Aufgabe 3: Tobis Turnier

Team-ID: 00537

Team-Name: LuC++

Bearbeiter dieser Aufgabe:
Lucas Schwebler

14. November 2020

Inhaltsverzeichnis

1 Lösungsidee **1**
1.1 Grundlegende Überlegungen . . . 1
1.2 Nummerierung und Spielplanerstellung . . . 1

2 Umsetzung **2**
2.1 Urnenmodell und Mischen . . . 2
2.2 Ligasystem . . . 2
2.3 K.O. . . . 2
2.4 K.O.x5 . . . 2

3 Ergebnisse und Auswertung **2**
3.1 Verwendung des Programms . . . 2
3.2 Beispielausgabe im K.O.-System . . . 3
3.3 Ergebnisse . . . 3
3.4 Erklärung der Ergebnisse . . . 4
3.5 Fazit / Antwort . . . 4

4 Quellcode **4**

1 Lösungsidee

1.1 Grundlegende Überlegungen

Die Lösungsidee zu dieser Aufgabe besteht darin, die vorgegebenen Turniervarianten zu implementieren. Dabei wird jede Variante n mal simuliert, wonach die Siege von Spieler i durch n geteilt werden, um die durchschnittliche Siegesquote zu erhalten. Um möglichst gute Ergebnisse zu erhalten, aber nicht länger als ein paar Sekunden auf diese warten zu müssen, wird $n = 10^6$ gewählt.

Nach Aufgabenstellung soll ermittelt werden, wie oft der spielstärkste Spieler gewinnt. Da die Aufgabenstellung aber nicht ausschließt, dass es mehrere Spieler mit maximaler Spielstärke gibt, kann es passieren, dass kein Spielstärkster Spieler existiert. In diesem Fall wird von denen spielstärksten Spielern der erste in der Liste genommen.

1.2 Nummerierung und Spielplanerstellung

Die Anordnung von Spielplänen kann bei K.O. eine wichtige Rolle spielen, wenn zum Beispiel der beste Spieler nur gegen gute Spieler spielt, wird er es schwerer haben als ein Spieler, der nur leichte Begegnungen

hat und erst im Finale auf den besten Spieler treffen kann. Daher wird diese Anordnung zufällig generiert, um einen zu großen Einfluss durch die Anordnung der Spieler in den Inputdateien zu erhalten.

Dasselbe gilt für das Ligasystem und die Nummerierungen. So heißt es zwar, dass Zeile i die Spielstärke von Spieler i enthält, womit die Nummerierung der Spieler bereits vorgegeben ist, aber diese Nummerierung hat einen großen Einfluss, da sie bei Gleichstand (was nicht selten Eintritt) über den Sieger entscheidet. Daher wird die Nummerierung bei jeder Simulation des Ligasystems zufällig neu erzeugt.

2 Umsetzung

2.1 Urnenmodell und Mischen

Um eine Partie zwischen Spielern mit Spielstärken a und b zu simulieren, wird eine Zufallszahl $z \in [1\ ;\ a+b]$ generiert, wobei alle möglichen z gleich wahrscheinlich sind.[1] Gehören nun die erstem a Kugeln dem ersten Spieler, so hat er gewonnen, wenn $z \leq a$. Sonst hat Spieler zwei gewonnen.

Zum Mischen einer Liste, was für die Generierung des K.O. Spielplans benötigt wird, wird ein Fisher-Yates Shuffle verwendet. Man iteriert über alle Indizes i der Liste und tauscht die Elemente an den Indices i und j, wobei j zufällig aus $[i\ ;\ n-1]$ gewählt wurde (n ist die Länge der 0-indizierten Liste).

2.2 Ligasystem

Sei $siege[i]$ die Anzahl an Siegen von Spieler i. Nun wird über alle Paare (u, v) mit $u < v$ iteriert[2] und die Partie zwischen u und v mit Sieger x simuliert. Danach wird $siege[x]$ um 1 erhöht.

Danach wird der Spieler mit kleinster Nummer i gesucht, für den $siege[i]$ maximal ist[3]. Dieser Spieler ist dann der Gewinner.

2.3 K.O.

Mit dem Fisher-Yates shuffle wird zunächst ein Turnierplan erstellt, indem die Spielernummern gemischt werden. In jeder Iteration spielt nun der Spieler an Position $2i$ gegen den auf $2i+1$ und dies für alle $i < \frac{n}{2}$, wobei n die Größe der Liste der Spieler ist, die noch am Turnier teilnehmen. Alle Spieler, die verlieren, werden aus der Liste entfernt[4]. Danach beginnt eine neue Iteration mit halbiertem n. Sobald $n = 1$ ist nur noch ein Spieler übrig, nämlich der Sieger.

2.4 K.O.x5

Es wird der allgemeine Fall betrachtet: Zwei Spieler mit Spielstärken a und b spielen k Spieler gegeneinander, wobei a genau x mal gewinnt. Ist nun $x > \frac{k}{2}$, so hat Spieler a mehr Spiele gewonnen, als Spieler b und kommt eine Runde weiter. Für K.O.x5 müssen wir also nur $k := 5$ setzen.

3 Ergebnisse und Auswertung

3.1 Verwendung des Programms

Zur Verwendung des Programms:

```
turnier.exe [ko | liga] <arg> <n> -v
- Bei ko legt arg die Anzahl an Spielen pro Runde fest. (1 oder 5)
- Bei liga ist arg ein boolean Wert -> Sollen die Nummern jedes Mal neu generiert werden?
- arg hat default 1
- n ist die Anzahl der Simulationen (default 1.000.000)
- -v ("verbose") gibt die Siegesrate für alle Spieler, nicht nur für den besten aus.
```

Die Spielstärken stehen in der Datei *turnier.in* und bei Erfolg wird die Ausgabe in *turnier.out* geschrieben.

[1] Verwendet wird hierzu der Mersenne-Twister mt19937 mit der uniform_int_distribution(1, $a+b$).
[2] Dies wird über zwei verschachtelte Schleifen realisiert.
[3] Hierzu wird in Linearzeit über die Liste iteriert und der aktuelle Sieger zwischengespeichert. Da die Nummern aufsteigend betrachtet werden, wird der Sieger nur dann aktualisiert, wenn der nächste Spieler mehr Siege hat.
[4] Um die Spieler zu entfernen, werden stattdessen die Sieger separat gespeichert, wonach die Liste durch diese ersetzt wird.

3.2 Beispielausgabe im K.O.-System

```
spielstaerken1.txt
turnier.exe ko 1 1000000 -v

Spielstärke  Siege
0:       0.00%
10:   0.84%
20:   3.36%
30:   7.11%
40:   11.41%
50:   16.06%
60:   20.92%
100:   40.29%


Ab jetzt: turnier.exe ko

spielstaerken2.txt

Siege: 30.15%

spielstaerken3.txt

Siege: 16.73%

spielstaerken4.txt

Siege: 6.91%
```

3.3 Ergebnisse

Beispiel	K.O.	K.O.x5	Liga (ohne Mischen)	Liga (mit Mischen)
spielstaerken1.txt	40,29%	60,11%	34,59%	45,84%
spielstaerken2.txt	30,15%	35,95%	21,01%	31,71%
spielstaerken3.txt	16,73%	27,75%	31,57%	26,01%
spielstaerken4.txt	6,91%	7,54%	11,45%	7,41%

Natürlich ist die Turniervariante, bei der der beste Spieler am häufigsten gewinnt diejenige, die am besten geeignet ist, um ein faires Turnier zu veranstalten. Es ist also offensichtlich, dass K.O.x5 besser ist als gewöhnliches K.O., da bei K.O.x5 bei allen Beispielen der beste Spieler häufiger gewinnt.

Bei Betrachtung der Ligen muss man jedoch vorsichtig sein, da bei den ersten Beispielen die Variante mit Mischen besser ist und bei den anderen die ohne Mischen. Dies liegt daran, dass in den ersten beiden Beispielen der beste Spieler die letzte Spielernummer hat und der bei den anderen weit vorne steht. Der Unterschied ist also auf den Einfluss von Unentschieden und dem Sieg des Spielers mit geringerer Nummer zurückzuführen. Wenn praktisch ein Turnier veranstaltet wird, so ist es egal auf welche der Ligavarianten zurückgegriffen wird, da sie bei einem einzigen Turnier äquivalent sind, weil gar kein Mischen zwischen den Runden stattfinden kann, wenn nur eine existiert. Da die Nummern aber in der Realität vermutlich zufällig gewählt werden, liefert die Variante mit Mischen bessere Ergebnisse, denn der Zufallsfaktor durch die Nummern wurde hier durch viele Simulationen mit unterschiedlicher Zuteilung entfernt.

Vergleicht man mit dem K.O. System, so ist das Ligasystem besser. Verglichen mit dem K.O.x5 System ist das Ligasystem aber schlechter.

Zusammenfassend lässt sich also sagen, dass die Ergebnisse nahelegen, dass diese dies die Rangliste der Turniersysteme ist (je besser, desto weiter oben)

1. K.O.x5
2. Liga
3. K.O.

3.4 Erklärung der Ergebnisse

Generell führen mehr Spiele zu einer besseren Chance für den besten Spieler, da eine Abweichung vom Erwartungswert, die groß genug ist, damit ein schlechterer Spieler gewinnt, immer unwahrscheinlicher wird. Zur Verdeutlichung: Lassen wir einen Spieler mit Spielstärke $a = 51$ gegen einen mit $b = 49$ antreten und gewonnen hat, wer von n Spielen mehr als $\frac{n}{2}$ gewinnt. Nach Bernoulli erhalten wir:

$$P(\text{Spieler 1 gewinnt}) = \sum_{k=\lfloor \frac{n}{2}+1 \rfloor}^{n} \binom{n}{k} \cdot 0.51^k \cdot 0.49^{n-k}$$

Bei $n = 1$ ist diese Wahrscheinlichkeit offensichtlich 0,51. Bei $n = 1001$ ist die Wahrscheinlichkeit, dass Spieler 1 gewinnt etwa 0,715. Dies liegt daran, dass nun häufiger der leicht unwahrscheinlichere Fall eintreten muss, dass Spieler 2 gewinnt, damit sich dieser auch den Gesamtsieg sichern kann.

Dies erklärt, wieso bei K.O.x5 häufiger der beste Spieler gewinnt, als bei K.O.: Es werden mehr Spiele gespielt.

Um einen Vergleich mit dem Ligasystem herstellen zu können, müssen wir berücksichtigen, dass die Anzahl der Spiele n, die der Sieger spielt, im K.O. System logarithmisch zur Anzahl der Spieler s wächst. Beim Ligasystem ist hingegen $n = s$. Das bedeutet, dass die Anzahl an Spielen bei kleiner Spieleranzahl ähnlich sein wird. Ferner ist $5 \cdot log_2(16) = 20 > 16$, womit bei den Testdaten sogar bei K.O.x5 immer mehr Spiele gespielt werden als beim Ligasystem., was ein Grund für den Vorteil von K.O.x5 ist. Gleichzeitig ist aber natürlich $s > log_2(s)$, was den Vorteil von Liga gegenüber K.O. erklärt. Bei deutlich größeren s könnte das Ligasystem aber im Vorteil sein, da dann $s >> 5 \cdot log_2(s)$.

3.5 Fazit / Antwort

Tobi sollte für das Turnier das K.O.x5 System verwenden, da hier der beste Spieler in der Regel die größten Gewinnchancen hat. Sollte er aber ein deutlich größeres Turnier mit tausenden oder millionen Teilnehmern planen, könnte es sein, dass das Ligasystem die bessere Wahl ist.

4 Quellcode

```
mt19937 mt(time(0));

bool wins(int a, int b)
{
    // Kugel element [1 ; a+b]
    uniform_int_distribution<int> dist(1, a + b);
    // Die ersten a Kugeln gehören Spieler 1
    return dist(mt) <= a;
}

// Fisher-Yates Shuffle
template<typename T>
void shuffle(vector<T>& v)
{
    for(int i = 0; i < v.size()-1; ++i)
    {
        int pos = uniform_int_distribution<int>(i, v.size()-1)(mt);
        swap(v[i], v[pos]);
    }
}

int n;
vector<pair<int, int>> spieler;

void read_input()
{
    ifstream fin("turnier.in");
    fin >> n;
    spieler.resize(n);
    for(int i = 0; i < n; ++i) fin >> spieler[i].first, spieler[i].second = i;
    fin.close();
}

int ko(int spiele)
```

```
{
    vector<pair<int, int>> sp = spieler;
    shuffle(sp);

    // Wenn nur noch ein Spieler übrig ist, hat dieser gewonnen
    while(sp.size() > 1)
    {
        vector<pair<int, int>> sieger;
        for(int i = 0; i < sp.size(); i += 2)
        {
            // Spiele mehrere Spiele (1 oder 5)
            int a = 0;
            for(int j = 0; j < spiele; ++j)
            {
                a += wins(sp[i].first, sp[i+1].first);
            }
            // Der Sieger kommt in die nächste Runde
            if(a > spiele / 2) sieger.push_back(sp[i]);
            else sieger.push_back(sp[i+1]);
        }
        sp = sieger;
    }

    return sp[0].second;
}

int liga(bool zufalls_nummern)
{
    vector<pair<int, int>> sp = spieler;
    if(zufalls_nummern) shuffle(sp);

    vector<pair<int, int>> siege(n);
    for(int i = 0; i < n; ++i)
    {
        siege[i].second = i;
    }

    // iteriere über alle ungeordneten Paare verschiedener Spieler
    for(int i = 0; i < n; ++i)
    {
        for(int j = i+1; j < n; ++j)
        {
            // Der Sieger erhält einen Punkt
            if(wins(sp[i].first, sp[j].first)) siege[i].first++;
            else siege[j].first++;
        }
    }

    int sieger = 0;
    for(auto& p : siege)
    {
        // Da die Spieler ihrer Spielernummer nach betrachtet werdem,
        // hat bei Gleichstand der Spieler mit gerigerer Nummer gewonnen.
        if(p.first > siege[sieger].first) sieger = p.second;
    }

    return sp[sieger].second;
}
```