Prof. Dr. Leif Kobbelt
Stefan Dollase, Ira Fesefeldt, Alexandra Heuschling, Gregor Kobsik

Lösung - Übung 5

Tutoraufgabe 1 (Quicksort):

Sortieren Sie das folgende Array mithilfe von Quicksort. Geben Sie dazu das Array nach jeder Partition-Operation an und markieren Sie das jeweils verwendete Pivot-Element.

2	3	9	6	7	4	1	5	8

Lösung

2	3	9	6	7	4	1	5	8
2	3	5	6	7	4	1	**8**	9
1	3	5	6	7	4	2	8	9
1	**2**	5	6	7	4	3	8	9
1	2	**3**	6	7	4	5	8	9
1	2	3	4	**5**	6	7	8	9
1	2	3	4	5	6	**7**	8	9

Tutoraufgabe 2 (Mergesort):

Sortieren Sie das folgende Array mithilfe von Mergesort. Geben Sie dazu das Array nach jeder Merge-Operation an.

2	3	9	6	7	4	1	5	8

Lösung

2	3	9	6	7	4	1	5	8
2	3	9	6	7	4	1	5	8
2	3	9	6	7	4	1	5	8
2	3	9	6	7	4	1	5	8
2	3	6	7	9	4	1	5	8
2	3	6	7	9	1	4	5	8
2	3	6	7	9	1	4	5	8
2	3	6	7	9	1	4	5	8
1	2	3	4	5	6	7	8	9

Tutoraufgabe 3 (Heapsort):

Sortieren Sie das folgende Array mithilfe von Heapsort. Geben Sie dazu das Array nach jeder Swap-Operation an.

2	3	9	6	7	4	1	5	8

Lösung

2	3	9	6	7	4	1	5	8
2	3	9	8	7	4	1	5	6
2	8	9	3	7	4	1	5	6
2	8	9	6	7	4	1	5	3
9	8	2	6	7	4	1	5	3
9	8	4	6	7	2	1	5	3
3	8	4	6	7	2	1	5	9
8	3	4	6	7	2	1	5	9
8	7	4	6	3	2	1	5	9
5	7	4	6	3	2	1	8	9
7	5	4	6	3	2	1	8	9
7	6	4	5	3	2	1	8	9
1	6	4	5	3	2	7	8	9
6	1	4	5	3	2	7	8	9
6	5	4	1	3	2	7	8	9
2	5	4	1	3	6	7	8	9
5	2	4	1	3	6	7	8	9
5	3	4	1	2	6	7	8	9
2	3	4	1	5	6	7	8	9
4	3	2	1	5	6	7	8	9
1	3	2	4	5	6	7	8	9
3	1	2	4	5	6	7	8	9
2	1	3	4	5	6	7	8	9
1	2	3	4	5	6	7	8	9

Tutoraufgabe 4 (Sortieren von Polarkoordinaten):

In dieser Aufgabe möchten wir Punkte $(x, y) \in \mathbb{Z}^2$ im 2-dimensionalen Raum nach verschiedenen Kriterien (Schlüsseln) sortieren.

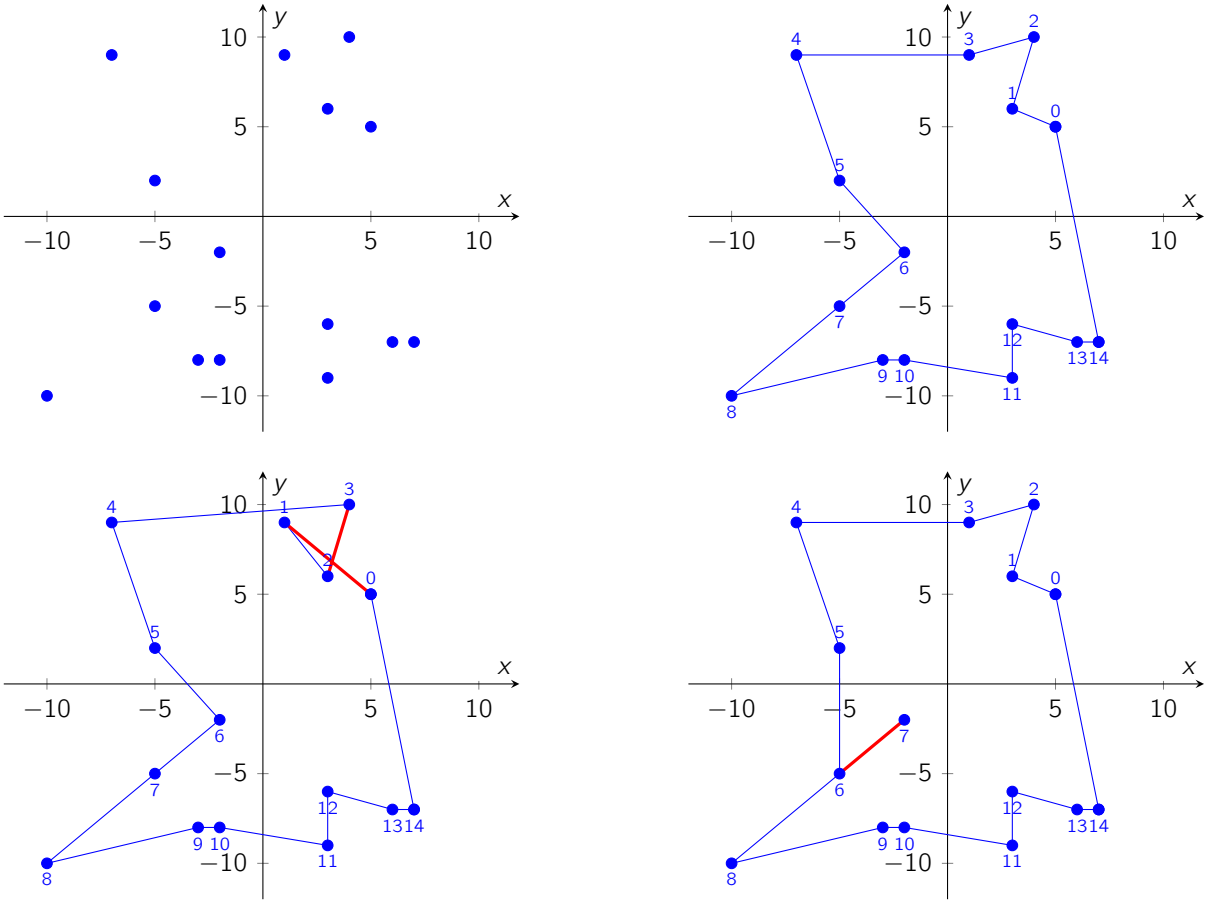
a) Beschreiben Sie, wie man die Sortierverfahren aus der Vorlesung abändern kann, um ein gegebenes Array von Punkten aufsteigend nach dem Euklidischen Abstand zum Ursprung zu sortieren.

b) Beschreiben Sie, wie man die Sortierverfahren aus der Vorlesung abändern kann, um ein gegebenes Array von Punkten aufsteigend nach dem Winkel zwischen der x-Achse und der Linie die den Ursprung mit einem Punkt verbindet zu sortieren.

c) Beschreiben Sie einen Algorithmus der als Eingabe ein Array `A` von n Punkten bekommt und dieses Array so sortiert, dass wir jeweils die Punkte `A[i]` und `A[i+1]` (für `i` $\in \{0, \ldots, n-2\}$ sowie `A[n-1]` und `A[0]` mit geraden Linien verbinden können, **ohne** dass sich die Linien kreuzen bzw. überlappen.

Wir gehen davon aus, dass sich in jedem Quadranten des Koordinatensystems mindestens ein Punkt befindet. Begründen Sie kurz, warum ihr Algorithmus korrekt ist.

Beispiel. In der unten stehenden Abbildung sehen wir eine mögliche Eingabe (oben links), eine korrekte Ausgabesortierung mit eingezeichneten Linien (oben rechts), sowie zwei falsche Ausgabesortierungen bei der sich die Linien sowohl kreuzen (unten links) als auch überlappen (unten rechts).



Lösung

a) Wir erstellen eine Funktion, die uns für einen gegebenen Punkt den Abstand zum Ursprung zurück gibt:

```
def dist(p):
  return sqrt(pow(p.x,2) + pow(p.y,2))
```

Nun wird im Sortieralgorithmus jeder Vergleich zwischen zwei Array Elementen `E[i] < E[j]` ersetzt durch `dist(E[i]) < dist(E[j])` (Analog für '`<=`', '`>`', '`==`', ...)

b) Gesucht wird hier der Polarwinkel, nach dem wir sortieren sollen. Ähnlich zur vorherigen Aufgabe erstellen wir nun eine Funktion, die uns für einen gegebenen Punkt den Polarwinkel zurück gibt. Hierzu benutzen wir den `arctan2`:

```
def polararc(p):
    return arctan2(p.y,p.x)
```

Nun wird genauso wie in der vorherigen Aufgabe im Sortieralgorithmus jeder Vergleich zwischen zwei Array Elementen `E[i] < E[j]` ersetzt durch `polararc(E[i]) < polarc(E[j])` (Analog für '`<=`', '`>`', '`==`', ...)

c)

```
def polarSort(E):
    Sortiere E mit Mergesort aufsteigend nach Euklidischen Abstand zum Ursprung
    Sortiere E mit Mergesort aufsteigend nach Polarwinkeln
    return E
```

Begründung: Wir erhalten ein Array das aufsteigend nach dem Polarwinkel sortiert ist. Dadurch das Mergesort stabil ist, gilt außerdem, dass Punkte mit gleichem Winkel nach ihrem Abstand zum Ursprung sortiert sind.

Man könnte dies tatsächlich auch mit einer lexikographischen Ordnung sortieren, bei der wir die Schlüssel `(polararc(p),dist(p))` nutzen.

Wenn wir eine Linie zwischen zwei aufeinander folgenden Punkten p_1 bzw. p_2 mit Polarwinkeln φ_1 bzw. φ_2 zeichnen, so hat jeder Punkt auf der Linie einen Polarwinkel $\varphi \in [\varphi_1, \varphi_2]$. Alle Punkte die auf zuvor gezeichneten Linien liegen können höchstens einen Polarwinkel $\varphi' \leq \varphi_1$ haben und sich deshalb nicht mit der Linie zwischen p_1 und p_2 kreuzen. Dies gilt insbesondere auch, da es in jedem Quadranten mindestens einen Punkt geben muss und wir uns deshalb beim Zeichnen der Linie nur gegen den Uhrzeigersinn um den Ursprung bewegen. Außerdem kann bei Punkten mit gleichem Polarwinkel keine Überlappung der Linien entstehen, da die Punkte dann nach ihren Abstand zum Ursprung sortiert sind.

Hinweise:

- Im zweiten Sortierschritt muss ein stabiles Verfahren verwendet werden.
- Das genaue Sortierverfahren im ersten Schritt ist nicht relevant.