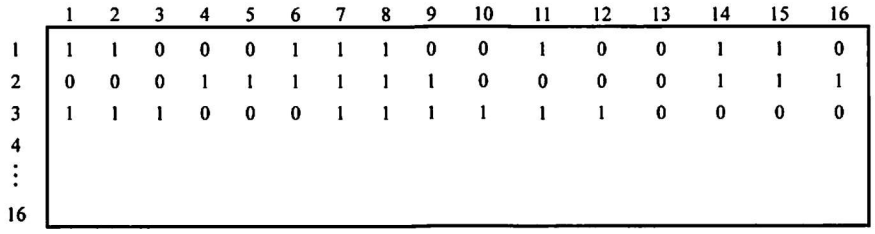
   阻塞I/O是指当用户进程调用I/O操作时，进程就被阻塞，需要等待I/O操作完成，进程才被唤醒继续执行。

   非阻塞I/O是指用户进程调用I/O操作时，不阻塞该进程，该I/O调用返回一个错误返回值，通常，进程需要通过轮询的方式来查询I/O操作是否完成。比如：打印机的输入和输出。

   大多数操作系统提供的I/O接口都是采用阻塞I/O。等待I/O操作完成，将数据读入内存等等

3. [文件管理]以磁盘外存空间为例，设计高效的空闲块分配、回收算法，给出设计思想，操作方法，数据结构。（10分）

- 位图
  - 位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，磁盘上所有的盘块都有一个二进制位与之对应。当其值为“0”时，表示对应的盘块空闲：为“1”时，表示已分配。这样，一个m×n位组成的位示图就可用来表示m×n个盘块的使用情况：



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

  - 盘块的分配：
    1)顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位。
    2)将找到的一个或一组二进制位，转换成与之对应的盘块号。若找到的其值为“0”的二进制位位于位示图的第i行、第j列，则其相应的盘块号应按下式计算(n为每行位数)：
    $b = n(i-1) + j.$
  - 盘块的回收：
    1)将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：
    $i = (b-1)DIVn + 1$
    $j = (b-1)MODn + 1$
    2)修改位示图，令$map[i,j] = 0.$
    空闲表法和空闲链表法都不适用于大型文件系统，因为这会使空闲表或空闲链表太大。
- 链表
  - 将所有空闲盘区拉成一条空闲链。根据构成链所用基本元素的不同，分为两种形式：
  - 1)空闲盘块链。将磁盘上的所有空闲空间以盘块为单位拉成一条链。当用户因创建文件而请求分配存储空间时，系统从链首开始，依次摘下适当数目的空闲盘块分配给用户。当用户因删除文件而释放存储空间时，系统将回收的盘块依次插入空闲盘块链的末尾。这种方法的优点是分配和回收一个盘块的过程非常简单，但在为一个文件分配盘块时可能要重复操作多次，效率较低。又因它是以盘块为单位的，空闲盘块链会很长。
  - 2)空闲盘区链。将磁盘上的所有空闲盘区（每个盘区可包含若干个盘块）拉成一条链。每个盘区除含有用于指示下一个空闲盘区的指针外，还应有能指明本盘区大小（盘块数）的信息。分配盘区的方法与内存的动态分区分配类似，通常采用首次适应算法。在回收盘区时，同样也要将回收区与相邻接的空闲盘区合并。这种方法的优缺点刚好与第一种方法的相反，即分配与回收的过程比较复杂，但效率通常较高，且空闲盘区链较短。
- 块组链表

  对空闲链表的一个改进是将n个空闲块的地址存在第一个空闲块中。这些块中的前n-1个确实为空，而最后一块包含另外n个空闲块的地址，如此继续。大量空闲块的地址可以很快地找到，这一点有别于标准链表方法。
- 第一空闲块+计数

  另外一种方法是利用这样一个事实：通常，有多个连续块需要同时分配或释放，尤其是在使用连续分配和采用簇时更是如此。因此，不是记录个空闲块的地址，而是可以记录第一块的地址和紧跟第一块的连续的空闲块的数量。这样，空闲空间表的每个条目包括磁盘地址和数量。虽然每个条目会比原来需要更多空间，但是表的总长度会更短，这是因为连续块的数量常常大于1。

4. [进程管理]在一个仓库中可以存放A和B两种产品，要求:
   ①每次只能存入一种产品。
   ②A产品数量-B产品数量 <M。
   ③B产品数量-A产品数量<N。
   其中，M,N是正整数，试用 P操作、V操作描述产品 A 与产品 B 的入库过程。

   [解答] 使用信号量mutex控制两个进程互斥访问临界资源（仓库），使用同步信号量Sa和Sb(分别代表产品A与B的还可容纳的数量差，以及产品B与A的还可容纳的数量差）满足条件2和条件3。代码如下:

```
Semaphore Sa=M-1,Sb=N-1;
Semaphore mutex=1;
//访问仓库的互斥信号量
process A(){                                          process B(){
    while(1){                                             while(1){
            P(Sa);                                            P(Sb);
            P(mutex);                                         P(mutex);
            A产品入库;                                         B产品入库;
            V(mutex);                                         V(mutex);
            V(Sb);                                            V(Sa);
        }                                                     }
    }                                                     }
```

输入/输出管理

① I/O的硬件（略）.

② I/O的方式. 轮询. 中断和DMA方式.（计组）.

③ I/O应用接口：阻塞（同步）和非阻塞（异步）.

Test 6 (Week 16)

Operating System

1. [I/O管理]I/O方式中，简述轮询、中断、DMA方式的机制和各自优缺点，并给出一个外部空闲空间分配和回收的方案。（10分）[2022]
   - 轮询：计算机从外部设备读取的每个字，CPU需要对外设状态进行循环检查，直到确定该字已经在I/O控制器的数据寄存器中。在程序直接控制方式中，由于CPU的高速性和I/O设备的低速性，致使CPU的绝大部分时间都处于等待I/O设备完成数据I/O的循环测试中，造成了CPU资源的极大浪费。在该方式中，CPU之所以要不断地测试I/O设备的状态，就是因为在CPU中未采用中断机构，使I/O设备无法向CPU报告它已完成了一个字符的输入操作。程序直接控制方式虽然简单且易于实现，但其缺点也显而易见，由于CPU和I/O设备只能串行工作，导致CPU的利用率相当低。
   - 中断：中断驱动方式的思想是，允许I/O设备主动打断CPU的运行并请求服务，从而“解放”CPU,使得其向I/O控制器发送读命令后可以继续做其他有用的工作。我们从I/O控制器和CPU两个角度分别来看中断驱动方式的工作过程。从I/O控制器的角度来看，I/O控制器从CPU接收一个读命令，然后从外部设备读数据。一旦数据读入I/O控制器的数据寄存器，便通过控制线给CPU发出中断信号，表示数据已准备好，然后等待CPU请求该数据。I/O控制器收到CPU发出的取数据请求后，将数据放到数据总线上，传到CPU的寄存器中。至此，本次I/O操作完成，I/O控制器又可开始下一次I/O操作。从CPU的角度来看，CPU发出读命令，然后保存当前运行程序的上下文（现场，包括程序计数器及处理机寄存器），转去执行其他程序。在每个指令周期的末尾，CPU检查中断。当有来自I/O控制器的中断时，CPU保存当前正在运行程序的上下文，转去执行中断处理程序以处理该中断。这时，CPU从I/O控制器读一个字的数据传送到寄存器，并存入主存。接着，CPU恢复发出I/O命令的程序（或其他程序）的上下文，然后继续运行。中断驱动方式比程序直接控制方式有效，但由于数据中的每个字在存储器与I/O控制器之间的传输都必须经过CPU,这就导致了中断驱动方式仍然会消耗较多的CPU时间。
   - DMA：在中断驱动方式中，I/O设备与内存之间的数据交换必须要经过CPU中的寄存器，所以速度还是受限，而DMA（直接存储器存取）方式的基本思想是在I/O设备和内存之间开辟直接的数据交换通路，彻底“解放”CPU。DMA方式与中断方式的主要区别是，中断方式在每个数据需要传输时中断CPU,而DMA方式则是在所要求传送的一批数据全部传送结束时才中断CPU;此外，中断方式的数据传送是在中断处理时由CPU控制完成的，而DMA方式则是在DMA控制器的控制下完成的。
   - 分配和回收的方案见题目3.

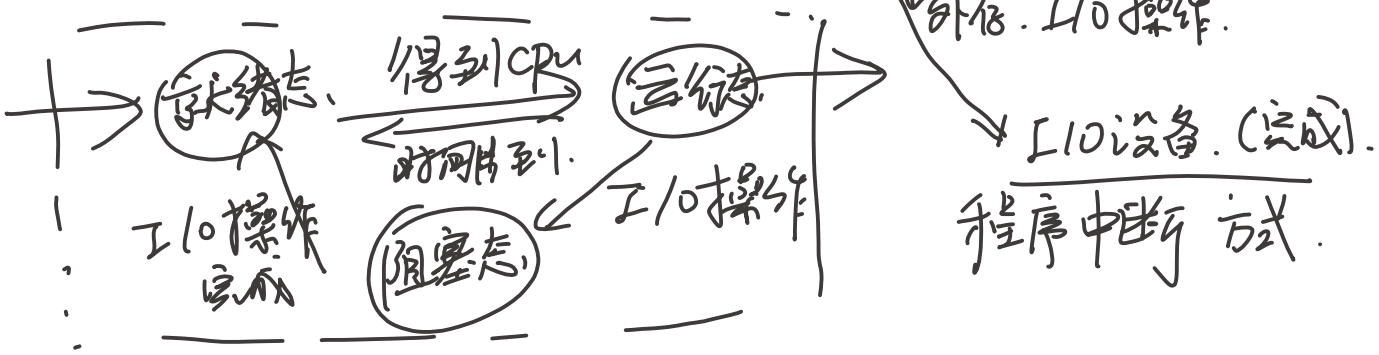
2. [I/O管理]非阻塞和阻塞I/O是什么，主要有什么不同，分别用在哪里.

操作系统的I/O接口还涉及两种模式：阻塞和非阻塞。

阻塞I/O是指当用户进程调用I/O操作时，进程就被阻塞，需要等待I/O操作完成，进程才被唤醒继续执行。

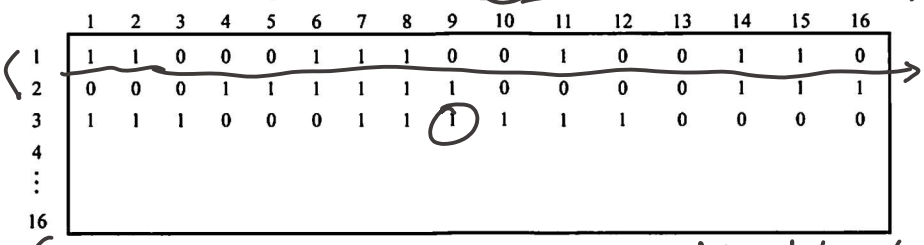
非阻塞I/O是指用户进程调用I/O操作时，不阻塞该进程，该I/O调用返回一个错误返回值，通常，进程需要通过轮询的方式来查询I/O操作是否完成。比如：打印机的输入和输出。

大多数操作系统提供的I/O接口都是采用阻塞I/O，等待I/O操作完成，将数据读入内存等等



3. [文件管理]以磁盘外存空间为例，设计高效的空闲块分配、回收算法，给出设计思想，操作方法，数据结构。（10分）

- 位图
  - 位示图是利用二进制的一位来表示磁盘中一个盘块的使用情况，磁盘上所有的盘块都有一个二进制位与之对应。当其值为“0”时，表示对应的盘块空闲；为“1”时，表示已分配。这样，一个m×n位组成的位示图就可用来表示m×n个盘块的使用情况：



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	1	1	0	0	0	1	1	1	0	0	1	0	0	1	1	0
2	0	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	1	1	1	1	1	1	0	0	0	0
4																
⋮																
16																

  - 盘块的分配：
    1)顺序扫描位示图，从中找出一个或一组其值为“0”的二进制位。
    2)将找到的一个或一组二进制位，转换成与之对应的盘块号。若找到的其值为“0”的二进制位位于位示图的第i行、第j列，则其相应的盘块号应按下式计算(n为每行位数)：
    $b = n(i-1) + j.$
  - 盘块的回收：
    1)将回收盘块的盘块号转换成位示图中的行号和列号。转换公式为：
    $i = (b-1)DIVn + 1$
    $j = (b-1)MODn + 1$
    2)修改位示图，令$map[i,j] = 0.$
    空闲表法和空闲链表法都不适用于大型文件系统，因为这会使空闲表或空闲链表太大。
- 链表
  - 将所有空闲盘区拉成一条空闲链。根据构成链所用基本元素的不同，分为两种形式：
  - 1)空闲盘块链。将磁盘上的所有空闲空间以盘块为单位拉成一条链。当用户因创建文件而请求分配存储空间时，系统从链首开始，依次摘下适当数目的空闲盘块分配给用户。当用户因删除文件而释放存储空间时，系统将回收的盘块依次插入空闲盘块链的末尾。这种方法的优点是分配和回收一个盘块的过程非常简单，但在为一个文件分配盘块时可能要重复操作多次，效率较低。又因它是以盘块为单位的，空闲盘块链会很长。
  - 2)空闲盘区链。将磁盘上的所有空闲盘区（每个盘区可包含若干个盘块）拉成一条链。每个盘区除含有用于指示下一个空闲盘区的指针外，还应有能指明本盘区大小（盘块数）的信息。分配盘区的方法与内存的动态分区分配类似，通常采用首次适应算法。在回收盘区时，同样也要将回收区与相邻接的空闲盘区合并。这种方法的优缺点刚好与第一种方法的相反，即分配与回收的过程比较复杂，但效率通常较高，且空闲盘区链较短。
- 块组链表

对空闲链表的一个改进是将n个空闲块的地址存在第一个空闲块中。这些块中的前n-1个确实为空，而最后一块包含另外n个空闲块的地址，如此继续。大量空闲块的地址可以很快地找到，这一点有别于标准链表方法。

- 第一空闲块+计数

另外一种方法是利用这样一个事实：通常，有多个连续块需要同时分配或释放，尤其是在使用连续分配和采用簇时更是如此。因此，不是记录n个空闲块的地址，而是可以记录第一块的地址和紧跟第一块的连续的空闲块的数量。这样，空闲空间表的每个条目包括磁盘地址和数量。虽然每个条目会比原来需要更多空间，但是表的总长度会更短，这是因为连续块的数量常常大于1。

4. [进程管理]在一个仓库中可以存放A和B两种产品，要求：
①每次只能存入一种产品。
②A产品数量-B产品数量 <M。
③B产品数量-A产品数量<N。
其中，M,N是正整数，试用 P操作、V操作描述产品 A 与产品 B 的入库过程。

[解答] 使用信号量mutex控制两个进程互斥访问临界资源（仓库），使用同步信号量Sa和Sb(分别代表产品A与B的还可容纳的数量差，以及产品B与A的还可容纳的数量差)满足条件2和条件3。代码如下：

```
Semaphore Sa=M-1,Sb=N-1;
Semaphore mutex=1;
//访问仓库的互斥信号量
process A(){
    while(1){
        P(Sa);
        P(mutex);
        A产品入库;
        V(mutex);
        V(Sb);
    }
}
process B(){
    while(1){
        P(Sb);
        P(mutex);
        B产品入库;
        V(mutex);
        V(Sa);
    }
}
```

学院 专业 级 学号 姓名 封 密 线

山东大学 数据库系统 课程试卷

09-10 学年 1 学期

题号	一	二	三	四	五	六	七	八	九	十	总分	总分人
得分												

得分	阅卷人

一、 名词解释（12 分）

1. 数据独立性

2. 正则覆盖

3. 两阶段封锁协议

4. 实体完整性约束

得分	阅卷人

二、简答（20 分）

1. 事务的 ACID 特性分别是什么？每个特性的用途是什么？

2. 死锁的发生是坏事还是好事？试说明理由。如何解除死锁状态？

3. 在嵌入式 SQL 中，什么情况下的 DML 语句不必涉及到游标操作？

4. 试述 ER 模型、层次模型、网状模型、关系模型和面向对象模型的主要特点

得分	阅卷人

三、设 R 和 S 是下图表示的关系, 计算下列关系代数表达式和元组表达式的值。(8 分)

A	B	C
2	4	6
3	2	1
5	4	4

R

A	D	E
3	6	9
3	4	5
2	4	7

S

1. R S

2. $\sigma_{A<E}(R \times S)$

3. $\{ t \mid \exists v \in S(\exists u \in R(u[C] > v[A] \wedge t[A]=u[B] \wedge t[B]=v[E] \wedge t[C]=u[A]))\}$

4. $\{ t \mid t \in R \wedge \forall u \in S(t[A] < u[E])\}$

得分	阅卷人

四、 假设某超市公司要设计一个数据库系统来管理该公司的业务信息。该超市公司的业务管理规则如下:

⑴该超市公司有若干仓库，若干连锁商店，供应若干商品。

⑵每个商店有一个经理和若干收银员，每个收银员只在一个商店工作。

⑶每个商店销售多种商品，每种商品可在不同的商店销售。

⑷每个商品编号只有一个商品名称，但不同的商品编号可以有相同的商品名称。每种商品可以有多种销售价格。

⑸超市公司的业务员负责商品的进货业务。

试按上述规则设计 ER 模型。(10 分)

得分	阅卷人

五、试解决下列问题（10 分）

1. 设关系模式 R（ABCD），F 是 R 上成立的 FD 集，F={A→B，B→C}，
   1) 试写出属性集 BD 的闭包$(BD)^+$。
   2) 试写出所有左部是 B 的函数依赖（即形为“B→？”）。(4 分)

2. 设关系模式 R（ABC），F 是 R 上成立的 FD 集，F={ C→B，B→A }。
   1) 试说明 R 不是 3NF 模式的理由。
   2) 试把 R 分解成 3NF 模式集。 (6 分)

得分	阅卷人

六、下图所示的调度是冲突可串行化的吗？如果是冲突可串行化的，请给出等价的串行调度序列；如果不是，请说明原因。 (5 分)

T1	T2	T3
Read(A)		
Write(A)		
		Read(A)
	Read(A)	
	Write(A)	
		Read(B)
Read(B)		
Write(B)		
	Read(B)	
	Write(B)	

姓名 学号 专业 学院

得分	阅卷人

七、设数据库中有三个关系：

职工表 EMP（E#，ENAME，AGE，SEX，ECITY），

其属性分别表示职工工号、姓名、年龄、性别和籍贯。

工作表 WORKS（E#，C#，SALARY），

其属性分别表示职工工号、工作的公司编号和工资。

公司表 COMP（C#，CNAME，CITY），

其属性分别表示公司编号、公司名称和公司所在城市。

试写出下列操作（35 分）：

1 分别使用 SQL 语句、关系代数和元组关系演算，检索超过 50 岁的男职工的工号和姓名。

2 假设每个职工可在多个公司工作，分别使用 SQL 语句、关系代数检索在编号为 C4 和 C8 公司兼职的职工工号和姓名。

3 假设每个职工可在多个公司工作，使用一 SQL 语句，检索每个职工的兼职公司数目和工资总数.显示（E#，NUM，SUM_SALARY），分别表示工号、公司数目和工资总数。

4 分别使用关系代数和 SQL 语句，求不在 C3 公司工作的职工姓名。

5 工号为 E6 的职工在多个公司工作，分别使用 SQL 语句、关系代数和元组关系演算，检索至少在 E6 职工兼职的所有公司工作的职工工号。

6 使用一 SQL 语句，检索联华公司中低于本公司平均工资的职工工号和姓名。

7 使用一 SQL 语句，在每一公司中为 50 岁以上职工加薪 100 元（若职工为多个公司工作，可重复加）。

8 使用一 SQL 语句，在 EMP 表和 WORKS 表中删除年龄大于 60 岁的职工有关元组。

数据库原理试题---A 卷答案

一. 名词解释（12 分）

1. 数据独立性：数据独立性是指应用程序与 DB 的数据结构之间相互独立。

评分细则：描述正确给全分

2. 正则覆盖：满足下列条件的函数依赖集 F 称为正则覆盖，记作 Fc：1）Fc 与 F 等价 2）Fc 中任何函数依赖都不含无关属性 3）Fc 中函数依赖的左半部都是唯一的

评分细则：每条 1 分

3. 两段锁协议：可以保证可串行性。两段锁协议要求每个事物分成两个阶段提出加锁和解锁申请：增长阶段：事物可以获得封锁，不能释放锁；缩减阶段：事物可以释放锁，但不能获得新锁。

评分细则：保证可串行 1 分，每个阶段 1 分

4. 实体完整性约束：关系的主码中的属性值不能为空值

评分细则：描述正确给全分

二. 简答（20 分）

1. 事务的 ACID 特性分别是什么？每个特性的用途是什么？

原子性(Atomicity) 事务中包含的所有操作要么全做，要么全不做

一致性(Consistency) 事务的隔离执行必须保证数据库的一致性

隔离性(Isolation) 系统必须保证事务不受其它并发执行事务的影响

持久性(Durability) 一个事务一旦提交之后，它对数据库的影响必须是永久的

评分细则： ACID 特性 1 分，用途每条 1 分

2. 死锁的发生是坏事还是好事？试说明理由。如何解除死锁状态？

答：在 DBS 运行时，死锁状态是我们不希望发生的，因此死锁的发生本身是一件坏事。但是坏事可以转换为好事。如果我们不让死锁发生，让事务任意并发做下去，那么有可能破坏 DB 中数据，或用户读了错误的数据。从这个意义上讲，死锁的发生是一件好事，能防止错误的发生。

在发生死锁后，系统的死锁处理机制和恢复程序就能起作用，抽取某个事务作为牺牲品，把它撤消，做 ROLLBACK 操作，使系统有可能摆脱死锁状态，继续运行下去。

评分细则： 好坏 1 分，理由 1 分，解除死锁 3 分

3. 在嵌入式 SQL 中，什么情况下的 DML 语句不必涉及到游标操作？

INSERT、DELETE 和 UPDATE 语句；

对于 SELECT 语句，如果已知查询结果肯定是单值时。

评分细则： INSERT、DELETE 和 UPDATE 语句各 1 分，select 语句 2 分

4. 试述 ER 模型、层次模型、网状模型、关系模型和面向对象模型的主要特点。

答：ER 模型直接表示实体类型及实体间联系，与计算机系统无关，充分反映用户的需求，用户容易理解。

层次模型的数据结构为树结构，记录之间联系通过指针实现，查询较快，但 DML 属于过程化的，操作复杂。

网状模型的数据结构为有向图，记录之间联系通过指针实现，查询较快，并且容易实现 M:N 联系，但 DML 属于过程化的语言，编程较复杂。

关系模型的数据结构为二维表格，容易为初学者理解。记录之间联系通过关键码实现。DML 属于非过程化语言，编程较简单。

面向对象模型能完整描述现实世界的数据结构，具有丰富的表达能力，能表达嵌套、递归的数据结构。但涉及的知识面较广，用户较难理解

评分细则：各 1 分

三. 设 R 和 S 是下图表示的关系, 计算下列关系代数表达式和元组表达式的值。（8 分）

A	B	C
2	4	6
3	2	1
5	4	4

R

A	D	E
3	6	9
3	4	5
2	4	7

S

1. $R \bowtie S$

A	B	C	D	E
2	4	6	4	7
3	2	1	6	9
3	2	1	4	5

2. $\sigma_{A<E}(R \times S)$

A	B	C	S_A	D	E
2	4	6	3	6	9
2	4	6	3	4	5
2	4	6	2	4	7
3	2	1	3	6	9
3	2	1	3	4	5
3	2	1	2	4	7
5	4	4	3	6	9
5	4	4	2	4	7

3. $\{ t \mid \exists v \in S(\exists u \in R(u[C]>v[A] \wedge t[A]=u[B] \wedge t[B]=v[E] \wedge t[C]=u[A]))\}$

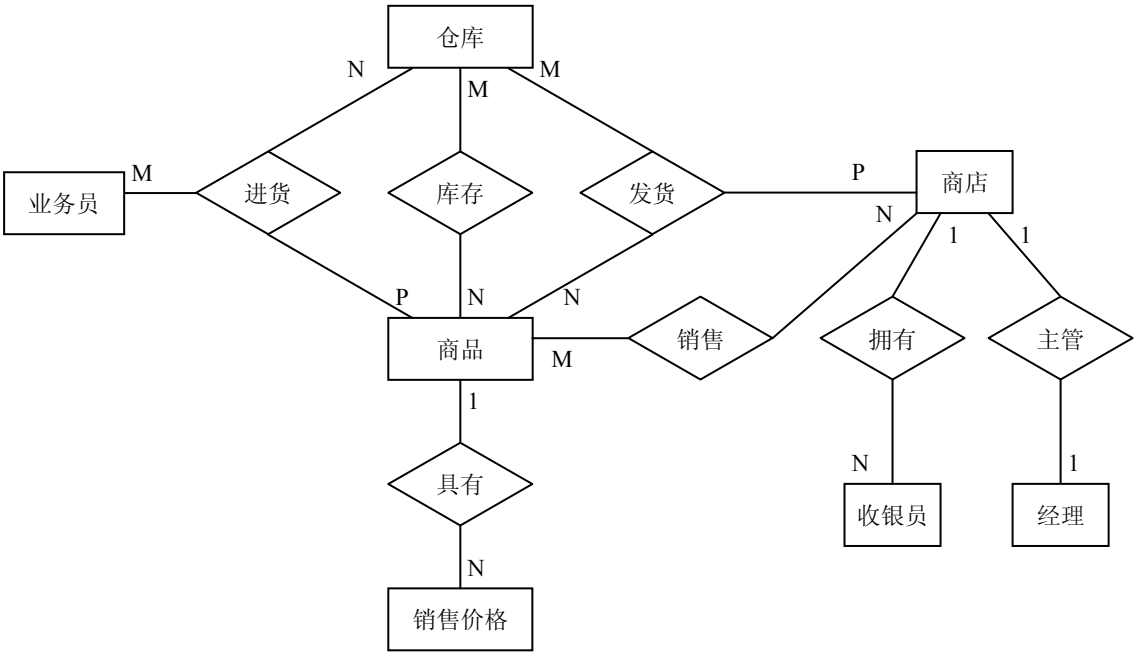
A	B	C
4	9	2
4	5	2
4	7	2
4	9	5
4	5	5
4	7	5

4. $\{ t \mid t \in R \wedge \forall u \in S(t[A] < u[E])\}$

A	B	C
2	4	6
3	2	1

评分细则：结果正确得全分，结果有错误得 0 分。

四. （10 分）



评分细则：实体集 5 分，联系集 5 分需要表明联系的映射基数

五. 试解决下列问题（10 分）

1. 解：①从已知的 F，可推出 BD→BCD，所以$(BD)^+$=BCD。

②由于 B^+=BC，因此左部是 B 的 FD 有四个：

B→ ϕ，B→B，B→C，B→BC。

评分细则：①2 分②2 分

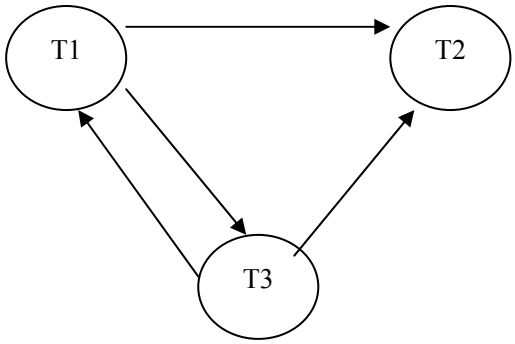
2. 解：①从已知 FD 集 F，可知 R 的候选键是 C。

从 C→B 和 B→A，可知 C→A 是一个传递依赖，因此 R 不是 3NF 模式。

②此时 R 应分解成 ρ ={ CB，BA }， ρ 是 3NF 模式集。

评分细则：①3 分②3 分

六.



有环，不可串行

评分细则：图不对写明是不可串行化的，得 3 分，图对写明是不可串行化的，得 5 分

七. 设数据库中有三个关系：

职工表 EMP（E#，ENAME，AGE，SEX，ECITY），

其属性分别表示职工工号、姓名、年龄、性别和籍贯。

工作表 WORKS（E#，C#，SALARY），

其属性分别表示职工工号、工作的公司编号和工资。

公司表 COMP（C#，CNAME，CITY），

其属性分别表示公司编号、公司名称和公司所在城市。

试写出下列操作：

1 分别使用 SQL 语句、关系代数和元组关系演算，检索超过 50 岁的男职工的工号和姓名。

2 假设每个职工可在多个公司工作，分别使用 SQL 语句、关系代数检索在编号为 C4 和 C8 公司兼职的职工工号和姓名。

3 假设每个职工可在多个公司工作，使用一 SQL 语句，检索每个职工的兼职公司数目和工资总数.显示（E#，NUM，SUM_SALARY），分别表示工号、公司数目和工资总数。

4 分别使用关系代数和 SQL 语句，求不在 C3 公司工作的职工姓名。

5 工号为 E6 的职工在多个公司工作，分别使用 SQL 语句、关系代数和元组关系演算，检索至少在 E6 职工兼职的所有公司工作的职工工号。

6 使用一 SQL 语句，检索联华公司中低于本公司平均工资的职工工号和姓名。

7 使用一 SQL 语句，在每一公司中为 50 岁以上职工加薪 100 元（若职工为多个公司工作，可重复加）。

8 使用一 SQL 语句，在 EMP 表和 WORKS 表中删除年龄大于 60 岁的职工有关元组。

1
```
SELECT E#, ENAME
FROM EMP
WHERE AGE>50 AND SEX='M';
```

$$\pi_{E\#,ENAME}\left(\Sigma_{age>50\wedge sex='m'}(EMP)\right)$$

$$\{t \mid \exists s \in EMP(t[E\#]=s[E\#] \wedge t[ENAME]=s[ENAME] \wedge s[AGE]>50 \wedge s[SEX]='M')\}$$

2
```
SELECT A.E#, A.ENAME
FROM EMP A, WORKS B, WORKS C
WHERE A.E#=B.E# AND B.E#=C.E#
  AND B.C#='C4' AND C.C#='C8';
```

$$\pi_{E\#,ENAME}\left(\Sigma_{works.c\#='c4'\wedge emp.e\#=works.e\#}(EMP\times WORKS)\right)\cap \pi_{E\#,ENAME}\left(\Sigma_{works.c\#='c8'\wedge emp.e\#=works.e\#}(EMP\times WORKS)\right)$$

3
```
SELECT E#, COUNT(C#) AS NUM, SUM(SALARY) AS SUM_SALARY
FROM WORKS
GROUP BY E#;
```

4
```
select ENAME
From  emp
Where e# not in (select e# from works where c#='C3')
```

$$\pi_{ENAME}(EMP)-\pi_{ENAME}\left(\sigma_{C\#='C3'\wedge emp.e\#=works.e\#}(EMP\times WORKS)\right)$$

5
```
SELECT X.E#
FROM WORKS X
WHERE NOT EXISTS
        (SELECT *
          FROM WORKS Y
```

```
                WHERE E#='E6'
                  AND NOT EXISTS
                      (SELECT *
                        FROM WORKS Z
                        WHERE Z.E#=X.E#
                          AND Z.C#=Y.C#));
```

$\Pi_{E\#,\ C\#}(WORKS) \div \Pi_{C\#}(\sigma_{E\#='E6'}(WORKS))$

$\{t \mid \exists w \in EMP\ (t[E\#]=w[E\#]) \wedge (\forall u \in WORKS(u[E\#]='E6' \Rightarrow \exists v \in WORKS(u[C\#]=v[C\#] \wedge t[E\#]=v[E\#]))\}$

6
```
SELECT A.E#, A.ENAME
FROM EMP A, WORKS B, COMP C
WHERE A.E#=B.E# AND B.C#=C.C#
  AND CNAME='联华公司'
  AND SALARY<(SELECT AVG(SALARY)
              FROM WORKS, COMP
              WHERE WORKS.C#=COMP.C#
                AND CNAME='联华公司');
```

7
```
UPDATE WORKS
SET SALARY=SALARY+100
WHERE E# IN (SELECT E# FROM EMP WHERE AGE>50);
```

8
```
DELETE FROM WORKS
WHERE E# IN (SELECT E# FROM EMP WHERE AGE>60);
DELETE FROM EMP
    WHERE AGE>60;
```

评分细则：结果正确得全分，结果有错误得 0 分。

姓名 学号 级 专业 学院 密 封 线

山东大学 数据库系统 课程试卷

09-10 学年 1 学期

题号	一	二	三	四	五	六	七	八	九	十	总分	总分人
得分												

得分	阅卷人

一、 名词解释（12 分）

1. 事务

2. 正则覆盖

3. 弱实体集

4. DBMS

得分	阅卷人

二、简答（20 分）

1. 举例说明参照完整性对数据有什么要求。

2. 你是如何理解空值(NULL)的？

3. 简述数据库系统三级模式结构及其同数据独立性之间的关系。

4. 简述函数依赖与多值依赖的联系与区别。

姓名 学号 级 专业 学院 密 封 线

得分	阅卷人

三、设 R 和 S 是下图表示的关系，计算下列关系代数表达式和元组表达式的值。（8 分）

A	B	C
1	2	3
4	5	6
7	8	9

R

A	D	E
1	2	3
1	4	6
4	6	9

S

1. $R \bowtie S$

2. $\sigma_{B>D}(R \times S)$

3. $\{ t \mid \exists v \in S(\exists u \in R(u[C]>v[D] \wedge t[A]=u[B] \wedge t[B]=v[E] \wedge t[C]=u[A]))\}$

4. $\{ t \mid t \in R \wedge \forall u \in S(t[C] > u[A])\}$

得分	阅卷人

四、 一个工厂有若干仓库；每一仓库有若干职工作为仓库管理员，职工之间有领导与被领导的关系；仓库中保存工厂生产的多种零件。用 E-R 图表示上述内容，关注仓库面积、仓库中保存零件的种类、每种零件的入库时间及入库数量，职工的姓名、职称、职务及工资待遇，零件的颜色、成本及出厂价。并将 E-R 图转换成相应的关系模型（10 分）

得分	阅卷人

五、试解决下列问题（10 分）

1. 假设有关系 R(B,O,S,Q,I,D)，其函数依赖集为{S→D, I→B, I S→Q, B→O} (6 分)
   1) 找出的关系模式 R 的候选码。
   2) 将关系模式 R 规范化为 BCNF。

2. 证明如果一个关系模式是 BCNF 则一定是 3NF。(4 分)

得分	阅卷人

六、下图所示的调度是冲突可串行化的吗？如果是冲突可串行化的，请给出等价的串行调度序列；如果不是，请说明原因。 (5 分)

T1	T2	T3
	Write(Q)	
		Read(Q)
Read(Q)		
Write(Q)		
		Write(Q)

得分	阅卷人

七、 有关系 S(SNO,SNAME,DEPT), C(CNO,CNAME), SC(SNO,CNO,SCORE)。关系 S、C 和 SC 分别表示学生信息、课程信息和学生选课情况。请按要求表达下列查询。(35 分)

其属性分别表示如下：

SNO—学生编号，SNAME—学生姓名，DEPT—学生所在系，CNO—课程编号，CNAME—课程名称，SCORE—成绩。

1. 分别使用 SQL 语句、关系代数和元组关系演算，求选修了课程号为 C4 的学生的学号及成绩。
2. 分别使用 SQL 语句和关系代数，求计算机系所有学生的成绩，包括 SNO,SNAME, CNO,CNAME, SCORE。
3. 使用一 SQL 语句，求数据库课程的平均成绩。
4. 分别使用关系代数和 SQL 语句，求没有学习 C1 课程的学生姓名。
5. 使用一 SQL 语句，求出有 2 门以上成绩为优(>=90)的学生学号。
6. 分别使用 SQL 语句、关系代数和元组关系演算，求选修了学生 s3 所选全部课程的学生学号。
7. 使用一 SQL 语句，将所有课程的分数加 5 分。
8. 使用一 SQL 语句，对计算机系学生的成绩，如低于本门课程平均成绩的一半，则提高 5%。

山东大学 数据库系统 课程试卷

09-10 学年 1 学期

姓名

学号

级

专业

学院

数据库原理试题---B 卷答案

一. 名词解释（12 分）

1. 弱实体集：如果一个实体集的所有属性都不足以形成主码，则称这样的实体集为弱实体集。

评分细则：描述正确给全分

2. 事务：是由一系列操作序列构成的程序执行单元，这些操作要么都做，要么都不做，是一个不可分割的工作单位。

评分细则：描述正确给全分

3. 正则覆盖：满足下列条件的函数依赖集 F 称为正则覆盖，记作 Fc：1）Fc 与 F 等价 2）Fc 中任何函数依赖都不含无关属性 3）Fc 中函数依赖的左半部都是唯一的

评分细则：每条 1 分

4. DBMS：系统软件，对数据库进行统一管理和控制

评分细则：描述正确给全分

二. 简答（20 分）

1. 举例说明参照完整性对数据有什么要求。

如果关系 R2 的外部码 Fk 与关系 R1 的主码 Pk 相对应，则 R2 中的每一个元组的 Fk 值或者等于 R1 中某个元组的 Pk 值，或者为空值。

评分细则：主外码相对应 1 分，可取空值 1 分，等于主码 1 分。

2. 你是如何理解空值(NULL)的？

空值就是表示“无意义”，当实体在某个属性上没有值时设为 null；或者表示“值未知”，即值存在，但目前没有获得该信息；当空值参与运算，结果为空值。

评分细则：无意义，值未知各 2 分，参与运算为空 1 分。

3. 简述数据库系统三级模式结构及其同数据独立性之间的关系。

为了提高数据的物理独立性和逻辑独立性，使数据库的用户观点，即用户看到的数据库，与数据库的物理方面，即实际存储的数据库区分开来，数据库系统的模式是分级的，美国数据系统语言协商会）提出模式、外模式、存储模式三级模式的概念。三级模式之间有两级映象；存储结构改变时，修改模式/内模式映象，使模式保持不变，从而应用程序可以保持不变，称为数据的物理独立性；当模式改变时，修改外模式/模式映象，使外模式保持不变，从而应用程序可以保持不变，称为数据的逻辑独立性

评分细则：数据库系统三级模式结构 3 分，与两种数据独立性之间的关系各 1 分

4. 简述函数依赖与多值依赖的联系与区别。

函数依赖规定某些元组不能出现在关系中，也称为相等产生依赖；多值依赖要求某种形式的其它元组必须在关系中，称为元组产生依赖。

X→Y 的有效性仅决定于 X、Y 属性集上的值；X→→Y 的有效性与属性集范围有关

评分细则：联系 3 分，区别 2 分

三. 设 R 和 S 是下图表示的关系，计算下列关系代数表达式和元组表达式的值。（8 分）

A	B	C
1	2	3
4	5	6
7	8	9

R

A	D	E
1	2	3
1	4	6
4	6	9

S

1. $R \bowtie S$

A	B	C	D	E
1	2	3	2	3
1	2	3	4	6
4	5	6	6	9

2. $\sigma_{B>D}(R \times S)$

A	B	C	S_A	D	E
4	5	6	1	2	3
4	5	6	1	4	6
7	8	9	1	2	3
7	8	9	1	4	6
7	8	9	4	6	9

3. $\{ t \mid \exists v \in S(\exists u \in R(u[C] > v[D] \wedge t[A] = u[B] \wedge t[B] = v[E] \wedge t[C] = u[A]))\}$

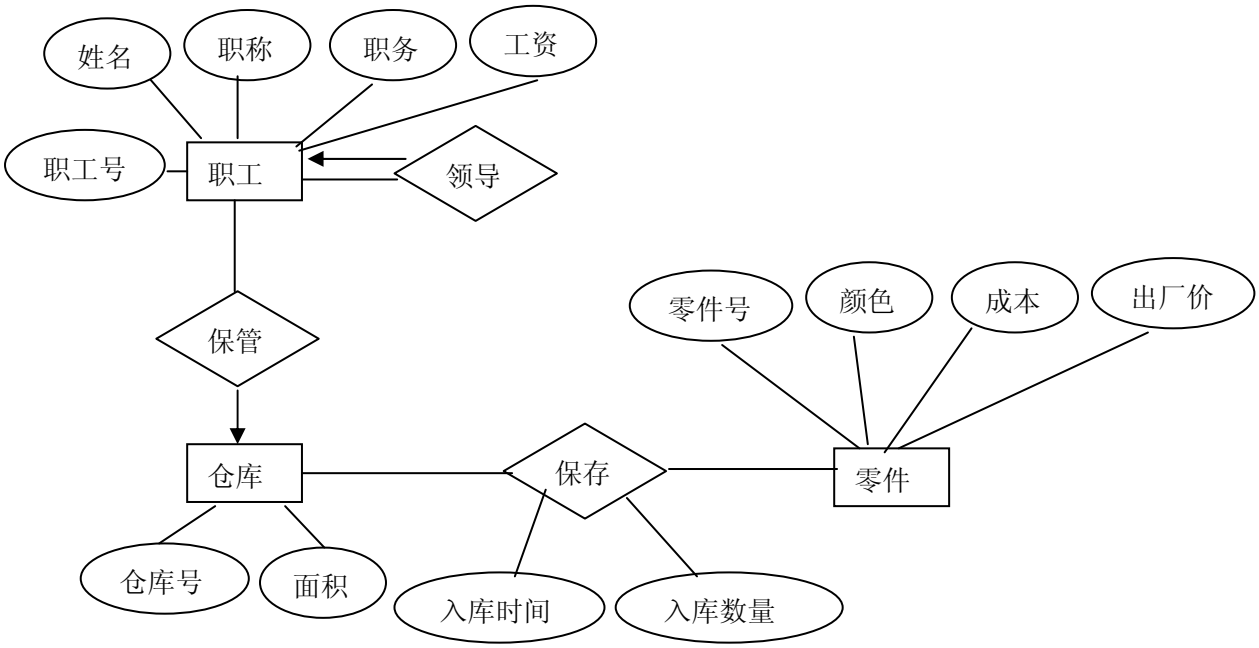
A	B	C
2	3	1
5	3	4
5	6	4
8	3	7
8	6	7
8	9	7

4. $\{ t \mid t \in R \wedge \forall u \in S(t[C] > u[A])\}$

A	B	C
4	5	6
7	8	9

评分细则：结果正确得全分，结果有错误得 0 分。

四. 一个工厂有若干仓库；每一仓库有若干职工作为仓库管理员，职工之间有领导与被领导的关系；仓库中保存工厂生产的多种零件。用 E-R 图表示上述内容，关注仓库面积、仓库中保存零件的种类、每种零件的入库时间及入库数量，职工的姓名、职称、职务及工资待遇，零件的颜色、成本及出厂价。并将 E-R 图转换成相应的关系模型（10 分）



职工(职工号、姓名、职称、职务、工资、领导、仓库号)
仓库(仓库号、面积)
零件(零件号、颜色、成本、出厂价)
保存(仓库号、零件号、入库时间、入库数量)
评分细则：画 E-R 图 5 分，需要表明联系的映射基数；关系模式 5 分，基于实体和基于联系的关系模式都需要。

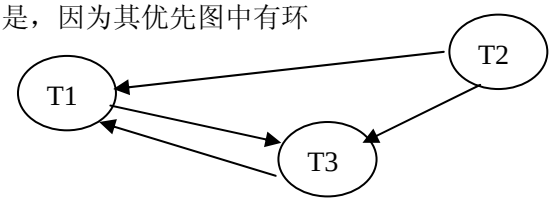
五. 试解决下列问题（10 分）

1. 假设有关系 R(B,O,S,Q,I,D), 其函数依赖集为{S→D, I→B, I S→Q, B→O} (6 分)
   1) 找出的关系模式 R 的所有候选码。 I S
   2) 将关系模式 R 规范化为 BCNF。 {SQI}{BO}{IB}{SD}
   评分细则：候选码 2 分 BCNF4 分。
2. 证明如果一个关系模式是 BCNF 则一定是第三范式。(4 分)
   定义证明或反证法
   评分细则：逻辑正确即可得全分。

六. 下图所示的调度是冲突可串行化的吗？如果是冲突可串行化的，请给出等价的串行调度序列；如果不是，请说明原因。 (5 分)

T1	T2	T3
	Write(Q)	
		Read(Q)
Read(Q)		
Write(Q)		
		Write(Q)

不是，因为其优先图中有环



评分细则：图不对写明是不可串行化的，得 3 分，图对写明是不可串行化的，得 5 分

七. 有关系 S(SNO,SNAME,DEPT), C(CNO,CNAME), SC(SNO,CNO,SCORE)。关系 S、C 和 SC 分别表示学生信息、课程信息和学生选课情况。请按要求表达下列查询。(35 分)
其属性分别表示如下：
SNO—学生编号，SNAME—学生姓名，DEPT—学生所在系，CNO—课程编号，CNAME—课程名称，SCORE—成绩。

1. 分别使用 SQL 语句、关系代数和元组关系演算，求选修了课程号为 C4 的学生的学号及成绩。

SELECT CNO, SCORE FROM SC WHERE CNO=C4

$$\pi_{CNO,\ SCORE}(\ \sigma_{CNO=C4}\ (SC))$$

$$\{t \mid \exists s \in SC\ (\ t[SNO] = s[SNO] \wedge t[SCORE] = s[SCORE] \wedge s[CNO]=C4\ )\}$$

2. 分别使用 SQL 语句和关系代数，求计算机系所有学生的成绩，包括 SNO,SNAME, CNO,CNAME, SCORE。

SELECT SNO,SNAME, CNO,CNAME, SCORE
FROM SC,S,C
WHERE DEPT='计算机系' AND S.SNO=SC.SNO AND C.CNO=SC.CNO

$$\pi_{SNO,SNAME,\ CNO,CNAME,\ SCORE}(\ \sigma_{DEPT='计算机系'\ \wedge\ S.SNO=SC.SNO\wedge\ C.CNO=SC.CNO}\ (SC\times S\times C))$$

3. 使用一 SQL 语句，求数据库课程的平均成绩。

SELECT AVG(SCORE)
FROM SC
WHERE CNO IN (SELECT CNO FROM C WHERE CNAME='数据库')

4. 分别使用关系代数和 SQL 语句，求没有学习 C1 课程的学生姓名。

SELECT SNAME
FROM S
WHERE SNO NOT IN
(SELECT SNO
FROM SC
WHERE CNO =C1)

$$\pi_{SNAME}(S) - \pi_{SNAME}(\ \sigma_{CN0\ =C1\ \wedge\ S.SNO=SC.SNO}\ (SC\times S))$$

5. 使用一 SQL 语句，求出有 2 门以上成绩为优(>=90)的学生学号。

Select sno
From sc
Where score>=90

Group by sno
Having count(*)>=2

6. 分别使用 SQL 语句、关系代数和元组关系演算，求选修了学生 s3 所选全部课程的学生学号。

```
select    SNAME
     from     S
     where    not exists    (select     CNO
                             from      C
                            where     exists    (select     *
                                                 from      SC
                                                where      SC.CNO = C.CNO
                                                 and     SC.SNO ='s3')
                            and  not  exists    (select     *
                                                 from      SC
                                                where      SC.CNO = C.CNO
                                                 and     SC.SNO = S.SNO)
```

$\Pi_{SNO,\ CNO}(SC) \div \Pi_{CNO}(\sigma_{SNO='S3'}(SC))$

$\{t \mid \forall u \in C\ (\exists s \in SC \wedge \exists w \in S\ (\ s[CNO] = u[CNO] \wedge w[SNO] = s[SNO] \wedge w[SNO] = '\ s3'\ ) \Rightarrow \exists s^1 \in SC \wedge \exists w^1 \in S\ (\ s^1[CNO] = u[CNO] \wedge w^1[SNO] = s^1[SNO] \wedge w^1[SNAME] = t[SNAME]\ ))\}$

7. 使用一 SQL 语句，将所有课程的分数加 5 分。

```
UPDATE    SC
 SET SCORE =SCORE + 5
```

8. 使用一 SQL 语句，对计算机系学生的成绩，如低于本门课程平均成绩的一半，则提高 5%。

```
UPDATE    SC
 SET SCORE =SCORE *1.05
WHERE SNO IN (SELECT SNO FRON S WHERE DEPT='计算机系')
      AND    SCORE < (SELECT AVG(SCORE)/2
                      FROM SC SC1
                      WHERE SC1.CNO =SC.CNO)
```

评分细则：结果正确得全分，结果有错误得 0 分。

山东大学
青年媒体中心
Youth Media Centre of SDU

算法设计与分析

1.如图m*n方格矩阵a[m][n]中摆放着价值不等的宝贝（价值可正可负），从左上角a[0][0]出发到达右下角a[m-1][n-1]，可以向右或向下走到相邻格子，并捡起当前格子的宝贝（无论价值的正负），每个格子只能走一遍，求能捡到宝贝价值之和的最大值。

2	−1	6	−2	6
−3	2	5	−5	1
4	8	3	−2	4
5	2	8	−4	7

（1）按动态规划算法的解题过程，写出递推关系式。（6分）
（2）根据递推关系式，写出递归型的动态规划函数。（6分）

（1）递推关系式：

在该问题中，我们可以使用以下递推关系式来计算dp数组的值：

dp[i][j] = max(dp[i-1][j], dp[i][j-1]) + a[i][j]

其中，dp[i][j]表示从左上角a[0][0]走到当前格子a[i][j]时能够捡到宝贝的最大价值之和，a[i][j]表示当前格子的宝贝价值。

（2）我们可以编写递归型的动态规划函数来解决该问题。以下是一个示例的c++代码实现：

```
int max_value_recursive(vector<vector<int>>& a, int i, int j) {
    if (i == m && j == n) {
        return a[m][n];
    } else if (i == m) {
        return max_value_recursive(a, i, j+1) + a[i][j];
    } else if (j == n) {
        return max_value_recursive(a, i+1, j) + a[i][j];
    } else {
        return max(max_value_recursive(a, i+1, j), max_value_recursive(a, i, j+1)) + a[i][j];
    }
}
```

a表示宝贝价值的矩阵，i和j表示当前格子的索引。当i == m && j == n时，即到达右下角格子，直接返回该格子的宝贝价值。否则，如果i==m，则只能向右走，调用递归函数计算右边格子的最大价值之和并加上当前格子的宝贝价值。如果j为n，则只能向下走，调用递归函数计算下方格子的最大价值之和并加上当前格子的宝贝价值。否则，可以向下或向右走到达当前格子，取两者中的较大值作为当前格子的最大价值之和。

2.解释“归约”的概念并证明顶点覆盖归约到集合覆盖

"归约"是指将一个问题转化为另一个问题的过程，使得解决一个问题的算法可以被用来解决另一个问题。如果问题A可以通过归约转化为问题B，并且问题B的解决方案可以直接应用于问题A，那么我们说问题B归约到问题A。

首先，我们来定义顶点覆盖和集合覆盖问题：

顶点覆盖问题（Vertex Cover Problem）：给定一个无向图G=(V, E)，顶点覆盖是指在图中选择一些顶点，使得每条边都至少与其中一个顶点相连。目标是找到具有最小顶点数的顶点覆盖集合。

集合覆盖问题（Set Cover Problem）：给定一个集合U和它的一些子集合S1, S2, ..., Sm，集合覆盖是指选择最少的子集合，使得它们的并集等于U。目标是找到具有最小子集合数的集合覆盖。

顶点覆盖的实例：G=(V,E)，k，我们要转化为S，C，k。

S=E，

对于每个点$v_i \in V$,与v_i相关联的边集为c_i，因此这些组成了集合C，k直接用即可。

很显然，规约时间是多项式。

3.假设有一个大小至少为k的顶点覆盖A，则A的每个顶点都会对应到C中的元素，

组成了集合C'，因为A是顶点覆盖，因此他覆盖了所有边，因此$\bigcup_{c \in C'} c = S$，且$|C'| \leq k$。

4.假设有C的子集C'，且$|C'| \leq k$,且$\bigcup_{c \in C'} c = S$，则因为C'的每个元素代表了G中某个顶点所关联的边，

因此C'元素对应顶点组成了集合A，一定能够覆盖G中全部的边，又因为$|C'| \leq k$，因此A是G的至多为k的顶点覆盖。

3.

```
int BSearch(elemtype a[], elemtype x, int low, int high)
    if (low > high) return -1;
    int mid = (low + high) / 2;
    if (x == a[mid])
        return mid;
    if (x < a[mid])
        return BSearch(a, x, low, mid - 1);
    else
        return BSearch(a, x, mid + 1, high);
```

分析算法时间复杂性，列出递归方程

根据给定的二分查找递归算法，我们可以列出递归方程来分析其时间复杂性。

让我们定义递归函数的时间复杂性为T(n)，其中n表示输入数组的大小。

对于每次递归调用，算法将数组的一半大小进行处理。因此，递归方程可以表示为：

$T(n) = T(n/2) + O(1)$

其中T(n/2)表示递归调用的时间复杂性，O(1)表示每个递归步骤的常量时间复杂性。

根据递归方程，我们可以观察到每次递归调用都将问题规模缩小一半。这意味着递归树的高度为$\log_2 n$。

递归树的每个层级的工作量都是O(1)，因为每次递归调用只需要常量时间。

因此，我们可以计算出递归函数的时间复杂性：

$T(n) = O(1) + O(1) + \ldots + O(1)$ (总共$\log_2 n$个O(1))

$= O(\log_2 n)$

所以，该二分查找递归算法的时间复杂性为$O(\log_2 n)$。

4.设有n项独立的作业，由m台相同的机器加工处理。作业i所需要的处理时间为ti。约定：任何一项作业可在任何一台机器上处理，但未完工前不准中断处理；任何作业不能拆分更小的子作业。为多机调度问题设计一种调度算法，使所给的n个作业在尽可能短的时间内由m台机器处理完。

该贪心算法的基本思想是将作业按照它们的处理时间进行排序，然后将每个作业分配给当前空闲时间最早的机器。

具体来说，算法的步骤如下：

1.初始化m台机器的空闲时间为0。

2.对作业列表按照处理时间进行递减排序。这样做的目的是让处理时间较长的作业优先分配给机器，以便尽早释放机器的空闲时间。

3.依次遍历排序后的作业列表。

4.对于每个作业，找到当前空闲时间最早的机器，即空闲时间最小的机器。将该作业分配给这台机器，并更新该机器的空闲时间。更新方法是将作业的处理时间加上该机器的当前空闲时间。

5.重复步骤4，直到所有作业都被分配给机器。

通过按处理时间排序并优先分配处理时间较长的作业，可以使得每台机器的空闲时间尽可能早地释放出来，从而缩短整体作业完成时间。

```
multiMachineScheduling(jobs, m):
    Sort jobs in descending order based on processing time
    Create an array machineTime with size m and initialize all elements to 0
    for each job in jobs:
        Find the machine with the minimum machineTime and assign the job to that machine
        Update the machineTime of the assigned machine by adding the processing time of the job
    return machineTime
```

排序操作需要花费O(n log n)的时间，其中n是作业的数量。

在for循环中，我们遍历了每个作业并进行了一些常数时间的操作。总共有n个作业，所以这部分的时间复杂度为O(n)。

因此，算法的总体时间复杂度为O(n log n)。

基于Ford－Fulkerson方法的最大流

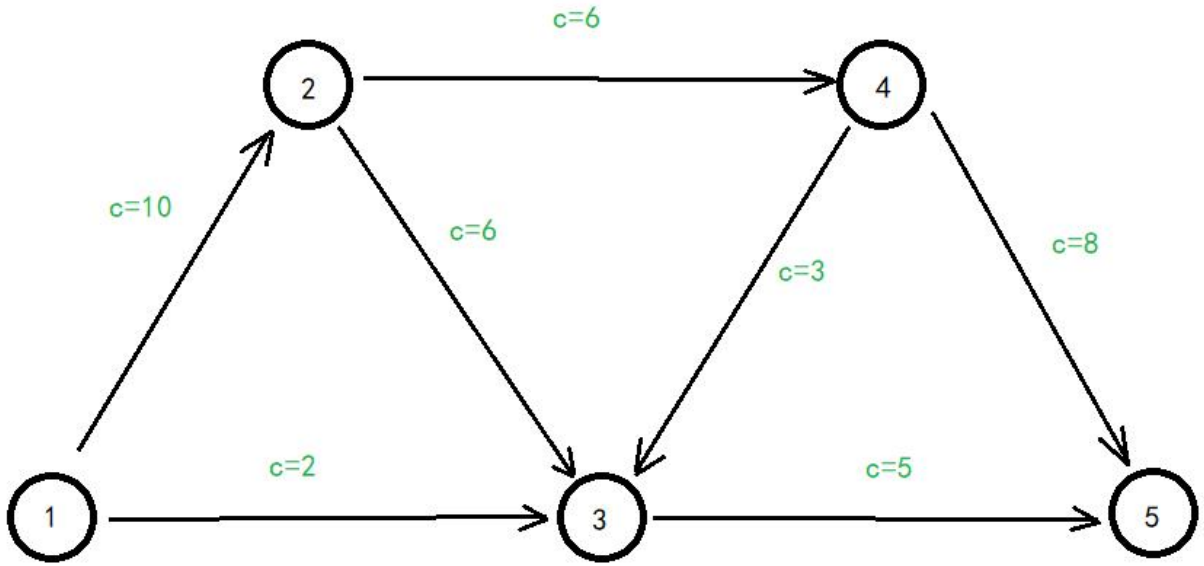
Ford-Fulkerson方法的一种实现寻找增广路径和删去 “ 割集中的边 ”。每次找到一条从源点到汇点的路径，记录这条路径的流的最大值即为路径上权值最小的边。并建立反边表示 “ 删去” 正向边，以便于在后来调整路径。当没有通向汇点的路径时，流饱和，退出搜索。

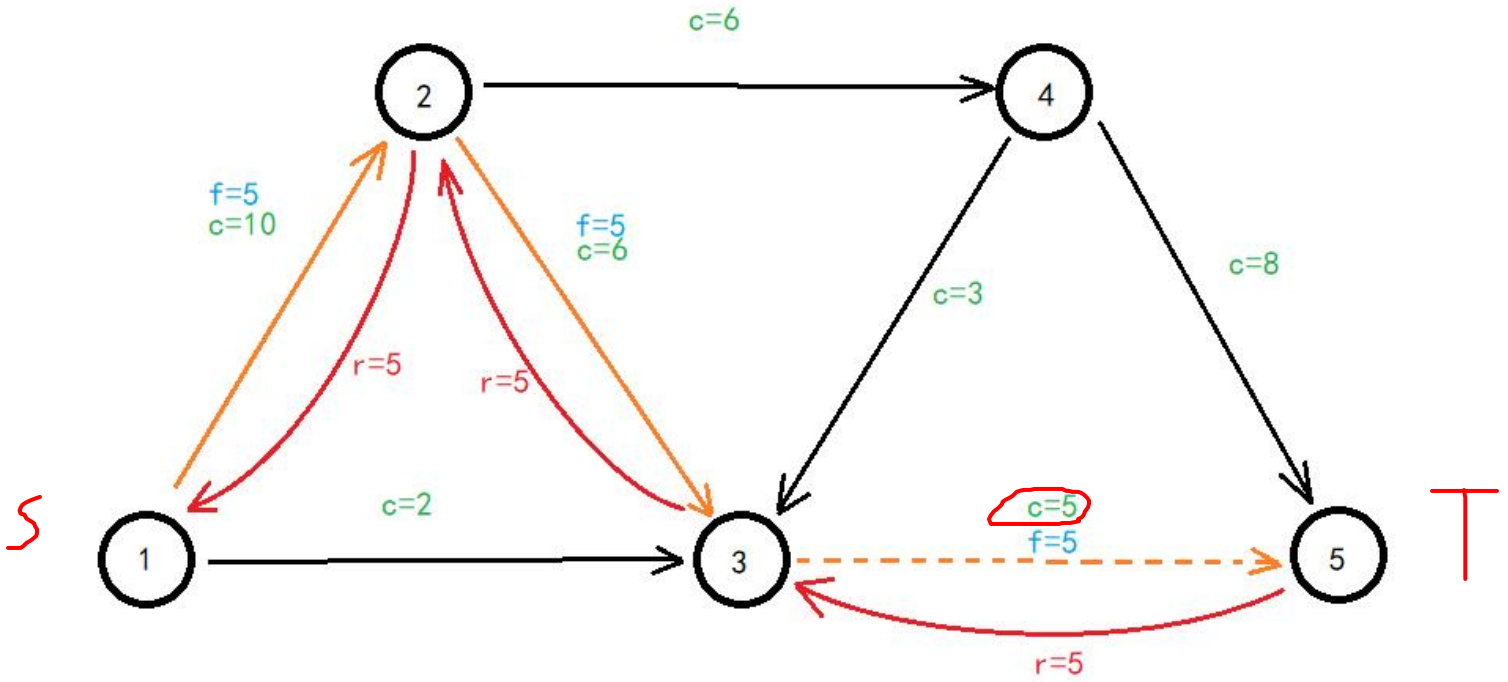
为了便于理解，我用一些图表现这一算法的思路。

设c为路的容量，r为反向边的容量，f为占用的流。

建立一张图，假设此时求点1到点5的最大流

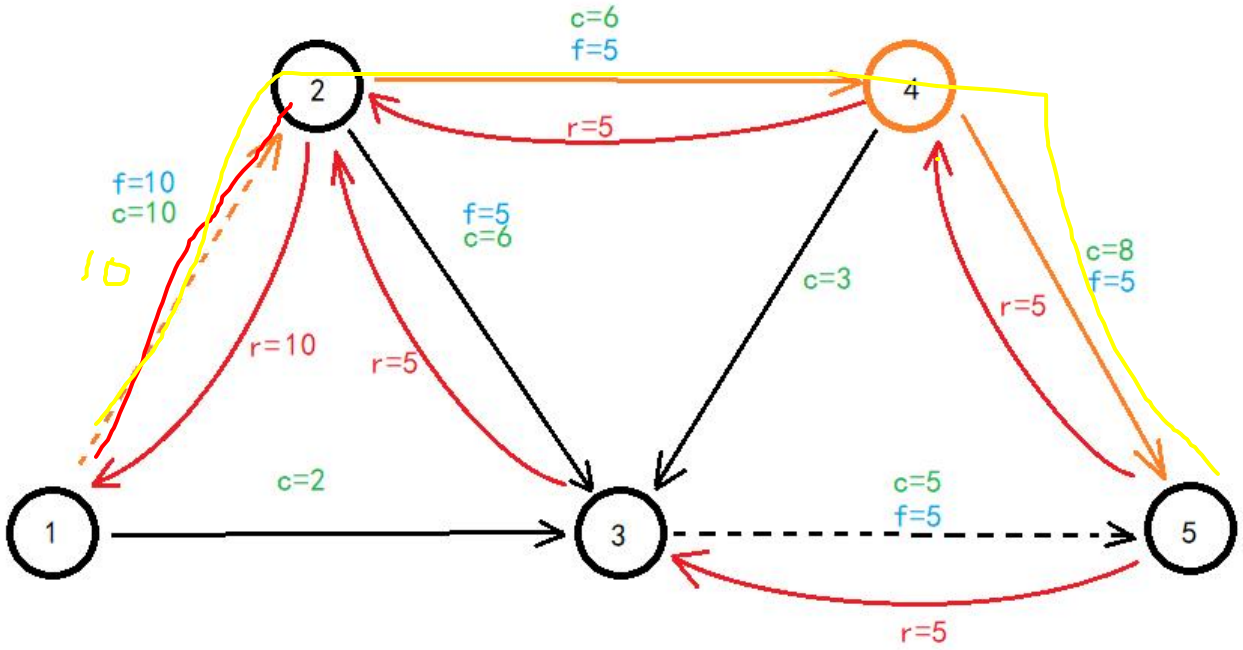
$f \leq c$





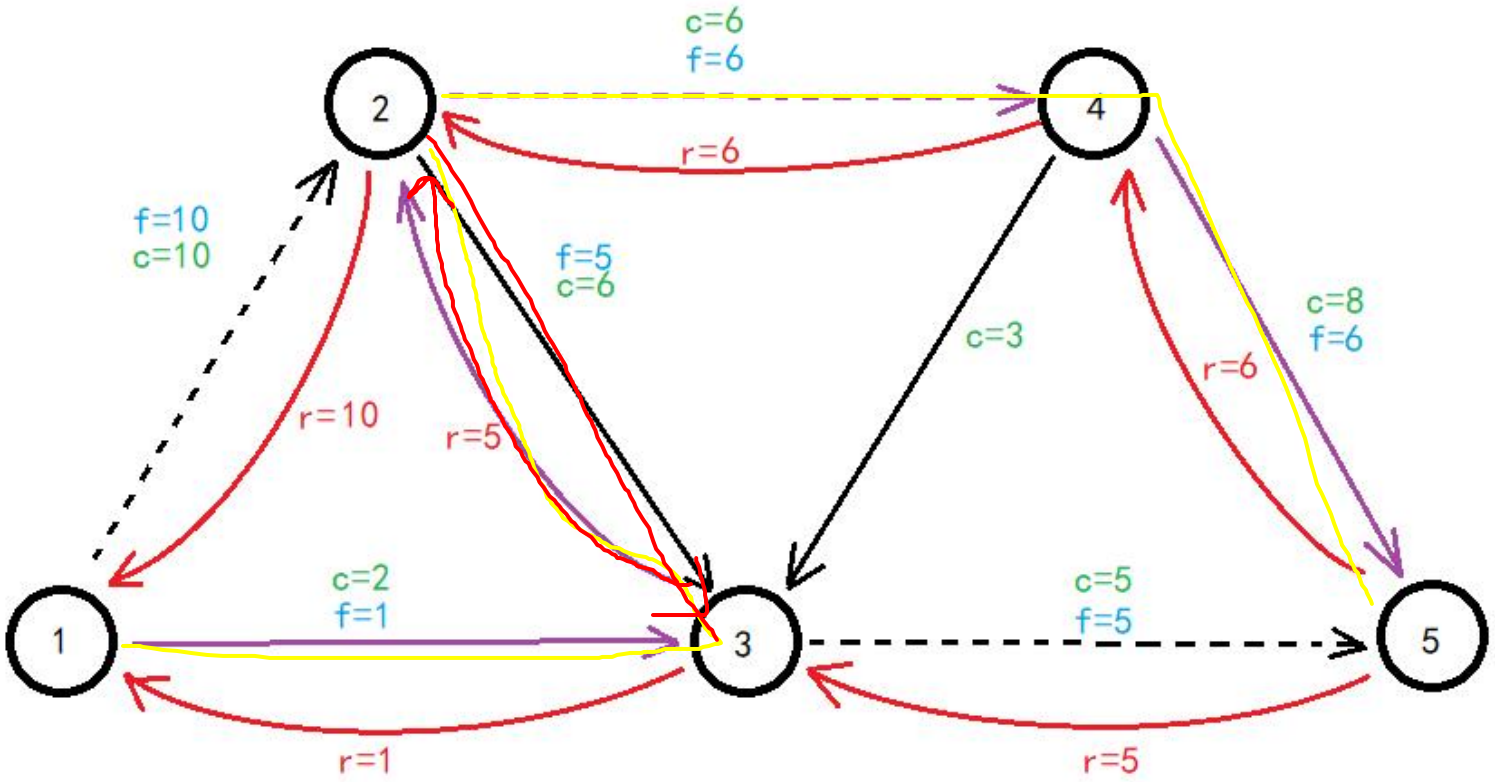
第一次DFS，找到一条通往点5的路径，此路径的流f=5，同时建立反边。

sumflow=5



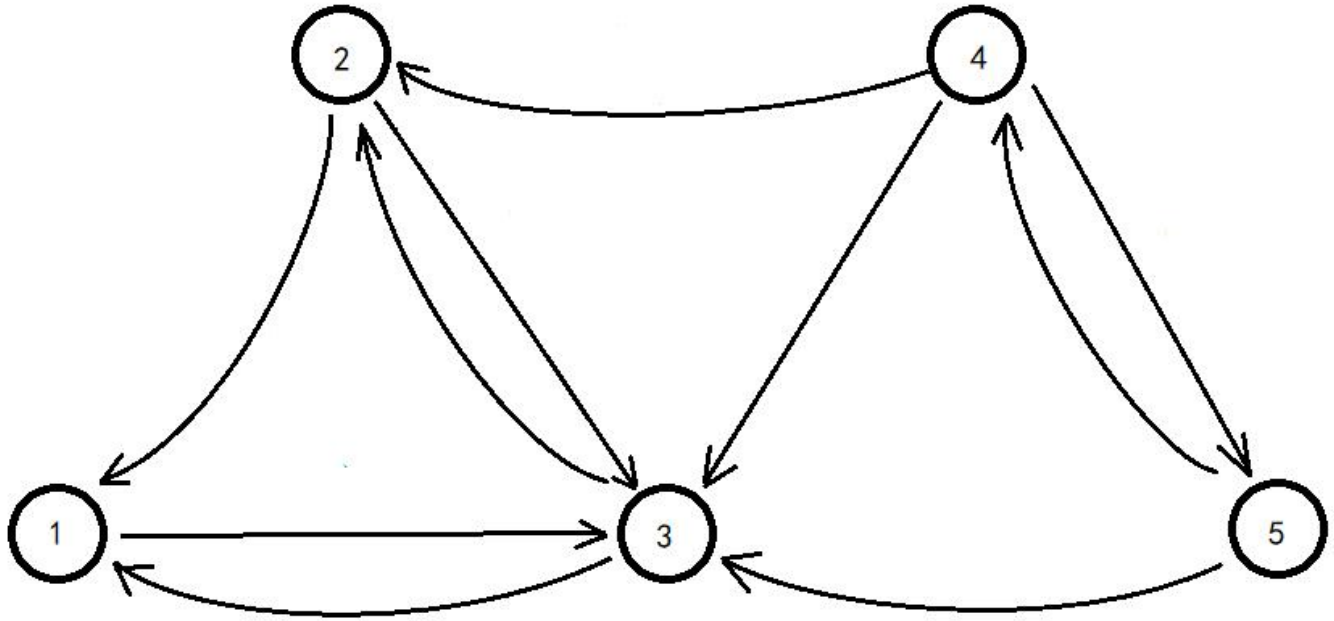
第二次DFS，找到路径1-2-4-5，得到新的流f=5，通过去掉虚线可以看出，仅用黑线和橙线已经无法连通道点5了。

sumflow=5+5=10



建立反边的意义就体现在此处，我们选择了1-3-2-4-5这条路，可以贡献流f=1

sumflow=5+5+1=11



至此，这张图已经不存在任何一条连通点1和点5的路径了

第1、2、3、4章：

- 算法复杂性，时间复杂性/空间复杂性；
- 函数的增长，渐进符号Θ, O, Ω；
- 分而治之：基本思想，归并排序、最大子数组；

第15章 动态规划

- 动态规划原理，最优子结构，设计和分析动态规划
- 钢条切割、矩阵链乘法、最长公共子序列问题。

第16章 贪心算法

- 贪心算法原理，设计和分析贪心算法；
- 活动选择问题；0/1背包问题，分数背包问题等；

第24章 单源最短路

- 最短路的性质和定理。
- Bellman-Ford算法，Dijkstra算法。

第25章 所有顶点对之间的最短路

- 按边数分解的（基于矩阵乘法的）动态规划算法。
- 按顶点编号分解的（Floyd-Warshall）动态规划算法。

第26章 最大流

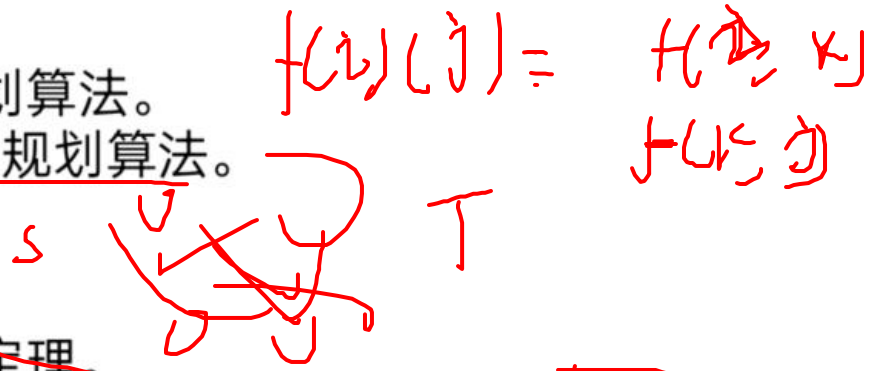
- 流网络的基本概念和性质。
- 最大流问题的增广路算法，最大流最小割定理。
- 使用最大流的方法计算最大匹配。

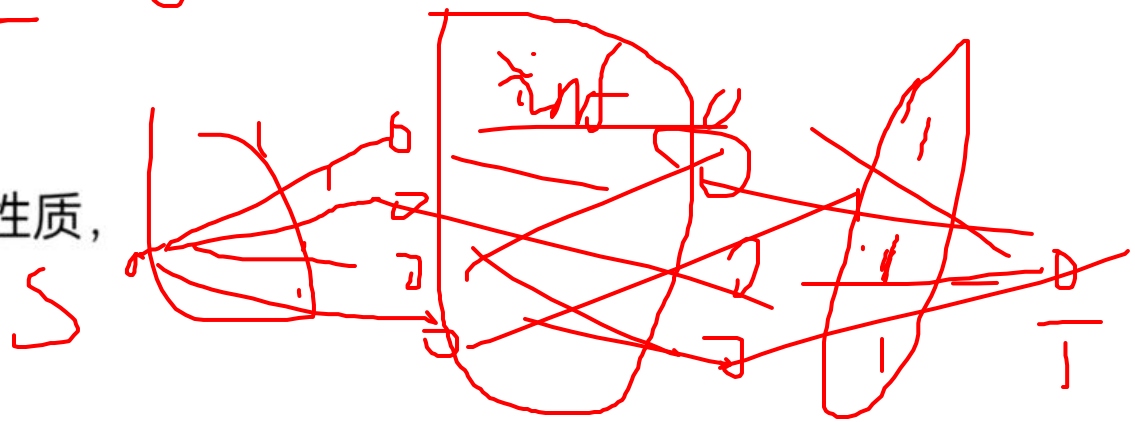
第34章 NPC

- P、NP、NPC、NP-hard、多项式归约的概念和性质，义。
- 基本、经典问题的NP完全性的证明。

第35章 近似算法

- 满足三角不等式的TSP问题的近似算法。
- 顶点覆盖问题的近似算法。



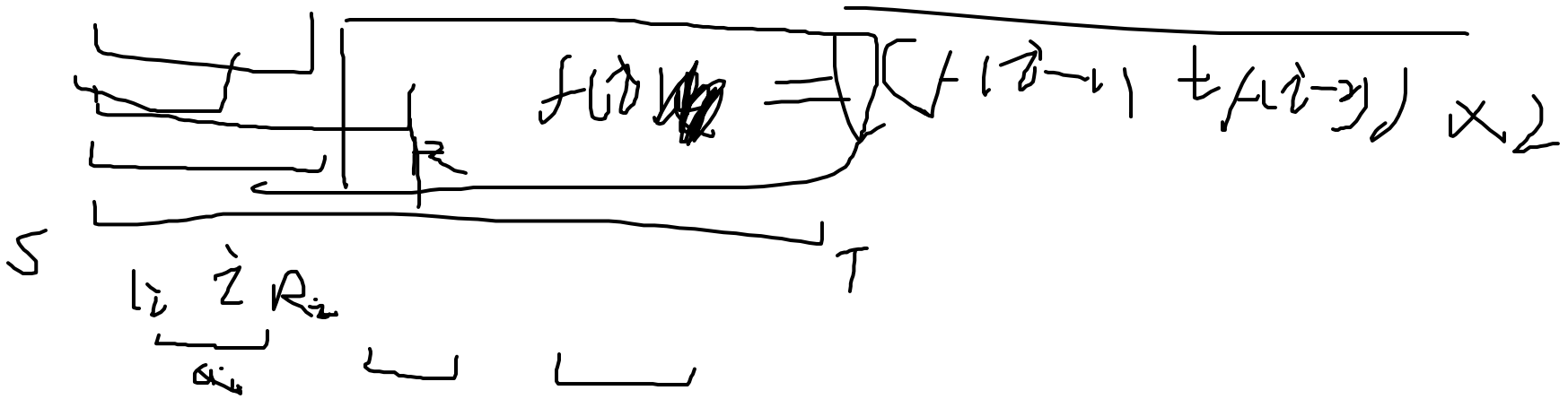


3.Floyd-Warshall算法，简述算法思想，说出时间复杂度，求解下一个最短路矩阵。

4.(1)写一种最小生成树的伪代码，分析时间复杂度。Prim或Kruskal均可；(2)在原图G中，加入一条边e，请设计一种高效的算法判断是否改变原来的最小生成树。简述算法思想，写出伪代码，分析时间复杂度，给出证明。

5.使用动态规划求解：一个字符串允许出现的字符可以是a、b或c，求长度为n且不包括两个连续a的字符串个数。简述算法思想，写出伪代码，分析时间复杂度、空间复杂度。

6.使用贪心算法求解：区间覆盖问题。简述算法思想，写出伪代码，分析时间复杂度，给出证明。



山东大学
青年媒体中心
Youth Media Centre of SDU

感谢您的倾听