

**“AZƏRBAYCAN HAVA YOLLARI” CJSC NATIONAL AVIATION ACADEMY**

**Individual Work № 4:**

**Topic: Heaps**

**Subject: Obyektyönümlü proqramlaşdırma**

**Teacher: Mamed Shahmaliyev**

**Group: 1459i Student: Mehrali Babayev**

**Date: 11.10.2021 Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_**

**­ Baku 2021**

**What is a Heap?**

Heap is a data structure that follows a complete binary tree's property and satisfies the heap property. Therefore, it is also known as a binary heap. As we all know, the complete binary tree is a tree with every level filled and all the nodes are as far left as possible. In the binary tree, it is possible that the last level is empty and not filled. Now, you must be wondering what is the heap property? In the heap data structure, we assign key-value or weight to every node of the tree. Now, the root node key value is compared with the children’s nodes and then the tree is arranged accordingly into two categories i.e., max-heap and min-heap. Heap data structure is basically used as a heapsort algorithm to sort the elements in an array or a list. Heapsort algorithms can be used in priority queues, order statistics, Prim's algorithm or Dijkstra's algorithm, etc. In short, the heap data structure is used when it is important to repeatedly remove the objects with the highest or the lowest priority.

## As learned earlier, there are two categories of heap data structure i.e. max-heap and min-heap. Let us understand them below but before that, we will study the heapify property to understand max-heap and min-heap.

## What is Heapify?

Before moving forward with any concept, we need to learn what is heapify. So, the process of creating a heap data structure using the binary tree is called Heapify. The heapify process is used to create the Max-Heap or the Min-Heap. Let us study the Heapify using an example below:

**What is Max Heap?**

When the value of each internal node is larger than or equal to the value of its children node then it is called the Max-Heap Property. Also, in a max-heap, the value of the root node is largest among all the other nodes of the tree. Therefore, if “a” has a child node “b” then

Key(a) >= key(b)

represents the Max-Heap Property. Let us display the max-heap using an array. Therefore, the root node will be arr[0]. So, for kth node i.e., arr[k]:

arr[(k - 1)/2] will return the parent node

arr[(2\*k) + 1] will return left child

arr[(2\*k) + 2] will return right child

**What is Min Heap?**

When the value of each internal node is smaller than the value of its children node then it is called the Min-Heap Property. Also, in the min-heap, the value of the root node is the smallest among all the other nodes of the tree. Therefore, if “a” has a child node “b” then

Key(a) < key(b)

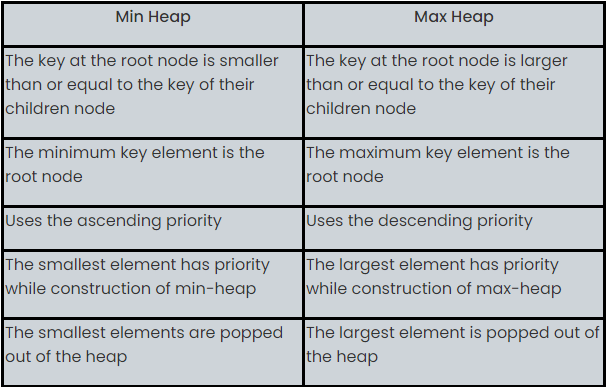
represents the Min Heap Property. Let us display the max heap using an array. Therefore, the root node will be arr[0]. So, for kth node i.e., arr[k]:

arr[(k - 1)/2] will return the parent node

arr[(2\*k) + 1] will return left child

arr[(2\*k) + 2] will return right child

**Min Heap vs Max Heap**



**Create a Heap**

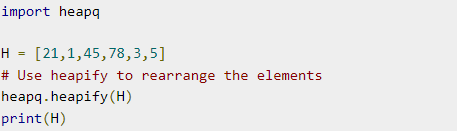
A heap is created by using python’s inbuilt library named heapq. This library has the relevant functions to carry out various operations on heap data structure. Below is a list of these functions.

* **heapify** − This function converts a regular list to a heap. In the resulting heap the smallest element gets pushed to the index position 0. But rest of the data elements are not necessarily sorted.
* **heappush** − This function adds an element to the heap without altering the current heap.
* **heappop** − This function returns the smallest data element from the heap.
* **heapreplace** − This function replaces the smallest data element with a new value supplied in the function.

## Creating a Heap

A heap is created by simply using a list of elements with the heapify function. In the below example we supply a list of elements and the heapify function rearranges the elements bringing the smallest element to the first position.

### **Example**



### **Output**

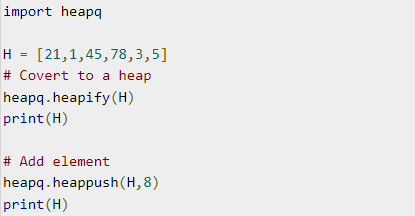
When the above code is executed, it produces the following result –



## Inserting into heap

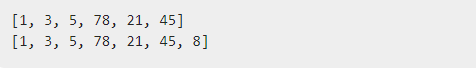
Inserting a data element to a heap always adds the element at the last index. But you can apply heapify function again to bring the newly added element to the first index only if it smallest in value. In the below example we insert the number 8.

### **Example**



### **Output**

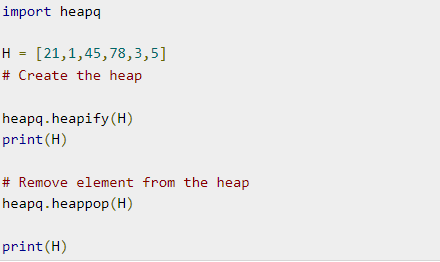
When the above code is executed, it produces the following result −



## Removing from heap

You can remove the element at first index by using this function. In the below example the function will always remove the element at the index position 1.

### **Example**



### **Output**

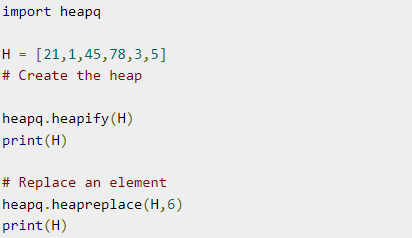
When the above code is executed, it produces the following result –



## Replacing in a Heap

The heap replace function always removes the smallest element of the heap and inserts the new incoming element at some place not fixed by any order.

### **Example**



### **Output**

When the above code is executed, it produces the following result −

