Einzelprüfung „Theoretische Informatik / Algorithmen (vertieft)“

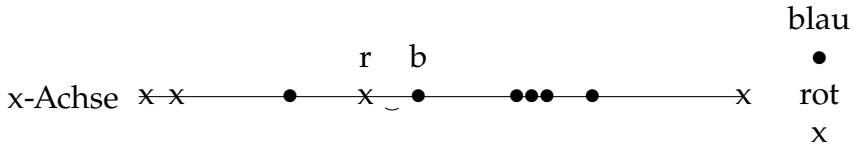
Einzelprüfungsnummer 66115 / 2020 / Frühjahr

Thema 2 / Aufgabe 8

(Nächstes rot-blaues Paar auf der x-Achse)

Stichwörter: Algorithmische Komplexität (O-Notation)

Gegeben seien zwei nichtleere Mengen R und B von roten bzw. blauen Punkten auf der x-Achse. Gesucht ist der minimale euklidische Abstand $d(r,b)$ über alle Punktepaare (r,b) mit $r \in R$ und $b \in B$. Hier ist eine Beispielinstanz:



Die Eingabe wird in einem Feld A übergeben. Jeder Punkt $A[i]$ mit $1 \leq i \leq n$ hat eine x-Koordinate $A[i].x$ und eine Farbe $A[i].color \in \{\text{rot}, \text{blau}\}$. Das Feld A ist nach x-Koordinate sortiert, d. h. es gilt $A[1].x < A[2].x < \cdots < A[n].x$, wobei $n = |R| + |B|$.

(a) Geben Sie in Worten einen Algorithmus an, der den gesuchten Abstand in $\mathcal{O}(n)$ Zeit berechnet.

Lösungsvorschlag

Pseudo-Code

Algorithmus 1: Minimaler Euklidischer Abstand

```
d_min := max ;                          // Setze d_min zuerst auf einen maximalen Wert.
for i in 0 ... vorletzter Index do ;    // Iteriere über die Indizes des Punkte-Arrays P bis
 zum vorletzten Index P[n − 1]
    if P[n].color ≠ P[n + 1].color then ; // Berechne den Abstand nur, wenn die Punkte
     unterschiedliche Farben haben
        d = P[n + 1].x − P[n].x
        if d < d_min then
            d_min = d
        end
    end
end
```

Java

```java
public double findMinimalDistance() {
  double distanceMin = Double.MAX_VALUE;
  for (int i = 0; i < latestIndex - 1; i++) {
    if (points[i].color != points[i + 1].color) {
      double distance = points[i + 1].x - points[i].x;
```

```
        if (distance < distanceMin) {
          distanceMin = distance;
        }
      }
    }
    return distanceMin;
  }
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java`

(b) Begründen Sie kurz die Laufzeit Ihres Algorithmus.

Lösungsvorschlag

Da das Array der Länge n nur einmal durchlaufen wird, ist die Laufzeit $\mathcal{O}(n)$ sichergestellt.

(c) Begründen Sie die Korrektheit Ihres Algorithmus.

Lösungsvorschlag

In d_{min} steht am Ende der gesuchte Wert (sofern nicht $d_{min} = Integer.MAX\_VALUE$ geblieben ist)

(d) Wir betrachten nun den Spezialfall, dass alle blauen Punkte links von allen roten Punkten liegen. Beschreiben Sie in Worten, wie man in dieser Situation den gesuchten Abstand in $o(n)$ Zeit berechnen kann. (Ihr Algorithmus darf also insbesondere nicht Laufzeit $\Theta(n)$ haben.)

Lösungsvorschlag

Zuerst müssen wir den letzten blauen Punkt finden. Das ist mit einer binären Suche möglich. Wir beginnen mit dem ganzen Feld als Suchbereich und betrachten den mittleren Punkt. Wenn er blau ist, wiederholen wir die Suche in der zweiten Hälfte des Suchbereichs, sonst in der ersten, bis wir einen blauen Punkt gefolgt von einem roten Punkt gefunden haben.

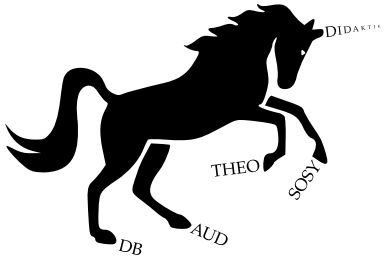
Der gesuchte minimale Abstand ist dann der Abstand zwischen dem gefundenen blauen und dem nachfolgenden roten Punkt. Die Binärsuche hat eine Worst-case-Laufzeit von $\mathcal{O}(\log n)$.

Additum: Komplette Java-Klasse

```
enum Color {
  RED, BLUE
}

class Point {
  double x;
  Color color;

  public Point(double x, Color color) {
```

```
    this.x = x;
    this.color = color;
  }
}

public class RedBluePairCollection {
  Point[] points;
  int latestIndex;

  public RedBluePairCollection(int size) {
    points = new Point[size];
  }

  public void addPoint(double x, Color color) {
    points[latestIndex] = new Point(x, color);
    latestIndex++;
  }

  public double findMinimalDistance() {
    double distanceMin = Double.MAX_VALUE;
    for (int i = 0; i < latestIndex - 1; i++) {
      if (points[i].color != points[i + 1].color) {
        double distance = points[i + 1].x - points[i].x;
        if (distance < distanceMin) {
          distanceMin = distance;
        }
      }
    }
    return distanceMin;
  }

  public static void main(String[] args) {
    RedBluePairCollection pairs = new RedBluePairCollection(11);

    pairs.addPoint(0, Color.RED);
    pairs.addPoint(0.2, Color.RED);
    pairs.addPoint(1.5, Color.BLUE);
    pairs.addPoint(3.1, Color.RED);
    pairs.addPoint(4.0, Color.BLUE);
    pairs.addPoint(4.2, Color.BLUE);
    pairs.addPoint(5.1, Color.RED);
    pairs.addPoint(6, Color.BLUE);
    pairs.addPoint(6.1, Color.BLUE);
    pairs.addPoint(6.2, Color.BLUE);
    pairs.addPoint(7.2, Color.RED);

    System.out.println(pairs.findMinimalDistance());
  }
}
```

Code-Beispiel auf Github ansehen: `src/main/java/org/bschlangaul/examen/examen_66115/jahr_2020/fruehjahr/RedBluePairCollection.java`



Die Bschlangaul-Sammlung
Hermine Bschlangaul and Friends

Eine freie Aufgabensammlung mit Lösungen von Studierenden für Studierende zur Vorbereitung auf die 1. Staatsexamensprüfungen des Lehramts Informatik in Bayern.

Hilf mit! Die Hermine schafft das nicht allein! Das ist ein Community-Projekt! Verbesserungsvorschläge, Fehlerkorrekturen, weitere Lösungen sind herzlich willkommen - egal wie - per Pull-Request oder per E-Mail an hermine.bschlangaul@gmx.net.Der TEX-Quelltext dieses Dokuments kann unter folgender URL aufgerufen werden: `https://github.com/bschlangaul-sammlung/examens-aufgaben/blob/main/Staatsexamen/66115/2020/03/Thema-2/Aufgabe-8.tex`