

Altklausuren Antworten

Landmesser Zusammenfassung

Aufgabe 1

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

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

Initialisierung:

```
begin
  for  $i := 1$  to  $n$  do
    vater[i]=0
    name[i]=i
    wurzel[i]=i
  end
end
```

FIND(x):

```
begin
  while  $vater[x] \neq 0$  do
    x = vater[x]
  end
  return name[x]
end
```

UNION(A,B,C):

```
begin
   $r_1 = \text{wurzel}[A]$ 
   $r_2 = \text{wurzel}[B]$ 
  vater[ $r_1$ ] =  $r_2$ 
  name[ $r_2$ ] = C
  wurzel[C] =  $r_2$ 
end
```

Aufgabe 2

Zusatzaufgabe: Perfektes Hashing

Aufgabe3

Aufgabe 4

Sei G ein planarer Graph mit $n \geq 3$ Knoten und m Kanten, dann gilt $m = 3n - 6$.
dh $m = O(n)$, also linear viele Kanten.

H. Ein maximal planarer Graph ist ein planarer Graph, der durch Hinzufügen einer Kante $(v, w) \notin E$ nicht-planar wird. Beobachtung: Alle Faces in jeder planaren Einbettung von G sind Dreiecke (Triangulierung). Jedes Face in einer Triangulierung hat 3 Rand-Kanten und jede Kante liegt am Rand von 3 Faces.

$$\Rightarrow 3f = 2m$$

Einstsetzen in Euler-Formel

$$n - m + \frac{2}{3}m = 2$$

$$m = 3n - 6$$

$m \leq 3n - 6$ für beliebige planare Graphen

□

Zusatzaufgabe

Sei G ein **bipartiter** planarer Graph Dann gilt $m \leq 2n - 4$.

Beweis: Keine Kreise ungerader Länge in bipartiten Graphen. Kleinstmögliche Fläche in einem bipartiten Graphen ist ein Viereck. □

Aufgabe 5

Split-Find-Problem Idee - Umkehrung von Union-Find Feld $name[1,...,n]$

Initialisierung:

```
begin
  for  $\forall i, 1 \leq i \leq n$  do
    |  $name[i] = 1$ 
  end
end
```

Find(i) begin

| return $name[i] \rightarrow \mathcal{O}(1)$

end

Split(i)

- Neuer Name des neuen Intervalls das durch Split entsteht ++count
- Relabel the smaller half
- Laufe parallel d.h. abwechselnd nach links und rechts von i aus, bis Intervallgrenze erreicht d.h. $name[i] \neq name[betrachtetesElement]$
- Nenne den Teil um, der kleiner ist $[a, i]$ oder $[i + 1, b]$ indem nochmals über diesen Teil gelaufen wird. $name[betrachtetesElement] = count$

a			i				b
1	1	1	1	1	1	1	
2	2	2					

\Rightarrow Kosten für 1 Split $\mathcal{O}(2 * \text{LaengedeskuerzerenIntervalls})$

Analyse $\mathcal{O}(\#Namensaenderungen) : \max \frac{\text{Laenge des umzubennendenes Intervall}}{2} \Rightarrow \mathcal{O}(\log n)$

Aufgabe 6

Algorithmus von Dijkstra:

```

begin
  foreach  $v \in V$  do
    DIST[v]  $\leftarrow \infty$ 
    PRED[v]  $\leftarrow \text{NULL}$ 
  end
  DIST[s]  $\leftarrow 0$ 
  PQ.insert(v,0)
  while not PQ.empty() do
    u  $\leftarrow$  PQ.delmin() //liefertInfo
    foreach  $v \in V$  mit  $(u,v) \in E$  do
      d  $\leftarrow$  DIST[u] + c(u,v)
      if  $d < \text{DIST}[v]$  then
        if DIST[v] =  $\infty$  then
          PQ.insert(v,d)
        end
        else
          PQ.decrease(v,d)
        end
        DIST[v]  $\leftarrow$  d
        PRED[v]  $\leftarrow$  u
      end
    end
  end
end

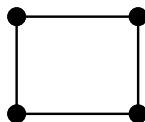
```

Laufzeitanalyse: $\mathcal{O}(\sum_{v \in V} (1 + \text{outdeg}(v)) + PQ_{Operationen})n * (T_{insert} + T_{delmin} + T_{empty}) + m * T_{decrease}$

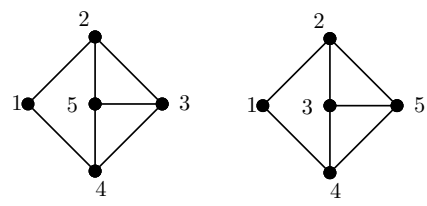
- Bei Binärem Heap: $\mathcal{O}(n * \log(n) + m * \log(n) = \mathcal{O}((n + m) * \log(n))$
- Fibonacci-Heap: Amortisierte Analyse ist ok, da Gesamtlaufzeit betrachtet. $\mathcal{O}(n * \log(n) + m)$, insert+empty = $\mathcal{O}(1)$, delmin = $\mathcal{O}(\log(n))$, decrease = $\mathcal{O}(1)$

Aufgabe 7

Eine Planare Einbettung ist genau dann eindeutig wenn diese 3-fach zusammenhängend ist:



Gleicher Graph verschiedene Einbettungen:



Aufgabe 8