Theory: Priority queue

11 minutes 7 / 7 problems solved

Start practicing

A **priority queue** is a data structure similar to a regular queue: we can also put and extract elements from it. The difference is that each element in a priority queue is associated with a value called **priority**. This value is used to order elements of a queue: elements with higher priority are retrieved before the elements with lower priority.

In other words, a priority queue is an abstract data type that provides the following operations:

- `is_empty(q)` — returns *true* if a queue doesn't contain any elements and *false* otherwise;
- `put(q, elem)` — put the specified element into a queue;
- `extract_max(q)` — extracts the element with the maximum priority.

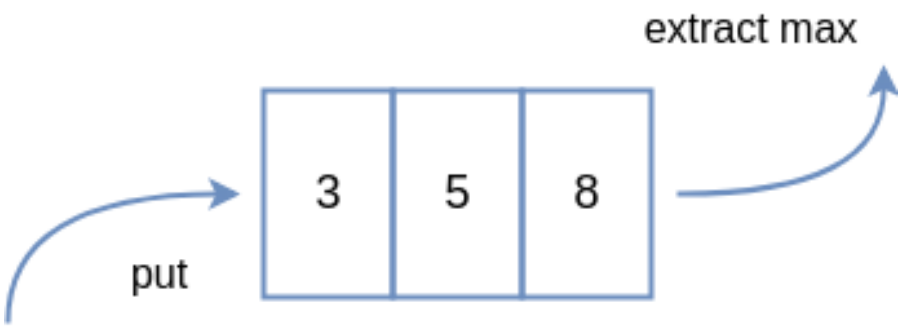
§1. An example

Let's see an example of how we can work with a priority queue. For simplicity, we will use integer numbers as elements. The priority of an element corresponds to the number itself.

First, let's put some values into a queue `q`:

```
put(q, 8)
put(q, 3)
put(q, 5)
```

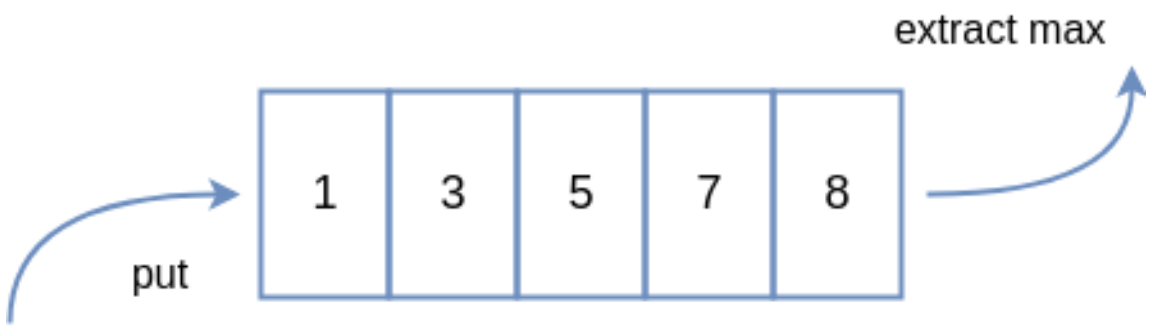
After the operations are done, the state of the queue can be depicted as follows:



There are three elements ordered according to their priorities. If we perform two more insertions:

```
put(q, 7)
put(q, 1)
```

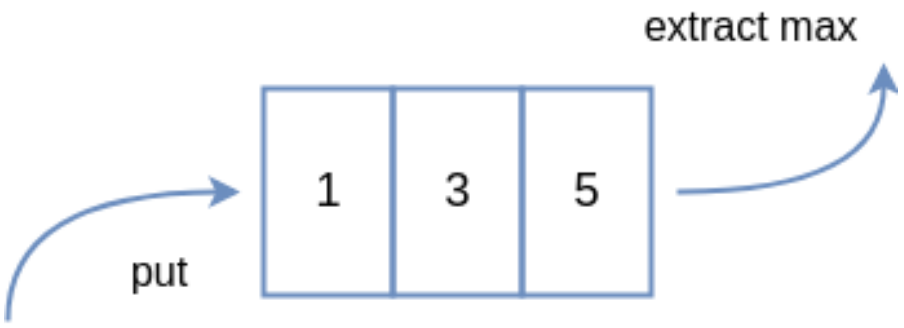
we will get the following state:



All the elements again are ordered according to their priorities. Now, let's extract some elements from the queue:

```
extract_max(q) # extracts 8
extract_max(q) # extracts 7
```

After that, we will get the following state:



When we extract the remaining elements:

```
extract_max(q) # extracts 5
extract_max(q) # extracts 3
extract_max(q) # extracts 1
```

we will have all of them extracted in the sorted order: 8, 7, 5, 3, 1.

The above priority queue is depicted as an ordered array. However, such an implementation is only one possible variant, which is usually not used due to inefficiency. More efficient implementations are based on a structure called a *heap*. Here, we will not describe the implementation details but concentrate only on the main concepts of how a priority queue works.

§2. Applications

A priority queue is used in many algorithms where some objects need to be accessed according to their priorities.

One example is Dijkstra's algorithm, that allows us to find the shortest paths between the nodes of a graph. At each step of the algorithm, we need to find the node with the minimum current distance. To do that, a priority queue might be used.

Another example is the Huffman algorithm that is used for data compression. The algorithm calculates the symbol frequencies in text and then use them to build an optimal code for each symbol. A priority queue is used here to retrieve symbols according to their frequencies.

Report a typo

45 users liked this theory. **0** didn't like it. What about you?

Start practicing

Comments (0) Hints (0) Useful links (0) Show discussion