SoSe 2020

Prof. Dr. Margarita Esponda

Objektorientierte Programmierung

4. Übungsblatt

Lernziel: Sortieralgorithmen in Python und Analyse deren Komplexität.

1. Aufgabe (9 Punkte)

a) Verbessern Sie den Shellsort-Algorithmus aus der Vorlesung, indem sie die Länge der Segmente aus der **Magic**-Liste wählen (siehe Vorlesungs-Folien).

b) Schreiben Sie eine Variante der Funktion, die die Anzahl der Vergleichs-Operationen zählt und als Rückgabewert zurückgibt.

c) Schreiben Sie ein Python-Programm, das bei gleichen Zahleneingaben die Anzahl der Vergleichs-Operation des Shellsort- und Insertsort-Algorithmus in eine tabellarische Ausgabe ausgibt. Sie sollen dabei sortierte und unsortierte Zahlen verwenden. Interessant wird, wenn die Eingabezahlen umgekehrt sortiert sind.

2. Aufgabe (16 Punkte)

a) Zeigen Sie mit Hilfe eines Zahlenbeispiels, dass der Heapsort-Algorithmus nicht stabil ist.

Nehmen Sie an, wir bekommen Nachrichten, mit Prioritäten in einem Wertebereich zwischen 1 und 50, die abgearbeitet werden müssen. Die Abarbeitung der Nachrichten soll streng nach Prioritäten erfolgen. Bei Nachrichten mit der gleichen Priorität soll die zeitliche Reihenfolge des Empfangs einer Nachricht respektiert werden (FIFO-Strategie).

$$(p_1, t_1, m_1),\ (p_2, t_2, m_2), \ldots, (p_i, t_i, m_i), \ldots$$

b) Programmieren Sie mit Hilfe der **yield**-Anweisung eine Funktion **next_message**, die bei jedem Aufruf der Funktion eine neue Nachricht mit zufälliger Priorität und mit entsprechendem Zeitstempel produziert.

Simulieren Sie unter Verwendung einer ***heap***-Struktur (siehe Vorlesungsfolien) eine Warteschlange, bei der die Nachrichten nach Priorität und Zeit abgearbeitet werden (*Priority Queue*).

Eine Prioritätswarteschlange speichert Objekte nach einer Prioritätseigenschaft und entfernt immer als erstes das Objekt mit der höchsten Priorität.

Vorgehensweise:

c) Definieren Sie zuerst folgende vier Hilfsfunktionen:

```
def insert ( priority_queue, message )
  """ Eine neue Nachricht wird in die Prioritätswarteschlange eingefügt """

def is_empty ( priority_queue )
  """ gibt einen Wahrheitswert als Rückgabewert zurück, je nachdem, ob die
      Prioritätswarteschlange leer oder nicht ist"""
```

```
def delete ( priority_queue )
  """ Die Nachricht mit der höchsten Priorität und die, die zeitlich zuerst produziert
      worden ist, wird aus der Prioritätswarteschlange entfernt und als Ergebnis der
      Funktion zurückgegeben """

def sort_messages ( priority_queue )
```

d) Schreiben Sie eine Funktion **sim_message_traffic**, die Nachrichten nach zufälligen Zeitabständen und Prioritäten einfügt bzw. entfernt. Die Nachrichten sollen mit Hilfe der **next_message** Funktion erzeugt werden. Wenn die Prioritätswarteschlange zwischendurch leer ist, soll die **delete**-Funktion den Wert **None** zurückgeben.

e) Analysieren Sie die Komplexität der Funktionen.

3. Aufgabe (5 Punkte)

a) Definieren Sie eine Variante des *Countingsort*-Algorithmus aus der Vorlesung, bei dem Sie die Elemente **in-Place** sortieren. Ihre *Countingsort* Variante muss nicht stabil sein. Sie dürfen als einzigen zusätzlichen Speicher das Hilfsrarray C verwenden (siehe Vorlesungsfolien). Das Hilfsarray **B** darf nicht mehr verwendet werden.

b) Analysieren Sie die Komplexität ihres Algorithmus.

4. Aufgabe (5 Punkte)

Was ist der effizienteste Weg, um eine Million 32-Bit-Ganzzahlen zu sortieren?. Die Frage hat Google an Obama im Jahr 2008 gestellt. Begründen Sie Ihre Antwort.

Implementieren Sie Ihre Lösung und testen Sie diese mit einer Million zufällig erzeugter 32-Bit Zahlen. Erläutern Sie Vor- und Nachteile Ihrer Lösung.

Wichtige Hinweise:

1) Verwenden Sie geeignete Namen für Ihre Variablen und Funktionen, die den semantischen Inhalt der Variablen oder die Funktionalität der Funktionen darstellen.
2) Verwenden Sie vorgegebene Funktionsnamen, falls diese angegeben werden.
3) Kommentieren Sie Ihre Programme.
4) Verwenden Sie geeignete Hilfsvariablen und Hilfsfunktionen in Ihren Programmen.
5) Löschen Sie alle Programmzeilen und Variablen, die nicht verwendet werden.
6) Schreiben Sie getrennte Test-Funktionen für alle Aufgaben.
7) Die Lösungen sollen elektronisch (Whiteboard-Upload) abgegeben werden.