Software Testing Problem

- The input domain of a program is its all possible inputs.
- Identify a subset T of the input domain so that the executions of the program with T reveals all bugs.
- In practice, you are likely to select a subset of T such that it reveals as many errors as possible within the testing budget.

软件测试问题

- 一个程序的输入域是其所有可能的输入。
- 确定输入域的一个子集 T，使得程序在 T 上的执行能够揭示所有缺陷。
- 在实践中，你可能会选择 T 的一个子集，使其在测试预算范围内尽可能多地发现错误。

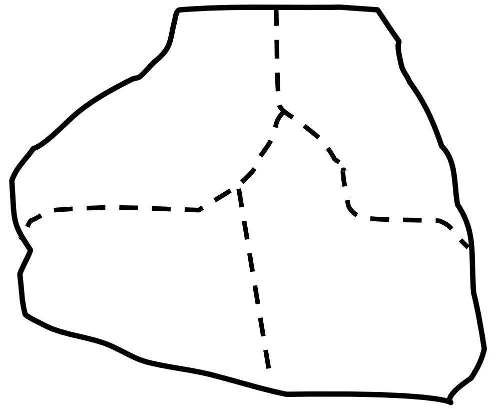
Equivalence Class Testing

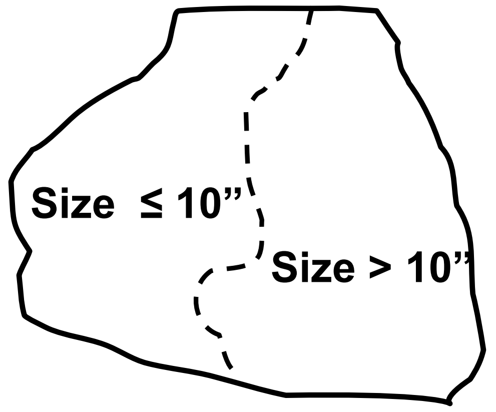
- The variable domain is partitioned into disjoint subsets.
- The system behaviour within each subset should be consistent i.e. each point in the subset shall result in the same behaviour of the program.
- Choose one test case (data point) from each sub-set
- Equivalence of outputs could be considered as well and could be used to refine the equivalence partitions of the input space e.g. if two different inputs of program cause outputs that belong to different equivalence classes of outputs, these inputs should be in different equivalence classes of input values, too
- Boundary and special values tests are based on the assumptions that bugs are likely when input or state values are at or very near to minimum or maximum.

等价类测试

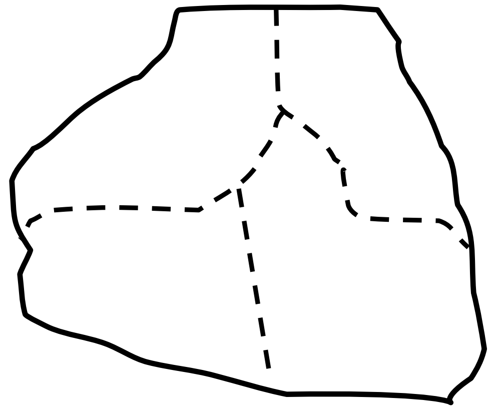
- 变量域被划分为互不相交的子集。
- 每个子集内的系统行为应保持一致，即子集中的每个点都应导致程序产生相同的行为。
- 从每个子集中选择一个测试用例（数据点）
- 也可以考虑输出的等价性，并可用于细化输入空间的等价划分；例如，如果程序的两个不同输入导致属于不同输出等价类的输出，则这些输入也应属于不同的输入值等价类。
- 边界值和特殊值测试基于以下假设：当输入或状态值处于最小值或最大值附近时，更容易出现缺陷。

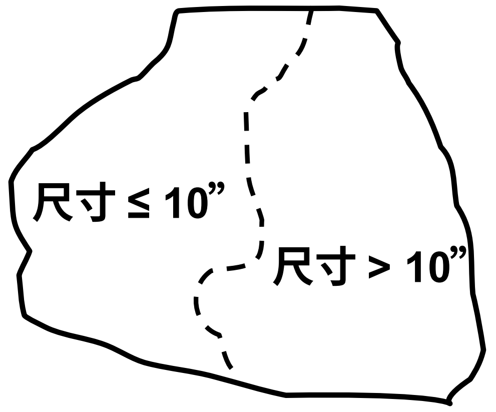
Partitioning





分区





Exhaustive Testing

How long would it take (approximately) to test exhaustively the following program?

- `int sum(int a, int b) { return a + b; }`
- $2^{32} \times 2^{32} = 2^{64} \sim 10^{19}$ tests
- If each test takes 10^{-9} secs, it is still 10^{10} secs

穷尽测试

彻底测试以下程序大约需要多长时间?

- `int sum(int a, int b) { return a + b; }`
- $2^{32} \times 2^{32} = 2^{64} \sim 10^{19}$ 次测试
- 如果每次测试耗时 10^{-9} 秒，仍然需要 10^{10} 秒

Weak Normal Equivalence testing

1. Assumes the ‘single fault’ or “independence of input variables.”
   - e.g. If there are 2 input variables, these input variables are independent of each other.
2. Partition the test cases of each input variable separately into one of the different equivalent classes.
3. Choose the test case from each of the equivalence classes for each input variable independently of the other input variable

弱正常等价类测试

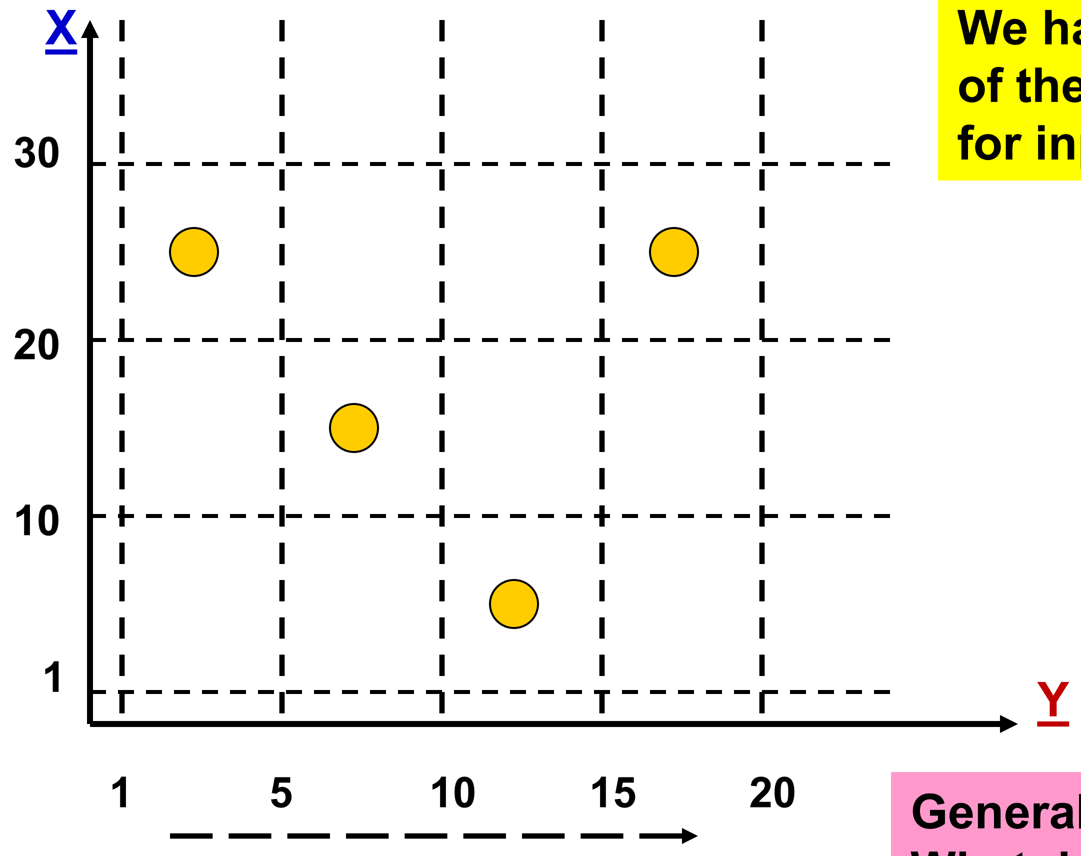
1. 假设存在“单一故障”或“输入变量的独立性”。
   - 例如，如果有两个输入变量，则这些输入变量彼此相互独立。
2. 将每个输入变量的测试用例分别划分为不同的等价类之一。
3. 从每个等价类中选择测试用例 针对每个输入变量独立于其他输入变量

Example of : Weak Normal Equivalence testing

示例：弱正常等价类测试

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30
and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11 to15; and 16 to 20

假设输入X的等价划分为：1 到 10；11 到 20，21 到 30，以及输入 Y 的等价划分为：
1 到 5；6 到 10；11 到 15；以及16 到 20

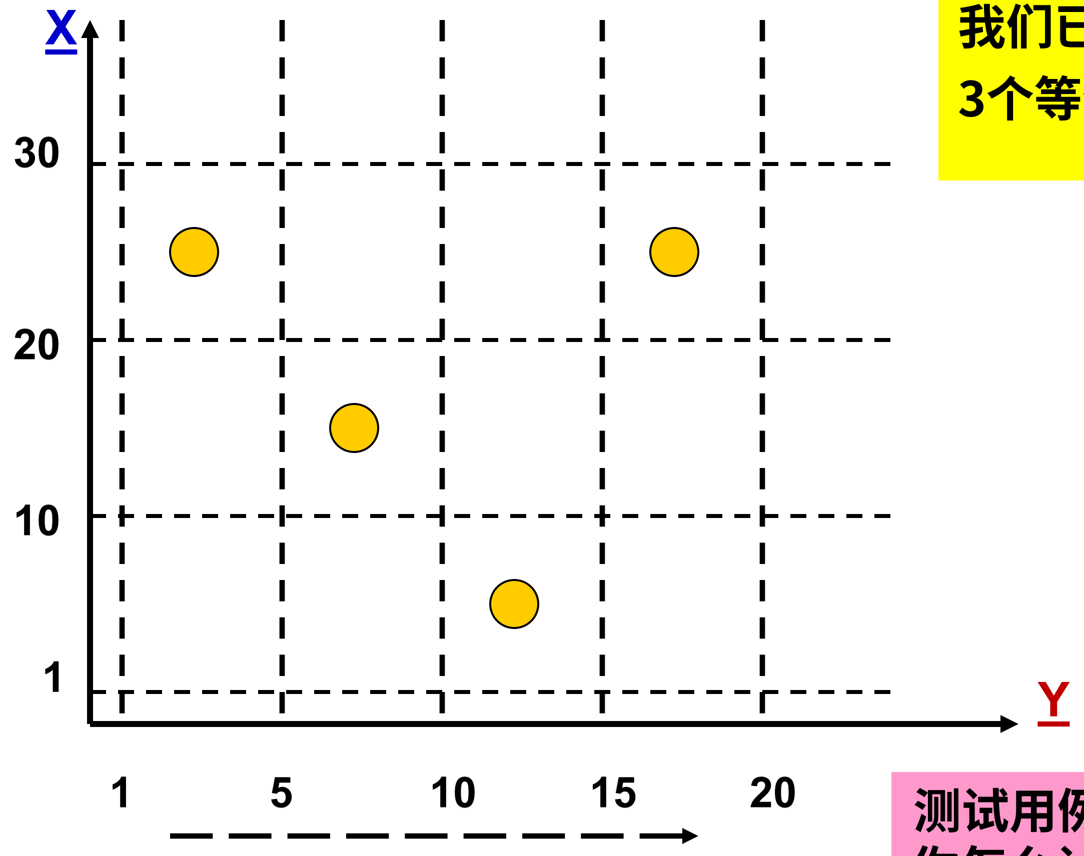


We have covered everyone of the 3 equivalence classes for input X.

For (x, y)
we have:
(24, 2)
(15, 8)
(4, 13)
(23, 17)

We have covered each of the 4 equivalence classes for input Y.

General rule for # of test cases?
What do you think?
# of partitions of the largest set?



我们已经覆盖了输入X的全部3个等价类。

对于 (x, y)
我们有：
(24, 2)
(15，8)
(4, 13)
(23, 17)

我们已经覆盖了全部4个输入Y的等价类。 –

测试用例数量的一般规则是什么?
你怎么认为?
最大集合的划分数量?

Strong Normal Equivalence testing

- This is the same as the weak normal equivalence testing except for

  “multiple fault assumption”

  or

  “dependence among the inputs”

- All the combinations of equivalence classes of the variables must be included.

强正常等价类测试

- 这与弱正常等价类测试相同，只是采用了“多重故障假设”

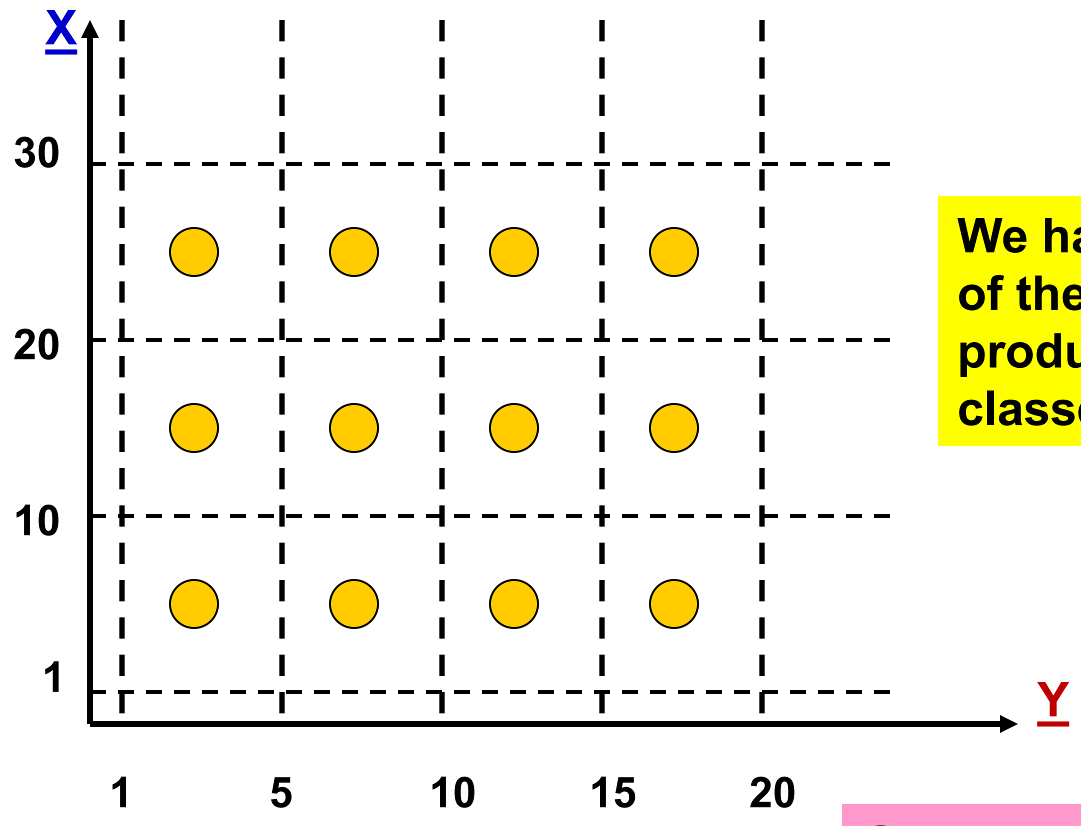
  or

  “输入之间的依赖关系”

- 所有变量的等价类的所有组合 必须全部包含。

Example of : Strong Normal Equivalence testing

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11;15; and 16 to 20

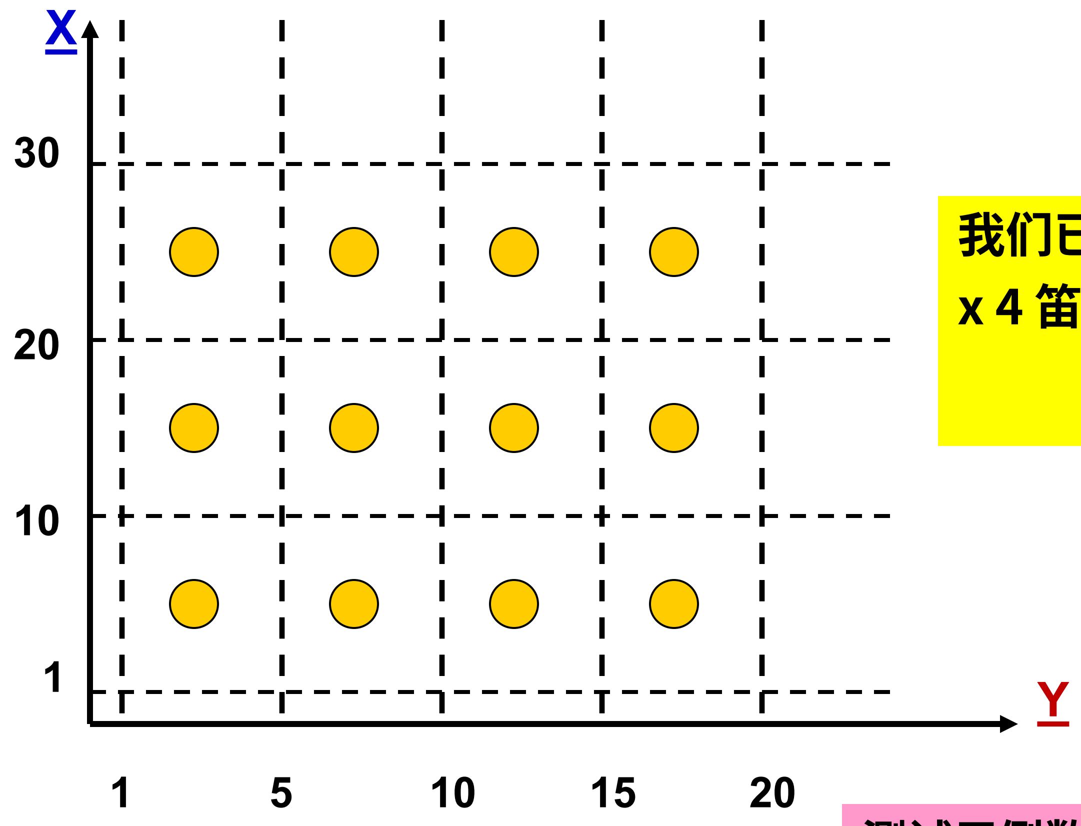


We have covered everyone of the 3 x 4 Cartesian product of equivalence classes

General rule for # of test cases? What do you think?

示例： 强健正常等价性测试

假设输入X的等价类划分为：1 到 10；11 到 20；21 到 30，而输入Y的等价类划分为：1 到 5；6 到 10；11 到 15；以及 16 到 20



我们已经覆盖了等价类的 3 x 4 笛卡尔积中的每一个情况

测试用例数量的一般规则是什么？你认为呢？

Weak Robust Equivalence testing

- Up to now we have only considered partitioning the ***valid*** input space.

- “Weak robust” is similar to “weak normal” equivalence test except that the ***invalid*** input variables are now considered.

A note about considering invalid input is that there may not be any definition “specified” for the various, different invalid inputs - - - making definition of the output for these invalid inputs a bit difficult at times. *(but fertile ground for testing*)

弱健壮等价测试

- 到目前为止，我们仅考虑了划分有效输入空间。

- “弱健壮”类似于“弱一般”等价测试，只是现在还考虑了无效输入变量。

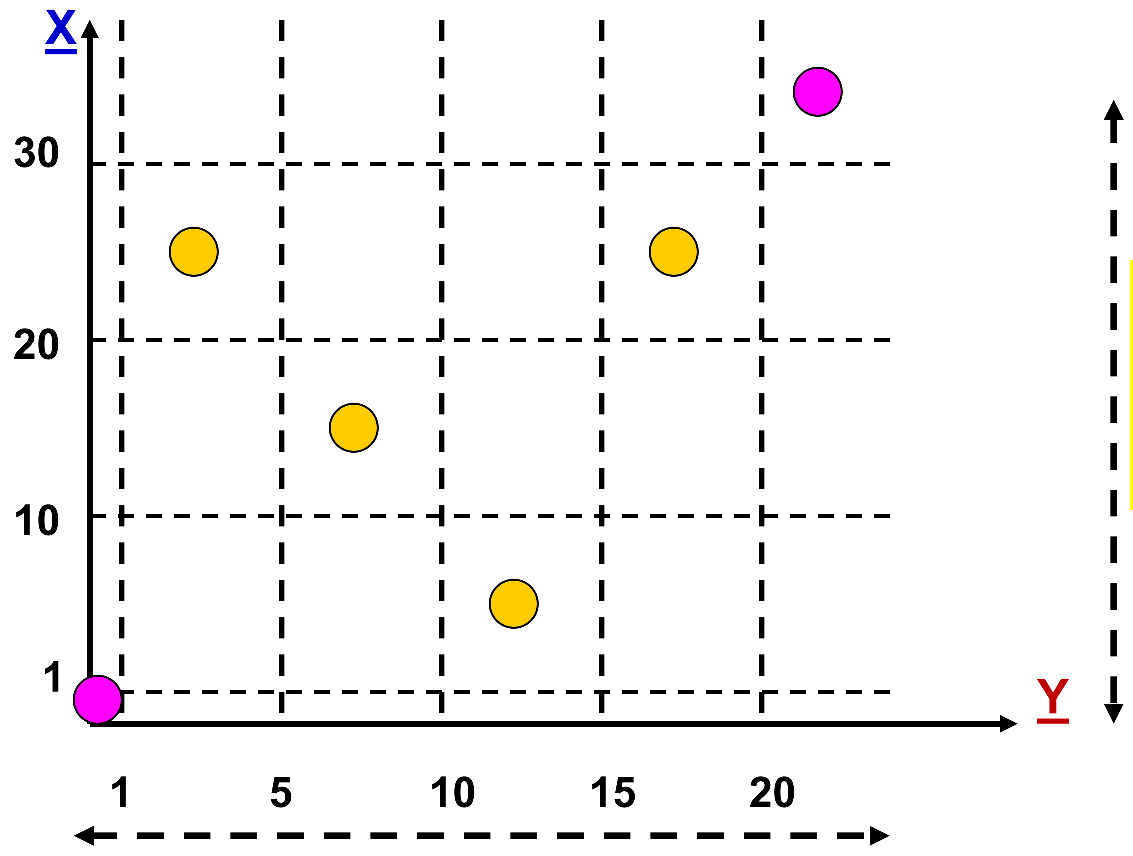
关于考虑无效输入的一点说明是，可能并未对各种不同的无效输入明确定义“指定”行为——这使得有时难以确定这些无效输入对应的输出。（但这是测试的有利切入点）

Example of : Weak Robust Equivalence testing

示例： 弱健壮等价类测试

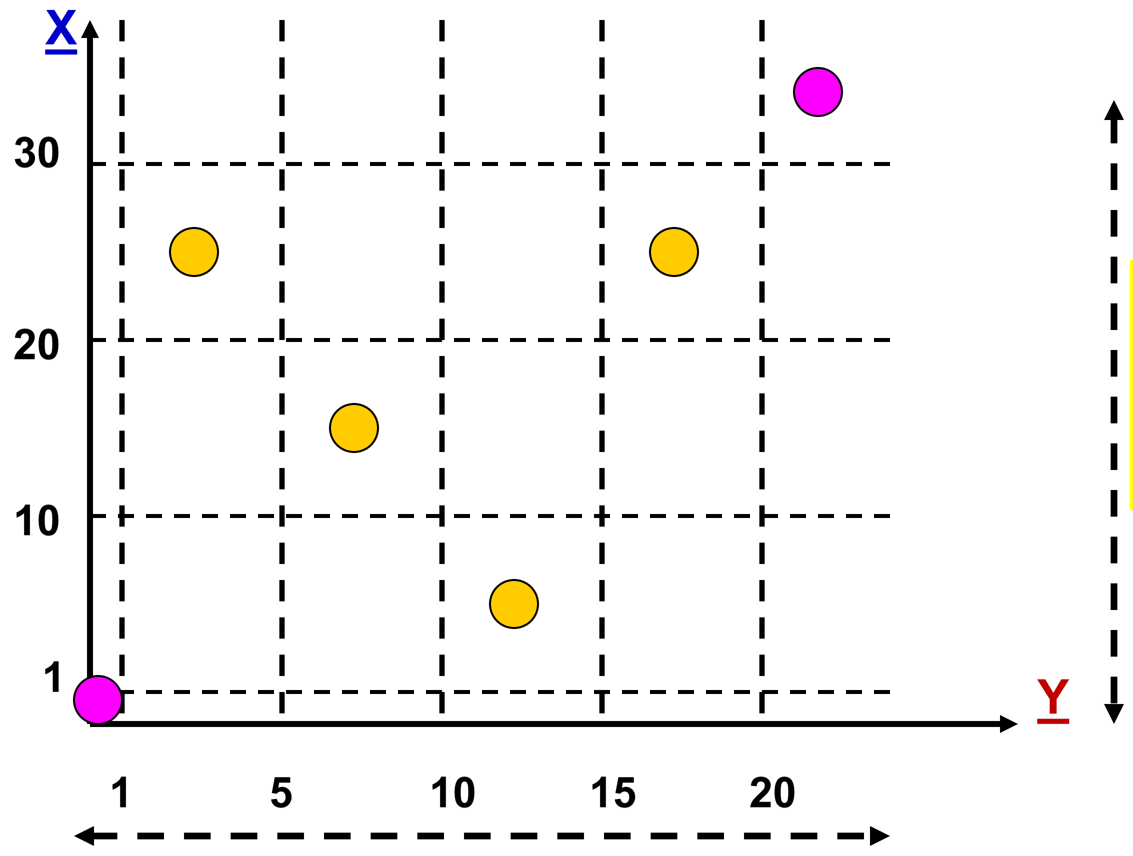
Assume the equivalence partitioning of input X is 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is 1 to 5; 6 to 10; 11;15; and 16 to 20

假设输入X的等价类划分为 1到10；11到20，21到30，且输入Y的等价类划分为 1到5；6到10；11到15；以及16到20输入 Y



We have covered everyone of the 5 equivalence classes for input X.

We have covered each of the 6 equivalence classes for input Y.



我们已经覆盖了输入X的全部5个等价类。

我们已经覆盖了每个输入Y的6个_等价类。

Strong Robust Equivalence testing

- Does not assume “single fault” - - - assumes dependency of input variables
- “Strong robust” is similar to “strong normal” equivalence test except that the invalid input variables are now considered.

强健壮等价性测试

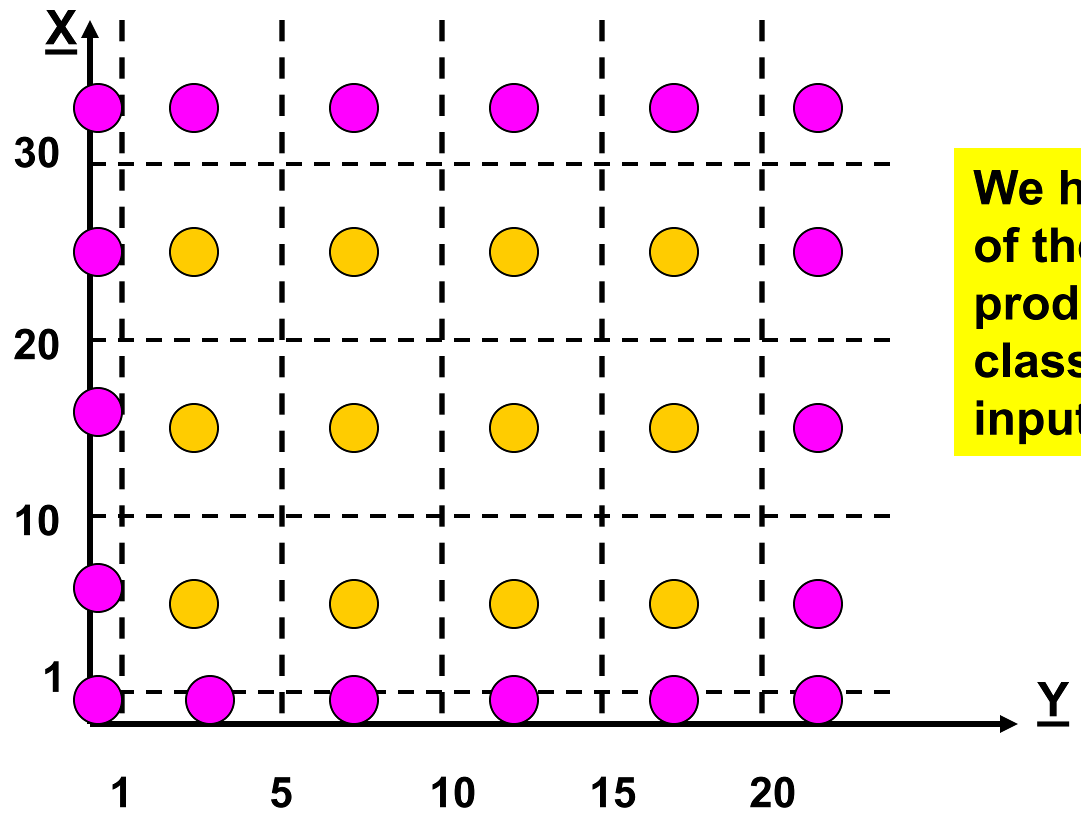
- 不假设“单故障”——假设依赖性输入变量的依赖性
- “强健壮”与“强正常”等价性测试类似，不同之处在于现在考虑了无效输入变量。

Example of : Strong Robust Equivalence testing

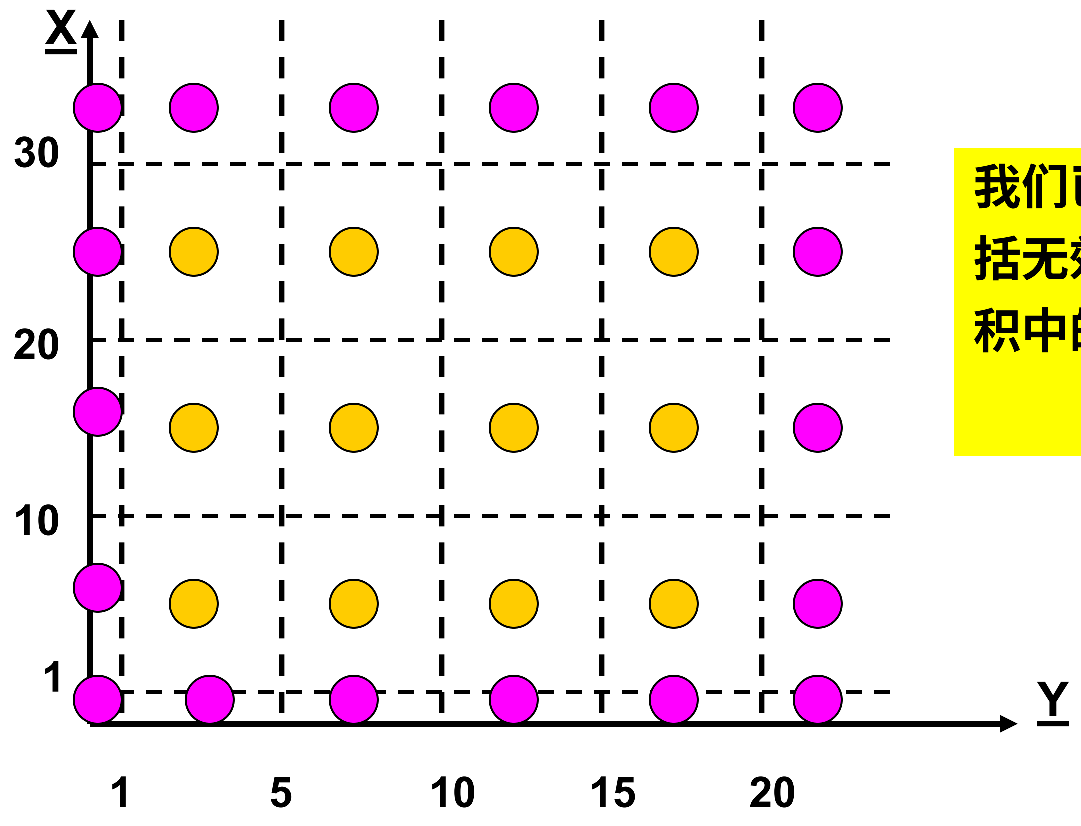
示例： 强健壮等价类测试

Assume the equivalence partitioning of input X is: 1 to 10; 11 to 20, 21 to 30 and the equivalence partitioning of input Y is: 1 to 5; 6 to 10; 11;15; and 16 to 20

假设输入X的等价划分为：1到10；11到20，21到30，输入Y的等价划分为：1到5；6到10；11到15；以及16到20



We have covered everyone of the 5 x 6 Cartesian product of equivalence classes (including invalid inputs)



我们已经涵盖了等价类（包括无效输入）的 5 x 6 笛卡尔积中的每一个情况

Next Day Program example

- After receiving Day of the month, Month and the year as three separate inputs, program produces the next on the calendar.

- One possible set of partitions are:
  - Day : 1 through 31 days
  - Month: 1 through 12
  - Year: 0001 through 3000

- In the above case, if we assume that day of the month, month and year values are independent and invalid values for the day of the month, month and year are prevented via input validation, for weak normal equivalence testing only needs 1 test case from each input. Example:
  - (year; month; day) : (2001, 9, 23)

- *Clearly lots of bugs could potentially be not exposed if only one potential input is considered in this case.*

- *What could be a better set of partitions?*

次日计划示例

- 在接收到“日”、“月”和“年”作为三个独立输入后，程序将生成日历上的下一个日期。

- 一 p可能的一组 p分区如下：
  - 日： 1 至 31 天
  - 月： 1 至 12
  - 年份： 0001 至 3000

- 在上述情况下，如果我们假设日期、月份和年份的取值相互独立，并且通过输入验证防止了日期、月份和年份中的无效值，那么对于弱正常等价类测试而言，每个输入只需一个测试用例。例如：
  - （年；月；日）：（2001，9，23）

- 显然，如果在此情况下仅考虑一个可能的输入，则可能会遗漏大量潜在的缺陷。

- 什么样的分区组合会更好?

Next Day Program example (cont.)

- Day: 1 through 28
- Day :29
- Day: 30
- Day: 31

- Month: those that <u>have 31</u> days or {1,3,5,7,8,10,12}
- Month: those that <u>have 30</u> days or {4,6,9,11}
- Month: that has <u>less than 30</u> days or {2}

- Year: leap years between 0001 and 3000
- Year: non-leap years between 0001 and 3000

i.e. 4 partitions of Day, 3 partitions of month and 2 partitions of year.

If we assume that inputs are independent of each other (weak normal), what faults could be missed?

If we assume the inputs are interdependent (weak robust), what inputs constitute redundant testing?

次日计划示例（续）

- 日期：1 至 28 日
- 日期：29 日
- 日期：30 日
- 日期：31 日

- 月份：包含 31 天的月份或 {1,3,5,7,8,10,12}
- 月份：有30天的月份或{4,6,9,11}
- 月份：少于30天的月份或{2}

- 年份：0001年至3000年之间的闰年
- 年份：0001年至3000年之间的平年

即天分为4个部分，月分为3个部分，年分为2个部分。

如果我们假设输入彼此独立（弱正常），可能会遗漏哪些故障?

如果我们假设输入之间是相互依赖的（弱健壮性），那么哪些输入构成了冗余测试?

Next Day Program example (cont.)

If we assume that inputs are independent of each other i.e. weak normal

1. How many test cases ?
2. What errors/faults/bugs could be missed out?

   Example Inputs:
   (Leap Year, Month, Day)
   (leap year, 1, 8)
   (leap year, 2, 28)
   (non-leap year, 3, 31)
   (non-leap year, 6, 29)

If we assume the inputs are interdependent i.e. weak robust

1. How many test cases?
2. Is redundant testing still possible?

第二天程序示例（续）

如果我们假设输入彼此相互独立，即弱正常情况

1. 有多少个测试用例?
2. 可能遗漏哪些错误/故障/缺陷?

   示例输入：
   （闰年，月份，日期）
   （闰年，1，8）
   （闰年，2，28）
   （非闰年，3， 31）
   （非闰年，6，29）

如果我们假设输入是相互依赖的，即较弱的情况 robust

1. 有多少个测试用例?
2. 仍然可能存在冗余测试吗?

Next Day Problem example (cont.)

We should have (2 years x 3 months x 4 days) = 24 test cases

(leap year, 10, 5)	*(non-leap year, 10, 5)*
(leap year, 10, 30)	*(non-leap year, 10, 30)*
(leap year, 10, 31)	*(non-leap year, 10, 31)*
(leap year, 10, 29)	*(non-leap year, 10, 29)*
(leap year, 6, 5)	*(non-leap year, 6, 5)*
(leap year, 6, 30)	*(non-leap year, 6, 30)*
(leap year, 6, 31)	*(non-leap year, 6, 31)*
(leap year, 6,29)	*(non-leap year, 6, 29)*
(leap year, 2, 5)	*(non-leap year, 2, 5)*
(leap year, 2, 30)	*(non-leap year, 2, 30)*
(leap year, 2, 31)	*(non-leap year, 2, 31)*
(leap year, 2, 29)	*(non-leap year, 2, 29)*

Can we still do better?

第二天问题示例（续）

我们应该有 (2 年 × 3 月 × 4 天) = 24 个测试用例

(闰年, 10, 5)	*(非闰年, 10, 5)*
(闰年, 10, 30)	*(非闰年, 10, 30)*
(闰年, 10, 31)	*(非闰年, 10, 31)*
(闰年, 10, 29)	*(非闰年, 10, 29)*
(闰年, 6, 5)	*(非闰年, 6, 5)*
(闰年, 6, 30)	*(非闰年, 6, 30)*
(闰年, 6, 31)	*(非闰年, 6, 31)*
(闰年, 6,29)	*(非闰年, 6, 29)*
(闰年, 2, 5)	*(非闰年, 2, 5)*
(闰年, 2, 30)	*(非闰年, 2, 30)*
(闰年, 2, 31)	*(非闰年, 2, 31)*
（闰年，*2*，*29*）	（非闰年，*2*，*29*）

我们还能做得更好吗?

Next Day Problem example (cont.)

– M1 = {month : 1 .. 12 | days(month) = 30 }
– M2 = {month : 1 .. 12 | days(month) = 31 ∧ month ≠ 12 }
– M3 = {month : {12} }
– M4 = {month : {2} }
– D1 = {day : 1 .. 27}
– D2 = {day : {28} }
– D3 = {day : {29} }
– D4 = {day : {30} }
– D5 = {day : {31} }
– Y1 = leap year
– Y2 = common year

Handles end of month and year better.
Year 2000 could be a separate partition for Y2K bugs.

次日问题示例（续）

– M1 = {month : 1 .. 12 | days(month) = 30 }
– M2 = {month : 1 .. 12 | days(month) = 31 ∧ month ≠ 12 }
– M3 = {month : {12} }
– M4 = {月 : {2} }
– D1 = {day : 1 .. 27}
– D2 = {day : {28} }
– D3 = {day : {29} }
– D4 = {day : {30} }
– D5 = {day : {31} }
– Y1 = 闰年
– Y2 = 平年

更好地处理月末和年末情况。2000 年可以单独划分为一个分区，以应对 Y2K 漏洞。

(In Class Assignment) Search Clients

Determine the equivalence partitions for the keywords, time window and location area based search of the photos from the folder in the external storage by the Photo Gallery app. Assume the following:

•user specifies a time window by picking a StartDate and an EndDate with the date range from MinDate to MaxDate

•for location based search, the user specifies a search area defined by {[TopLeft.Lat, TopLeft.Long], [BottomRight.Lat, BottomRight.Long]}, bounded by {[90, -180], [-90, 180]}

•For keyword based search the user specifies a keyword of up to 10 printable characters and all printable characters (including escape characters) are treated equally

（课堂作业）搜索客户

确定照片库应用程序根据外部存储中文件夹内的照片的关键词、时间范围和位置区域进行搜索时的等价类划分。假设以下情况：

•用户通过选择开始日期和结束日期来指定时间窗口，日期范围从最小日期到最大日期

•对于基于位置的搜索，用户指定由{[左上纬度、左上经度], [右下纬度、右下经度]}定义的搜索区域，该区域由{[90, -180], [-90, 180]}限定

•对于基于关键词的搜索，用户指定最多10个可打印字符的关键词，所有可打印字符（包括转义字符）均被同等对待

(In Class Assignment) – Search Photos (cont.)

Time Window based search:

- Valid Equivalence Class: MinDate <= StartDate <= EndDate <= MaxDate.
- InValid Equivalence Classes: StartDate = null; EndDate = null; or EndDate < StartDate.

Location Area based search:

- Valid Equivalence Class: 90 >= TopLeft.Lat >= BottomRight.Lat >= -90 and -180 <= TopLeft.Long <= BottomRight.Long <= 180.
- InValid Equivalence Classes: At least one of TopLeft.Lat, TopLeft.Long, BottomRight.Left, BottomRight.Long is null; TopLeft.Lat < BottomRight.Lat; or TopLeft.Long > BottomRight.Long;
- Note: Additional equivalence classes e.g. search along a latitude, or search along a longitude could also be specified.

Keyword based search:

- Valid Equivalence Class: a string of up to 10 printable characters.
- InValid Equivalence Class: Null or empty string.

（课堂作业）—— 搜索照片（续）

基于时间窗口的搜索：

- 有效等价类：MinDate <= StartDate <= EndDate <=MaxDate。
- 无效等价类：StartDate = null；EndDate = null；或 EndDate < StartDate。

基于位置区域的搜索：

- 有效等价类：90 >= TopLeft.Lat >= BottomRight.Lat >= -90 且 -180 <= TopLeft.Long <= BottomRight.Long <= 180。
- 无效等价类：TopLeft.Lat、TopLeft.Long、BottomRight.Left、BottomRight.Long 中至少有一个为 null；TopLeft.Lat < BottomRight.Lat；或 TopLeft.Long > BottomRight.Long；
- 注意：也可以指定其他等价类，例如沿纬度搜索，或沿经度搜索。

基于关键字的搜索：

- 有效等价类：最多包含 10 个可打印字符的字符串。
- 无效等价类：空值或空字符串。

Independent Testing

- Programmers have a hard time believing they made a mistake
- a vested interest in not finding mistakes
- Design and programming are constructive tasks
- Testers must seek to break the Software

独立测试

- 程序员很难相信自己犯了错误
- 有既得利益而不愿发现错误
- 设计和编程是建设性的工作
- 测试人员必须设法破坏软件

Equivalence Testing

- Several tries may be required before the “right” equivalence relation is discovered
- Coverage reports can be used to know if more partitions needed
- Input validation can eliminate testing for invalid partitions.

等价类测试

- 在发现“正确”的等价关系之前，可能需要多次尝试
- 可以利用覆盖率报告来判断是否需要更多的划分
- 输入验证可以消除对无效划分的测试。