Assignment 7: Heap

July 2025 | CSE 106

Deadline: 14 February

Objective

In this assignment, you will implement a **Min Heap** data structure using an array representation. You will implement various heap operations including building a heap from an unsorted array (heapify).

Introduction to Min Heap

A **Min Heap** is a complete binary tree where each parent node is **less than or equal to** its child nodes. The root node contains the **smallest** value.

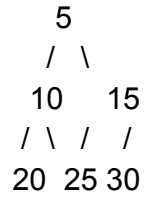

Figure: An example of a Min Heap

Array Representation of Min Heap

A min heap can be represented as an array (Arr).

- The root element will be at Arr[0]
- For any node Arr[i]:
    - **Left child** is stored at index 2i + 1
    - **Right child** is stored at index 2i + 2

- **Parent** is stored at index ⌊(i - 1) / 2⌋

Example: The heap shown above is represented as: [5, 10, 15, 20, 25, 30]

Min Heap Operations

Basic Operations

1. **insert(x):**
   Inserts a new element x into the heap. The heap property is restored using **sift-up**.
   Example: If the heap is [5, 10, 15] and insert(3) is called, the heap becomes [3, 10, 5, 15].
2. **find-min():**
   Returns the minimum element (root) of the heap without removing it.
   Example: If the heap is [5, 10, 15, 20], calling find-min() returns 5.
3. **extract-min():**
   Removes and returns the minimum value (the root) from the heap. After removal, **sift-down** restores the heap property.
   Example: For the heap [5, 10, 15], extract-min() returns 5 and modifies the heap to [10, 15].
4. **decrease-key(index, new_value):**
   Decreases the value of the element at index to new_value and restores the heap property using **sift-up**.
   Example: If the heap is [5, 10, 15] and decrease-key(2, 3) is called, the heap becomes [3, 10, 5].
5. **delete-key(index):**
   Deletes the element at the given index. This can be done by decreasing the key to negative infinity (or INT_MIN) and then extracting the minimum.
   Example: For the heap [5, 10, 15], delete-key(1) removes 10 and rebalances to [5, 15].

Inspection Operations

6. **get-size():**
   Returns the number of elements in the heap.
7. **is-empty():**
   Returns true if the heap is empty, false otherwise.

8. **print-heap():**
   Prints all elements of the heap in array order.
9. **is-valid-min-heap():**
   Returns true if the Min Heap property is preserved, false otherwise.

Advanced Operations

10. **heapify(array, n):**
    Builds a heap from an unsorted array of n elements in **O(n)** time using the bottom-up approach.
    Example: Given array [30, 10, 20, 5, 15], after heapify, the heap becomes [5, 10, 20, 30, 15].
11. **heap-sort():**
    Returns all elements of the heap in sorted (ascending) order. The original heap should remain unchanged.
    Example: For heap [5, 10, 15], heap-sort() returns [5, 10, 15].
12. **replace-min(x):**
    Removes the minimum element and inserts a new element x in a single operation (more efficient than extract + insert).
    Example: For heap [5, 10, 15], replace-min(12) returns 5 and the heap becomes [10, 12, 15].

Internal Operations

13. **sift-up(index):**
    Moves the element at index upwards to restore the heap property after insertion or key decrease.
14. **sift-down(index):**
    Moves the element at index downwards to restore the heap property after deletion or replacement.

Input Format

Each line of input contains a command. The command mapping is as follows:

Command	Operation	Description
1 x	insert(x)	Insert value x into the heap
2	extract-min()	Extract and return the minimum element
3	find-min()	Return the minimum value without removing
4	get-size()	Return the size of the heap
5	is-empty()	Return true if heap is empty
6 i	delete-key(i)	Delete element at index i
7 i x	decrease-key(i, x)	Decrease value at index i to x
8	print-heap()	Print the heap array
9	is-valid-min-heap()	Check if Min Heap property holds
10 n v1 v2 ... vn	heapify(arr, n)	Build heap from n values
11	heap-sort()	Return sorted elements
12 x	replace-min(x)	Replace min with x

Instructions

- You have been given a header file named MinHeap.h. You only need to edit this file in the marked places: /**Write your code here**/. Implement all the operations.
- A main.cpp file is provided to evaluate your implementations. **Do not modify this file.**
- Input and output are handled through files. Enter inputs in input.txt, and outputs will be generated in output.txt.

Sample I/O

Sample Input (input.txt)

```
1 20
1 10
1 15
8
9
3
1 5
8
2
8
10 5 40 30 10 20 25
8
9
11
12 8
8
```

Sample Output (output.txt)

```
Inserted 20 into the heap.
Inserted 10 into the heap.
Inserted 15 into the heap.
Heap: 10 20 15
Min Heap property is preserved.
Min: 10
Inserted 5 into the heap.
Heap: 5 10 15 20
Extracted Min: 5
Heap: 10 20 15
Heap built from array.
Heap: 10 20 25 40 30
Min Heap property is preserved.
Sorted: 10 20 25 30 40
Replaced Min: 10 with 8
Heap: 8 20 25 40 30
```

Submission Guidelines

1. Create a directory with your **7-digit student ID** as its name.
2. Place all source files (.cpp, .h) into that directory.
3. Compress the directory in .zip format (.rar, .7z, etc. are **not** acceptable).
4. Upload the .zip file to Moodle.

Marks Distribution

Task	**Marks**
sift-up(index)	15
sift-down(index)	15
insert(x)	8
extract-min()	8
delete-key(i)	8
decrease-key(i, x)	8
heapify(arr, n)	10
is-valid-min-heap()	8
heap-sort()	5
replace-min(x)	5
Utility functions: find-min, get-size, is-empty, print-heap	10
Total	**100**