

10.10.2023 9:56:57

priority\_queue\_test.py

Page 1/2

```

1
2 # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3 # Path : uebung04/ml/aufgabe02
4 # Version: Tue Oct 10 09:56:57 CEST 2023
5
6 import sys
7 import random
8 from queue import PriorityQueue as PythonPQ
9 from uebung04.ml.aufgabe02.priority_queue_adv import PriorityQueueAdv
10 from uebung04.ml.aufgabe02.priority_queue import PriorityQueue
11
12
13 def stress_test():
14     print("\nStress-Test: ... ", end = "")
15     NUMBER_OF_TESTS = 20000
16     LENGTH_RANGE = 10
17     DATA_RANGE = 10
18     i = 0
19     while i < NUMBER_OF_TESTS:
20         length = random.choice(range(1, LENGTH_RANGE))
21         randoms = []
22         j = 0
23         while j < length:
24             randoms.append(int(random.uniform(0, DATA_RANGE)))
25             j += 1
26         ourPQ = PriorityQueue(length)
27         pythonPQ = PythonPQ(length)
28         for r in randoms:
29             ourPQ.insert(r, "Value_" + str(r))
30             pythonPQ.put(r)
31         j = 0
32         while j < length:
33             if ourPQ.size() != pythonPQ.qsize():
34                 print("ERROR: wrong size!")
35                 print("randoms: " + str(randoms))
36                 sys.exit(1)
37             if ourPQ.remove_min().get_key() != pythonPQ.get():
38                 print("ERROR: wrong removeMin()!")
39                 print("randoms: " + str(randoms))
40                 sys.exit(2)
41             j += 1
42         if ourPQ.remove_min() != None:
43             print("ERROR: removeMin() != None")
44             print("randoms: " + str(randoms))
45             sys.exit(3)
46         i += 1
47     print("o.k.")
48
49
50

```

10.10.2023 9:56:57

priority\_queue\_test.py

Page 2/2

```

50
51 if __name__ == '__main__':
52
53     pq = PriorityQueue(7)
54     # pq = PriorityQueueAdv(7, "Uebung 4:PQ", 2, 2)
55
56     print("insert()'s: ")
57     pq.print()
58     pq.insert(4, "D")
59     pq.print()
60     pq.insert(5, "E")
61     pq.print()
62     pq.insert(3, "C")
63     pq.print()
64     pq.insert(2, "B")
65     pq.print()
66     pq.insert(1, "A")
67     pq.print()
68     print("\nmin(): " + str(pq.min()))
69     while pq.size() > 1:
70         print("remove_min(): " + str(pq.remove_min()))
71         pq.print()
72
73     stress_test()
74
75
76 """ Session-Log::
77
78 insert()'s:
79 [None, None, None, None, None, None, None, None]
80 [None, [(4,D),1], None, None, None, None, None, None]
81 [None, [(4,D),1], [(5,E),2], None, None, None, None, None]
82 [None, [(3,C),1], [(5,E),2], [(4,D),3], None, None, None, None]
83 [None, [(2,B),1], [(3,C),2], [(4,D),3], [(5,E),4], None, None, None]
84 [None, [(1,A),1], [(2,B),2], [(4,D),3], [(5,E),4], [(3,C),5], None, None]
85
86 min(): (1,A)
87 remove_min(): (1,A)
88 [None, [(2,B),1], [(3,C),2], [(4,D),3], [(5,E),4], None, None, None]
89 remove_min(): (2,B)
90 [None, [(3,C),1], [(5,E),2], [(4,D),3], None, None, None, None]
91 remove_min(): (3,C)
92 [None, [(4,D),1], [(5,E),2], None, None, None, None, None]
93 remove_min(): (4,D)
94 [None, [(5,E),1], None, None, None, None, None, None]
95
96 Stress-Test: ... o.k.
97
98 """

```

10.10.2023 9:56:57

priority\_queue.py

Page 1/2

```

1
2 # HSLU / ICS/AI ML : Modul ADS : Algorithmen & Datenstrukturen
3 # Path : uebung04/ml/aufgabe02
4 # Version: Tue Oct 10 09:56:57 CEST 2023
5
6 import functools
7 from uebung04.ml.aufgabe02.full_priority_queue_exception import FullPriorityQueueException
8
9
10 class PriorityQueue:
11     """ A heap-based (array-implementation) Priority-Queue with fixed length. """
12
13     @functools.total_ordering
14     class _PQEntry:
15
16         def __init__(self, key, value):
17             self._key = key
18             self._value = value
19
20         def get_key(self):
21             return self._key
22
23         def get_value(self):
24             return self._value
25
26         def __lt__(self, other):
27             if other == None:
28                 return False
29             return self._key < other._key
30
31         def __eq__(self, other):
32             if other == None:
33                 return False
34             return self._key == other._key
35
36         def __str__(self):
37             return "(" + str(self._key) + "," + str(self._value) + ")"
38
39
40     def __init__(self, max_size):
41         self._heap_array = [None] * (max_size+1)
42         self._last = 0 # Points to the last element in the heap.
43
44     def insert(self, key, value):
45         if self._last == (len(self._heap_array) - 1):
46             raise FullPriorityQueueException("Maximum reached: " + str(len(self._heap_array)))
47         self._last += 1
48         e = PriorityQueue._PQEntry(key, value)
49         self._heap_array[self._last] = e
50         self._upheap(self._last)
51         return e
52
53     def min(self):
54         return self._heap_array[1]
55
56     def remove_min(self):
57         if self._last == 0:
58             return None
59         result = self._heap_array[1]
60         self._heap_array[1] = self._heap_array[self._last]
61         self._heap_array[self._last] = None
62         self._last -= 1
63         self._downheap(1)
64         return result
65
66     def is_empty(self):
67         return self.size() == 0
68
69

```

10.10.2023 9:56:57

priority\_queue.py

Page 2/2

```

69
70     def size(self):
71         return self._last
72
73     def print(self):
74         print(self.__str__())
75
76     def __str__(self):
77         string = "["
78         for i in range(len(self._heap_array)):
79             entry = self._heap_array[i]
80             if entry != None:
81                 string += "[" + str(entry) + "," + str(i) + "]"
82             else:
83                 string += "None"
84             if i < len(self._heap_array)-1:
85                 string += ", "
86             string += "]"
87         return string
88
89     def _swap(self, parent_index, child_index):
90         """ Swaps a child-node with its parent-node.
91
92         parentIndex Index of the parent-node.
93
94         childIndex Index of the child-node.
95         """
96         tmp = self._heap_array[parent_index]
97         self._heap_array[parent_index] = self._heap_array[child_index]
98         self._heap_array[child_index] = tmp
99
100     def _upheap(self, current_index):
101         if current_index == 1:
102             return # no further _upheap-swaps possible
103         parent = current_index // 2
104         if self._heap_array[parent] > self._heap_array[current_index]:
105             self._swap(parent, current_index)
106             self._upheap(parent)
107
108     def _downheap(self, current_index):
109         left_child_index = current_index * 2
110         right_child_index = left_child_index + 1
111         left_child_is_smaller = self._check_for_potential_swap(current_index, left_child_index)
112         right_child_is_smaller = self._check_for_potential_swap(current_index, right_child_index)
113         if left_child_is_smaller and right_child_is_smaller:
114             if self._heap_array[left_child_index] <= self._heap_array[right_child_index]:
115                 self._swap_and_downheap(current_index, left_child_index)
116             else:
117                 self._swap_and_downheap(current_index, right_child_index)
118         elif left_child_is_smaller:
119             self._swap_and_downheap(current_index, left_child_index)
120         elif right_child_is_smaller:
121             self._swap_and_downheap(current_index, right_child_index)
122
123     def _check_for_potential_swap(self, parent, child):
124         return (child <= self._last) and (self._heap_array[parent] > self._heap_array[child])
125
126     def _swap_and_downheap(self, self, parent, child):
127         self._swap(parent, child)
128         self._downheap(child)
129

```

10.10.2023 9:56:57

**full\_priority\_queue\_exception.py**

Page 1/1

```
1
2 # HSLU / ICS/AIML : Modul ADS : Algorithmen & Datenstrukturen
3 # Path   : uebung04/ml/aufgabe02
4 # Version: Tue Oct 10 09:56:57 CEST 2023
5
6 class FullPriorityQueueException(Exception):
7
8     def __init__(self, err):
9         super().__init__(err)
10
11
```