期末重点

选择10题，每题2分，共20分
论述4题，每题8分，论述要点突出，不要水字数，共32分

- 软件维护类型
- 画PAD图
- 写PDL语言
- A测和B测的基本原则
- 软件再工程
- 产生软件危机的原因
  应用题3题，10+20+18共48分
  建模UML，数据流图，设计PDL

大题：软件定义阶段(ER图&数据流图&系统流程图)，软件测试(白盒测试)，项目管理(甘特图&工程网络图)

UML类图
用例图

第一章 概述

传统方法学（生命周期方法，结构化范型）

- 结构化分析、结构化设计、结构化实现
- 每个阶段的开始和结束都有严格标准，文档作为通信工具
  面向对象方法学
- 数据和行为同样重要
- 主动反复迭代
  软件危机

1. **需求不明确**
2. **开发方法落后**

3. **管理不善**
4. **复杂度高**
5. **维护困难**

软件生命周期
三大时期：软件定义时期、软件开发时期、软件维护时间
可行性研究：压缩和简化了系统分析和设计过程
需求分析：用正式文档记录对目标系统的需求
集成测试与单元测试：测试文档也是软件配置的组成部分
软件维护(4类维护活动)：改正性、适应性、完善性、预防性

1. **改正性维护**：修复软件中的错误和缺陷。
2. **适应性维护**：调整软件以适应环境或需求的变化。
3. **完善性维护**：改进软件功能或性能以满足用户新需求。
4. **预防性维护**：优化软件结构或代码以防止未来问题。

典型软件过程

- 瀑布模型
- 快速原型模型
- 螺旋模型
- 增量模型

第二章 可行性研究

本章重点是**系统流程图**和**数据流图**

第三章 需求分析

重点是**E-R图**
要会绘制

第五章 总体设计

设计原理：模块化、抽象
耦合和内聚：高内聚低耦合

耦合：**数据**<控制<公共环境<**内容**
内聚：**偶然**<逻辑<时间<通信<过程<顺序<**功能**

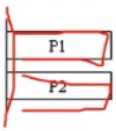
模块化的核心原则

1. **高内聚**：模块内部功能紧密相关
2. **低耦合**：模块之间依赖尽可能少
3. **独立性**：模块功能明确且独立
4. **可重用性**：模块设计便于复用
5. **可维护性**：模块易于理解和修改

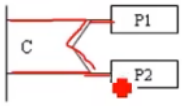
第六章 详细设计

过程设计的工具
问题分析图**PAD**
控制结构:

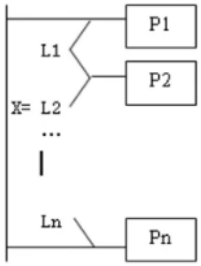
1. 顺序（先执行 P1 后执行 P2）



2. 选择（IF C THEN P1 Else P2）

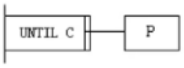


3. CASE 型多分支



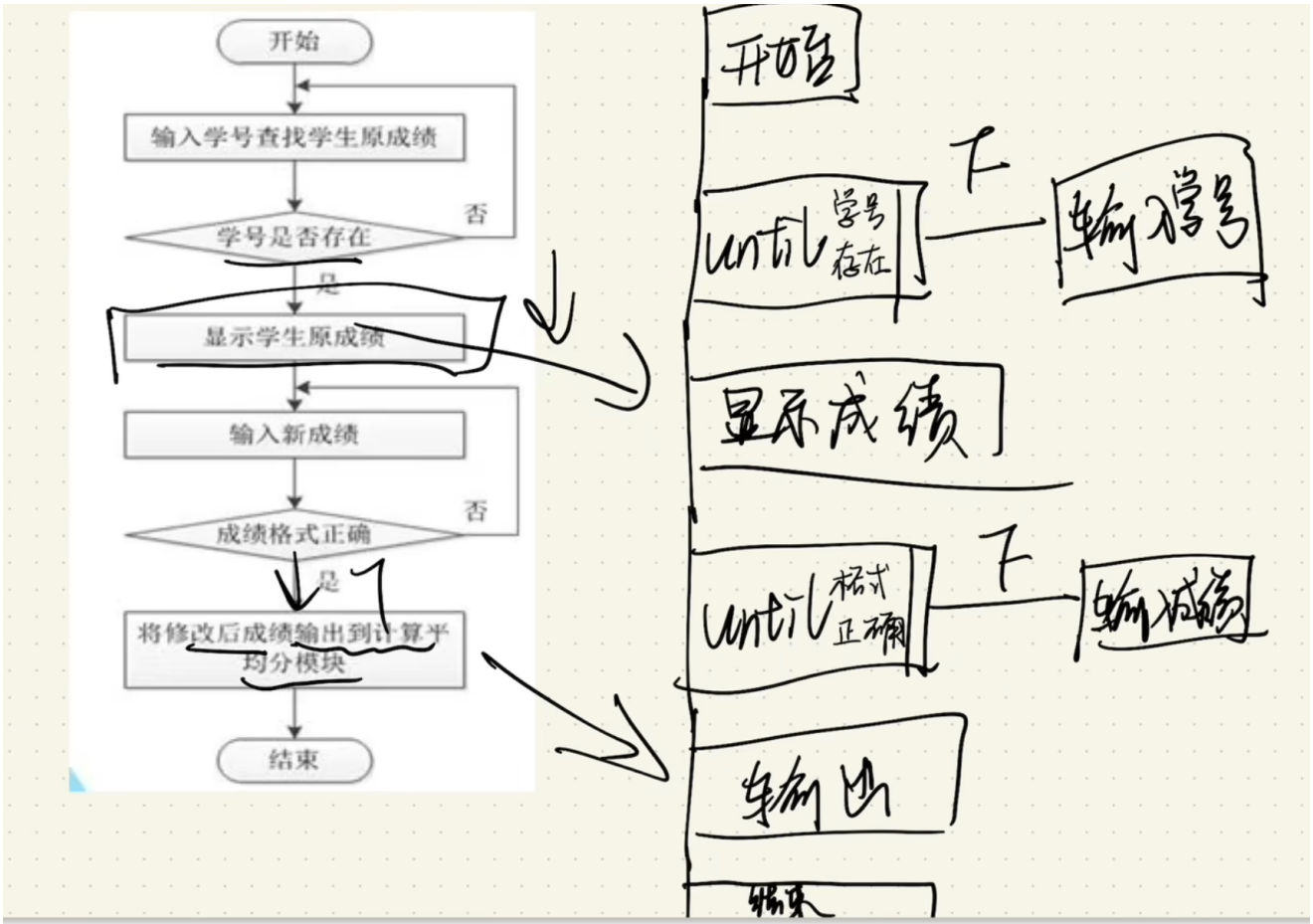
4. WHILE 型循环（WHILE C DO P）

5. UNTIL 型循环（RFPEAT P UNTIL C）



6. 语句标号

7. 定义



过程设计语言**PDL**
写伪代码就是了

第七章 实现

软件测试
单元测试与**集成测试**
模块测试（单元测试），发现编码和详细设计错误
子系统测试（集成测试），测试模块接口问题
系统测试（集成测试），发现设计错误和需求错误
验收测试/确认测试/交付测试

A测和B测的基本原则（验收测试）

- **A测**：内部环境，早期阶段，全面测试，快速反馈，非公开
- **B测**：外部环境，后期阶段，用户反馈，公开测试，优化改进
  Alpha测试：用户在开发者的场所和指导下进行测试
  Beta测试：用户在用户的场所测试，再报给开发者

白盒测试/逻辑覆盖
黑盒测试：等价划分（具体怎么划分看情况）（大概率是边界值分析）
如果规定了输入条件的**取值范围**或输入个数，则可划分出一个有效的等价类，两个无效的等价类

第八章 维护

预防性维护与软件再工程

需要通过实例理解

预防性维护

为了改进软件未来的可维护性和可靠性，或者为了给未来的改进提供更好的基础，而对软件进行的修改，通常称作预防性维护

改正性维护是改正软件中原有的错误，所以对软件的修改一般不会导致文档的修改，而适应性和完善性维护则不同，这两类维护一般要导致文档的修改

软件再工程过程

软件再工程也是一种**预防性维护**

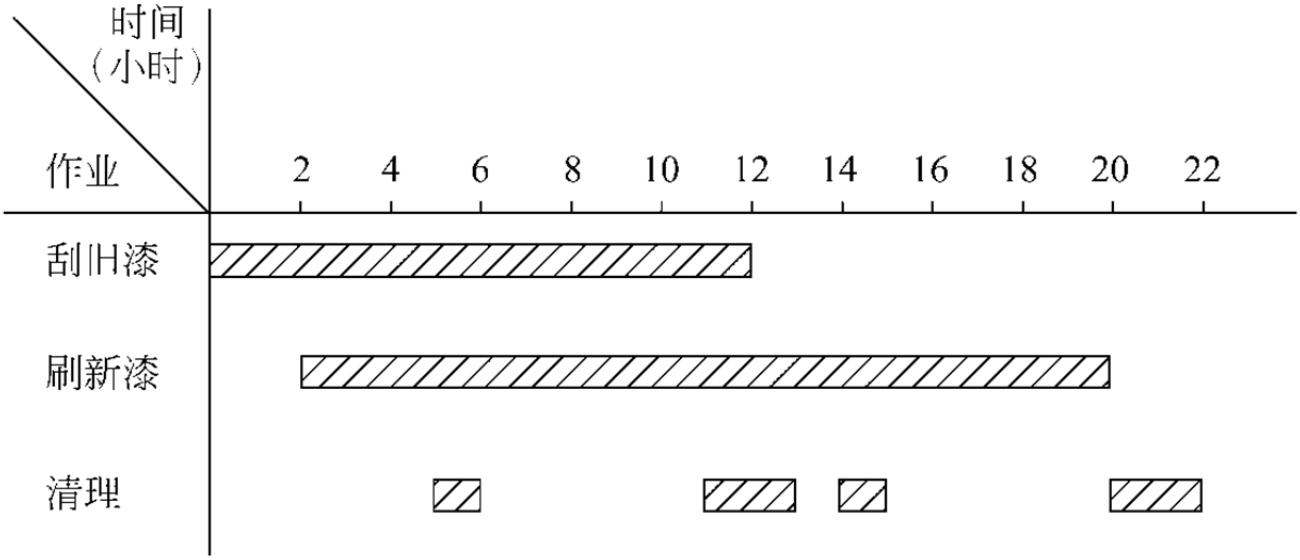
1. 库存目录分析
2. 文档重构
3. 逆向工程
4. 代码重构
5. 数据重构
6. 正向工程

- 然后从头循环这个过程

第十三章 软件项目管理

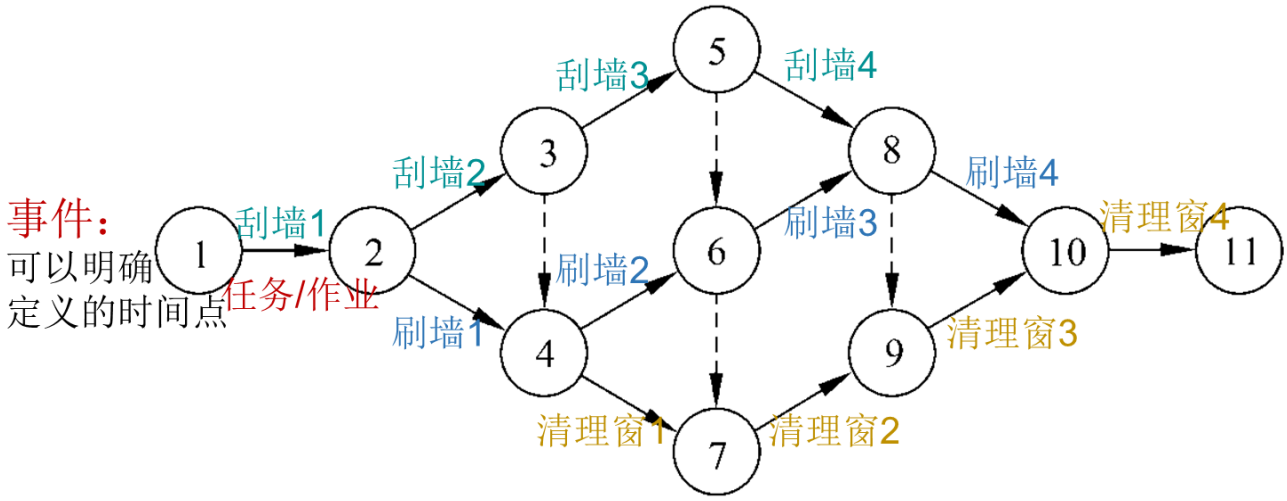
Gantt图

进度计划，直观展示每个子任务的起止时间，但表现不出依赖关系



工程网络PERT，要会计算机动时间

一定要自己会画

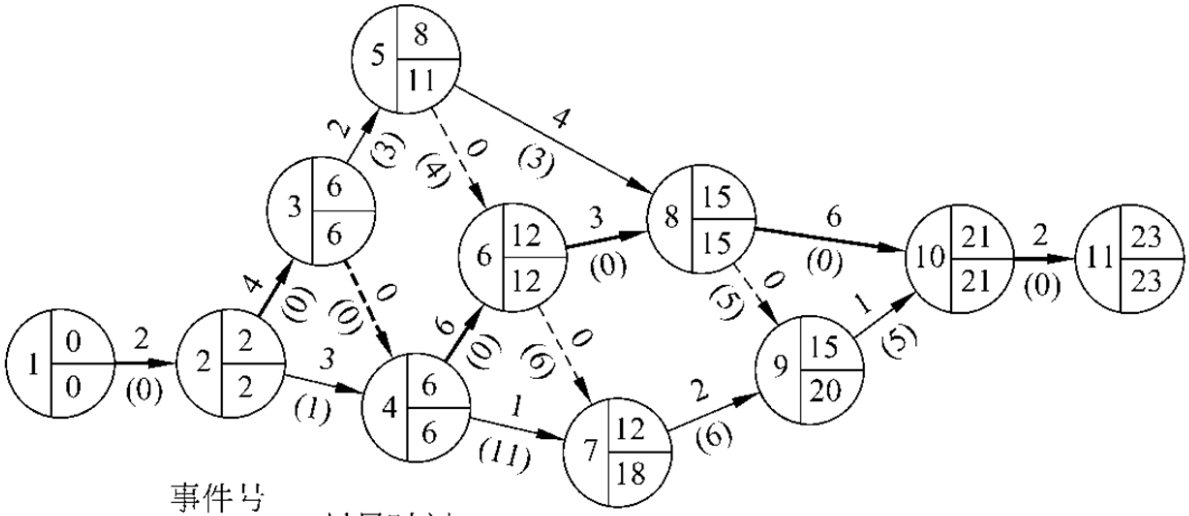


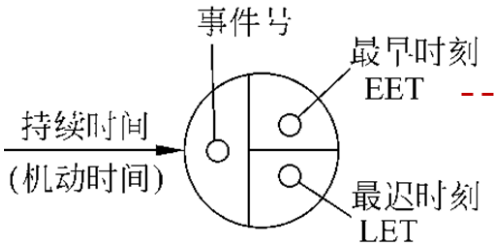
以上图是旧木板房刷漆工程的工程网络

图中表示刮第1面墙上旧漆的作业开始于事件1，结束于事件2。

用开始事件和结束事件的编号标识一个作业，因此“刮第1面墙上旧漆”是作业1-2

机动时间

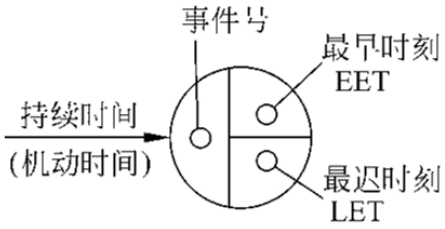




(1) 考虑进入该事件的所有作业。

(2) 对于每个作业都计算它的持续时间与起始事件的EET之和。

(3) 选取上述和数中的最大值作为该事件的最早时刻EET。



(1) 考虑离开该事件的所有作业。

(2) 从每个作业的结束事件的最迟时刻中减去该作业的持续时间。

(3) 选取上述差数中的最小值作为该事件的最迟时刻LET。

最早时刻是**最早开始时间**/最迟时刻是**最迟开始时间**

上最早下最迟

图中箭头上的数字就是**持续时间**

括号里的就是计算出来的机动时间

机动时间=后一个事件的最迟开始时间-前一个事件最早开始时间-这两个事件的持续时间

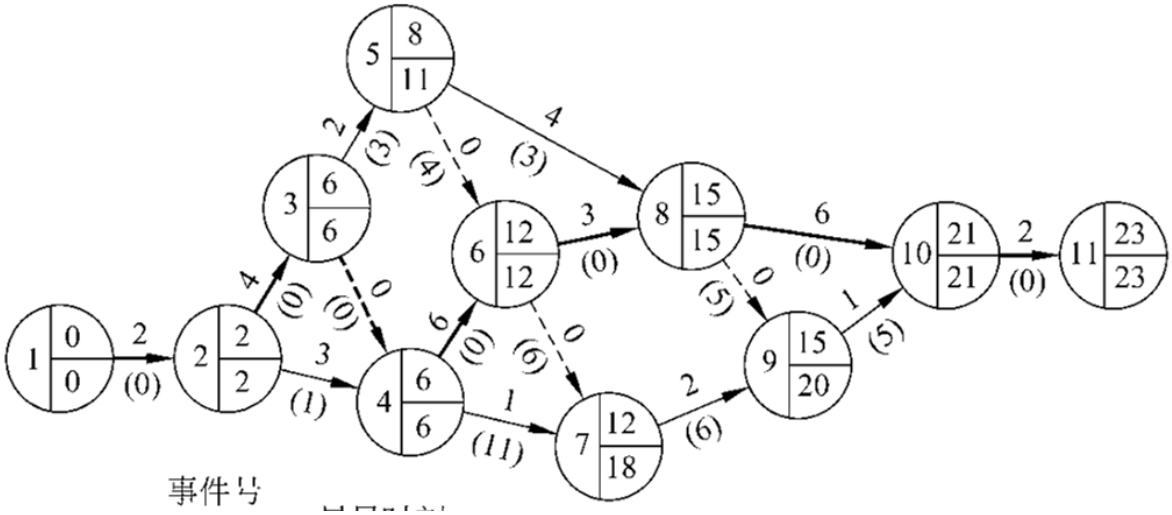
机动时间=(LET)结束-(EET)开始-持续时间

机动时间多意味着这些作业可以晚些开始或者持续时间长一些(少用一些资源)

关键路径

- **关键路径**是工程网络图中从项目开始到结束的最长路径
- 它决定了项目的最短总工期
- 关键路径上的所有活动都没有机动时间
  关键路径可以有多个
  非关键路径有一定的机动时间，不影响工程结束的时间

计算方法:
EET：从出发开始算，看不同路径算最长的时间
LET：结束事件的LET=结束事件，然后从结束开始往回算，看事件出去的箭头中减去持续事件，选择时间最小的



像这里事件8，有到事件10和事件9的箭头
事件10这里最迟21，21-6=15
事件9这里最迟20，20-0=20
因此时间8的最晚时间取min{15,20}=15

有依赖的情况下注意在甘特图中有表现出**依赖的先后关系**

软件质量保证和软件配置