Aufgabe 3: Tobis Turnier

00759 - TranshumanTechnocrator - Erik Klein

1 Aufgabe

Gegeben sind N Spielstärken zwischen 0...100. Gefragt ist, wie wahrscheinlich es für drei gegebene Turniervarianten ist, dass ein bester Spieler (ein Spieler mit maximaler Spielstärke) gewinnt. Dann ist noch herauszufinden, welche Turniervariante für die Eingabe am wahrscheinlichsten einen besten Spieler herausfindet.

2 Lösungsidee

Meine Idee war, die Monte-Carlo-Methode anzuwenden, und das Spiel $ITER$ Mal zu simulieren. Für jeden Durchlauf will ich für jede Turniervariante die Spielstärke des gewinnenden Spielers wissen, um dann zu zählen, wie oft ein Spieler mit maximaler Spielstärke gewonnen hat, und die Wahrscheinlichkeiten zu berechnen.

Für die Liga gehe ich alle Paare von Spielern durch, lasse sie spielen und verwalte für jeden Spieler eine Zähler, wie oft er schon gewonnen hat. Dann finde ich den Spieler mit dem kleinsten Index mit dem größten Zähler und gebe seine Stärke zurück. Für den K.O. Modus arbeite ich nur mit Stärken, und verwalte einen Turnierbaum in einem Heap. Index 1 ist der Gewinner, Indices N...2N-1 die anfänglichen Spielstärken. Ich lasse jeweils Indices $2 \cdot i$ und $2 \cdot i + 1$ gegeneinander Spielen um den Gewinner an Index i zu ermitteln. Ich lasse die Spieler entweder 1 oder 5 Mal pro Match spielen, und gebe die Stärke die am häufigsten gewonnen hat, zurück.

3 Umsetzung

Die Lösungsidee wurde in C++ implementiert. Erst werden die Stärken eingelesen, wobei auch die maximale Spielstärle $smax$ berechnet wird. Zum simulieren eines gegeneinander Spielens, dem Herausziehen von einer Kugel aus einer Urne, wird die Funktion $rand()$ verwendet. Mit $rand()\%(s_i + s_j)$ kann das Herausnehmen aus einer Urne geeigneter Größe simuliert werden. Anfangs wurde mit $srand(time(NULL))$ ein Seed gesetzt. Außerdem werden in jedem Durchlauf mit $random\_shuffle()$ die Paarungen neu gewürfelt.

```
cin >> N;
for(int i = 0; i < N; i++) {
    cin >> s[i];      // spielstaerken einlesen
    smax = max(smax,s[i]); // maximale staerke
}
srand(time(NULL));
```

Die Funktion $liga()$ simuliert einen Durchlauf im Liga-Modus wie oben und in den Quelltextkommentaren beschrieben:

```
int liga() {
    // gibt staerke des gewinnenden spielers
    vector<int> siege (N);
    for(int j = 0; j < N; j++)
        for(int i = 0; i < j; i++) {
            // spieler i gegen spieler j
            // s[i]+s[j] kugeln liegen in urne
            // kugeln nummeriert 0...s[i]+s[j]-1
            // kugel ziehen:
            int rd = rand()%(s[i]+s[j]);
```

```
            if(rd < s[i]) siege[i]++; // i gewinnt
            else          siege[j]++; // j gewinnt
        }
    // besten spieler finden
    int best = 0; // wie oft er gewonnen hat
    int loc = 0;  // sein index
    for(int i = 0; i < N; i++)
        if (best < siege[i]) {
            best = siege[i];
            loc = i;
        }
    return s[loc]; // seine staerke zurueckgeben
}
```

Die Funktion *match*() lässt zwei Spielstärken 1 oder 5 Mal gegeneinander spielen und gibt den Gewinner zurück:

```
int match(int si, int sj, int gamespermatch) {
    // gibt staerke des gewinnenden spielers
    int cnti = 0, cntj = 0; // counter
    for(int g = 0; g < gamespermatch; g++) {
        // alle spiele spielen
        int rd = rand()%(si+sj);
        if(rd < si) cnti++; // i gewinnt
        else        cntj++; // j gewinnt
    }
    // am ende pruefen, wer oefter gewonnen
    // seine staerke zurueckgeben
    if(cnti > cntj) return si;
    else            return sj;
}
```

Die Funktion *ko*() simuliert wie in der Lösungsidee beschrieben (mit Turnierbaum und Heap) einen Durchlauf in K.O.-Modus, und benutzt dabei *match*().

```
int ko(int gamespermatch) {
    // gibt staerke des gewinnenden spielers
    vector<int> turnier (2*N);
    for(int i = 0; i < N; i++)
        turnier[i+N] = s[i];
    for(int i = N-1; i > 0; i--)
        turnier[i] = match(turnier[2*i],turnier[2*i+1],gamespermatch);
    return turnier[1];
}
```

Und die Hauptschleife der Simulation kombiniert alle diese Funktionen und zählt, wie oft für jeden Modus ein bester Spieler gewonnen hat:

```
// simulieren
for(int it = 0; it < ITER; it++) {
    // monte carlo iteration
    // damit einer der besten spieler
    // gewonnen, muss staerke des
    // gewinneneden spielers
    // maximalstaerke sein
    random_shuffle(s,s+N); // im ko-modus
                    // paarungen zufaellig
    cnt_liga += (smax == liga());
    cnt_ko   += (smax == ko(1));
    cnt_ko5  += (smax == ko(5));
```

```
13 }
```

Schließlich werden die Ergebnisse ausgegeben.

4 Beispiele

Hier die Ausgaben für die Beispiele von der BwInf-Webseite:

spielstaerken1.txt
Simulation abgeschlossen, 100000 Iterationen
Wahrscheinlichkeit, dass bester Spieler gewinnt
Liga: 0.45726
K.O.: 0.40726
KO 5: 0.60119
Beste Turniervariante KO 5

spielstaerken2.txt
Simulation abgeschlossen, 100000 Iterationen
Wahrscheinlichkeit, dass bester Spieler gewinnt
Liga: 0.3209
K.O.: 0.30061
KO 5: 0.35977
Beste Turniervariante KO 5

spielstaerken3.txt
Simulation abgeschlossen, 100000 Iterationen
Wahrscheinlichkeit, dass bester Spieler gewinnt
Liga: 0.25919
K.O.: 0.16448
KO 5: 0.27812
Beste Turniervariante KO 5

spielstaerken4.txt
Simulation abgeschlossen, 100000 Iterationen
Wahrscheinlichkeit, dass bester Spieler gewinnt
Liga: 0.07453
K.O.: 0.06945
KO 5: 0.07565
Beste Turniervariante KO 5

Man siehr hier, dass KO 5 immer am besten ist.
Eigene Beispiele:

4
1 1 1 100
Liga: 0.97968
K.O.: 0.98029
KO 5: 0.99998
Beste Turniervariante KO 5
4
99 99 99 100
Liga: 0.25301
K.O.: 0.25126
KO 5: 0.25478
Beste Turniervariante KO 5