黑盒测试：

也称功能测试、数据驱动测试，它将被测软件看作一个打不开的黑盒，主要根据功能需求设计测试用例，进行测试。

概念：

黑盒测试是从一种从软件外部对软件实施的测试，也称功能测试或基于规格说明的测试。其基本观点是：任何程序都可以看作是从输入定义域到输出值域的映射，这种观点将被测程序看作一个打不开的黑盒，黑盒里面的内容(实现)是完全不知道的，只知道软件要做什么。因无法看到盒子中的内容，所以不知道软件是如何实现的，也不关心黑盒里面的结构，只关心软件的输入数据和输出结果。

检测软件功能能否按照需求规格说明书的规定正常工作，是否有功能遗漏；

检测是否有人机交互错误，是否有数据结构和外部数据库访问错误，是否能恰当地接收数据并保持外部信息（如数据库或文件）等的完整性；

检测行为、性能等特性是否满足要求等； 检测程序初始化和终止方面的错误等。

优点：

① 与软件具体实现无关，如果软件实现发生了变化，测试用例仍可用；

② 设计黑盒测试用例可以和软件实现同时进行，因此可压缩项目总开发时间。

黑盒测试常用方法

1. 等价类划分
2. 边界值分析
3. 因果图
4. 决策表分析

等价类划分

完全不考虑程序的内部结构，只根据程序规格说明书对输入范围进行划分，把所有可能的输入数据，即程序输入域划分为若干个互不相交的子集，称为等价类，然后从每个等价类中选取少数具有代表性的数据作为测试用例，进行测试。

划分原则：区间、数值、数值集合、限制条件或规则、细分等价类

边界值分析

边界值和等价类密切相关，输入等价类和输出等价类的边界是要着重测试的边界情况。在等价类的划分过程中产生了许多等价类边界。边界是最容易出错的地方，所以，从等价类中选取测试数据时应该关注边界值。

在等价类划分基础上进行边界值分析测试的基本思想是，选取正好等于、刚刚大于或刚刚小于等价类边界的值作为测试数据，而不是选取等价类中的典型值或任意值做为测试数据。

对于一个 n 变量的程序，边界值分析测试会产生 4n+1 个测试用例。

因果图

（1）确定软件规格中的原因和结果。分析规格说明中哪些是原因（即输入条件或输入条件的等价类），哪些是结果（即输出条件），并给每个原因和结果赋予一个标识符。

（2）确定原因和结果之间的逻辑关系。分析软件规格说明中的语义，找出原因与结果之间、原因与原因之间对应的关系，根据这些关系画出因果图。

（3）确定因果图中的各个约束。由于语法或环境的限制，有些原因与原因之间、原因与结果之间的组合情况不可能出现。为表明这些特殊情况，在因果图上用一些记号表明约束或限制条件。

（4）把因果图转换为决策表。

（5）根据决策表设计测试用例。

决策表分析

在所有的黑盒测试方法中，基于决策表的测试是最严格，最具有逻辑性的测试方法。

决策表是把作为条件的所有输入的各种组合值以及对应输出值都罗列出来而形成的表格。

它能够将复杂的问题按照各种可能的情况全部列举出来，简明并避免遗漏。因此，利用决策表能够设计出完整的测试用例集合。

步骤：

（1）列出所有的条件桩和动作桩。

（2）确定规则的个数。

（3）填入条件项。

（4）填入动作项，得到初始决策表。

（5）简化决策表，合并相似规则。

对于 n 个条件的决策表，相应有 2n 个规则

决策表合并原则：即若表中有两条以上规则具有相同的动作，并且在条件项之间存在极为相似的关系，便可以合并。

白盒测试：

也称结构测试或逻辑驱动测试，它是知道产品内部工作过程，可通过测试来检测产品内部动作是否按照规格说明书的规定正常进行，按照程序内部的结构测试程序，检验程序中的每条通路是否都有能按预定要求正确工作，而不顾它的功能。

白盒测试常用方法

本覆盖标准：逻辑驱动、循环、基路测试等，主要用于软件验证。

“白盒”法全面了解程序内部逻辑结构、对所有逻辑路径进行测试。

“白盒”法是穷举路径测试。在使用这一方案时，测试者必须检查程序的内部结构，从检查程序的逻辑着手，得出测试数据。贯穿程序的独立路径数是天文数字。但即使每条路径都测试了仍然可能有错误。

原因：

第一，穷举路径测试决不能查出程序违反了设计规范，即程序本身是个错误的程序。

第二，穷举路径测试不可能查出程序中因遗漏路径而出错。

第三，穷举路径测试可能发现不了一些与数据相关的错误。

白盒测试逻辑驱动方法

1. 语句覆盖
2. 判定覆盖
3. 条件覆盖
4. 判定/条件覆盖
5. 条件组合覆盖
6. 路径覆盖