Informatik II

Übung 4

FS 2019

Heutiges Programm

1. Feedback letzte Übung
2. Erklärung Ants Colony Optimization Algorithmus
3. [Code]Expert

1. Feedback letzte Übung

Sortieren

Bubblesort	min	max
Vergleiche	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Sequenz	egal	egal
Vertauschungen	0	$\mathcal{O}(n^2)$
Sequenz	$1, 2, \ldots, n$	$n, n-1, \ldots, 1$

Sortieren

InsertionSort	min	max
Vergleiche	$\mathcal{O}(n)$	$\mathcal{O}(n^2)$
Sequenz	$1, 2, \ldots, n$	$n, n-1, \ldots, 1$
Vertauschungen	0	$\mathcal{O}(n^2)$
Sequenz	$1, 2, \ldots, n$	$n, n-1, \ldots, 1$

SelectionSort	min	max
Vergleiche	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$
Sequenz	egal	egal
Vertauschungen	0	$\mathcal{O}(n)$
Sequenz	$1, 2, \ldots, n$	$n, n-1, \ldots, 1$

Sortieren

QuickSort	min	max
Vergleiche	$\mathcal{O}(n \log n)$	$\mathcal{O}(n^2)$
Sequenz	kompliziert	$1, 2, \ldots, n$
Vertauschungen	$\mathcal{O}(n)$	$\mathcal{O}(n \log n)$
Sequenz	$1, 2, \ldots, n$	kompliziert

kompliziert: Folge muss so gestaltet sein, dass der Pivot die Daten in jedem Schritt in zwei etwa gleich grosse Teile aufteilt. Zum Beispiel ($n = 7$): $4, 5, 7, 6, 2, 1, 3$

Bubble Sort in Python

```
values = inputInts()
swapped = True
while(swapped):
  swapped = False
  for i in range(1,len(values)):
    if values[i] > values[i−1]:
      values[i−1], values[i] = values[i], values[i−1]
      swapped = True
```

Binary Search Tree – Node Class

```
class Node:
  def __init__(self, k, l=None, r=None):
    """Constructor that takes a key k,
    and optionally a left and right node."""
    self.key, self.left, self.right, self.flagged = k, l, r, False
```

Binary Search Tree – Search

```
def search(k):
  """Searches the node with key k in the tree."""
  global tree # allow to access and modify the global variable tree
  v = tree
  while v != None:
    if k == v.key:
      return v
    elif k < v.key:
      v = v.left
    else:
      v = v.right
```

Binary Search Tree – add

```
def add(k):
  """Add key k to the tree."""
  global tree # allow to access and modify the global variable tree
  if tree is None:
    tree = Node(k)
    return True
   ...
```

Binary Search Tree – add (ctd)

```
t = tree;
while k != t.key:
  if k < t.key:
    if t.left is None:
      t.left = Node(k)
      return True
    else:
      t = t.left
  else:
    if t.right is None:
      t.right = Node(k)
      return True
    else:
      t = t.right
return False
```

Binary Search Tree – print in pre-order

```
def preorderprint(tree):
  """Print the keys of the tree in pre-order"""
  if tree is None:
    return
  print(tree.key, end=" ")
  preorderprint(tree.left)
  preorderprint(tree.right)
```

2. Erklärung Ants Colony Optimization Algorithmus

3. [Code]Expert

Fragen oder Anregungen?