11S20018

Ruth Aulya Silalahi

6.24 A queue can be implemented by using an array and maintaining the current size. The queue elements are stored in consecutive array positions, with the front item always in position 0. Note that this is not the most efficient method. Do the following:

a. Describe the algorithms for `getFront`, `enqueue`, and `dequeue`.
b. What is the Big-Oh running time for each of `getFront`, `enqueue`, and `dequeue` using these algorithms?
c. Write an implementation that uses these algorithms using the protocol in Figure 6.28.

Jawab:

a) Describe the algorithms for getFront, enqueque, and dequeque

getFront

```
algoritma getFront

//Method element()
public AnyType element()
//Check condition
if (isEmpty())
//Throw exception
throw new QueueException();
//Otherwise
else
//Return queue front
return Arr[qFront%qCapacity];
```

enqueue

```
algoritma enqueue

//Method enqueue()
public void enqueue (AnyType data)
//Check condition
if (isFull())
//Call function
doubleSize();
//Increment queue back
qBack++;
//Assign data to array
Arr[qBack%qCapacity] = data;
//Increment queue current
qCurrent++;
```

dequeue

```
algoritma dequeue

//Method remove()
public AnyType remove()
//Check condition
if (isEmpty())
//Throw exception
throw new QueueException();
//Update data
AnyType data = element();
//Update
Arr[qFront%qCapacity] = null;
//Increment queue front
qFront++;
//Decrement queue current
qCurrent--;
//Return data
return data;
```

b) What is the Big-Oh running time for each of getFront, enqueue, and dequeue using these algorithms?
Kompleksitas waktu getFront: O(1)
Kompleksitas waktu enqueue: O(1)
Kompleksitas waktu Dequeue: O(N)

c) Write an implementation that uses these algorithms using the protocol in Figure 6.28

figure 6.28
Possible Queue interface

```
1  package weiss.util;
2
3  /**
4   * Queue interface.
5   */
6  public interface Queue<AnyType> extends Collection<AnyType>
7  {
8      /**
9       * Returns but does not remove the item at the "front"
10      * of the queue.
11      * @return the front item of null if the queue is empty.
12      * @throws NoSuchElementException if the queue is empty.
13      */
14     AnyType element( );
15
16     /**
17      * Returns and removes the item at the "front"
18      * of the queue.
19      * @return the front item.
20      * @throws NoSuchElementException if the queue is empty.
21      */
22     AnyType remove( );
23 }
```

6.24\ArrayQueue.java

6.24\Collection.java

6.24\Iterator.java

6.24\Queue.java

6.24\QueueException.java

6.24\QueueTester.java

6.25 The operations that are supported by the `SortedSet` can also be implemented by using an array and maintaining the current size. The array elements are stored in sorted order in consecutive array positions. Thus `contains` can be implemented by a binary search. Do the following:

a. Describe the algorithms for `add` and `remove`.

b. What is the running time for these algorithms?

c. Write an implementation that uses these algorithms, using the protocol in Figure 6.1.

d. Write an implementation that uses these algorithms, using the standard `SortedSet` protocol.

Jawab:

a) Algoritma add

```
//Algorithm add()
add(Object x);
```

Algoritma remove

```
//Algorithm remove()
remove(Object x);
```

b) What is the running time for these algorithms?
Kompleksitas waktu add: O(log n)
Kompleksitas waktu : O(log n)

c) Write an implementation that uses these algorithms, using the protocol in Figure 6.1

```
1  package weiss.nonstandard;
2
3  // SimpleContainer protocol
4  public interface SimpleContainer<AnyType>
5  {
6      void insert( AnyType x );
7      void remove( AnyType x );
8      AnyType find( AnyType x );
9
10     boolean isEmpty( );
11     void makeEmpty( );
12 }
```

figure 6.1

A generic protocol for many data structures

6.25\SimpleContainer.java
6.25\SortedSet.java
6.25\TesterC.java

d) Write an implementation that uses these algorithms, using the standard SortedSet protocol
6.25\TesterD.java

3.

6.32 A `MultiSet` is like a `Set`, but allows duplicates. Consider the following interface for a `MultiSet`:

```
public interface MultiSet<AnyType>
{
    void add( AnyType x );
    boolean contains( AnyType x );
    int count( AnyType x );
    boolean removeOne( AnyType x );
    boolean removeAll( AnyType x );
    void toArray( AnyType [] arr );
}
```

There are many ways to implement the `MultiSet` interface. A `TreeMultiSet` stores items in sorted order. The data representation can be a `TreeMap`, in which the key is an item that is in the multiset, and the value represents the number of times the item is stored. Implement the `TreeMultiSet`, and make sure `toString` is provided.

Jawab:

6.32\Multiset.java

6.32\SetUtils.java

4.

6.37 Write a method that takes a `Map<String,String>` as a parameter and returns a new `Map<String,String>` in which keys and values are swapped. Throw an exception if there are duplicate values in the map that is passed as a parameter.

Jawab:

6.37\InterchangeMapKeyValue.java