HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

test

Max Mustermann

Fachhochschule Trier

19. Juli 2015

Table of content

eins

Komplexität

Implementierung

eins

test

Komplexität

Komplexität

Ursprüngliche Implementierung

Komplexität

HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

Ursprüngliche Implementierung

- Einzelschritte:

Komplexität

HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

Ursprüngliche Implementierung

- Einzelschritte:
  - Initialisieren der Arrays je $O(m)$

Komplexität

Ursprüngliche Implementierung

- Einzelschritte:
  - Initialisieren der Arrays je $O(m)$
  - Abarbeiten der grauen Knoten $O(m)$

Komplexität

Ursprüngliche Implementierung

- Einzelschritte:
  - Initialisieren der Arrays je $O(m)$
  - Abarbeiten der grauen Knoten $O(m)$
  - Bestimmen des Minimums $O(m)$

Komplexität

Ursprüngliche Implementierung

- Einzelschritte:
  - Initialisieren der Arrays je $O(m)$
  - Abarbeiten der grauen Knoten $O(m)$
  - Bestimmen des Minimums $O(m)$
  - Aktualisieren der Nachfolger $O(deg(v)) \rightarrow O(k)$

Komplexität

Ursprüngliche Implementierung

- Einzelschritte:
  - Initialisieren der Arrays je $O(m)$
  - Abarbeiten der grauen Knoten $O(m)$
  - Bestimmen des Minimums $O(m)$
  - Aktualisieren der Nachfolger $O(deg(v)) \rightarrow O(k)$
- insgesamt Komplexität $O(m^2)$

Komplexität

HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

Implementierung mit Heap

- Vorteile:

Komplexität

Implementierung mit Heap

- Vorteile:
  - Heapoperationen in $O(log m)$

Komplexität

Implementierung mit Heap

- Vorteile:
  - Heapoperationen in $O(logm)$
  - Bestimmen des Minimums in $O(logm)$

Komplexität

Implementierung mit Heap

- Vorteile:
  - Heapoperationen in $O(logm)$
  - Bestimmen des Minimums in $O(logm)$
- insgesamt Komplexität $O(k * logm)$

Implementierung

Eigenschaften

Implementierung

Eigenschaften

- Programmiersprache: Python

Implementierung

Eigenschaften

- Programmiersprache: Python
- Umsetzung mit Heap (Priority Queue)

Implementierung

Eigenschaften

- Programmiersprache: Python
- Umsetzung mit Heap (Priority Queue)
- keine Speicherung der Farbstufen wie bei Dijkstra

Implementierung

Eigenschaften

- Programmiersprache: Python
- Umsetzung mit Heap (Priority Queue)
- keine Speicherung der Farbstufen wie bei Dijkstra
  - kürzerer und übersichtlicherer Code

HOCHSCHULE TRIER
Trier University of Applied Sciences
Informatik - Computer Science

Kompletter Code

```
from heapq import heappush, heappop

def dijkstra_pq(G,s):
    m = len(G)                                  #O(1)
    pq = []         #priority queue             #O(1)
    d = [None]*m    #kosten                     #O(m)
    p = [None]*m    #vorgaenger               #O(m)
    d[s] = 0                                    #O(1)
    heappush(pq, (0,s))                         #O(log(m))
    while pq:                                   #O(m)
        (_,v) = heappop(pq)                     #O(log m)
        for u in G[v]:                          #O(deg(v)) --> O(k)
            alt = d[v] + G[v][u]                #O(1)
            if d[u]== None or alt < d[u]:       #O(1)
                d[u] = alt                    #O(1)
                p[u] = v                          #O(1)
            heappush(pq, (alt,u))               #O(log m)
    return d,p

def shortest_path(s,v,p):
    if v == None:
        return []
    else:
        return shortest_path(s,p[v],p) + [v]

#Graph:

def define_G():
    G = [   {1:1, 4:4, 2:2},    # Nachfolger von s
            {3:3, 4:1},         # von u
            {4:2, 5:3},         # von x
            {},                 # von y
            {3:1, 5:2},         # von v
            {}                  # von z
        ]
    return G


G = define_G()
d,p = dijkstra_pq(G,0)
print( shortest_path(0,5,p))
print( shortest_path(0,3,p))
```

Implementierung

Eingabe

- leer

Implementierung

Algorithmus

- Initialisierung

Implementierung

Algorithmus

- Initialisierung
- Erweitern

Implementierung

Algorithmus

- Initialisierung
- Erweitern
- Aktualisieren

Implementierung

Rekursives Bestimmen des Pfades

- leer

Implementierung

Aufruf

- leer