Aufgabe 3: Voll daneben

Team-ID: 00342

Team-Name: Pied Piper

Bearbeiter/-innen dieser Aufgabe:
Jerrit Gläsker

29. September 2018

Inhaltsverzeichnis

1 Lösungsidee

Grundsätzlich geht es in dieser Aufgabe darum, möglichst viele Tipps der Besucher im Bereich 1-1000 mit 10 von Al Capone platzierten Markern abzudecken, um einen möglichst großen Gewinn zu erhalten. Hierbei ist das bilden eines Mittelwerts hilfreich, da dieser ein Indikator für die Konzentration der Tipps auf der gegebenen Skala ist. Wichtig zu sagen ist, dass hierbei das arithmetische Mittel herangezogen wird, nicht etwa der Medianwert, da dieser Ausreißer, beispielsweise 50 Leute die auf 13 tippen, außer Acht lassen würde, was aber in dieser Aufgabe nicht förderlich ist, da gerade diese Ausreißer beachtet werden müssen. Um zu verhindern, dass alle 10 "Marker" von Al Capone an derselben Stelle landen, wird jeder Marker in einen eigenen Bereich verschoben, für den dann jeweils das arithmetische Mittel berechnet wird, welches dann die Position des Markers des jeweiligen Bereiches bildet.

2 Umsetzung

Das Programm wurde in der Sprache C++ implementiert. Zum Speichern der Tipps der Besucher, sowie der Tipps von Al Capone werden dynamische Arrays (std::vector) verwendet. Die Aspekte des ermitteln der Wahl von Al Capone, das Berechnen des Gewinns, das Auswählen des nächsten von Al Capones Tipps sowie das Berechnen des arithmetischen Mittels in einem bestimmten Interval werden als Funktionen separiert. Ein Programmabschnitt wird dem Ausführen der BWINF-Beispiele gewidmet, ein anderer zum Durchführen von beliebig vielen Versuchen mit zufällig gewählten Tipps dient der Quantifizierung des Erfolgs der Strategie von Al Capone, d.h. über eine beliebige Anzahl an Versuchen wird der durchschnittliche Gewinn ermittelt. Ist dieser positiv, hat die Strategie Erfolg, ebenso andersherum.

3 Beispiele

3.1 BWINF-Beispiel 1:

Die Anzahl der Teilnehmer beträgt 25, daher wird als Anfangseinnahme 4975 errechnet. Danach folgt die Ausgabe von Al Capones Wahl: 50, 147, 247, 347, 447, 547, 647, 747, 847, 947. Der Auszuzahlende Betrag lautet in diesem Fall 4950. Die Differenz aus den Anfangseinnahmen und dem auszuzahlenden Betrag ist +25, daher hat Al Capone Gewinn erzielt. Die Zahlen in diesem Beispiel sind uniform verteilt, daher würde auch eine einfache uniforme Verteilung (50, 150, 250 . . . , 950) der Wahl von Al Capone funktionieren. Der Grund warum die Zahlen nicht exakt verteilt sind, ist dass das Berechnen des arithmetischen Mittels die untere Grenze einschließt, die obere aber ausschließt.

3.2 BWINF-Beispiel 2:

Die Anzahl der Teilnehmer ist hier 100, daher ist die Anfangseinnahme 2500. Al Capones Wahl ist in diesem Beispiel: 55, 158, 244, 354, 432, 549, 666, 763, 850, 942. Aus dieser Wahl ergibt sich der Auszahlungsbetrag von 2275. Hier ist die Differenz +225, wodurch auch in diesem Beispiel Gewinn gemacht wurde. In diesem Beispiel sind die Zahlen nicht mehr uniform verteilt, wodurch das erdachte Verfahren den Gewinn optimieren kann. Würde Al Capone seine Wahl uniform verteilen, müsste er in diesem Beispiel 2488 auszahlen, wodurch der Gewinn lediglich 12 wäre.

3.3 BWINF-Beispiel 3:

Auch hier ist die Anzahl der Teilnehmer 100, daher die Anfangseinnahmen von 2500. Al Capone wählt in diesem Fall: 50, 135, 245, 344, 441, 541, 647, 733, 840, 937. Hiermit muss er 2357 auszahlen, wodurch ein Gewinn von 143 entsteht. Würde Al Capone in diesem Beispiel seine Wahl uniform verteilen, müsste er 2540 auszahlen und hätte somit 40 Verlust gemacht.

3.4 Quantifiziertes Beispiel:

Testet man den zweiten Teil des Programms und führt 1 Millionen Durchläufe mit jeweils 100 zufällig gewählten Teilnehmerzahlen durch, erhält man einen durchschnittlichen Gewinn von 187,978. Verändert man den Versuch um zufällige Teilnehmerzahlen und auf 100,000 Durchläufen erhält man einen durchschnittlichen Gewinn von 200,289. In beiden Versuchen erzielt Al Capone deutlichen Gewinn, was den Erfolg dieser Technik untermalt.

3.5 Spezialfall:

Für den Fall, dass sich alle Teilnehmer absprechen und ihre Zahlen uniform, genau an den Grenzen der Abschnitte für Al Capones Wahlen platzieren, entsteht ein Problem für dieses Verfahren. In dem getesteten Beispiel wird auf 0,99,100,199,200, 299, . . . ,900,999 gesetzt. Dadurch wird jede von Al Capones Wahlen in die Mitte des jeweiligen Bereichs gesetzt, wodurch in diesem Beispiel ein Verlust entsteht (490). In der Aufgabe ist jedoch nicht geschildert, dass dieser Fall der Absprache möglich ist und das Erreichen dieser Konstellation durch Zufall ist mehr als unwahrscheinlich, daher ist im Sinne der Wirtschaftlichkeit dieser Fall zu vernachlässigen.

4 Quellcode

Haupblock, Einzelfall-Betrachtung:

```
    cout << "Anfangseinnahmen: " << entries.size() * 25 << endl << endl;

    cout << "Al Capones Wahl: " << endl;
    for (int i = 0; i < 10; i++)
    {
      //Unterteilung von 1000 in 10 gleichgrosse Bereiche, mit dem Marker in der Mitte
      //Start des Markers in der Mitte seines Bereiches
      int marker = i * 100 + 50;

      //Um moeglichst dicht an moeglichst vielen Leuten zu liegen
      //wird die durchschnittlich gewaehlte Zahl in diesem Bereich errechnet
      float avInRange = averageInRange(entries, marker - 50, marker + 50);
      cout << (int)avInRange << ", ";

      picks.push_back((int)avInRange);
    }

    cout << endl << endl;
    cout << "Auszuzahlen: " << toPay(entries, picks) << endl;
```

Hauptblock, Beliebige Durchläufe:

```
    cout << "Durchlaeufe?" << endl;
    int iterations;
    cin >> iterations;

    float dGewinn = 0;
    for (int i = 0; i < iterations; i++)
    {
      printProgress((float)i / (float)iterations);
      entries.clear();
      fillRandomly(entries);
      float einzelGewinn = evaluate(entries);
      //cout << einzelGewinn << endl;
      dGewinn += einzelGewinn;
    }

    //Durchschnittlicher Gewinn wird berechnet
    dGewinn /= iterations;

    cout << endl << "Durchschnittlicher Gewinn: " << dGewinn << endl;
```

Durchschnittsfunktion:

```
float averageInRange(vector <int> &entries, int low, int high)
{
  float av = 0;
  int amount = 0;
  for (int i : entries)
  {
    if (i >= low && i < high)
    {
      av += i;
      amount++;
    }
  }
  return(av / (float)amount);
}
```

Ermitteln der nächsten von Al Capones Wahlen:

```
int getClosest(int a, vector <int> &picks)
{
  float recdiff = 100000;
  float closestPick = 0;

  for (int pick : picks)
  {
    float newDiff = abs(pick - a);
    if (newDiff < recdiff)
    {
      recdiff = newDiff;
      closestPick = pick;
    }
  }
  return closestPick;
}
```

Gewinnfunktion:

```
float toPay(vector <int> &entries, vector <int> &picks)
{
  int sum = 0;
  for (int bet : entries)
  {
    int closest = getClosest(bet, picks);

    int diff = abs(closest - bet);

    sum += diff;
  }
  return sum;
}
```