Algorithmen und Datenstrukturen (Master) WiSe 19/20

Benedikt Lüken-Winkels

February 8, 2020

Contents

1 Aufgabe 1:

Beschreiben Sie jeweils eine Lösung für das Union-Find-Problem mit Laufzeit

1. $O(\log n)$ (amortisiert) für UNION und $O(1)$ für FIND
2. $O(1)$ für UNION und $O(\log n)$ für FIND

wobei n die Anzahl der Elemente ist. Begründen Sie in beiden Fällen die entsprechenden Laufzeiten.

Lösung 1.)

Union in $O(\log n)$, Find in $O(1)$. **Idee:** Relable the smaller half, sodass jedes Element nur maximal $\log n$ geändert wird:

Datenstruktur

```
name[x]: Name des Blocks, der x enthält
size[A]: Größe des Blocks A (Init 1)
list[A]: Liste der Elemente in Block A
```

Algorithmus 1 : Initialisierung

```
foreach x ∈ N do
    name[x]← x;
    size[x]← 1;
    list[x]← {x};
end
```

Algorithmus 2 : Find(x)

```
return name[x];
```

Algorithmus 3 : Union(A,B)

```
if size[A] ≥ size[B] then
    foreach x ∈ list[B] do
        name[x] ← A;
    size[A] ← size[A]+size[B];
    list[A].append(list[B]);
else
    foreach x ∈ list[A] do
        name[x] ← B;
    size[B] ← size[A]+size[B];
    list[B].append(list[A]);
```

Laufzeit

Find: $O(1)$. Lookup im Array.

Union: $O(\log n)$. Jedes x kann maximal $\log n$ mal seinen Namen ändern, da es sich nach jeder Namensänderung in einer doppelt so großen Liste befindet.

Lösung 2.)

Union in $O(1)$ und Find in $O(\log n)$. **Idee:** Bei Union Anhängen des kleineren Teilbaums an den Größeren.
Das ergibt die Abschätzung size[x] $\geq 2^{höhe(x)}$, bzw $\log_2$(size[x]) $\geq$ höhe(x), also wird der Baum nie tiefer, als $\log n$

Datenstruktur

```
name[x]: Name des Blocks mit Wurzel x (nur, wenn x eine Wurzel relevant)
size[x]: Anzahl der Knoten im Unterbaum mit Wurzel x
wurzel[x]: Wurzel des Blocks mit Namen x
vater[x]: Vaterknoten des Knotens x. 0, wenn x Wurzel
```

Algorithmus 4 : Initialisierung

```
foreach x ∈ N do
    name[x]← x;
    size[x]← 1;
    wurzel[x]← x;
    vater[x]← 0;
end
```

Algorithmus 5 : Find(x)

```
while vater[x] ≠ 0 do
    x ← vater[x] ;
end
return name[x];
```

Algorithmus 6 : Union(A, B, C)

```
a ← wurzel[A];
b ← wurzel[B];
if size[a] ≥ size[b] then
    vater[b] ← a;
    name[a] ← C;
    wurzel[C] ← a;
    size[a] ← size[a] + size[b];
else
    analog;
end
```

Laufzeit

Find: $O(\log n)$. Höhe des Baums bleibt maximal $\log n$, da der kleinere Teilbaum immer an die Wurzel des Größeren gehangen wird und sich die Tiefe des Baums durch seine Größe abschätzen lässt.

Union: $O(1)$. Lediglich die Wurzel muss umgeschrieben werden.