Chapter 2

Math Prerequisites of Software Testing

2.1 Set Theory

2.2 Functions

2.3 Relations

2.4 Propositional Logic

2.5 Probability Theory

2.6 Graph Theory

2.7 Finite State Machines

2.8 Petri Net

2.9 State Charts

2.1 Set Theory

- **Set Definition**

  A={elements list }; B={elements list };

  $a \in A$, $b \notin B$;

- **Set Operations:**

  $A \cap B = \{x: x \in A \wedge x \in B\}$;

  $A \cup B = \{x: x \in A \vee x \in B\}$.

  $A' = \{x: x \notin A\}$;

  $A-B=\{x \in A \wedge x \notin B\}$;

  $A \oplus B=(A \cup B)-(A \cap B)$;

  $A \times B=\{\langle x, y\rangle : x \in A \wedge y \in B\}$

- **Set Relations**

  A is a subset of B: $A \subseteq B$;

  A is a proper subset of B,: $A \subset B$;

  A=B: iff $A \subseteq B \wedge B \subseteq A$;

- **Set Partitions**

  Given a set A, and a set of subsets A_1, A_2, …, An of A, the subsets are a partition of A iff:

  $A_1 \cup A_2 \cup \cdots \cup A_n = A$,

  and $i \neq j \Rightarrow A_i \cap A_j = \emptyset$.

2.2 Functions

- **Definition**

Given sets A and B, a function f is a subset of A × B such that for

$$a_i, a_j \in A, \quad b_i, b_j \in B, \text{ and } f(a_i)=b_i, \quad f(a_j)=b_j,$$

$$\text{If } b_i \neq b_j \Rightarrow a_i \neq a_j$$

or $f: A \rightarrow B$

$f \subseteq A \times B$

- **Function Types**

f is a function from A onto B iff f(A) = B （A到B上）

f is a function from A into B iff f(A) ⊂ B (note the proper subset here! （A到B中）

f is a one-to-one function from A to B iff,
for all ai, aj ∈ A, ai ≠ aj ⇒ f(ai) ≠ f(aj)

f is a many-to-one function from A to B iff,
there exists ai, aj ∈ A, ai ≠ aj such that f(ai) = f(aj).

2.3 Relations

- **Definition:**

Given two sets A and B, a relation R is a subset of the Cartesian product $A \times B$.

A relation $R \subseteq A \times A$ is

Reflexive： iff for all $a \in A$, $\langle a, a \rangle \in R$

Symmetric ： iff $\langle a, b \rangle \in R \Rightarrow \langle b, a \rangle \in R$

Antisymmetric： $\langle a, b \rangle, \langle b, a \rangle \in R \Rightarrow a = b$

Transitive： iff $\langle a, b \rangle, \langle b, c \rangle \in R \Rightarrow \langle a, c \rangle \in R$：

A relation $R \subseteq A \times A$ is an ordering relation if R is reflexive, antisymmetric, and transitive.

Fore example: $\geqslant$

A relation $R \subseteq A \times A$ is an equivalence relation if R is reflexive, symmetric, and transitive.

Fore example:

（相等，同余 ）

2.4 Propositional Logic

- **Definition:**

  A proposition is a sentence that is either true or false.

For example：6>2、Earth is a star；

but：water is cold, The mountain is very high, are not propositions.

- **Logical Operators**

  and ： $\wedge$

  or ： $\vee$

  not ： $\neg$

  exclusive–or ： $\oplus$

  IF-THEN： $\rightarrow$

p	q	$p \oplus q$	$p \rightarrow q$
T	T	F	T
T	F	T	F
F	T	T	T
F	F	F	T

2.5 Probability Theory

- **定义**

命题p为真的概率Pr(p), is

$$|T(p)|/|U|.$$

给定命题p,p的真集T是论域U中所有使p为真的元素的集合，记作T(p)

例如：一个袋子中有10个红球，10个兰球；从中任取一个球，是红球的几率？

再取一个是红球的几率？

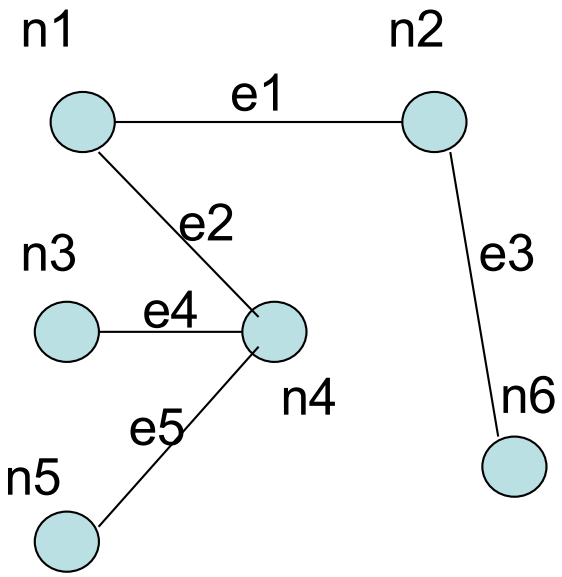
连续2个是红球的几率？

2.6 Graph Theory

- **Definition**

A graph G = (V, E) is composed of a finite (and nonempty) set V of nodes and a set E of unordered pairs of nodes.

Example 1：



V={n1,n2,n3,n4,n5};

E={e1,e2,e3,e4,e5}

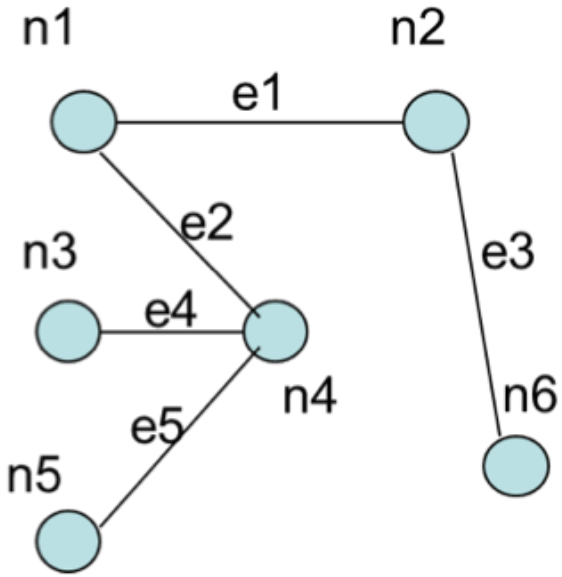
={(n1,n2),(n1,n4),(n2,n6),(n3,n4),(n4,n5)};

- Degree of a node: the number of edges that have that node as an endpoint.

deg(n1)=2, deg(n2)=2,…deg(n6)=1

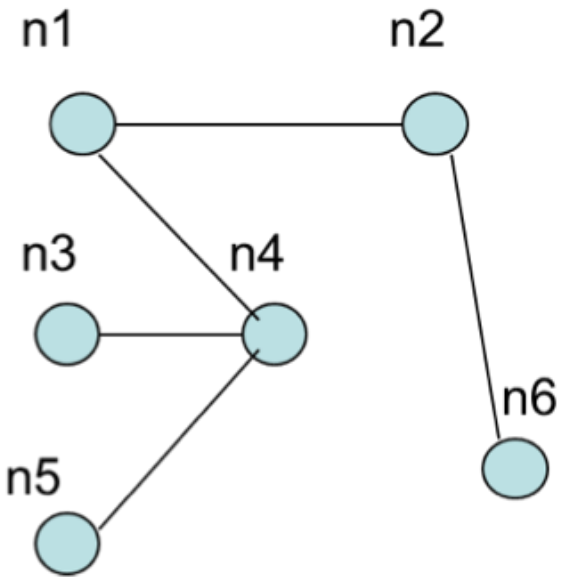
- **Incidence Matrices**： with m nodes and n edges is an m × n matrix, where the element in row i, column j is a 1 if and only if node i is an endpoint of edge j; otherwise, the element is 0.

	e1	e2	e3	e4	e5
n1	1	1	0	0	0
n2	1	0	1	0	0
n3	0	0	0	1	0
n4	0	1	0	1	1
n5	0	0	0	0	1
n6	0	0	1	0	0

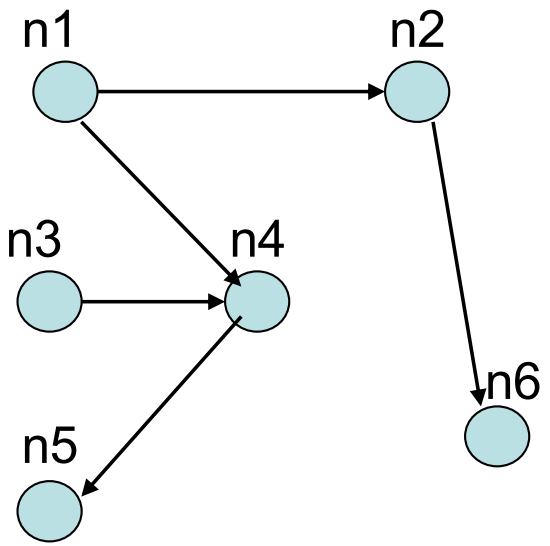


- **Adjacency Matrices**： The adjacency matrix of a graph G = (V, E) with m nodes is an m × m matrix, where the element in row i, column j is a 1 if and only if an edge exists between node i and node j; otherwise, the element is 0.

	n1	n2	n3	n4	n5	n6
n1	0	1	0	1	0	0
n2	1	0	0	0	0	1
n3	0	0	0	1	0	0
n4	1	0	1	0	1	0
n5	0	0	0	1	0	0
n6	0	1	0	0	0	0



- Path：a sequence of edges such that for any adjacent pair of edges ei, ej in the sequence, the edges share a common (node) endpoint.
- Connectedness: two nodes are connected if and only if they are in the same path.
- Component of a graph(分图): a maximal set of connected nodes.
- Directed Graphs
- Indegrees
- Outdegree

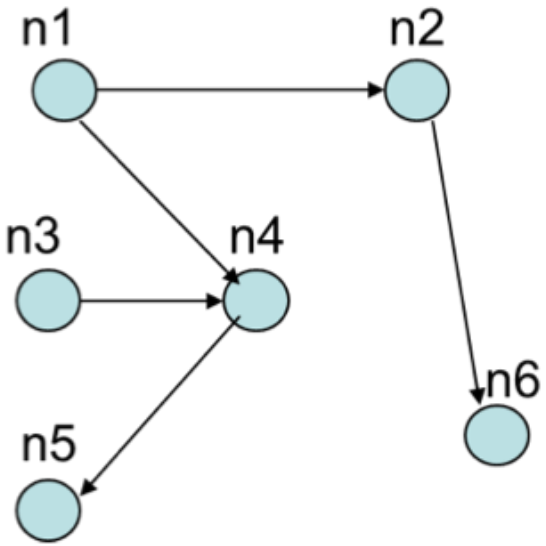


indeg(n1) = 0 outdeg(n1) = 2

indeg(n2) = 1 outdeg(n2) = 1

Adjacency Matrix of a Directed Graph

	n1	n2	n3	n4	n5	n6
n1	0	1	0	1	0	0
n2	0	0	0	0	0	1
n3	0	0	0	1	0	0
n4	0	0	0	0	1	0
n5	0	0	0	0	0	0
n6	0	0	0	0	0	0



• Paths and Semipaths

(directed) path： a sequence of edges such that, for any adjacent pair of edges ei, ej in the sequence, the terminal node of the first edge is the initial node of the second edge.

Cycle：a directed path that begins and ends at the same node.

Chain：a sequence of nodes such that each interior node has indegree = 1 and outdegree = 1.

(directed) semipath：a sequence of edges such that for at least one adjacent pair of edges ei, ej in the sequence, the initial node of the first edge is the initial node of the second edge or the terminal node of the first edge is the terminal node of the second edge.

For example:

A path from n1 to n6；A semipath between n1 and n3；

A semipath between n2 and n4；A semipath between n5 and n6

- **n-Connectedness**

0-connected: iff no path exists between ni and nj.

1-connected: iff a semipath but no path exists between ni and nj.

2-connected: iff a path exists between ni and nj.

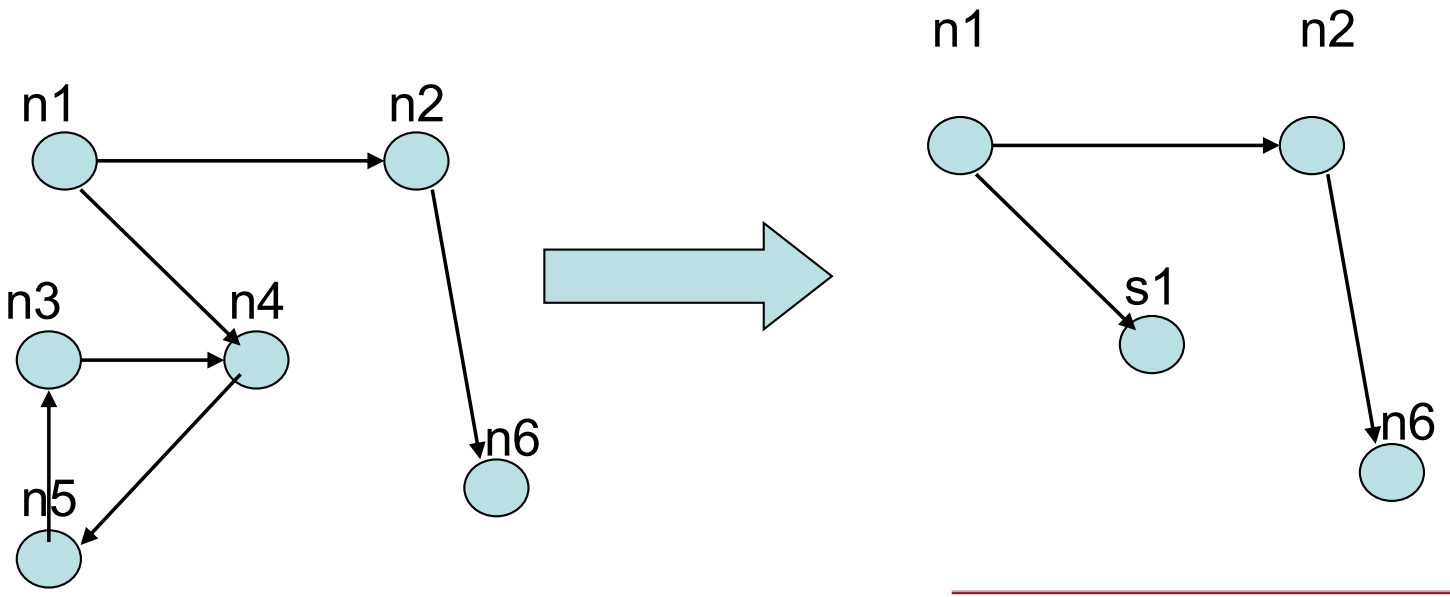
3-connected: iff a path goes from ni to nj and a path goes from nj to ni.

- **Strong Components（强分图）**

  A strong component of a directed graph is a maximal set of 3-connected nodes.

For example:

Strong Components are {n3,n4,n5}，we could removing loops and isolated nodes.



The large number of execution paths comes from nested loops. Condensation graphs eliminate loops (or at least condense them down to a single node); therefore, we can use this as a strategy to simplify situations that otherwise are computationally untenable.

- **Program Graphs（omit）**

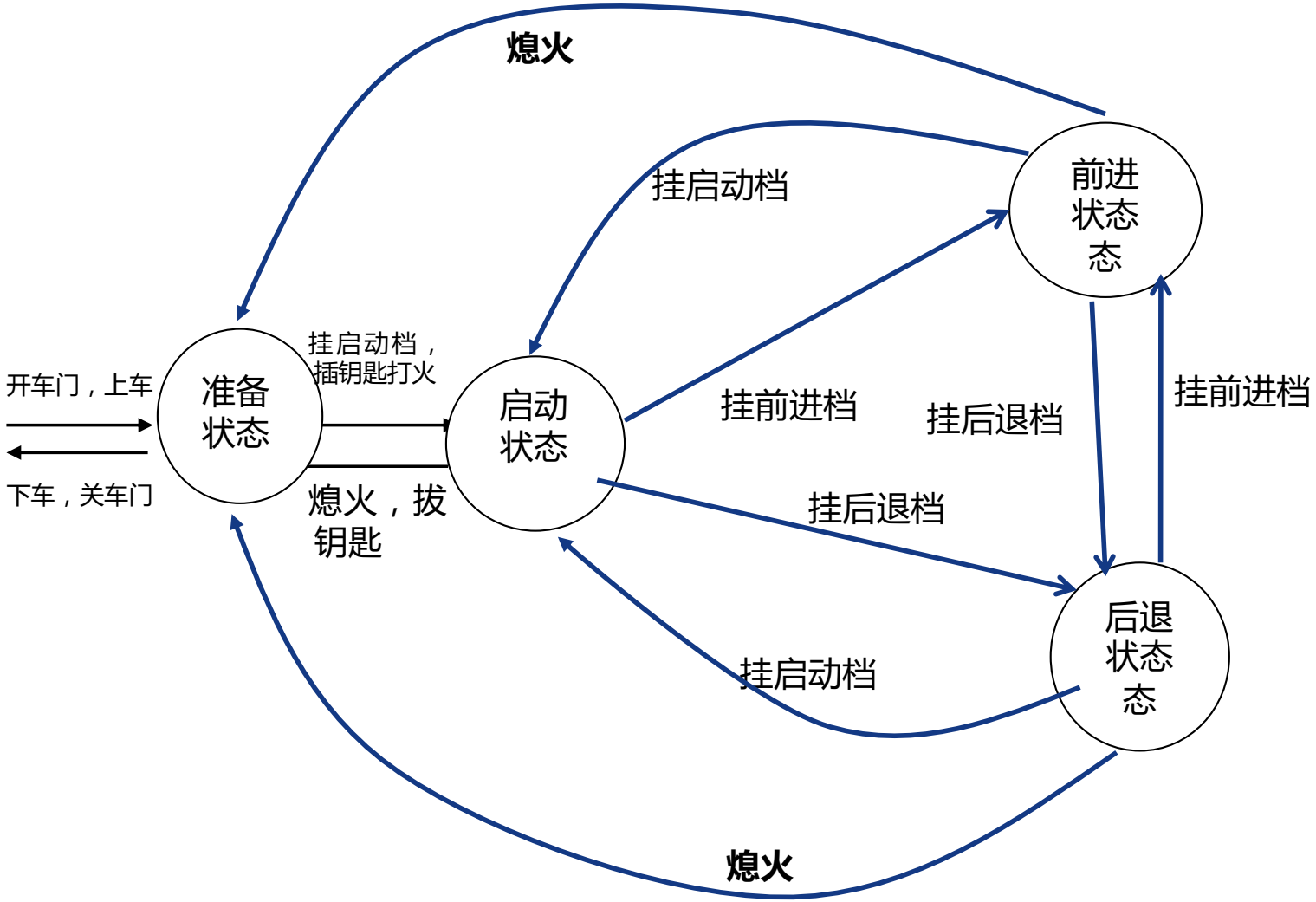
2.7 Finite State Machines

有限（穷）状态机是形式化描述的有效方法，在计算机调度、管理中得到了广泛的应用。

我们通过一个简单的例子来说明有穷状态机的基本概念。

司机驾驶汽车，有以下状态。打开车门；车挡在启动档，插钥匙，打火发动汽车；挂前进档，汽车前行；挂后退档，汽车后退；挂启动档，熄火，拔钥匙；下车关门。

当然，油门和刹车也是两个重要的动作，一个是为了提速，一个是为了减速，但它们都不影响汽车的状态，这里予以简化。空档是一个过渡状态，为了简洁，也予以简化。


熄火
挂启动档
前进
状态
态
挂启动档，
插钥匙打火
开车门，上车
准备
状态
启动
状态
挂前进档
挂后退档
挂前进档
下车，关车门
熄火，拔
钥匙
挂后退档
后退
状态
态
挂启动档
熄火

状态转换图在实际设计中往往被状态转换表所代替，下表是汽车驾驶的状态转换表。

当前状态 / 状态 / 执行动作	准备状态	启动状态	前进状态	后退状态
（挂启动档）打火	启动状态	启动状态	无效	无效
挂前进档	准备状态	前进状态		前进状态
挂后退档	准备状态	后退状态	后退状态	
挂启动挡	准备状态	启动状态	启动状态	启动状态
熄火	准备状态	准备状态	准备状态	准备状态

汽车驾驶的状态转换表

从上面的例子我们可以看到，一个有穷状态机包括以下5个部分：状态集J、输入（动作）集K、状态转换函数（转换表）T、初始态S和终止态F。对于上面的例子有穷状态机的各个部分如下：

- 状态集J：{准备状态，启动状态，前进状态，后退状态}；
- 输入集K：{打火，熄火，挂前进档，挂后退档，挂启动档}；
- 转换函数T：如表13-1所示；
- 初始态S：{准备状态}；
- 终止态F：{准备状态}；

由上面的说明，我们可以将有穷状态机表示为一个5元组（J，K，T，S，F），其中：

- J 是一个有穷的非空状态集；
- K是一个有穷的非空输入集；
- T是一个从（J-F）×K到J的转换函数；
- S ∈J，是唯一的初始态；
- F J，是若干个终止态。

有穷状态机(FSM)在计算机系统中应用的十分广泛。如操作系统中的作业调度与管理，GUI图形用户界面的驱动等。

2.8 Petri Net 说明

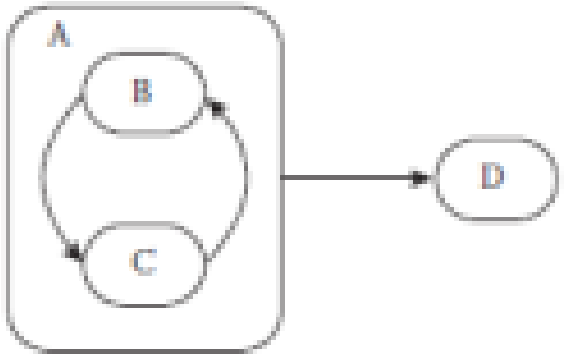
- Petri Net 的介绍与第11讲 交互测试密切关联，因此放在第11讲介绍。

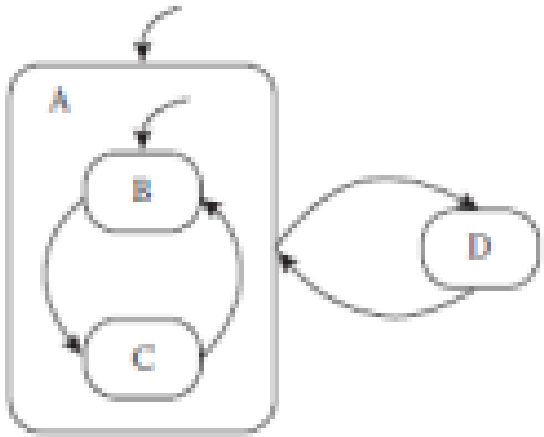
2.9 State Charts

StateChart notation: to devise a visual notation that combined the ability of Venn diagrams to express hierarchy and the ability of directed graphs to express connectedness (Harel, 1988).

If a state contains substates, as state A does, the edge “refers” to all substates.

Thus, the edge from A to D means that the transition can occur either from state B or from state C.





其它

对于软件测试而言，并不是仅仅需要本节介绍的数学知识。

形式化描述、数学验证、数据结构与算法等都是可能用到的数学。

在本课程中，将以软件测试的实践为主，不再过多介绍数学的知识。

作业说明

- 教材第二章需要自己阅读。
- 第四讲课程的内容，请大家预习
- Java
- Eclipse
- JUNIT
- EMMA

书上的7个例子程序需要自己熟悉！

Q&A