

**Assignment 4**

**Course:** Software Testing

**Instructor:** Dr. Shaarbaf

**Student:** Alireza Dastmalchi Saei

**Student ID:** 993613026

Contents

[**Introduction** 3](#_Toc154244578)

[**Objectives** 3](#_Toc154244579)

[**Code** 4](#_Toc154244580)

[**Control Flow Graph (CFG)** 5](#_Toc154244581)

[**Definition Use Paths (DU Paths)** 6](#_Toc154244582)

[**DU Graph** 6](#_Toc154244583)

[**DU Pair** 7](#_Toc154244584)

[**DU Paths** 8](#_Toc154244585)

[**DU Paths** 9](#_Toc154244586)

[**Test Cases** 10](#_Toc154244587)

[**Test Case 1** 10](#_Toc154244588)

[**Test Case 2** 10](#_Toc154244589)

[**Test Case 3** 10](#_Toc154244590)

# **Introduction**

This assignment delves into control flow graphs, data-flow analysis, and the nuanced art of test case generation. Our focal point is a code snippet extracted from game logic, and our mission is two-fold. Initially, we construct a Control Flow Graph (CFG) that visually encapsulates the program's intricate control flow structure. Subsequently, we navigate the terrain of Data-Flow Analysis to unveil the Def-Use (DU) paths within the code. The final leg of this intellectual odyssey entails crafting three discerning test cases tailored to the identified DU paths. Through these endeavors, we aim to fortify our understanding of program flow dynamics, decipher the nuanced relationships between data elements, and cultivate the skill of formulating compelling test scenarios to validate the robustness of the provided code.

**Objectives**

1. **Constructing Control Flow Graph (CFG):** You will begin by creating a visual representation of the program's control flow using a Control Flow Graph. This will provide insights into the order in which statements and branches are executed.
2. **Data-Flow Analysis and DU Paths:** After building the CFG, you'll delve into Data-Flow Analysis to identify Def-Use (DU) paths within the program. Understanding these paths is crucial for uncovering how data is manipulated and used across various code sections.
3. **Test Case Generation:** The final part of the assignment involves generating three meaningful test cases based on the identified DU paths. These test cases will aim to provide adequate coverage to ensure the robustness of the code.

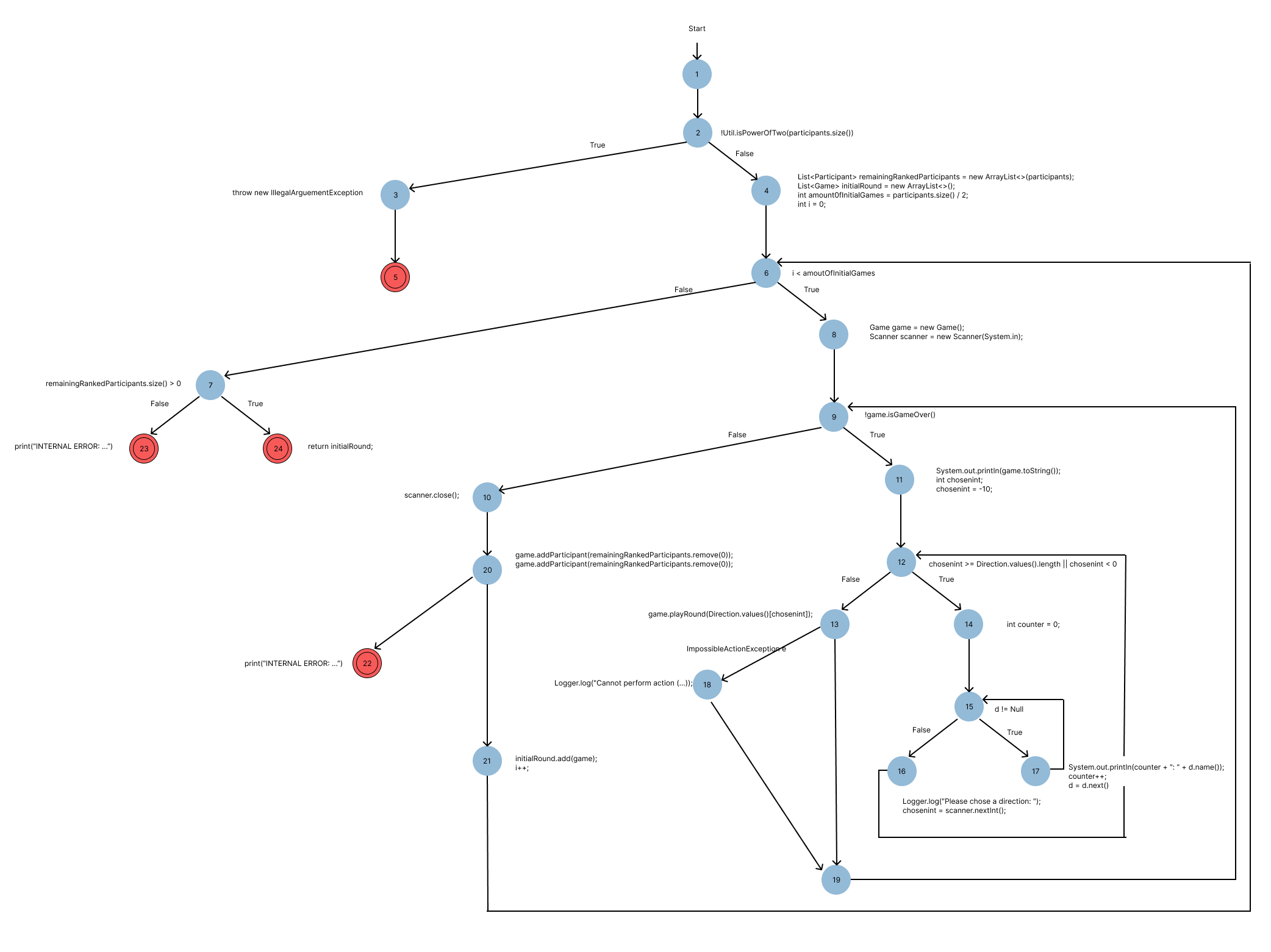
# **Code**

The code given in the assignment is represented below.



# **Control Flow Graph (CFG)**

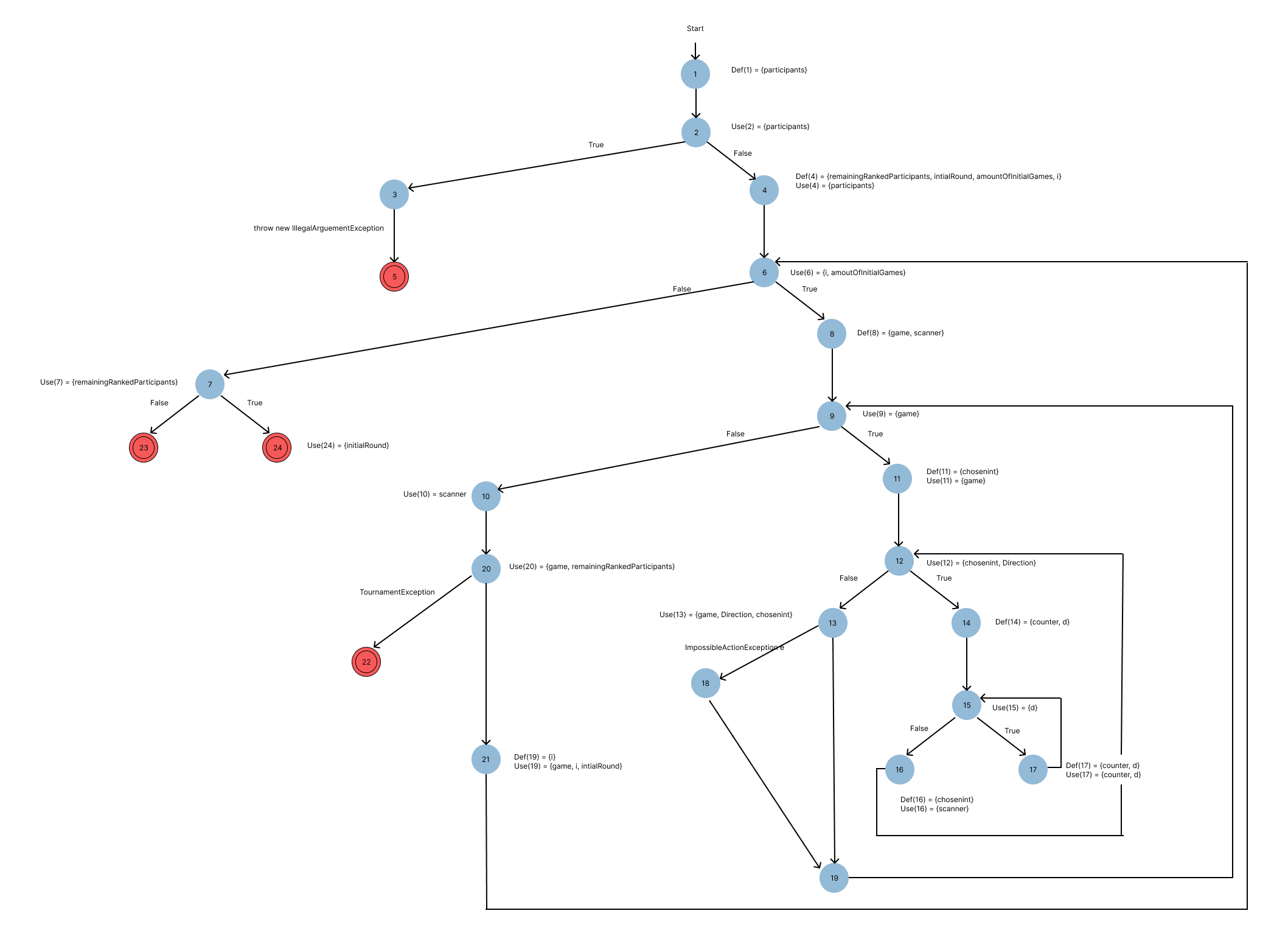
The following graph is constructed from the code given in the previous part. It has four end nodes, three due to the Exceptions. One end node contains the “***return****”* statement.



# **Definition Use Paths (DU Paths)**

## **DU Graph**

In this part, we will extract all Def-Use Paths from the constructed CFG in the previous part. First, all variable definitions and their uses must be identified.



## **DU Pair**

In this step, the Def-Use table from the previous graph is created:

|  |  |  |
| --- | --- | --- |
| Node | Def | Use |
| 1 | participants | - |
| 2 | - | participants |
| 3 | - | - |
| 4 | remainingRankedParticipants, intialRound, amountOfInitialGames, i | participants |
| 5 | - | - |
| 6 | - | i, amoutOfInitialGames |
| 7 | - | remainingRankedParticipants |
| 8 | game, scanner | - |
| 9 | - | game |
| 10 | - | scanner |
| 11 | chosenint | game |
| 12 | - | chosenint, Direction |
| 13 | - | game, Direction, chosenitn |
| 14 | counter, d | - |
| 15 | - | d |
| 16 | chosenint | scanner |
| 17 | counter, d | counter, d |
| 18 | - | - |
| 19 | - | - |
| 20 | - | game, remainingRankedParticipants |
| 21 | i | game, initialRound, i |
| 22 | - | - |
| 23 | - | - |
| 24 | - | intialRound |

# **DU Paths**

After creating the DU table, we can create DU pairs for further test case creation.

|  |  |
| --- | --- |
| Variable | DU PAIRs |
| dIRECTION | - |
| participants | [1, 2], [1, 4] |
| remainingRankedParticipants | [4, 7], [4, 20] |
| InitialRound | [4, 21], [4, 24] |
| amountOfInitialGames | [4, 6] |
| i | [4,6], [4,21], [21,6], [21,21] |
| game | [8,9], [8,11], [8,13], [8,20], [8,21] |
| scanner | [8,10], [8,16] |
| chosenint | [11,12], [11,13], [16,12], [16,13] |
| counter | [14,17], [17,17] |
| d | [14,15], [14,17], [17,15], [17,17] |
| e | - |

# **DU Paths**

Now that we have the DU Pair Table, we can create the DU Paths Table from it.

|  |  |
| --- | --- |
| Variables | DU Paths |
| dIRECTION | - |
| participants | [1,2], [1,2,4] |
| remainingRankedParticipants | [4,6,7], [4,6,8,9,10,20] |
| InitialRound | [4,6,7,24], [4,6,8,9,10,20,21] |
| amountOfInitialGames | [4, 6] |
| i | [4,6], [4,6,8,9,10,20,21],  [21,6], [21,6,8,9,10,20,21] |
| game | [8,9], [8,9,11],  [8,9,10,20], [8,9,11,12,13],  [8,9,10,20,21] |
| scanner | [8,9,10], [8,9,11,12,14,15,16] |
| chosenint | [11,12], [11,12,13],  [16,12], [16,12,13] |
| counter | [14,15,17], [17,15,17] |
| d | [14,15], [14,15,17],  [17,15], [17,15,17] |
| e | - |

# **Test Cases**

## **Test Case 1:**

* **Input:** [Player1, Player2]
* **Test-Path:** [1, 2, 4, 6, 8, 9, 10, 20, 21, 6, 8, 9, 10, 20, 21, 6, 7, 24]
* **Expected Output:** intialRound

## **Test Case 2:**

* **Input:** [Player1, Player2]
* **Test-Path:** [1, 2, 4, 6, 8, 9, 11, 12, 14, 15, 17, 15, 17, 15, 16, 12, 13, 19, 9, 10, 20, 22]
* **Expected Output:** “INTERNAL ERROR: …”

## **Test Case 3:**

* **Input:** [Player1, Player2, Player3]
* **Test-Path:** [1, 2, 3, 5]
* **Expected Output:** “The number of participants is not a power of two.”