**深 圳 大 学 实 验 报 告**

**课程名称：­ Python程序设计**

**实验项目名称： 作业2**

**学院： 计算机与软件学院**

**专业： 软件工程**

**指导教师： 潘浩源**

**报告人： 郑彦薇 学号： 2020151022**

**实验时间： 2022/6/6~2022/6/10**

**实验报告提交时间： 2022/6/10**

**教务部制**

|  |
| --- |
| **一、实验目的：**  **利用所学基础知识与数据结构，解决实际问题。** |
| 1. **实验方法步骤** 2. **读题，对每个问题提出解决问题的思路** 3. **对需要编程解决的题目按照得到的思路编写源代码** 4. **运行代码，调试错误，直到输出正确结果** |
| **三、实验过程及内容：**  **（一）完成课程讲义中状态机的实现**  **1. 解题思路**   * **状态机SM**  1. 父类状态机SM的函数编写，主要包括init函数和transduce函数；而对于函数transition\_fn   以及output\_fn，则在子类中根据相应功能进行具体编写。   1. Init函数：初始化私有成员start\_state的值为None即可。 2. Transduce函数：对于每一个子类的不同输入，定义可一致化的函数。访问输入中的每一项，对其进行相应功能操作（即transition\_fn），以输入长度设置循环，不断更新状态值（即start\_state），最后对其进行输出即可。   **代码及细节解释如下：**  **1654499784246**  根据输入列表长度设置循环，访问列表每一项，进行相应的功能操作，不断更新state值   * **累加器Accumulator**  1. Start\_state的初始化：该功能是实现对列表元素的相加，只需记录每次相加的结果，因此在该类中，将start\_state初始化为0。 2. Transition\_fn函数：累加器功能是实现对输入列表中每一位元素的相加。根据功能以及父类transduce函数方法，只需返回每次执行结果（即列表中第i个元素x与前面元素求和结果s的和）即可。 3. Output\_fn函数：对最终结果进行返回即可。   **代码及细节解释如下：**  **1654500115307**   * **二进制相加Binary\_Addition**  1. Start\_state的初始化：在进行二进制相加中，除了对当前对应位置值的相加结果进行记录，还需记录进位状态；另外根据例子可以知道，二进制相加结果将每一位作为列表元素以列表形式存储。因此在该类中，将start\_state初始化为（0，[]），其中start\_state[0]表示进位值，start\_state[1]表示当前结果，因为二进制逆序存储，因此直接向结果列表添加元素即可。 2. Transition\_fn函数：   在对当前位置二进制数进行相加时，会产生四种情况：  ①当前位置相加结果为2，且前一个位置相加已发生进位，即进位值与相加结果值的和为3：当前结果为1（即结果列表添加元素1），且仍需向前进一位（即进位值更新为1）  ②当前位置相加结果为2，且前一个位置相加不发生进位，或当前位置相加结果为1，且前一个位置相加发生进位，即进位值与相加结果值的和为2：当前结果为0（即结果列表添加元素0），且需要向前进一位（即进位值更新为1）  ③当前位置相加结果为1，且前一个位置相加不发生进位，或当前位置相加为0，且前一个位置相加发生进位，即进位值与相加结果值的和为1：当前结果为1（即结果列表添加元素1），且无需向前进位（即进位值为0）  ④当前位置相加结果为0，且前一个位置相加不发生进位，即进位值与相加结果值的和为0：当前结果为0（即结果列表添加元素0），且无需进位（即进位值为0）  根据上述四种情况设置四个判断条件，对相应结果状态进行更新即可。   1. Output\_fn函数：首先判断当前结果的进位值是否为1，为1则表示二进制相加结果需要再添加元素1，否则直接将结果列表进行返回。如111+111得到的状态值为（1，110），根据状态值为1，结果列表添加元素1，得到最终正确结果：1110。   **代码及细节解释如下：**  **1654501227736**  最后一步对进位值的判断，得到最终结果  根据所描述四种情况确定state值   * **反向器Reverser**  1. Start\_state的初始化：根据题目要求，首先需要根据seq1的长度输出对应个数None；其次需要根据end的出现位置停止对seq1的读取以及seq1长度的更新；接着需要继续读入seq2，并不断更新最终输出列表长度。因此在该类中，将start\_state初始化为（0，0，0，[]），其中start\_state[0]的值为0或1，表示seq1是否读取完毕；start\_state[1]记录seq1的长度，作为首先输出多少个None的依据；start\_state[2]记录需要进行输出的列表总长度，在完成对seq1元素的反向输出后，确定仍需输出多少个None的依据；start\_state[3]记录已输入列表（即根据读取input\_seq的结果，同步更新该列表） 2. Transition\_fn函数：根据上述所说state每一个位置的值所表示的意义，设置判断条件：若seq1读取未结束即状态值为0，则seq1长度增加；若当前元素为end，表示seq1读取完毕，修改状态；在整个读取过程中，输入列表的长度不断增加，记录列表也不断更新。 3. Output\_fn函数：根据transition\_fn函数得到的列表长度，设置循环将对应结果添加到结果列表中：seq1长度个None、seq1元素的反向输出、列表剩余元素（根据要求，均为None）。   **代码及细节解释如下：**  **1654503796057**  已输入列表，最终与input\_seq相同  已输入列表长度  seq1列表长度  状态值，判断seq1是否读取完毕   1. **程序运行结果展示**   主函数中设定输入值：  **1654504336379**  对应输出结果：  1654504444129   1. **纸牌游戏** 2. **解题思路**  * **Card类的补充**   根据类中函数的功能，可以知道待补充的三个函数为比较函数，分别进行大于、小于或等于的比较，在类中进行补全即可。  **代码及细节解释如下：**  **1654677126243**   * **Hand类的补充**  1. receive\_cards函数：玩家收到多张牌时，直接将收到的卡片列表添加到原有的手牌列表中即可。 2. receive\_card函数：玩家收到一张牌时，可以通过列表的append功能实现对手牌列表牌的补充。 3. give\_card函数：玩家给出牌前已经通过shuffle函数实现“洗牌”，首先判断手牌是否已空，如果已空，则返回None，否则直接给出第一张牌，同时将该牌从手牌列表中进行删除。 4. give\_cards函数：玩家给出多张牌，只需直接将玩家的手牌进行清空即可。   **代码及细节解释如下：**  **1654677685667**  1654677824246  需要使用copy复制列表信息，如果只是简单的赋值：temp=self.cards，对self.cards操作时，也对temp进行了操作，导致错误  将第一张牌从手牌列表中删除  临时变量，记录第一张手牌   * **Deck类的补充**   进行52张牌的分配，只需利用花色和点数设置循环，按照（点数，花色）的格式将牌信息进行添加即可。  **代码及细节解释如下：**  **1654678506322**   * **Player类的补充**   在该类中只需对玩家及其所持手牌进行初始化即可，注意手牌的初始化为列表的拷贝，需要使用copy()函数进行深拷贝，否则会出错。（与上述hand类中的give\_cards函数中的copy拷贝同理）。  **代码及细节解释如下：**  **1654686468639**   * **Game类的补充**  1. Deal函数：实现两个玩家的轮流取牌，直到牌堆被取完为止。 2. Turn函数：该函数的目的是获取每一轮的胜者。首先当两个玩家手中都还有牌时，游戏就仍在继续，因此可以设置循环条件执行每一轮，每一轮的获胜条件是当前玩家出牌大于另一个玩家，那么会有3种情况：第一个玩家出牌更大、第二个玩家出牌更大以及两人出牌大小一致；当循环结束时，同样会有3种情况：玩家1的牌数为0，玩家2获胜；玩家2的牌数为0，玩家1获胜；两人牌数都为0，达成平手。 3. Play函数：首先进行发牌，接着使用变量winner记录每一轮的获胜者并得到最后的获胜者（即最后一轮的获胜者），同时在两人每次出牌后，都会进行一次洗牌，也就是需要使用shuffle函数重新进行随机排序。   **代码及细节解释如下：**  **1654688557204**  在牌堆数为0之前，给两个玩家轮流发牌  **1654688682303**  将每一轮的获胜信息返回给play函数的winner  两个玩家手中牌清空前持续执行每一轮  借用中间变量temp完成每一轮牌的比较和转移  **1654688850252**  两个玩家手中牌清空，进行最后赢家的判断  **1654689023132**  记录每一轮赢家  玩家手中还有牌，洗牌  **2. 程序运行结果展示**   * **Test\_card():**   **1654689298077**   * **Test\_hand():**   **1654689330100**   * **Test\_deck():**   **1654689357996**   * **Test\_player():**   **1654689388999**   * **Test\_game():**   **1654689422622**   1. **SimpleGraph** 2. **解题思路**  * **Vertex()类的补充**   该类为顶点类，作为图中的顶点数据类型以及边中的起点和终点数据类型，将变量进行初始化即可。  **代码及细节解释如下：**  **1654788189167**   * **Edge()类的补充**   该类为边类，边中的起点和终点数据类型均为Vertex类，因此需要继承Vertex类，然后同样将变量进行初始化即可。  **代码及细节解释如下：**  **1654788288773**   * **SimpleGraph()类的补充**  1. Init函数：使用列表存储顶点信息和边信息，列表元素分别为Vertex类和Edge类，由于题中需要使用dfs求解路径，因此设置访问数组visit并初始化为0. 2. Add\_vertex函数：该函数是往图中添加顶点信息，只需将顶点以Vertex类的形式传入函数，将其添加到列表中即可。 3. Add\_edge函数：该函数需要往图中添加边，首先根据输入初始化边信息，然后将边添加到列表中，再进行邻接表的创建；遍历顶点列表，确定当前边起点，将边终点添加到起点存放邻居的列表edges中。 4. Contains\_vertex函数：遍历顶点列表，判断列表中是否存在点的名称与传入函数的字符串相同，若相同，则点存在，返回True，否则返回False。 5. Contains\_edge函数：遍历边列表，判断列表中是否存在边起点名称和边终点名称与传入参数相同，若存在相同，则边存在，返回True，否则返回False。 6. Get\_neighbors函数：遍历边列表，找到边起点名称与传入参数相同的边，设置临时变量temp，用于存放这些边的终点名称，返回该列表即可得到顶点v的邻居顶点信息。 7. Is\_empty函数：判断图是否为空，只需判断顶点列表和边列表是否为空。 8. Remove\_vertex函数：删除图中的顶点v，首先需要将v从顶点列表中删除，遍历顶点列表，找到名称与传入参数相同的点信息，使用remove函数将其中顶点列表中进行删除既可；其次需要将与v有关的边删除，遍历边列表，找到边的起点或终点名称与传入参数相同的边，对该边进行删除即可。 9. Remove\_edge函数：删除图中的边，首先需要将边从边列表中删除，只需对应找到边列表中，起点和终点名称与传入参数相同的边，然后进行删除；其次需要对邻接表中的信息进行删除，在顶点列表中找到边的起点，遍历起点的邻居列表，找到边的终点，对起点邻居列表中对应信息进行删除。 10. Is\_neighbor函数：判断两条边是否相邻，只需遍历边列表判断是否有某边的起点和终点都对应传入参数。 11. Is\_reachable函数：该函数目的是查找是否有从v1到达v2的路径。补充index函数用于获得顶点的下标；补充dfs函数对图的点进行深度优先遍历，连通的点更新访问值，访问值相同则说明点连通；补充visitinit函数对visit数组进行初始化。在is\_reachable函数中，判断是否存在v1到v2的路径，首先将v1的访问值设置为1，对v1进行深度有点遍历，然后判断v2的访问值是否发生改变，若发生改变，则说明v2与v1连通，即存在v1到v2的路径。   **代码及细节解释如下：**  **1654789652718**  **1654789692689**  创建邻接表，由于是有向图，只需“单向”补充  **1654789787142**  **1654789823420**  临时变量，存放邻居列表  **1654789894071**  找到边(v1, v2)在邻接表中的存储信息，进行删除  删除顶点v，与v有关的边也要进行删除  **1654789976409**  **1654790042520**  初始化visit数组，在每一次判断中都需进行初始化，避免错误  使用DFS实现对visit的修改，再根据visit判断是否存在路径  添加功能，获得顶点v的下标  **1654790166001**  图的信息输出函数  调用clear功能对列表进行清空   1. **程序运行结果展示**   设置test()函数，使用add\_vertex函数和add\_edge函数添加顶点和边信息，初始化图信息，同时测试这两个函数的功能。  1654794853451  1654794887062  得到图信息输出如下：  1654794963782  测试相应的功能：  1654795003818  1654795024709  得到运行结果如下：  1654795053381  测试完毕。 |

|  |
| --- |
| **四、作业完成总结**  通过本次作业更加熟悉了类函数的编写格式、实现类功能时对功能函数的编写以及私有成员的正确调用，对python类以及类的继承有了更深的认识。 |
| **指导教师批阅意见：**  **成绩评定：**  **指导教师签字：**  **年 月 日** |
| **备注：** |

**注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。**

**2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内**。