Prioritätswarteschlange

Eine Prioritätswarteschlange
zur Verwaltung
der günstigsten Alternativen

Prioritätswarteschlange

- Die Prioritätswarteschlange ist eine der Datenstrukturen, die zur Verwaltung von Alternativen bei Suchverfahren verwendet wird:
  - Stapel (stack) bei der Tiefensuche

    last in first out
  - Warteschlange (queue) bei der Breitensuche

    first in first out

  Beide ordnen die Elemente nach ihrem Auftreten ein.

Prioritätswarteschlange

- Die Prioritätswarteschlange ist eine der Datenstrukturen, die zur Verwaltung von Alternativen bei Suchverfahren verwendet wird:
  - Stapel *...*
  - Warteschlange *...*
  - Prioritätswarteschlange (priority queue) zB bei Dijkstra

    Elemente werden geordnet nach ihrer Bewertung eingefügt.

Prioritätswarteschlange

Allgemein

- Prinzipiell handelt es sich um ein objekt-orientiertes Vorgehen:
  Daten mit zugehörigen Zugriffsfunktionen
- Hier wird sie funktional realisiert, d.h. das Programm ***`prioritaetswarteschlange.py`***
  - stellt die zur Verwaltung einer Prioritaetswarteschlange notwendigen Funktionen bereit;
  - es benötigt ein Vergleichsprädikat ***`vor`*** , das vom einbindenden Programm kommt.

Prioritätswarteschlange

- fuege_eines_ein
  - fügt ein Element in die Prio-Schlange ein
- Kopf

```
def fuege_eines_ein(eines, vor, priows):
    return 'undefiniert'
```

Prioritätswarteschlange

- fuege_eines_ein
  - fügt ein Element in die Prio-Schlange ein
- Elementarfall Schlange leer, beispielsweise weil das Element an das Ende gehört

```
def fuege_eines_ein(eines, vor, priows):
    if len(priows)==0: return [eines]
    else: return 'undefiniert'
```

Prioritätswarteschlange

- fuege_eines_ein
  - fügt ein Element in die Prio-Schlange ein
- Elementarfall Element kommt hier hin

```
def fuege_eines_ein(eines, vor, priows):
    if len(priows)==0: return [eines]
    if vor(eines, priows[0]):
        return [eines]+priows
    else: return 'undefiniert'
```

Prioritätswarteschlange

- Sonst Position rekursiv weiter suchen, beim Backtracking Prio-WS wieder aufbauen

```
def fuege_eines_ein(eines, vor, priows):
    if len(priows)==0: return [eines]
    if vor(eines, priows[0]):
        return [eines]+priows
    else:
        return priows[0]+
               fuege_eines_ein(eines,
                               vor,
                               priows[1:]))
```

Prioritätswarteschlange

- Kurzversion

```
def fuege_eines_ein(eines, vor, priows):
    if len(priows)==0: return [eines]
    if vor(eines, priows[0]):
        return [eines]+priows
    return priows[0]+
           fuege_eines_ein(eines,
                           vor,
                           priows[1:]))
```

Prioritätswarteschlange

- fuege_eines_ein
  - fügt ein Element in die Prio-Schlange ein
- iterative Version der Funktion

```
def fuege_eines_ein(eines, vor, priows):
    for i in range(len(priows)):
        if vor(eines, priows[i]):
            return priows[:i]+[eines]+priows[i:]
    return priows + [eines]
```

Prioritätswarteschlange

- fuege_alle_ein
  - für Python passend keine Rekursion, sondern Einsatz einer for-Schleife

```
def fuege_alle_ein(alle, vor , priows):
    for eines in alle:
        priows=fuege_eines_ein(eines,
                               vor,
                               priows)

    return priows
```

Prioritätswarteschlange

Konkret bei Dijkstra für die Datenstruktur Kantenliste:

- Ordnung nach den Kantenbewertungen wird allein durch das Übergeben
  - des Vergleichsprädikats ***vor***
  - das Übergeben der einzufügenden Kanten
  - und das Halten der Prioritaetswarteschlange in Aufruf und Rückgabe

  realisiert.