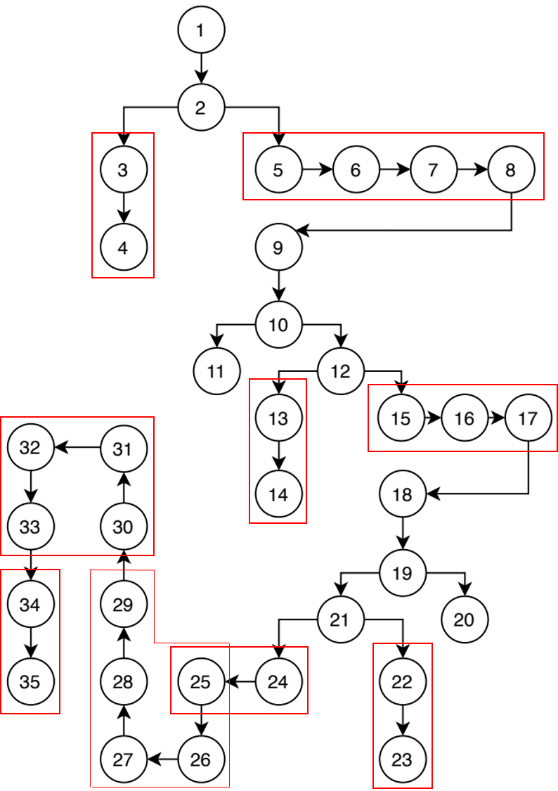
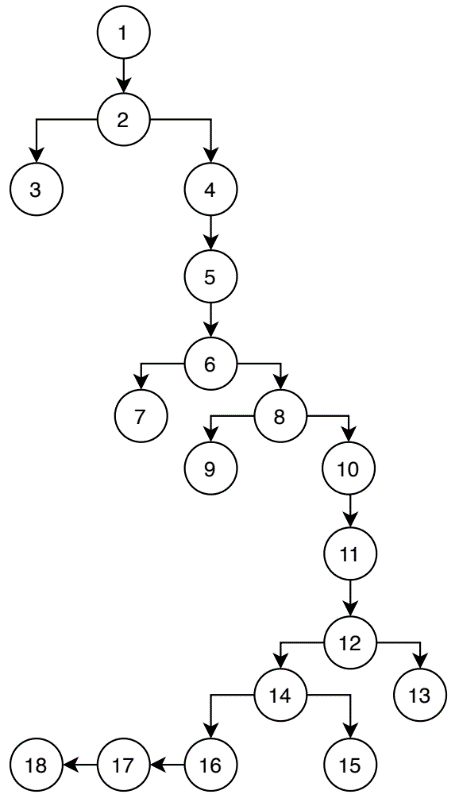
对于SellInfo和File模块，我们采用基于数据流的测试用例生成方法。SellInfo模块一共有四个函数，File模块一共有三个函数，我们选取其中的SellInfo. Query函数作为代表，详细介绍基于数据流的测试用例的生成过程。而其他函数的测试用例生成过程略去，具体材料我们放在附件之中。

需要注意的是，为了方便测试和展示，在本小节中，我们对所有的目标测试函数进行行号重排，即行号自函数头从1开始重新计数。

1. 绘制目标函数的程序图

在对SellInfo.Query函数进行行号重排后，我们可以画出对应的程序图。为了展示方便，且因为源代码中存在无意义的空行或括号单行，同时援引教材9.1.6中的方法，我们对原始的程序图进行节点合并与化简，得到最终可用的程序图（见图**？？？**）。

图**？？？** SellInfo.Query的原始程序图和简化程序图

1. 统计变量的定义/使用节点

在对程序图进行简化和绘制后，我们可以对SellInfo.Query中的所有变量的定义/使用节点进行统计，具体可以见表**？？？**。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量 | 定义节点  原始程序图 | 定义节点  简化程序图 | 使用节点  原始程序图 | 使用节点  简化程序图 |
| ctx | 1 | 1 | - | - |
| req | 1 | 1 | 2, 7 | 2, 4 |
| rsp | 1, 25, 26, 27, 28, 29, 30, 31, 32, 33 | 1, 16, 17 | - | - |
| info | 6 | 4 | 9, 16, 25, 26, 27, 28, 29 | 5, 10, 16 |
| err | 9, 18 | 5, 11 | 10, 12, 13, 19, 21, 22 | 6, 8, 9, 12, 14, 15 |
| good | 15 | 10 | 18, 30, 31, 32, 33 | 11, 17 |

表**？？？** SellInfo.Query的变量的定义/使用节点

1. 列出变量的定义-使用路径

在列出所有变量的定义/使用节点后，我们可以依据简化程序图的节点情况，依次列出所有变量的定义-使用路径，并且注明其是否为定义-清除路径，具体可见表**？？？**。同时，因为部分路径存在重合部分，因为我们对路径进行了整理，比如用<p1, 4>来代表p2的路径。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 变量 | 路径 | 完整路径 | 简化路径 | 是否为定义-清除路径 |
| req | p1 | 1, 2 | 1, 2 | Y |
| p2 | 1, 2, 4 | p1, 4 | Y |
| info | p3 | 4, 5 | 4, 5 | Y |
| p4 | 4, 5, 6, 8, 10 | p3, 6, 8, 10 | Y |
| p5 | 4, 5, 6, 8, 10, 11, 12, 14, 16 | p4, 11, 12, 14, 16 | Y |
| err | p6 | 5, 6 | 5, 6 | Y |
| p7 | 5, 6, 8 | P6, 8 | Y |
| p8 | 5, 6, 8, 9 | P7, 9 | Y |
| p9 | 5, 6, 8, 10, 11, 12 | P7, 10, p12 | N |
| p10 | 5, 6, 8, 10, 11, 12, 14 | P9, 14 | N |
| p11 | 5, 6, 8, 10, 11, 12, 14, 15 | P10, 15 | N |
| p12 | 11, 12 | 11, 12 | Y |
| p13 | 11, 12, 14 | p14, 14 | Y |
| p14 | 11, 12, 14, 15 | p15, 15 | Y |
| good | p15 | 10, 11 | 10, 11 | Y |
| p16 | 10, 11, 12, 14, 16, 17 | P15, 12, 14, 16, 17 | Y |

表**？？？** SellInfo.Query的变量的定义-使用路径

1. 选取测试覆盖指标，生成测试用例

在这一步，我们选用**全使用准则**作为测试覆盖指标，即选择每一个定义节点到所有使用节点以及其后续节点的定义-清除路径。而对于所有的定义-使用路径，依照表**？？**中的简化路径可知，我们可以用尽可能少的测试用例去覆盖尽可能多的目标路径，比如对于路径p8, p9, p10，我们只需要生成一个可以覆盖p10的测试用例即可。而具体的生成的测试用例，我们放在附件中予以展示，此处略去。

1. 编写代码，进行测试

步骤1~4就是基于数据流的测试用例生成过程，接下来我们可以编写代码，进行测试。值得一提的是，因为对于数据流测试的测试覆盖指标本身就是我们测试用例生成的依据，因此按照全使用准则，我们的覆盖率必然是100%，但是在其他覆盖率（比如条件覆盖率、分支覆盖率等）语境下，覆盖率会有所下降。

结束了对数据流测试的用例生成过程的介绍，接下来是整个作业过程中，我们对于数据流测试的感想：

相比于基于路径的测试方法，数据流测试方法的实践过程更加繁琐，一旦代码量增加，或者是出现循环、变量多处定义/使用，我们就很难用手工的方式来实现。而书上也提到，目前很少有针对数据流测试的商业软件，因此这种白盒测试方法似乎显得有点鸡肋。我们认为，基于数据流的测试方法应该是作为辅助的角色，帮助基于路径的测试方法，提供多样化的测试用例，提高测试覆盖率和效果。

最明显的例子就是SellInfo.Find，这个函数中针对变量req有连续10个独立的分支判断，导致总的定义-使用路径达到了条！虽然可以用归纳的方法和少量的用例将这些高度重合的路径全部覆盖，但是的确很难直接手写出所有的定义-使用路径。

因此，为了达到老师要求的95%的代码覆盖率（这里指的是总体的代码覆盖率，因此不采用全使用准则），在覆盖率不够的情况下，我们会在基于数据流测试的基础上，继续用基于DD路径的方法进行额外的用例生成，选择尽可能少的测试用例，使覆盖率达标。