

komplexität

DATAStruKtuRn

Union-Find

Schalarithmetik:

Flosskungen \rightarrow Pseudorandomen
 1. $(\text{log} n) \cdot \text{int}(\text{log} n) \cdot \text{int}(\text{log} n) \cdot \text{int}(\text{log} n)$
 2. $n \cdot (\text{log} n) \cdot (\text{log} n) \cdot \text{int}(\text{log} n) \cdot \text{int}(\text{log} n)^2$

push(a) legt a auf stack

$O(n)$ entfernt erstes Element und gibt es aus

$O(n)$ findet(x) gibt Menge zurück die x enthält

union(x,y) fügt beide Mengen zusammen

Imp: 3ere Teilmenge als liste

zusammengebracht in großer überlappender

Sortiert Partitionierung einer Menge in disjunkte Teilmengen

Indizes: Szenen: Verschiedene mit einer ID (z.B. 0, 1, 2, 3, 4)

binäre Suchbaum für dynamische Prozesse

Hash tabellen

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur verschiedene auf diese gespiennt.

Zahlen:
 1. **Querels:**
 2. **Sortiert - Algorithmen** (statisch)
 3. **Merge - Sort** (statisch)
 4. **Heapsort** (statisch)

Querels: nicht langsame *front zeigt auf erstes Element

1. **Insertion - Sort** $O(n^2)$
 2. **Selection - Sort** $O(n)$

1. Bereiche beim 2. Element

1. 2. Selektion Element innerhalb der

und rückwärts geben

→ zwischen \rightarrow Vorsäger

1. Zeiger auf ersten Wert (z.B. Zeiger auf ersten Wert hier)

2. Zeiger auf extra Stelle im Array

Codex: Kollisionen in nächster freier Amoar Platz speichern

(bzw. $h(k_i) = h(k_i + 1) \mod n$)

Doppel: $h(k_i) = h(k_i + 1) \mod n$

→ Anzahl das

Sonduren: Kollisionen nicht als extra liste speichern,

sondern an extra Stelle im Array

Querels: Kollisionen in nächster freier Amoar Platz speichern

(bzw. $h(k_i) = h(k_i + 1) \mod n$)

Folger: $h(k_i) = h(k_i + 1) \mod n$

...Anzahl das

Hash tabellen

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

...Folger: bestimmte Position eines Objekts durch Hashfunktion bestimmt wird

...Da es zu Kollisionen kommen kann, werden in Tabellen nur

verschiedene auf diese gespiennt.

Union-Find
 Verarbeitet Partitionierung einer Menge in disjunkte Teilmengen

Sequenzliches Suchen: Feld durchlaufen vergleichen $O(n)$

Indizes: Suchen: Verschiedene Vergleichen mit einer ID (z.B. 0, 1, 2, 3, 4)

Binäres Suchbaum für dynamische Prozesse

Hash tabellen

Sortiert - Algorithmen (statisch)

Insertion - Sort $O(n^2)$

Selection - Sort $O(n)$

Priority Queue P

Inseration Sort (A)

Merge - Sort (statisch)

Heapsort (statisch)

QuickSort $O(n^2)$ / Avg. $O(n \cdot \log n)$

Quicksort (A)

if ($\rightarrow \text{minLength}$)
int pivotPos = n/2;
swap(A[lowPos], A[n-1]);
pivotPos = n-1;
int pivot = A[pivotPos];
int i = 0;
int j = n-2;
do {
while (i <= A[i] < pivot) : i++;
while (j >= A[j] > pivot) : j--;
if (i < j) swap(A[i], A[j]);
i++; j--;
} while (i < j);

Quicksort (A, splitPoint, minLength)

Quicksort (A, splitPoint+1, n-splitPoint-1, minLength)

Graphen

Bellman-Ford SSSP Binärer Suchbaum

$G = (E, V)$: E = Knoten, V = Kanten
Wie Dijkstra, jedoch darf jedes Knoten mehrfach besucht werden zur Wegüberprüfung

gerichtet: Knoten $E = \{E_1, E_2, \dots\}$
gerichtet: Knoten $E = \{E_1, E_2, \dots\}$
 v adjazent zu u : $(u, v) \in E / \{u \neq v\}$

Knoten v incident zu Knoten u :

$$e = (x_u, v) / e = \{x_u\}$$

Ein einfacher Pfad: alle Knoten unterhalb zyklen können Knoten oft bei beinhaltet, weise haben Knoten nur einen Vorgänger

Zusammenhangskomponente: Topolog DFS in Graph $\rightarrow (1, 1), (2, 1), \dots, (5, 1) \cup \{(6 \rightarrow 6), (5, 1) \dots, 6\}$

Dann Zusätzlich bis auf nlin geschlossen wird. Nun bedeutet ~~zwei~~ ZHK (wege Vommen über mehrere Komponenten) mit Städte verbunden

Minimales Spannbaum

Voraussetzung: 1. Sortiere alle Knoten aufsteigend (Unterfach)

noch Gewicht

2. Wähle immer nächst schnelleste

Kante, falls sie keine Schleife bildet

Tree-Order: Wurzel - links - rechts

In-Order: Links - rechts - Wurzel

Post-Order: Rechts - rechts - Wurzel

Tiefen Suche OTS

- Stack

- solange tiefer deskriptivieren

bis nicht mehr geht (1. Zahl)

→ dann Rückwärtsmarkieren (wenn möglich neue obzw. alte Knoten statt schon berechneter)

Rufen auf Zeichen: bei ersten Würfeln Heap

Fest Vollständiger Brüderbaum mit Verzweigung von oben nach unten und Wurzel auch rechts

Max-Heap: Wurzel hat größten Wert

Besuch Knoten grau (Wurzel einfügen: auf unterstem Niveau so

→ Heap-Baum nicht weiterstellen

Wurzel ist Zulass. Voraussetzung

Zeichen: Tauschen

Brüder-OTOTS

→ zu den Knoten aus jedem Nachbarn

in Queue → Queue bearbeiten mit

schritt Knoten c und entferne jedem abgedankten Knoten tritt n in

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ nur positive Kontenreihen

→ jedem Knoten c und entferne jedem abgedankten Knoten tritt n in

→ jedem abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit

unterstem Tauschen

→ jeden abgedankten Knoten markieren

→ Heaps-Ein. wiederherstellen:

mit Kinder wiederein mit