黑盒测试 (black —box testing) 又称功能测试、 数据驱动测试或基于规范的测试。用这种方法进行测试时， 被测程序被当作看不见内部的黑盒。 在完全不考虑程序内部结构和内部特性的情况下， 测试者仅依据程序功能的需求规范考虑确定测试用例和推断测试结果的正确性。 因此黑盒测试是从用户观点出发的测试， 黑盒测试直观的想法就是既然程序被规定做某些事， 那我们就看看它是不是在任何情况下都做的对。完整的“ 任何情况 ”是无法验证的，为此黑盒测试也有一套产生测试用例的方法，以产生有限的测试用例而覆盖足够多的“ 任何情况 ”。由于黑盒测试不需要了解程序内部结构， 所以许多高层的测试如 确认测试、 系统测试、 验收测试 都采用黑盒测试。

黑盒测试首先是程序通常的 功能性测试 。要求：

每个软件特性必须被一个测试用例或一个被认可的异常所覆盖； 用数据类型和数据值的 最小 集测试；用一系列真实的数据类型和数据值运行，测试超负荷、饱和及其他“ 最坏情况 ”的结果；用 假想的数据类型和数据值 运行，测试排斥不规则输入的能力；对影响性能的关键模块，如基本算法、应测试单元性能 (包括精度、时间、容量等)。

不仅要考核“ 程序是否做了该做的 ？”还要考察“ 程序是否没做不该做的 2 ”同时还要考察程序在其他一些情况下是否正常。 这些情况包括数据类型和数据值的异常等等。 下述几种方法： (a) 等价类划分， (b) 因果图方法， (c) 边值分析法， (d) 猜错法， (e) 随机数法 ，就是从更广泛的角度来进行黑盒测试。 每一个方法都力图能涵盖更多的“ 任何情况 ”，但又各有长处，综合使用这些方法，会得到一个较好的测试用例集。

1. 等价类划分

等价类划分是一种典型的黑盒测试方法。 等价类是指某个输入域的集合。 它表示对揭露程序中的错误来说， 集合中的每个输入条件是等效的 。因此我们只要在一个集合中选取一个测试数据即可。 等价类划分的办法是把程序的输入域划分成若干等价类， 然后从每个部分中选取少数代表性数据当作测试用例。 这样就可使用少数测试用例检验程序在一大类情况下的反映。

在考虑等价类时，应该注意区别以下两种不同的情况：

有效等价类： 有效等价类指的是 对程序的规范是有意义的、 合理的输入数据所构成的集合。在具体问题中 ，有效等价类可以是一个，也可以是多个。

无效等价类：无效等价类指对程序的规范是不合理的或无意义的输入数据所构成的集合。对于具体的问题，无效等价类至少应有一个，也可能有多个。

确定等价类有以下几条原则：

如果输入条件规定了取值范围或值的个数， 则可确定一个有效等价类和两个无效等价类。例如，程序的规范中提到的输入条包括“ ,, 项数可以从 1 到 999,, ” ，则可取有效等价类为 “ l 考项数＜ 999 ”，无效等价类为“ 项数＜ l ，，及“ 项数＞ 999 ”。

输入条件规定了输入值的集合，或是规定了“ 必须如何 ”的条件，则可确定一个有效等价类和一个无效等价类。 如某程序涉及标识符， 其输入条件规定 “ 标识符应以字母开头 ,, ” 则“ 以字母开头者 ”作为有效等价类，“ 以非字母开头 ”作为无效等价类。

如果我们确知， 已划分的等价类中各元素在程序中的处理方式是不同的， 则应将此等价类进一步划分成更小等价类。

输入条件　有效等价类　无效等价类

○ ○ ○ ○ ○ ○

○ ○ ○ ○ ○ ○　　○ ○ ○ ○ ○ ○

○ ○ ○ ○ ○ ○　　○ ○ ○ ○ ○ ○

○ ○ ○ ○ ○ ○

根据已列出的等价类表，按以下步骤确定测试用例：

为每个等价类规定一个唯一的编号；

设计一个测试用例，　使其尽可能多地覆盖尚未覆盖的有效等价类。　重复这一步，最后使得所有有效等价类均被测试用例所覆盖；

设计一个新的测试用例，　使其只覆盖一个无效等价类。　重复这一步，　使所有无效等价类均被覆盖。　这里强调每次只覆盖一个无效等价类。　这是因为一个测试用例中如果含有多个缺陷，有可能在测试中只发现其中的一个，另一些被忽视。

等价类划分法能够全面、　系统地考虑黑盒测试的测试用例设计问题，　但是没有注意选用一些“ 高效的 ”、　“ 有针对性的 ” 测试用例。后面介绍的边值分析法可以弥补这一缺点。

2. 因果图

等价类划分法并没有考虑到输入情况的各种组合。　这样虽然各个输入条件单独可能出错的情况已经看到了，　但多个输入情况组合起来可能出错的情况却被忽略。采用因果图方法能帮助我们按一定步骤选择一组高效的测试用例，　同时，还能为我们指出程序规范的描述中存在什么问题。

利用因果图导出测试用例需要经过以下几个步骤：

分析程序规范的描述中哪些是原因，　哪些是结果。原因常常是输入条件或是输入条件的等价类。结果是输出条件。

分析程序规范的描述中语义的内容，　并将其表示成连接各个原因与各个结果的“ 因果图 ”。

由于语法或环境的限制，　有些原因和结果的组合情况是不可能出现的。　为表明这些特定的情况，　在因果图上使用持殊的符号标明约束条件。　把因果图转换成判定表。把判定表的每一列写成一个测试用例。

3. 边值分析法

边值分析法是列出单元功能、　输入、状态及控制的合法边界值和非法边界值，设计测试用例，包含全部边界值的方法。典型地包括　IF 语句中的判别值，定义域、值域边界，空或畸形输入，末受控状态等。边值分析法不是一类找一个例子的方法，而是以边界情况的处理作为主要目标专门设计测试用例的方法。另外，边值分析不仅考查输入的边值，　也要考虑输出的边值。　这是从人们的经验得出的一种有效方法。　人们发现许多软件错误只是在下标、　数据结构和标量值的边界值及其上、下出现，运行这个区域的测试用例发现错误的概率很高。

用边值分析法设计测试用例时，有以下几条原则：

如果输入条件规定了取值范围，　或是规定了值的个数，　则应以该范围的边界内及刚刚超出范围的边界外的值，　或是分别对最大、　最小及稍小于最小、　稍大于最大个数作为测试用例。如有规范“ 某文件可包含　l 至 255 ” 个记录 ,, “ ，则测试用例可选　1 和 255 及 0 和 256 等。

针对规范的每个输出条件使用原则〔　a〕。

如果程序规范中提到的输入或输出域是个有序的集合 (如顺序文件、表格等) 就应注意选取有序集的第一个和最后一个元素作为测试用例。

分析规范，尽可能找出可能的边界条件。 一个典型的边值分析例子是三角形分类程序。选取 a，b，c 构成三角形三边，“任意两边之和大于第三边”为边界条件。边值分析相等价类划分侧重不同， 对等价类划分是一个补充。 如上述三角形问题，选取 a＝3，b＝4，c＝5，a＝2，b＝4，c＝7 则覆盖有效和无效等价类。如果能在等价类划分中注入边值分析的思想。 在每个等价类中不只选取一个覆盖用例，而是进而选取该等价类的边界值等价类划分法将更有效， 最后可以用边值分析法再补充一些测试用例。

4. 猜错法

猜错法在很大程度上是凭经验进行的， 是凭人们对过去所作的测试工作结果的分析，对所揭示的缺陷的规律性作直觉的推测来发现缺陷的。

一个采用两分法的检索程序，典型地可以列出下面几种测试情况：

被检索的表只有一项或为空表；

表的项数恰好是 2 的幂次；

表的项数比 2 的幂次多 1 等。

猜错法充分发挥人的经验， 在一个测试小组中集思广益， 方便实用， 特别在软件测试基础较差的情况下，很好地组织测试小组 (也可以有外来人员) 进行错误猜测，是有效的测试方法。

5. 随机数法

即测试用例的参数是随机数。 它可以自动生成， 因此自动化程度高。 使用大量随机测试用例测试通过的程序会提高用户对程序的信心。 但其关键在于随机数的规律是否符合使用实际。