Aufgabe 3: Tobis Turnier

Teilnahme-Id: 56043

Bearbeiter/-in dieser Aufgabe:
Sebastian Brunnert

21. September 2020

Inhaltsverzeichnis

Lösungsidee

Tobi möchte wissen, bei welcher der drei Turnierformen der spielstärkste Spieler, am häufigsten gewinnt. Diese Aufgabe wäre eigentlich genau stochastisch lösbar. Doch da zu viele Variblen auftreten (z.B. die Reihenfolge der Spieler bei einem Turnier oder jedes einzelne Spiel-Ergebnis in der Liga) und dadurch die Wahrscheinlichkeit beeinflussen, ist (beispielsweise) das Baumdiagramm zu groß und komplex, um dieses durch Algorithmen und einem Computer lösen zu lassen. Aufgrund dessen, wurde der Ansatz gewählt, sich durch besonders viele Versuche sich in etwa an der tatsächlichen Wahrscheinlichkeit anzunähern. Also wird konkret anfangs der spielstärkste Spieler ermittelt und jede Turnierform wird 100.000 mal oder öfters wiederholt. Dabei wird gemessen, wie oft der zuvor ermittelte stärkste Spieler gewinnt. Daraus kann die zu suchende Wahrscheinlichkeit errechnet werden und es kann auch abgeleitet werden, welche Turnierform Tobi wählen sollte.

Lösungsidee K.O.-Turnier

Nun muss konkret für jede der drei Turnierformen überlegt werden, wie ein einzelner Durchlauf simuliert werden sollte. Bei dem K.O.-Turnier werden also die Spieler anfangs aufgelistet. Da es sich bei der Anzahl der Spieler immer um eine Zweierpotenz handelt, können immer zwei Spieler zu einem eindeutig Paar zugeordnet werden können. Die Paare werrden linear gebildet. Das heißt, dass die ersten beiden Spieler aus der Liste ein Paar bilden; die Spieler mit den Positionen 3 und 4 ergeben ein Paar usw. Die Verlierer der Partien werden anschließend entfernt, sodass neue Paare aus dieser Liste gebildet werden können, bis schließlich nur noch zwei Spieler in der Liste enthalten sind: Das Finale kann nun also aussgetragen werden, um den Sieger zu ermitteln.

Lösungsidee K.O.x5-Turnier

Letztlich kann ein solches Turnier exakt wie ein einfaches K.O.-Turnier ausgetragen werden, mit dem Unterschied, dass die Gewinner einer Zweier-Partie nicht durch eine Partie ermittelt werden, sondern – wie beschrieben – durch fünf Partien. Es muss also für jede Partie gesichert werden, wie oft einer der beiden Spieler gewonnen hat.

Lösungsidee Liga

Um ein Liga Turnier austragen zu können, muss jeder Spieler gegen jeden Spieler spielen. Daher muss die Liste der Spieler ineinander zwei mal iteriert werden. Dabei muss geprüft werden, ob eine Partie nicht möglich ist

auszutragen, da entweder die Partie schon ausgetragen wurde (A gegen B entspricht B gegen A) oder, da ein Spieler gegen sich selber spielen soll. Andernfalls wird ermittelt, welcher Spieler gewinnt und diesem wird ein Sieg zugesprochen.

Umsetzung

Die Umsetzung dieser Aufgabe geschieht in der Programmiersprache Java (11). Zu Beginn des Programmes werden die Spielerstärken aus der angegeben Spielerdatei ausgelesen. Für jeden Spieler wird eine Objekt-Instanz Spieler erstellt. Diese enthält neben einem Namen (bestehend aus „Spieler“ und zuordbarer Index-Number des Spielers) und natürlich der Spielerstärke des Spielers, die Hilfs-Methode gegenSpieler(Spieler gegner). Diese gibt den Spieler (Objekt selbst oder Spieler gegner) zurück, welcher durch das beschriebene Zufallsexpirement gewinnt. Dazu wird eine Zufallszahl zwischen (Spielerstärke Spieler 1 + Spielerstärke Spieler 2) generiert und überprüft, in welchem Intervall (0 bis Spielerstärke Spieler 1 bzw. Spielerstärke Spieler 2 bis Ende) liegt. Daraus resultiert auch der Gewinner des Zufallsexpirmentes. Darüber hinaus existiert die Methode gegenSpielerGibVerliererZurueck(Spieler gegner), welche ebenso funktioniert, aber den Verlierer der Partie zurückgibt.

Nachdem nun alle Spieler eingelesen worden sind, wird der spielstärkste Spieler durch einen for-Lop ermittelt. Also der Spieler, der möglichst am häufigsten gewinnen sollte.

Nun beginnt der Hauptteil des Programmes: Alle drei Turnierformen werden 100.000 mal simuliert und der Sieger wird ermittelt. Gewinnt der zuvor ermittelte spielstärkste Spieler, so wird eine Zahl n um 1 erhöht. Mit dieser Zahl der Siege kann nun die angenäherte Wahrscheinlichkeit der Siege des spielstärksten Spielers ermittelt und ausgegeben werden.

Alle drei Turnierformen werden als eigenständige Klassen angesehen. Bei dem K.O.-Turnier werden im Konstruktor alle aus der Datei ermittelten Spieler in die objekt-interne Liste aktiveSpieler kopiert und mittels java.utils.Collections gemischt, damit eine zufällige Einteilung der anfänglichen Partien erreicht wird. Um jetzt nun den Sieger zu ermitteln, wird erst jener Teil des Turniers bis zum Finale simuliert (also solange mehr als zwei Spieler „überlebt“ haben). In jedem Abschnitt des Turniers (Achtelfinale, Viertelfinale …) wird außerdem eine leere Liste verliererListe geniert, die als Buffer-Liste dient, um zu sichern, welche Spieler ausgeschieden sind. Diese dürfen erst nach dem Austragen aller Partien dieses Abschnittes des Turniers entfernt werden, da sonst die Liste aktiveSpieler variiert wird. Es wird also durch diese Liste mittels einer for-Schleife iteriert, welche bei jedem Durchgang um zwei erhöht wird. Dadurch werden die Partien klar abgegrenzt und diese können nun ausgetragen. Der Verlierer der Partie wird zur Buffer-Liste verliererListe hinzugefügt, um später entfernt zu werden. Wenn schlussendlich nur noch zwei Spieler „überlebt“ haben, wird der Final-Sieger ermittelt und zurückzugeben.

Die Turnierform K.O.x5-Turnier wird letztlich genau so ausgetragen. Beim Durchführen einer Partie wird allerdings der lokale Integer spieler1 eingeführt. Dieser stellt die Siege des vorne liegenden Spielers dar. Es wird also nach dem fünfmaligen gegeneinander Antretens geprüft, ob spieler1 > 2 ist, also ob dieser erste Spieler gewonnen hat.

Die Turnierform Liga wird so durchgeführt, das anfangs alle Spieler in eine Map als Key hinzugefügt werden. Der Value ist die Anzhal der Siege (anfangs 0). Um nun den Sieger ermitteln zu können werden alle Spieler (als spieler1 und spieler2) in zwei ineinander verschachtelten for-Loops iteriert. Es wird nun geprüft, ob diese Partie bereits durchgeführt wurde (ein String der die Partie eindeutig beschreibt wird in der Liste durchgeführtePartien gespeichert) oder spieler1 == spieler2 gilt. Wenn beides nicht der Fall ist, wird nun die Partie durchgeführt und dem Sieger wird in seinem Value in der Map ein Sieg zugesprochen. Zuletzt wird dann der Spieler mit den meisten Siegen ermittelt und zurückgegeben.

Alle Ergebnisse werden in der Konsole ausgegeben.

Beispiele

Empfhelung: Auf Basis der Ergebnisse kann gesagt werden, dass im einfachen K.O.-Turnier der spielstärkste Spieler eher weniger oft gewinnt. Bei vielen eher stärkeren Spielern kann gesagt werden, dass die Liga zu einem genaueren Ergebnis für den spielstärksten Spieler führt.

spielstaerken1.txt	Es wurden erfolgreich 9 Spieler eingelesen worden. Spielstärkster Spieler ist: Spieler 9 100000 mal K.O.-Turnier durchführen... In 38.199% der Fällen gewinnt der spielstärkste Spieler bei der K.O.-Turnierform. 100000 mal K.O.x5-Turnier durchführen... In 51.932% der Fällen gewinnt der spielstärkste Spieler bei der K.O.x5-Turnierform. 100000 mal Liga durchführen... In 48.364000000000004% der Fällen gewinnt der spielstärkste Spieler bei Liga-Turnierform.
spielstaerken2.txt	Es wurden erfolgreich 9 Spieler eingelesen worden. Spielstärkster Spieler ist: Spieler 9 100000 mal K.O.-Turnier durchführen... In 28.971999999999998% der Fällen gewinnt der spielstärkste Spieler bei der K.O.-Turnierform. 100000 mal K.O.x5-Turnier durchführen... In 32.534% der Fällen gewinnt der spielstärkste Spieler bei der K.O.x5-Turnierform. 100000 mal Liga durchführen... In 34.435% der Fällen gewinnt der spielstärkste Spieler bei Liga-Turnierform.
spielstaerken3.txt	Es wurden erfolgreich 17 Spieler eingelsesen worden. Spielstärkster Spieler ist: Spieler 5 100000 mal K.O.-Turnier durchführen... In 14.431% der Fällen gewinnt der spielstärkste Spieler bei der K.O.-Turnierform. 100000 mal K.O.x5-Turnier durchführen... In 21.833% der Fällen gewinnt der spielstärkste Spieler bei der K.O.x5-Turnierform. 100000 mal Liga durchführen... In 21.363% der Fällen gewinnt der spielstärkste Spieler bei Liga-Turnierform.
spielstaerken4.txt	Es wurden erfolgreich 17 Spieler eingelesen worden. Spielstärkster Spieler ist: Spieler 2 100000 mal K.O.-Turnier durchführen... In 6.557% der Fällen gewinnt der spielstärkste Spieler bei der K.O.-Turnierform. 100000 mal K.O.x5-Turnier durchführen... In 6.938% der Fällen gewinnt der spielstärkste Spieler bei der K.O.x5-Turnierform. 100000 mal Liga durchführen... In 11.458% der Fällen gewinnt der spielstärkste Spieler bei Liga-Turnierform.

Quellcode

Der vollständige Quellcode, ist in der Ordner Quellcode zu finden.

TobisTurnier.java:

```
System.out.println("Es wurden erfolgreich " + spieler.size() + " Spieler eingelesen worden.");

// Nun, wo alle Spieler erfolgreich eingelesen worden sind, soll ermittelt werden, welcher der spielstärkste Spieler ist. Also der,
// der über viele Wiederholungen am häufigsten gewinnen soll. Dafür wird durch alle Spieler einmal durchgegangen und ...
for(Spieler spieler : getSpieler()){
   if(staerksterSpieler == null) {
      // ... wenn noch kein Spieler als stärkster Spieler definiert wurde, wird es
      // Hier wird also der Spieler, der in der Liste an vorderster Stelle steht erstmal definiert
      staerksterSpieler = spieler;
   } else {
      // ... nun wird überprüft ob ein Spieler nach dem ersten Spieler eine höhere Spielstärke hat als der gerade stärkste Spieler
      if(staerksterSpieler.getSpielerStaerke() < spieler.getSpielerStaerke()){
         staerksterSpieler = spieler;
      }
   }
}

System.out.println("Spielstärkster Spieler ist: " + staerksterSpieler.getName());

// Nun wird jede Turnierform 1000 mal durchgeführt, dabei wird gezählt wie oft bei der Turnierform jeweils der spielstärkste Spieler gewinnt:

double siegeFürSpielstaerkstenSpieler = 0;

System.out.println("\n100000 mal K.O.-Turnier durchführen...");
for(int i = 0; i < 100000; i++){
   KoTurnier koTurnier = new KoTurnier();
```

```
        // Wenn bei dieser Turnier-Form der spielstärkste Spieler gewinnt wird siegeFürSpielstaerkstenSpieler um 1 erhöht
        if(koTurnier.ermittleSieger().equals(staerksterSpieler)){
            siegeFürSpielstaerkstenSpieler++;
        }
    }
    System.out.println("In " + (siegeFürSpielstaerkstenSpieler / 100000) * 100 + "% der Fällen gewinnt der spielstärkste Spieler bei der K.O.-Turnierform.");


    siegeFürSpielstaerkstenSpieler = 0;
    System.out.println("\n100000 mal K.O.x5-Turnier durchführen...");
    for(int i = 0; i < 100000; i++){
        Kox5Turnier kox5Turnier = new Kox5Turnier();
        if(kox5Turnier.ermittleSieger().equals(staerksterSpieler)){
            siegeFürSpielstaerkstenSpieler++;
        }
    }
    System.out.println("In " + (siegeFürSpielstaerkstenSpieler / 100000) * 100 + "% der Fällen gewinnt der spielstärkste Spieler bei der K.O.x5-Turnierform.");


    siegeFürSpielstaerkstenSpieler = 0;
    System.out.println("\n100000 mal Liga durchführen...");
    for(int i = 0; i < 100000; i++){
        Liga liga = new Liga();
        if(liga.ermittleSieger().equals(staerksterSpieler)){
            siegeFürSpielstaerkstenSpieler++;
        }
    }
    System.out.println("In " + (siegeFürSpielstaerkstenSpieler / 100000) * 100 + "% der Fällen gewinnt der spielstärkste Spieler bei Liga-Turnierform.");
```

Spieler.java:gegenSpieler(Spieler gegner):

```
// Der Spieler besitzt eine Methode mit der direkt ein Spiel gegen einen anderen Spieler simuliert werden kann
public Spieler gegenSpieler(Spieler gegner){

    // Es wird dabei erst die gesammte Anzahl an Kugeln (die gezogen werden können) errechnet
    // Nun wird eine zufällige Zahl von 0-Total generiert
    int random = TobisTurnier.getRandom().nextInt(getSpielerStaerke() + gegner.getSpielerStaerke()) + 1;

    // Nun wird geschaut in welchem Intervall diese zufällig gezugene Kugel liegt
    if(random > this.getSpielerStaerke()){
        // Außerhalb: Gegner hat gewonnen und wird zurückgegeben
        return gegner;
    } else {
        return this;
    }
}
```

KoTurnier.java:

```
private List<Spieler> aktiveSpieler;

public KoTurnier() {
    // Kopiere alle Spieler in die Liste mit den (noch) "überlebenden" Spielern
    this.aktiveSpieler = new ArrayList<>(TobisTurnier.getSpieler());
    // Diese Liste wird durchgemischt, um zufällige Gegner am Anfang zu ermitteln.
    Collections.shuffle(this.aktiveSpieler);
}

public Spieler ermittleSieger() {
    // Ein Schleifendurchgang symboliesiert hier eine Runde
    // Solange bis nur noch zwei Finalisten überleben...
    while (aktiveSpieler.size() > 2){

        // Da die Verlierer erst entfert werden dürfen, nachdem alle Verlierer einer Runde klar sind, wird eine Buffer-Liste eingeführt
```

```
        List<Spieler> verliererListe = new ArrayList<>();

        // Jede Partie in dieser Runde symbolisiert einen Schleifendurchgang, deshalb wird jeweils +2 zu i gerechnet
        for(int i = 0; i < aktiveSpieler.size()-1; i += 2){
            verliererListe.add( aktiveSpieler.get(i).gegenSpielerGibVerliererZurueck(aktiveSpieler.get(i+1)) );
        }

        // Verlierer werden entfernt
        for(Spieler verlierer : verliererListe){
            aktiveSpieler.remove(verlierer);
        }

    }

    // Der Gewinner des Finales wird zurückgegeben
    return aktiveSpieler.get(0).gegenSpieler(aktiveSpieler.get(1));

  }
```

Liga.java:

```
  // HashMap mit den Spielern in dieser Liga
  // Key: Spieler-Instanz
  // Value: Siege
  private HashMap<Spieler, Integer> spieler;

  public Liga() {
    this.spieler = new HashMap<Spieler,Integer>();
    // Alle Spieler werden in die HashMap geladen
    for(Spieler spieler : TobisTurnier.getSpieler()){
      this.spieler.put(spieler,0);
    }
  }
```

```
public Spieler ermittleSieger() {
    List<String> durchgeführtePartien = new ArrayList<>();

    // Durch die Map wird zweimal durchgegangen
    for(Spieler spieler1 : this.spieler.keySet()){
        for(Spieler spieler2 : this.spieler.keySet()){
            // Da ein Spieler nicht gegen sich selbst spielen kann, wird geprüft, ob spieler1 ungleich spieler2
            if(spieler1 != spieler2){
                // Außerdem wird geprüft, ob diese Partie schon durchgeführt wurde
                if(!(durchgeführtePartien.contains(spieler1.getName() + " vs " + spieler2.getName()) ||
durchgeführtePartien.contains(spieler2.getName() + " vs " + spieler1.getName()))){
                    // Jede Partie bekommt einen nachvollziehbaren Namen und wird zur Liste durchgeführtePartien
hinzufügegt
                    durchgeführtePartien.add(spieler1.getName() + " vs " + spieler2.getName());

                    // Der Gewinner der Partie wird ermittelt und ein Sieg wird hinzugefügt
                    Spieler gewinnerPartie = spieler1.gegenSpieler(spieler2);
                    this.spieler.put(gewinnerPartie, this.spieler.get(gewinnerPartie)+1);

                }
            }
        }
    }

    // Ermitteln des Spielers mit den meisten Siegen bzw. mit der kleinsten Spielernummer nach Verfahren wie
in TobisTurnier:55 beschrieben
    Spieler meisteSiege = null;
    for(Spieler spieler : this.spieler.keySet()){
        if(meisteSiege == null){
            meisteSiege = spieler;
        }
        // Die Spielernummer braucht nicht verglichen werden, da die Map der Spieler bereits sortiert ist, da sie
nie gemischt wurde
        if(this.spieler.get(spieler) > this.spieler.get(meisteSiege)){
            meisteSiege = spieler;
```

```
            }
        }

        return meisteSiege;

    }
```