

Forum: [Hack Forums](#)

Thread: [Data structures](#)

JakeDarkblade - April 6, 2018, 11:14 am

I'm struggling on some data structures practice questions.

Quote: A group of chess players meet every week for chess games. They want to record who has won a game against whom.

The following games have been played:

Player 1 vs Player 2 – winner = Player 2

Player 2 vs Player 4 – winner = Player 4

Player 2 vs Player 6 – winner = Player 6

Player 3 vs Player 4 – winner = Player 3

Player 3 vs Player 5 – winner = Player 3

Player 5 vs Player 2 – winner = Player 2

Player 5 vs Player 6 – winner = Player 5

Player 6 vs Player 1 – winner = Player 1

What would be the best data structure to store this in and how would it be stored?

Also, with the chosen data structure if i want to list the players a given player has beaten or lost to how would it work?

I'm studying ahead and this is really confusing to me.

m0noxide - April 6, 2018, 1:11 pm

When you have problems like this, think through every data structure you know and try to apply it to the problem at hand.

So what basics do we know? Linked Lists, arrays, trees, graphs, hashtable etc etc.

For example:

"Hmm, maybe a tree because each game can have two players and then link to an output node"

Once you think you have a good idea, try to minimize the space and time complexity

"Well we need efficient lookup time to get the winners and the losers, we don't want to have to traverse whatever structure we have in linear time if we can avoid it"

I get that this is in the Java section but I made a quick prototype in Python to show you one way you get the winners and losers given a player. I decided to use a hashmap but you could just use simple arrays instead.

Code class Game:

```
def __init__(self, p1, p2, winner):
    self.p1 = p1
    self.p2 = p2
    self.winner = winner
```

class Tournament:

```
def __init__(self, player_max):
    self.outcomes = { n : {"wins": [], "losses": []} for n in range(1, player_max + 1)}
```

```
def add(self, game):
    if game.winner == game.p1:
        self.outcomes[game.p1]["wins"].append(game.p2)
        self.outcomes[game.p2]["losses"].append(game.p1)
    else:
        self.outcomes[game.p1]["losses"].append(game.p2)
        self.outcomes[game.p2]["wins"].append(game.p1)
```

```
t = Tournament(6) # max player
```

```
#add our games, with winners
```

```
t.add(Game(1,2,2))
t.add(Game(2,4,4))
t.add(Game(2,6,6))
t.add(Game(3,4,3))
t.add(Game(3,5,3))
t.add(Game(5,2,2))
t.add(Game(5,6,5))
t.add(Game(6,1,1))
```

```
#see whom player 2 has beaten
```

```
print(t.outcomes[2]["wins"])
```

If you need to also be able to traverse the tournament and see games from top to bottom, then a tree like structure can be used where each game node has two players and that links to another game node.

Willful - April 7, 2018, 11:33 pm

(Apr 6, 2018 01:11 PM)m0noxide Wrote: [CITE](#) I get that this is in the Java section but I made a quick prototype in Python to show you one way you get the winners and losers given a player. I decided to use a hashmap but you could just use simple arrays instead.

@OP, here's the Java equivalent of what m0noxide did with some slight modifications to the Game object.

```
Code import java.util.ArrayList;
```

```
public class Player {
    private ArrayList<Integer> wins;
    private ArrayList<Integer> losses;

    public Player() {
        this.wins = new ArrayList<Integer>();
        this.losses = new ArrayList<Integer>();
    }

    public void addWin(Integer loser) {
        this.wins.add(loser);
    }

    public void addLoss(Integer winner) {
        this.losses.add(winner);
    }
}
```

```

    public ArrayList<Integer> getWins() {
        return this.wins;
    }

    public ArrayList<Integer> getLosses() {
        return this.losses;
    }
}

public class Game {
    private Integer winner;
    private Integer loser;

    public Game(Integer winner, Integer loser) {
        this.winner = winner;
        this.loser = loser;
    }

    public Integer getWinner() {
        return winner;
    }

    public Integer getLoser() {
        return loser;
    }
}

import java.util.HashMap;

public class Tournament {
    private HashMap<Integer, Player> outcomes;

    public Tournament(Integer maxPlayers) {
        this.outcomes = new HashMap<Integer, Player>();
        for (int playerNumber = 1; playerNumber < maxPlayers + 1; playerNumber++) {
            this.outcomes.put(playerNumber, new Player());
        }
    }

    public void add(Game game) {
        Integer winner = game.getWinner();
        Integer loser = game.getLoser();
        this.outcomes.get(winner).addWin(loser);
        this.outcomes.get(loser).addLoss(winner);
    }

    public HashMap<Integer, Player> getOutcomes() {
        return this.outcomes;
    }
}

public class TournamentRunner {

```

```
public static void main(String[] args) {  
    Tournament t = new Tournament(6);  
    t.add(new Game(2, 1));  
    t.add(new Game(4, 2));  
    t.add(new Game(6, 2));  
    t.add(new Game(3, 4));  
    t.add(new Game(3, 5));  
    t.add(new Game(2, 5));  
    t.add(new Game(5, 6));  
    t.add(new Game(6, 1));  
  
    //see whom player 2 has beaten  
    System.out.println(t.getOutcomes().get(2).getWins().toString());  
}  
}
```