Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Ein deterministischer Algorithmus

1. Falls $n < 30$: Sortiere und finde so das Ergebnis.
2. Bilde $m = \lfloor n/5 \rfloor$ Gruppen mit je fünf Schlüsseln. Bis zu vier Schlüssel bleiben übrig.
3. Berechne den Median jeder Gruppe.
4. Berechne rekursiv den Median p dieser $\lfloor n/5 \rfloor$ Mediane.
5. Verwende p als Pivotelement und führe damit Quickselect aus.

Welchen Rang r hat p?

p ist der Median von $m = \lfloor n/5 \rfloor$ vielen kleinen Medianen
$\rightarrow$ mindestens $m/2 - 1$ kleine Mediane sind kleiner als p
Jeder kleine Median hat zwei Schlüssel die kleiner sind
$\rightarrow$ mindestens $3m/2 - 1$ kleinere Schlüssel als p

Ebenso: Mindestens $3m/2 - 1$ größere Schlüssel als p
Das sind jeweils $3\lfloor n/5 \rfloor/2 - 1 \geq 3n/10 - 3/2 \geq n/4$

Deterministisches Selektieren – Analyse

Die Anzahl der Vergleiche ist jetzt

$$C_n \leq n + 1 + C_{\lfloor n/5 \rfloor} + C_{\lfloor 3n/4 \rfloor}$$

falls $n > 30$ und $O(1)$ falls $n \leq 30$:

- $C_{\lfloor n/5 \rfloor}$ für das rekursive Finden der Mediane
- $C_{\lfloor 3n/4 \rfloor}$ für die nächste Suche

Es folgt $C_n = O(n)$, da $1/5 + 3/4 < 1$.

Wir können also den Schlüssel mit Rang k in linearer Zeit finden.

Deterministisches Selektieren – Analyse

Die Anzahl der Vergleiche ist jetzt

$$C_n \leq n + 1 + C_{\lfloor n/5 \rfloor} + C_{\lfloor 3n/4 \rfloor}$$

falls $n > 30$ und $O(1)$ falls $n \leq 30$:

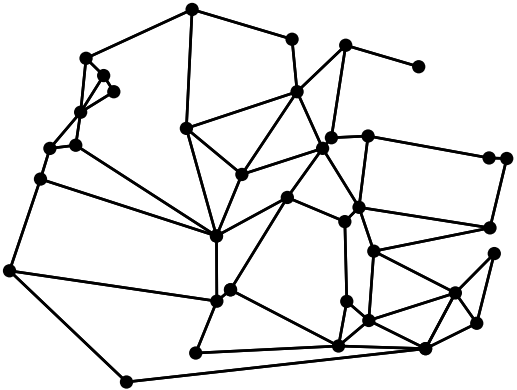
- $C_{\lfloor n/5 \rfloor}$ für das rekursive Finden der Mediane
- $C_{\lfloor 3n/4 \rfloor}$ für die nächste Suche

Es folgt $C_n = O(n)$, da $1/5 + 3/4 < 1$.

Wir können also den Schlüssel mit Rang k in linearer Zeit finden.

Übersicht

Graphen



Graphen

Definition

Ein ungerichteter Graph ist ein Paar (V, E), wobei V die Menge der Knoten und $E \subseteq \binom{V}{2}$ die Menge der Kanten ist.

Definition

Ein gerichteter Graph ist ein Paar (V, E), wobei V die Menge der Knoten und $E \subseteq V \times V$ die Menge der Kanten ist.

Oft betrachten wir Graphen mit Knoten- oder Kantengewichten.

Dann gibt es zusätzlich Funktionen $V \to \mathbf{R}$ oder $E \to \mathbf{R}$.

Graphen

Definition

Ein ungerichteter Graph ist ein Paar (V, E), wobei V die Menge der Knoten und $E \subseteq \binom{V}{2}$ die Menge der Kanten ist.

Definition

Ein gerichteter Graph ist ein Paar (V, E), wobei V die Menge der Knoten und $E \subseteq V \times V$ die Menge der Kanten ist.

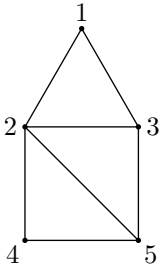
Oft betrachten wir Graphen mit Knoten- oder Kantengewichten.

Dann gibt es zusätzlich Funktionen $V \to \mathbf{R}$ oder $E \to \mathbf{R}$.

Übersicht

Darstellung von Graphen

Adjazenzmatrix

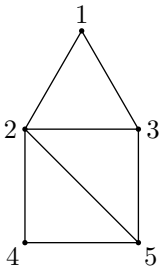


$$\begin{pmatrix} \cdot & 1 & 1 & 0 & 0 \\ \cdot & \cdot & 1 & 1 & 1 \\ \cdot & \cdot & \cdot & 0 & 1 \\ \cdot & \cdot & \cdot & \cdot & 1 \\ \cdot & \cdot & \cdot & \cdot & \cdot \end{pmatrix}$$

Speicherbedarf: $\Theta(|V|^2)$

Für gerichtete Graphen wird die ganze Matrix verwendet.

Darstellung von Graphen



Adjazenzliste

1	2, 3
2	1, 3, 4, 5
3	1, 2, 5
4	2, 5
5	2, 3, 4

Speicherbedarf: $\Theta(|V| + |E|)$.

In $O(n^2)$ Schritten kann zwischen beiden Darstellungen konvertiert werden.

Darstellung von Graphen

Java

```
public class SimpleGraph<V> implements Graph<V> {
  protected Set<V> nodes;
  protected Map<V, List<V>> edges;
  protected boolean directed = false;
  public SimpleGraph() {
    nodes = new HashSet<V>();
    edges = new HashMap<V, List<V>>();
  }
```

Wir wählen die Darstellung durch eine Adjazenzliste.

Java

```
public class Edge<V> {
  public V s, t;
  private boolean directed;
  protected Edge(V s, V t, boolean directed) {
    this.s = s;
    this.t = t;
    this.directed = directed;
  }
```

Java

```
public void addNode(V u) {
  nodes.add(u);
  edges.put(u, new LinkedList<V>());
}
```

Java

```
public void addEdge(V s, V t) {
  List<V> adjlist = edges.get(s);
  adjlist.add(t);
  if(!directed) {
    adjlist = edges.get(t);
    adjlist.add(s);
  }
}
```

Java

```
public void delEdge(V s, V t) {
  List⟨V⟩ adjlist = edges.get(s);
  adjlist.remove(t);
  if(!directed) {
    adjlist = edges.get(t);
    adjlist.remove(s);
  }
}
```

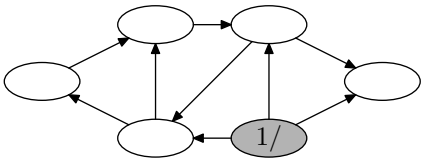
Übersicht

Tiefensuche

Tiefensuche ist ein sehr mächtiges Verfahren, das iterativ alle Knoten eines gerichteten oder ungerichteten Graphen besucht.

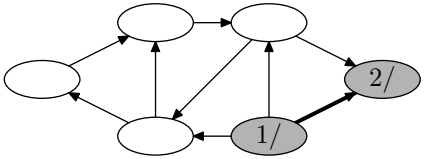
- Sie startet bei einem gegebenen Knoten und färbt die Knoten mit den Farben weiß, grau und schwarz.
- Sie berechnet einen gerichteten Tiefensuchwald, der bei einem ungerichteten Graph ein Baum ist.
- Sie ordnet jedem Knoten eine Anfangs- und eine Endzeit zu.
- Alle Zeiten sind verschieden.
- Die Kanten des Graphen werden als Baum-, Vorwärts-, Rückwärts- oder Querkanten klassifiziert.

Tiefensuche – Beispiel



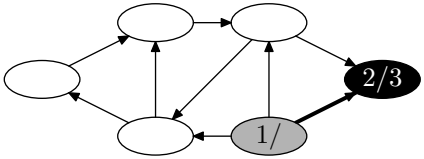
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



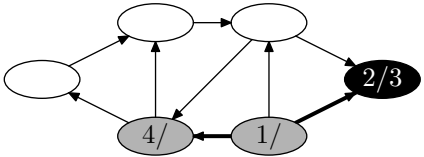
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



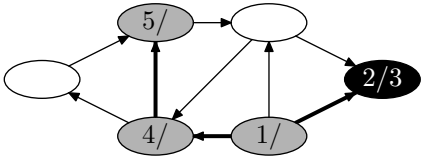
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



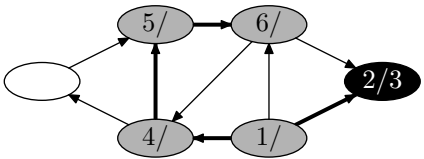
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



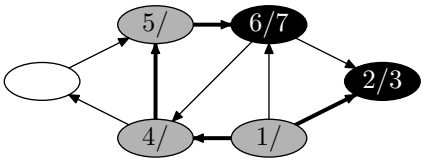
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



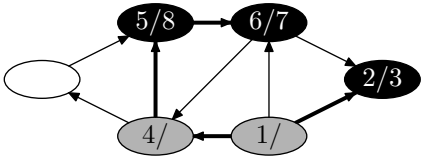
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



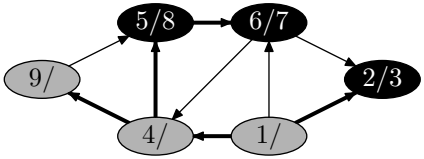
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



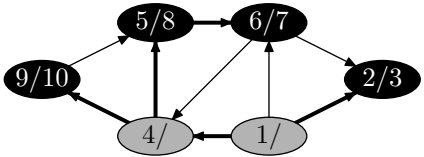
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



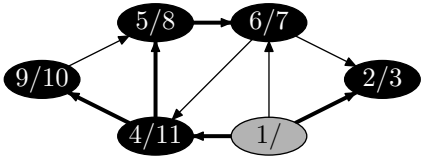
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



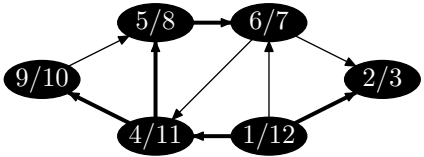
- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Tiefensuche – Beispiel



- Die Kanten des Tiefensuchwaldes sind dick dargestellt.
- Ein Knoten ist anfangs weiß.
- Ein Knoten ist grau, während er aktiv ist.
- Danach wird er schwarz.

Noch ein Beispiel

Java

```
public static<V> void DFS(Graph<V> G, Map<V, Integer> d,
    Map<V, Integer> f, Map<V, V> p) {
  Map<V, Integer> color = new HashMap<V, Integer>();
  for(V u : G.allNodes())
    color.put(u, WHITE);
  int time = 0;
  for(V u : G.allNodes())
    if(color.get(u) == WHITE)
      time = DFS(G, u, time, color, d, f, p);
}
```

Java

```
public static⟨V⟩ int DFS(Graph⟨V⟩ G, V u, int t, Map⟨V, Integer⟩ c,
        Map⟨V, Integer⟩ d, Map⟨V, Integer⟩ f, Map⟨V, V⟩ p) {
  d.put(u, ++t);
  c.put(u, GRAY);
  for(V v : G.neighbors(u))
    if(c.get(v) == WHITE) {
      p.put(v, u);
      t = DFS(G, v, t, c, d, f, p);
    }
  f.put(u, ++t);
  c.put(u, BLACK);
  return t;
}
```