Seoul National University

Data Structure

Fall 2015, Kang

Programming Assignment 3 : Non-Binary Tree (Chapter 6)

Due: Nov. 17, 11:00 am, submit at eTL

Reminders

- The points of this homework add up to 100.
- Like all homeworks, this has to be done individually.
- Lead TA: Minsoo Jung (qtyp456987@gmail.com)
- Write a program in Java.
- Do not use Java Collection Framework.

1. How to submit the programming assignment

1) Create a ***JAR*** file including 'src' folder that contains your sources files, but without 'release' folder. (Refer to '1 – Introduction.pptx' in the first lab session)
   - We will run your ***Main*** class in the JAR file to grade your programming assignments. Before submitting the JAR file, please check if your Main class in the JAR file works correctly.
   - You **MUST** obey the I/O specification of the programming assignment, and rules for the submission of the programming assignment.
   - Before submitting, check if your JAR file runs properly in your terminal with the following command line: java -classpath ./PA_03_2015-12345.jar ds.test.Main.

2) Compress the JAR file with a readme file and the project folder. The readme file needs to include your student id, name, how to execute your program, and any other information TA needs to know.
   - The format of the JAR file is "*PA_##_(StudentID).jar*" where '##' is the programming assignment ID, and *(StudentID)* is your student ID.
     - ex) PA_03_2015-12345.jar
   - All documents should be written in English.

3) The name of the compressed file (zip file) should be '*PA_##_(StudentID).zip*' where '##' is the programming assignment ID, and (StudentID) is your student ID.
   - ex) PA_03_2015-12345.zip
     - *03: the third programming assignment*
     - *2015-12345: your student ID*

4) Submit the compress file to the eTL (http://etl.snu.ac.kr/) .

2. How to grade your programming assignment

1) We made a grading machine to automatically grade your programming assignment. The machine will run your program, and compare answers and outputs that your program generates for given inputs. If your program can not generate correct answers for an input file, it will not give you the point corresponding to the input. Our machine will consider the following scenarios:
   - (***Accept***) When your program generates exact outputs for an input file, the machine will give you the point of the input.
   - (***Wrong Answer***) When your program runs normally, but generates incorrect outputs for an input file, including typos, the machine will not give you the point of the input.
   - (***Run Error***) When your program does not run, or is terminated suddenly for some reasons, the machine will not give you the point of an input file because it can not generate any outputs.
   - (***Time Limit***) When your program runs over a predefined execution time for an input file, our machine will stop your program, and it will not give you the point of the input. The time limit of the execution is ***5 seconds***.

2) We will generate 10 input files, and assign 10 points for each input file. For example, if your program gets 9 accepts, and 1 wrong answer by the machine, the total point will be 90 points. Hence, before submitting your programming assignment, please be sure that your program makes correct answers in reasonable time for any input case.

3. Problem

Mr. John is a game designer at a game company. He recently planned a game called ‘Power Game’. In this game, each team has a power. One team can increase its power by merging other teams. The team with the highest power is the winner of the game. The specific rules of the ‘Power Game’ are as follows:

- A player logs in to the game. Then, the player creates a new team, and becomes the leader of the team. (Note that the player is the only member of the team.) When the player creates the team, the team power is given.
- Two different teams can be merged. The leader of the merged team is the leader of the more powerful team. The merged team’s power is the sum of the two teams’ power. If the two teams have the same power, we cannot merge the two teams.

Your mission is to implement a program for the game. There are additional requirements for the program.

- Your program should be able to print the leader and the power of a team.
- If we cannot merge two teams (the two teams have the same power), your program should print the message “Merge Failed”.
- You need to use ‘Path Compression’ technique to implement Find() operation in ‘ParPtrTree’ class.

Here are several assumptions for clarity.

- A player logs in only once, and does not log off or leave the team.
- The maximum number of players in the game is 100,000.

Complete “PowerGame” class supporting the above requirements. You need to fill in the “PowerGame.java”, the “ParPtrTree.java”, and “Main.java” of “PA_03” java project. You can modify the skeleton codes if you want.

4. ADT of Data structure

1) Merge

Function
boolean Merge(int player1, int player2)
Description
- This function merges the team of “player1” and the team of “player2”. - The sum of the two teams’ power becomes the merged team’s power. - The leader of more powerful team becomes the merged team’s leader. - If the two teams have the same power, print the message “Merge Failed”.

2) Login

Function
void Login(int player, int power)
Description
- This function makes a new team of “player” with “power”. - The new team’s leader is “player”. - The new team’s power is “power”.

3) PrintLeader

Function
void PrintLeader(int player)
Description
- This function prints the leader of the team that “player” belongs to.

4) PrintPower

Function
Void PrintPower(int player)
Description
- This function prints the power of the team that “player” belongs to.

5. Specification of I/O

1) Merge

Input form	Output form
`merge (player1) (player2)`	`MERGE: (player1) (player2)`

Description

- Merge the team of (player1) and the team of (player2).
- If the two teams have the same power, print the message “Merge Failed”.

Example Input	Example Output
`Merge 1 2`	`MERGE: 1 2` `Merge Failed`

2) Login

Input form	Output form
`login (player) (power)`	`LOGIN: (player)`

Description

- (player) is the game player who logs in the game.
- (power) is power of the team that (player) belongs to.

Example Input	Example Output
`login 1 10`	`LOGIN: 1`

3) PrintLeader

Input form	Output form
`printleader (player)`	`LEADER: (leader)`

Description

- (player) is a game player.
- (leader) is the leader of the team that (player) belongs to.

Example Input	Example Output
`printleader 1`	`LEADER: 2`

4) PrintPower

Input form	Output form
`printpower (player)`	`POWER: (power)`

Description

- (player) is a game player.
- (power) is the power of the team that (player) belongs to.

Example Input	Example Output
`printpower 1`	`POWER: 40`

6. Sample Input

```
login 1 10

login 2 30

login 3 40

merge 1 2

printleader 1

printpower 1

merge 1 3
```

7. Sample Output

```
LOGIN: 1

LOGIN: 2

LOGIN: 3

MERGE: 1 2

LEADER: 2

POWER: 40

Merge Failed
```