哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2021 年春季学期

计算学部《软件构造》课程

Lab 2 实验报告

姓名	陈广焕
学号	1190501614
班号	1903006
电子邮件	2718458514@qq.com
手机号码	15778541719

目录

1 实验目标概述

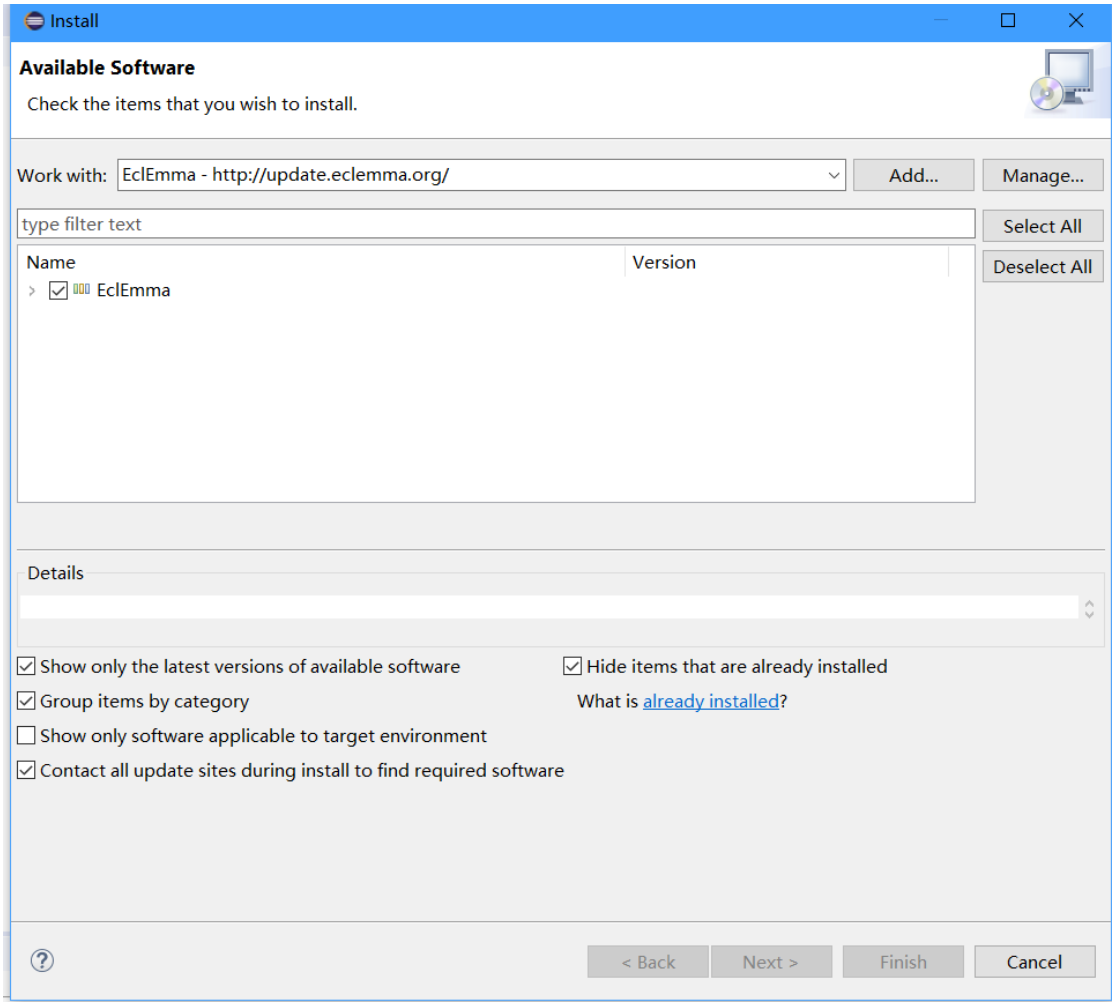
本次实验训练抽象数据类型（ADT）的设计、规约、测试，并使用面向对象编程（OOP）技术实现 ADT。具体来说:

- 针对给定的应用问题，从问题描述中识别所需的 ADT;
- 设计 ADT 规约（pre-condition、post-condition）并评估规约的质量;
- 根据 ADT 的规约设计测试用例;
- ADT 的泛型化;
- 根据规约设计 ADT 的多种不同的实现;针对每种实现,设计其表示(representation)、表示不变性（rep invariant)、抽象过程（abstraction function）
- 使用 OOP 实现 ADT，并判定表示不变性是否违反、各实现是否存在表示泄露（rep exposure)；
- 测试 ADT 的实现并评估测试的覆盖度;
- 使用 ADT 及其实现，为应用问题开发程序;
- 在测试代码中，能够写出 testing strategy 并据此设计测试用例。

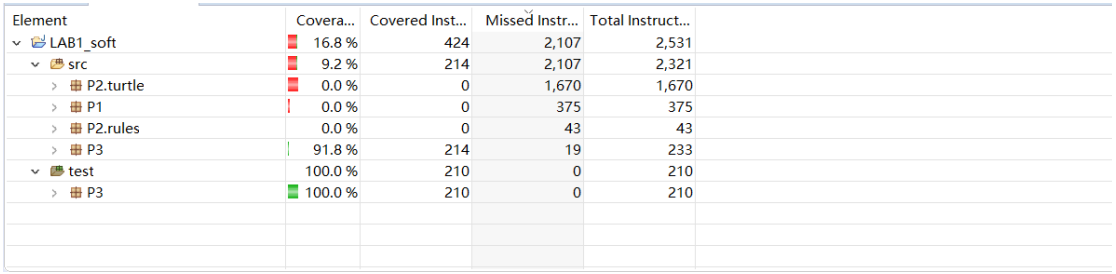
2 实验环境配置

2.1 安装 EclEmma

打开 eclipse，在菜单栏中选择 Help，然后点击 Install New Software,在打开的对话框中输入安装网址，选择 EclEmma，点击下一步即可（这里之前已经装好了，所以无法点击下一步）



安装效果：（这里用第一个实验的工程文件演示）



Element	Covera...	Covered Inst...	Missed Instr...	Total Instruct...
LAB1_soft	16.8 %	424	2,107	2,531
src	9.2 %	214	2,107	2,321
P2.turtle	0.0 %	0	1,670	1,670
P1	0.0 %	0	375	375
P2.rules	0.0 %	0	43	43
P3	91.8 %	214	19	233
test	100.0 %	210	0	210
P3	100.0 %	210	0	210

在这里给出你的 GitHub Lab2 仓库的 URL 地址（Lab2-学号）

https://github.com/ComputerScienceHIT/HIT-Lab2-1190501614

3 实验过程

3.1 Poetic Walks

该任务主要是通过实现 Graph 接口类中的方法并可额外增加一些方法来完成 Graph 的基本设计，然后通过 Graph 来实现 GraphPoet 类。该任务着重于抽象数

据类型的训练，包括设计、实现、测试；同时注重不可变类型和可变类型的规范

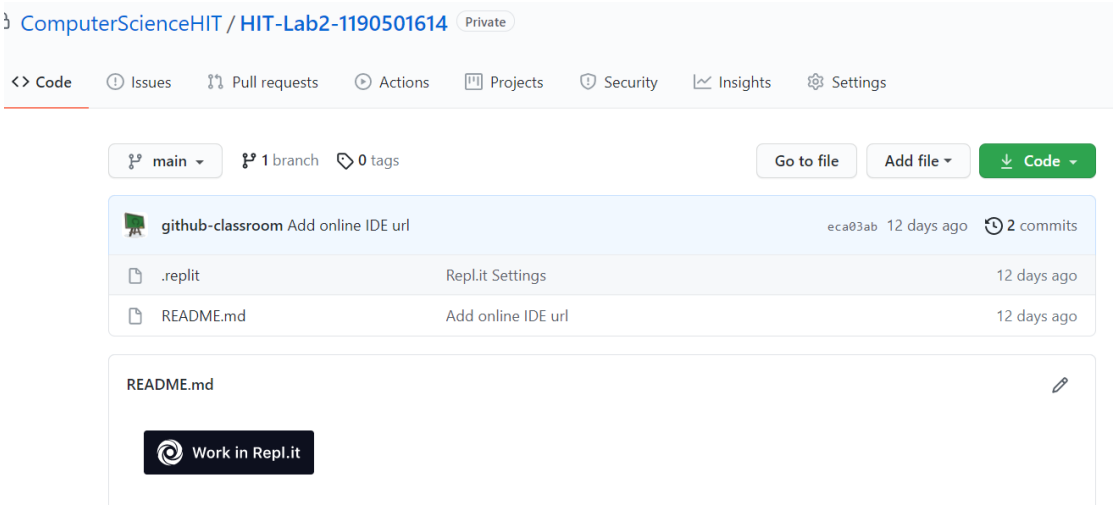
3.1.1 Get the code and prepare Git repository

由于网速问题，这里通过老师给出的另一条链接

https://github.com/rainywang/Spring2020_HITCS_SC_Lab2/tree/master/P1

去 Github 直接下载压缩包即可

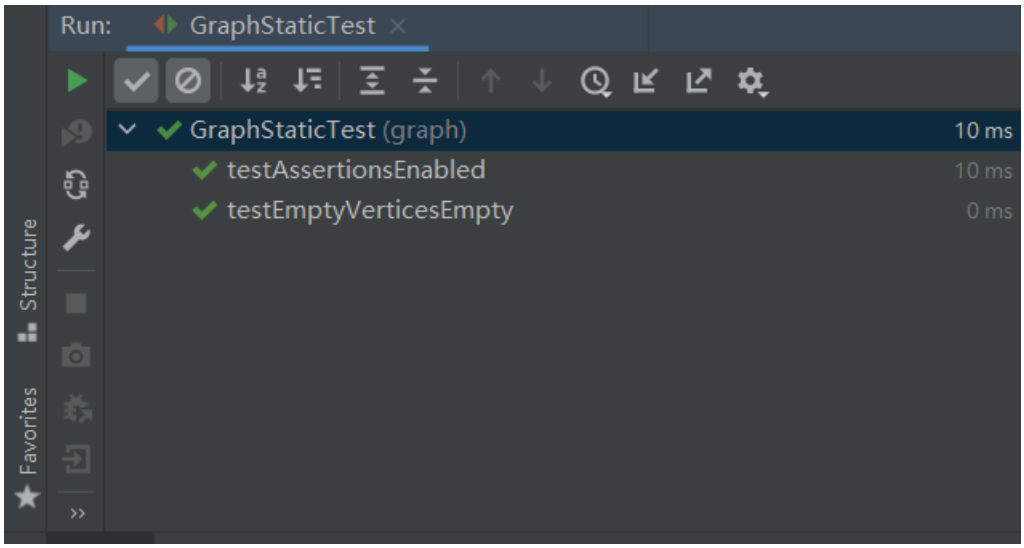
然后通过访问 GitHub Classroom 的 URL 地址建立仓库即可



3.1.2 Problem 1: Test `Graph <String>`

测试静态方法生成 String 类型的 Graph。

测试结果：



3.1.3 Problem 2: Implement Graph <String>

通过重写 Graph 接口中的方法并额外增加一些方法来实现 Graph<string>的基本功能

3.1.3.1 Implement ConcreteEdgesGraph

1. ConcreteEdgesGraph 实现
   - void checkRep()

```
for (L vertex : vertices)
        assert (vertex != null);
for (Edge<L> edge : edges)
        assert (edge != null);
```

   - boolean add(L vertex)
     如果 vertices 包含点 vertex，返回 false，否则添加 vertex，返回 true
   - int set(L source, L target, int weight)
     如果 weight<0，抛出异常；如果 vertices 不包含 source 或 target 则添加 source 或 target 到 vertices 中；如果 source 和 target 的值相同，则返回 0；然后使用迭代器遍历 edges，找到边(source,target)，如果 weight>0,将其值改为 weight，返回这条边之前的 weight，否则 remove 这条边，返回 0；如果找不到这条边并且 weight 大于 0，则添加这条边，返回 0，否则直接返回 0。
   - boolean remove(L vertex)

```
if (!vertices.contains(vertex))
      return false;
edges.removeIf(edge -> edge.source().equals(vertex) || edge.target().equals(vertex));
vertices.remove(vertex);
checkRep();
return true;
```

   - Set<L> vertices()

```
return new HashSet<>(vertices);
```

   - Map<L, Integer> sources(L target)
     遍历 edges，如果这条边的 target 等于参数 target，则将这条边的 source 和 weight 放进一个 Map<L,Integer>的容器中，最后返回这个容器
   - Map<L, Integer> targets(L source)
     同上，和 Map<L, Integer> sources(L target)类似
   - String toString()
     返回关于 vertices 和 edges 的基本信息的字符串

2. Edge 实现
   - Edge(L source, L target, int weight) //构造函数

```
this.source = source;
this.target = target;
this.weight = weight;
checkRep();
```

- void checkRep()

```
assert (weight > 0 && !source.equals(target));
```

- int getWeight()

```
return this.weight;
```

- L getSource()

```
 return this.source;
```

- L getTarget()

```
return this.target;
```

- boolean equaledge(L source,L target)

```
if(source.equals(this.source) && target.equals(this.target))
     return true;
return false;
```

- String toString()
  返回 source，target，weight 的基本信息

3.1.3.2 Implement ConcreteVerticesGraph

1. ConcreteVerticesGraph 实现
   - void checkRep()
     检查 vertices 中的每个点的 source 和 target 的 key 和 value
   - boolean add(L vertex)
     遍历 vertices，如果其中有一个点的 ver 等于参数 vertex，则返回 false，否则 new Vertex<L>(vertex),并将其加入到 vertices，返回 true
   - int set(L source, L target, int weight)
     首先检查 weight 是否小于 0，小于 0 则抛出异常；然后检查 vertices 是否包含 source 和 target;接下来检查 source 和 target 是否相同;然后遍历 vertices 找到 ver 等于的 source 的 Vertex 和 ver 等于 target 的 Vertex，然后分别设置出边表和入边表，返回之前的边的 weight
   - boolean remove(L vertex)
     通过迭代器遍历 vertices，找到其值等于参数 Vertex 的点，然后遍历更改与这个点有关的出边和入边，最后移除这个点
   - Set<L> vertices()
     返回 vertices 的一个副本即可
   - Map<L, Integer> sources(L target)
     放回以 target 为终点的边的<source，weight>的容器,简单遍历即可
   - Map<L, Integer> sources(L source)
     同上，基本类似
   - String toString()
     放回一个关于 vertices 大小信息的字符串

2. Vertex

- 构造函数

```
Vertex(L vertex){ this.ver=vertex; }
```

- void checkRep()

```
for (L key : inEdges.keySet())
        assert (inEdges.get(key) > 0);
for (L key : outEdges.keySet())
        assert (outEdges.get(key) > 0);
```

- L getVer()

```
return this.ver;
```

- Map<L, Integer> getsources()

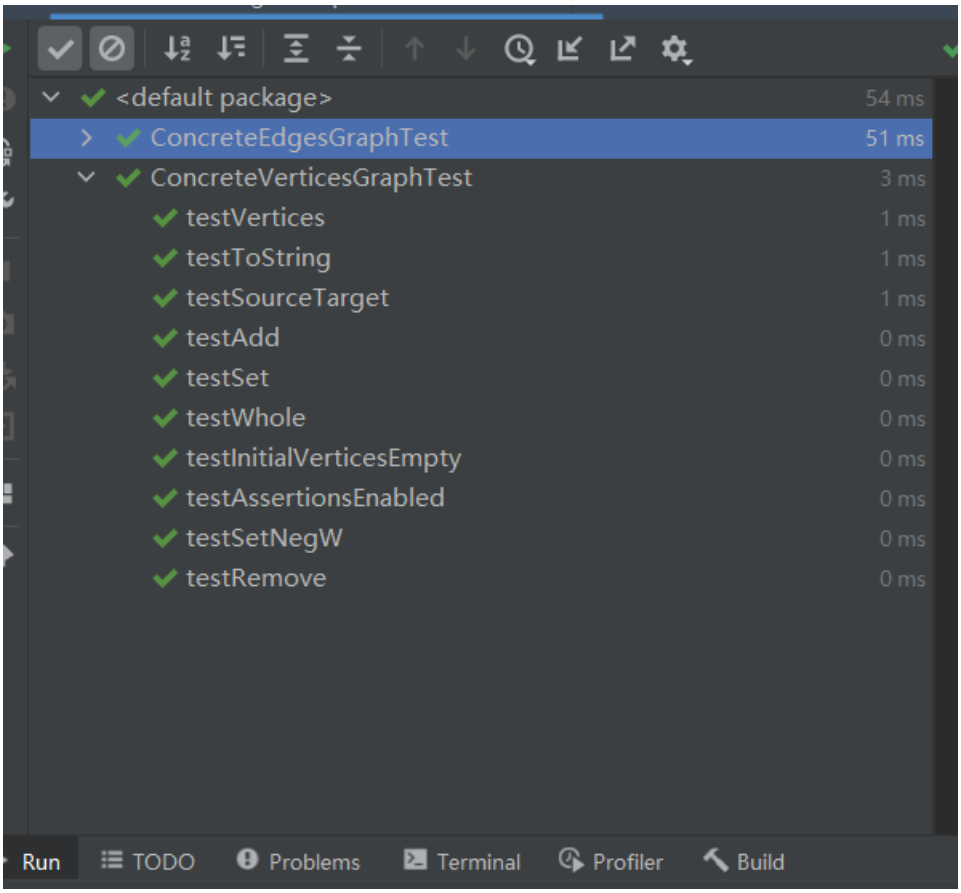
```
Map<L, Integer> sources = new HashMap<>();
        sources.putAll(inEdges);
        return sources;
```

- Map<L, Integer> gettargets()

```
Map<L, Integer> targets = new HashMap<>();
        targets.putAll(outEdges);
        return targets;
```

- int set_outEdge(L target, int weight)
  设置出边，遍历 outEdges 找到出边表中 key 等于 target 的元素，然后更改 weight，返回之前的 weight 值。注意检查 weight 是否为正数
- int setInEdge(L source, int weight)
  设置入边，同上类似
- int removeInEdge(L source)
  判断入边表中的 key 值是否包含 source，后直接调用内置函数移除即可
- int removeOutEdge(L target)
  同上，类似
- String toString()
  返回 ver 的信息

Junit 测试结果：



3.1.4 Problem 3: Implement generic `Graph<L>`

3.1.4.1 Make the implementations generic

在程序中选择“重构”或“String”并选择更改所有匹配项（要注意 toString），即可实现泛化。

3.1.4.2 Implement `Graph.empty()`

```
public static Graph<String> empty() {
        return new ConcreteEdgesGraph();
}
```

3.1.5 Problem 4: Poetic walks

根据给定文本中的信息生成一个单词图，然后给定一条输入，在图中搜索词之间的关系，自动补全语句。对每一个不一样的单词看作一个顶点，相邻的单词之间，建立一条有向边，这条有向边的权值等于相邻单词对出现的次数。在输入信息时，检查相邻单词 A 和 B，如果存在一个单词 C，在图中可以由前一个单词 A 通过这个单词 C 到达单词 B，那么就在

A 和 B 之间补全 C，补全的优先级按照权值越大者优先

3.1.5.1 Test GraphPoet

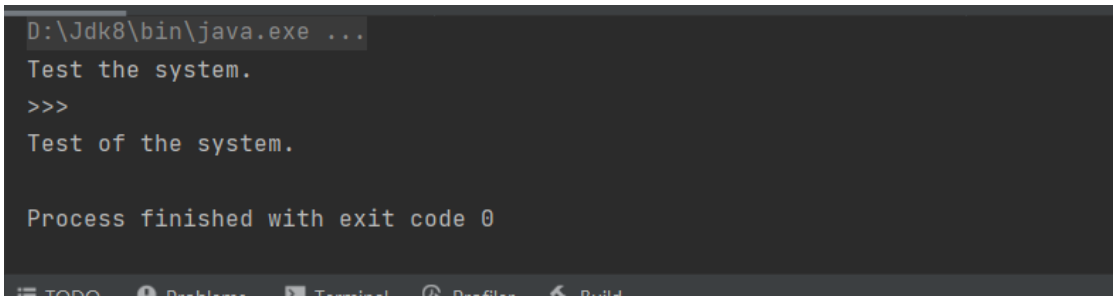
在基本测试之后，增加特殊的测试用例和边界检查，包括单连接，多连接，孤立点以及结尾的符号

3.1.5.2 Implement GraphPoet

1. void checkRep()
   检查 vertices 中的每一点 vertex 是否非空
2. GraphPoet(File corpus)
   根据文件内容构造单词关系图，首先读入文件，然后分割字符串，将字符串传给 graph，然后调用函数设置单词之间的边
3. String poem(String input)
   根据输入内容完善原来的单词图，首先分割输入的字符串，然后判断单词是否已经存在，若不存在，直接加入；若存在，则调用 target 和 source 函数找到其中的点的信息，然后修改
4. String toString()
   return "an instance of GraphPoet, hashcode: " + this.hashCode();

3.1.5.3 Graph poetry slam

文件样例测试：

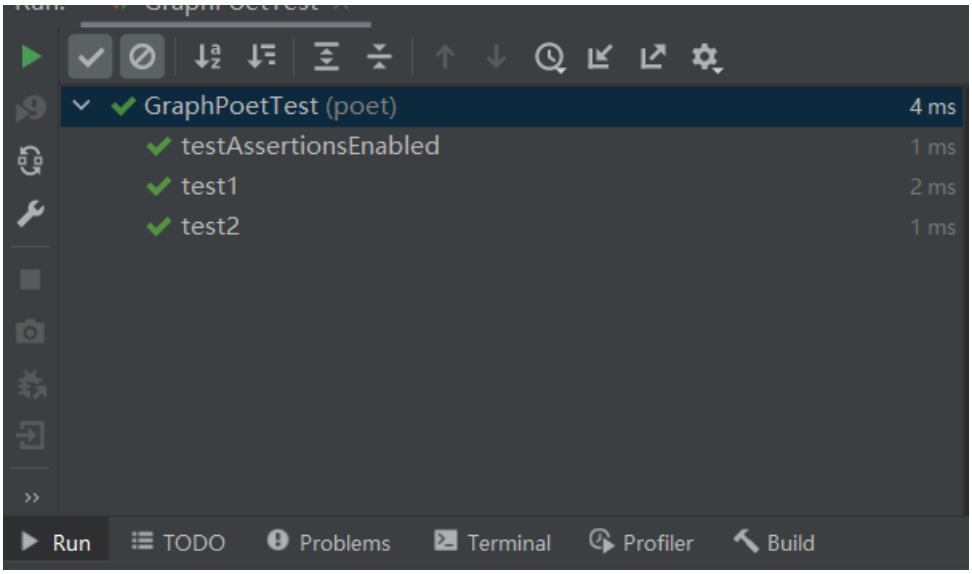


边界检查：This is a system Omni Theater native system test of.（去掉所有的 the）

测试用例 2：This is the a of system Omni Theater native system test of the.（制造回路）

```
D:\Jdk8\bin\java.exe ...
Test the of a system
>>>
Test of the a of the a of system

Process finished with exit code 0
```

Junit 测试：



3.1.6 使用 Eclemma 检查测试的代码覆盖度

Element	Covera...	Covered Inst...	Missed Instr...	Total Instruct...
SC_LAB2	42.4 %	1,241	1,688	2,929
src	39.8 %	610	923	1,533
test	45.2 %	631	765	1,396
P1.graph	62.3 %	631	382	1,013
GraphInstanceTest.java	64.7 %	450	246	696
ConcreteEdgesGraphTest.j	36.4 %	40	70	110
GraphStaticTest.java	40.9 %	38	55	93
ConcreteVerticesGraphTes	90.4 %	103	11	114

test (2021-6-10 15:10:12)

Element	Covera...	Covered Inst...	Missed Instr...	Total Instruct...
SC_LAB2	50.9 %	1,490	1,439	2,929
src	44.9 %	689	844	1,533
test	57.4 %	801	595	1,396
P1.graph	62.3 %	631	382	1,013
P2	49.8 %	155	156	311
P1.poet	20.8 %	15	57	72

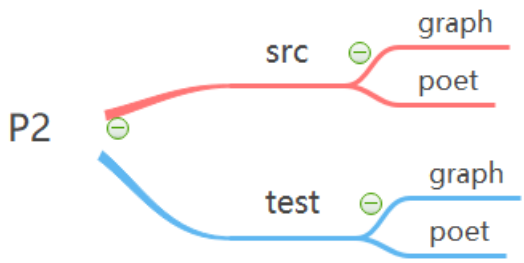
3.1.7 Before you’re done

```
陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity (master)
$ git init
Reinitialized existing Git repository in E:/localreposity/.git/

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity (master)
$ git clone https://github.com/ComputerScienceHIT/HIT-Lab2-1190501614.git
Cloning into 'HIT-Lab2-1190501614'...
```

```
MINGW64:/e/localreposity/HIT-Lab2-1190501614

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ dir
README.md

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git add .

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ dir
README.md  bin  pom.xml  src  target  test

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git commit -m "提交文件"
[master 05d4d42] 提交文件
 51 files changed, 1576 insertions(+)
 create mode 100644 .classpath
 create mode 100644 .project
 create mode 100644 .settings/org.eclipse.jdt.core.prefs
 create mode 100644 .settings/org.eclipse.m2e.core.prefs
 create mode 100644 bin/P1/graph/ConcreteEdgesGraph.class
```

```
陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git push
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 8 threads
Compressing objects: 100% (63/63), done.
Writing objects: 100% (65/65), 36.95 KiB | 2.46 MiB/s, done.
Total 65 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/ComputerScienceHIT/HIT-Lab2-1190501614.git
   eca03ab..05d4d42  master -> master

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$
```

在这里给出你的项目的目录结构树状示意图。



3.2 Re-implement the Social Network in Lab1

该任务要求我们用 P1 的 ADT 把第一次实验中的 FriendshipGraph 重新实现一遍，而对于已经写好的 FriendshipGraph 中的方法，要用 3.1 中的 Graph ADT 中的方法来实现它们。

3.2.1 FriendshipGraph 类

1. private final Graph<Person<L>> friendshipgraph = Graph.empty();
   初始化一个为空的 graph
2. boolean addVertex(Person<L> person)

```
return friendshipgraph.add(person);
```

3. int addEdge(Person<L> a, Person<L> b)
   调用 graph.set()两次，添加双向边，默认权值为 1，并返回旧边的权值。
4. int getDistance(Person<L> source, Person<L> target)
   用 dijkstra 算法即可（由于情况特殊，也可以用广度优先搜索）

3.2.2 Person 类

该类每一个人对应到一个 Person 对象，并存储名字的信息。为了保护信息被破坏，我将 String Name 设置为私有且不可变的。在构造函数中将 Name 初始化。

3.2.3 客户端 main()

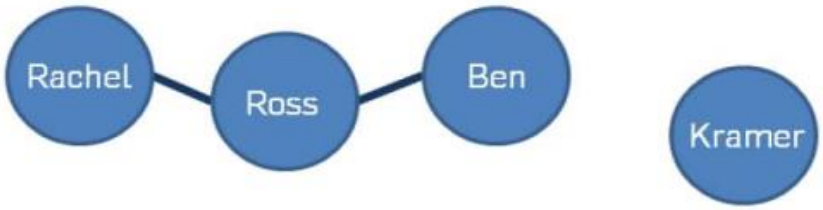
```
public class FriendshipGraphTest {

    /**
     * esstensial Network Test
     */
    @Test
    public void teste1() {
        final FriendshipGraph graph = new FriendshipGraph();

        final Person rachel = new Person("Rachel");
        final Person ross = new Person("Ross");
        final Person ben = new Person("Ben");
        final Person kramer = new Person("Kramer");

        assertEquals(true, graph.addVertex(rachel));
        assertEquals(true, graph.addVertex(ross));
```

```
        assertEquals(true, graph.addVertex(ben));
        assertEquals(true, graph.addVertex(kramer));

        assertEquals(0, graph.addEdge(rachel, ross));
        assertEquals(1, graph.addEdge(ross, rachel));
        assertEquals(0, graph.addEdge(ben, ross));
        assertEquals(1, graph.addEdge(ben, ross));

        assertEquals(2, graph.getDistance(rachel, ben));
        assertEquals(1, graph.getDistance(rachel, ross));
        assertEquals(0, graph.getDistance(rachel, rachel));
        assertEquals(-1, graph.getDistance(rachel, kramer));
    }

    /**
     * flexible Test
     */
    @Test
    public void testf2() {
        final FriendshipGraph graph = new FriendshipGraph();

        final Person a = new Person("1");
        final Person b = new Person("2");
        final Person c = new Person("3");
        final Person d = new Person("4");
        final Person e = new Person("5");
        final Person f = new Person("6");
        final Person g = new Person("7");
        final Person h = new Person("8");

        assertEquals(true, graph.addVertex(a));
        assertEquals(true, graph.addVertex(b));
        assertEquals(true, graph.addVertex(c));
        assertEquals(true, graph.addVertex(d));
        assertEquals(true, graph.addVertex(e));
        assertEquals(true, graph.addVertex(f));
        assertEquals(true, graph.addVertex(g));
        assertEquals(true, graph.addVertex(h));

        assertEquals(0, graph.addEdge(a, b));
        assertEquals(0, graph.addEdge(a, d));
        assertEquals(0, graph.addEdge(b, d));
        assertEquals(0, graph.addEdge(c, d));
        assertEquals(0, graph.addEdge(d, e));
```

```
        assertEquals(0, graph.addEdge(c, f));
        assertEquals(0, graph.addEdge(e, g));
        assertEquals(0, graph.addEdge(f, g));

        assertEquals(2, graph.getDistance(a, e));
        assertEquals(1, graph.getDistance(a, d));
        assertEquals(3, graph.getDistance(a, g));
        assertEquals(3, graph.getDistance(b, f));
        assertEquals(2, graph.getDistance(d, f));
        assertEquals(-1, graph.getDistance(f, h));
    }

}
```

3.2.4 测试用例

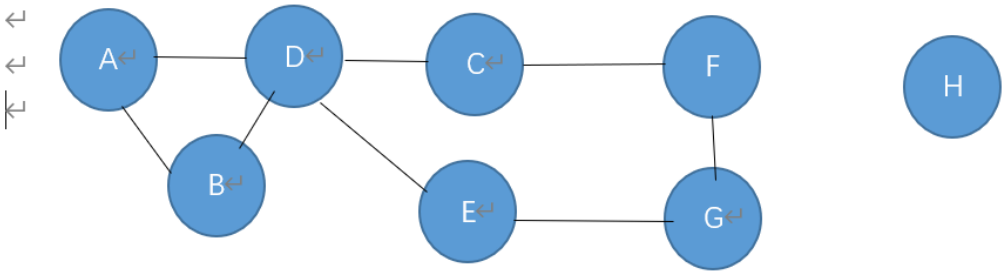
Ppt 提供的简单图测试：

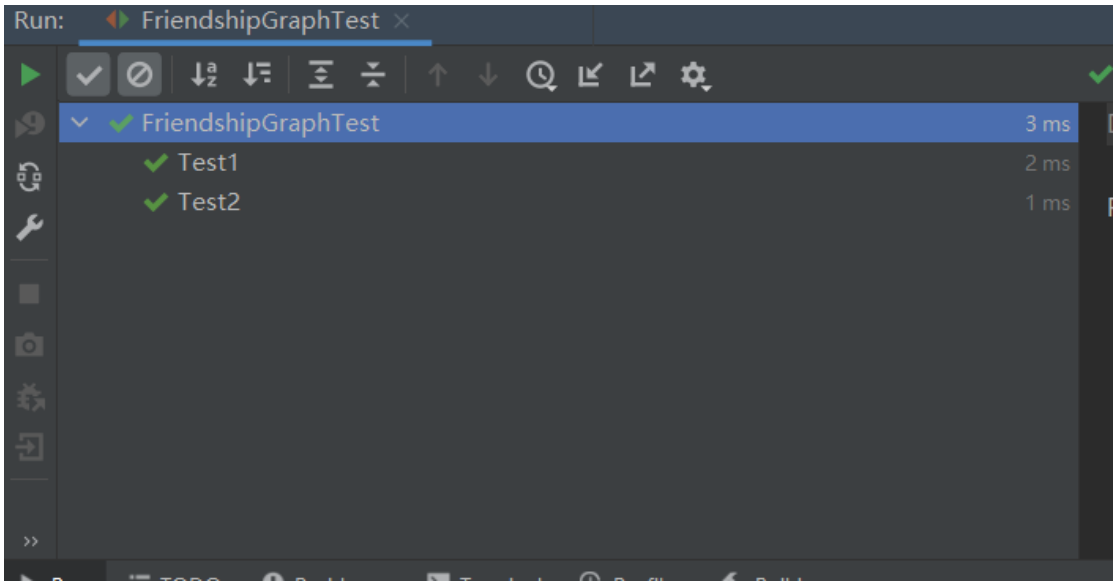


测试内容：

1. Rachel 和 Ross 距离是 1，Rachel 和 Ben 距离是 2
2. Rachel 和 Rachel 距离是 0
3. Rachel 和 Kramer 距离是-1

复杂图测试：



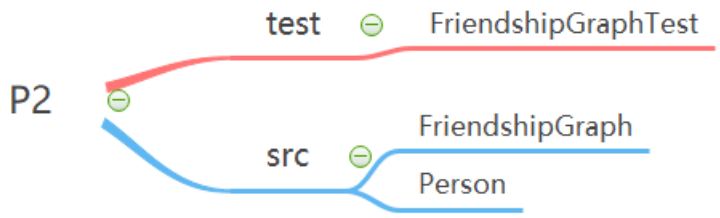


3.2.5 提交至 Git 仓库

```
陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity (master)
$ git init
Reinitialized existing Git repository in E:/localreposity/.git/

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity (master)
$ git clone https://github.com/ComputerScienceHIT/HIT-Lab2-1190501614.git
Cloning into 'HIT-Lab2-1190501614'...
```

```
陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ dir
README.md

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git add .

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ dir
README.md  bin  pom.xml  src  target  test

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git commit -m "提交文件"
```

```
陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$ git push
Enumerating objects: 66, done.
Counting objects: 100% (66/66), done.
Delta compression using up to 8 threads
Compressing objects: 100% (63/63), done.
Writing objects: 100% (65/65), 36.95 KiB | 2.46 MiB/s, done.
Total 65 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), done.
To https://github.com/ComputerScienceHIT/HIT-Lab2-1190501614.git
   eca03ab..05d4d42  master -> master

陈广焕@LAPTOP-B1P24AV2 MINGW64 /e/localreposity/HIT-Lab2-1190501614 (master)
$
```



4 实验进度记录

请使用表格方式记录你的进度情况，以超过半小时的连续编程时间为一行。

日期	时间段	计划任务	实际完成情况
6-06	下午、晚上	完成 P1graph	完成
6-07	晚上	完成 P1.poet	未完成
6-08	下午、晚上	完成 P1 和测试部分	完成
6-9	下午	完成 P2 构建	完成
6-10	下午	完成全部	完成

5 实验过程中遇到的困难与解决途径

遇到的难点	解决途径
用 set 的函数的代码实现出现 bug	使用 contains 函数 网上查找资料，使用 Map 的容器的一些内置函数
用 Dijstra 算法实现求路径出现问题	改用广度优先搜索

6 实验过程中收获的经验、教训、感想

6.1 实验过程中收获的经验和教训

要认真写 spec 很重要，包括注释
学会编写测试用例，检验程序正确性

6.2 针对以下方面的感受

(1) 面向 ADT 的编程和直接面向应用场景编程，你体会到二者有何差异？

面向 ADT 的编程更加抽象

(2) 使用泛型和不使用泛型的编程，对你来说有何差异？

没有太多差异

(3) 在给出 ADT 的规约后就开始编写测试用例，优势是什么？你是否能够适应这种测试方式？

能把 RI 和 AF 考虑全面，方便后面编程

(4) P1 设计的 ADT 在多个应用场景下使用，这种复用带来什么好处？

减低开发的时间成本

(5) 为 ADT 撰写 specification, invariants, RI, AF，时刻注意 ADT 是否有 rep exposure，这些工作的意义是什么？你是否愿意在以后编程中坚持这么做？

减少程序出现错误

(6) 关于本实验的工作量、难度、deadline。

还行

(7) 《软件构造》课程进展到目前，你对该课程有何体会和建议？

无