DSA ASSIGNMENT 5

BT22CSH011

Devashish Athawale

Problem Statement:

Q1) Build a min heap from array and make sure that heap order property is maintainedafterevery input. The input is checked if it is greater than it's parent, if it's not, it is swapped. You can take the sample array as: arr = [1,5,6,8,9,7,3]

Write a generalized program so that user can input any set of values. Your insertion functionshould take O(n) time as compared to O(n log n) time.

Q2) Write a delete_max function to delete the element with the maximum key in a min-maxheap. It should take O (log n).

Q3) Write a program to implement Heap sort by building a heap for the given set of elementsand then deleting one element at a time. Show the complexity of your code.

Q1)

```
#include <iostream>
#include <vector>


void adjustHeap(std::vector<int>& data, int size, int index) {
int smallest = index;
int left = 2 * index + 1;
int right = 2 * index + 2;
if (left < size && data[left] < data[smallest])
{
smallest = left;
}
if (right < size && data[right] < data[smallest])
{
smallest = right;
}
if (smallest != index)
{
std::swap(data[index], data[smallest]);
adjustHeap(data, size, smallest);
}
}
void buildHeap(std::vector<int>& data)
{
int size = data.size();
for (int i = size / 2 - 1; i >= 0; i--)
{
adjustHeap(data, size, i);
}
}
void insertValue(std::vector<int>& arr, int data)
{
arr.push_back(data);
int i = arr.size() - 1;
while (i > 0 && arr[i] < arr[(i - 1) / 2])
{
std::swap(arr[i], arr[(i - 1) / 2]);
i = (i - 1) / 2;
}
}

int main() {
std::vector<int> data = { 1, 5, 6, 8, 9, 7, 3 };
buildHeap(data);
std::cout << "Min Heap: \n";
std::cout << "Original: ";
for (int i = 0; i < data.size(); i++)
{
std::cout << data[i] << " ";
}
std::cout << std::endl;
int n = 4;
insertValue(data, n);
std::cout << "After inserting " << n << ": ";
for (int i = 0; i < data.size(); i++)
{
std::cout << data[i] << " ";
}
std::cout << std::endl;
return 0;
}
```

Output:

```
Min Heap:
Original: 1 5 3 8 9 7 6
After inserting 4: 1 4 3 5 9 7 6 8
```

Q2)

```
#include <iostream>
#include <vector>


void heapify(std::vector<int>& arr, int size, int index, bool isMinLevel) {
int largeOrSmall = index;
int left = 2 * index + 1;
int right = 2 * index + 2;
if (isMinLevel) {
if (left < size && arr[left] < arr[largeOrSmall]) {
largeOrSmall = left;
}
if (right < size && arr[right] < arr[largeOrSmall]) {
largeOrSmall = right;
}
}
else {
if (left < size && arr[left] > arr[largeOrSmall]) {
largeOrSmall = left;
}
if (right < size && arr[right] > arr[largeOrSmall]) {
largeOrSmall = right;
}
}
if (largeOrSmall != index) {
std::swap(arr[index], arr[largeOrSmall]);
heapify(arr, size, largeOrSmall, !isMinLevel);
}
}

int deleteMaxElement(std::vector<int>& arr)
{
if (arr.empty())
{
std::cout << "Heap is empty!" << std::endl;
return -1; // Return a sentinel value to indicate an empty heap.
}
int maxElement = arr[0];
int lastIndex = arr.size() - 1;
std::swap(arr[0], arr[lastIndex]);
arr.pop_back();
bool isMinLevel = true;
heapify(arr, arr.size(), 0, isMinLevel);
return maxElement;
}
void main()
{
std::vector<int> arr = { 9, 8, 6, 7, 5, 1, 3 };
std::cout << "Original Min-Max Heap: ";
for (int i = 0; i < arr.size(); i++)
{
std::cout << arr[i] << " ";
}
std::cout << "\n";
std::cout << "Element Deleted: " << deleteMaxElement(arr) << std::endl;
std::cout << "Remaining Min-Max Heap: ";
for (int i = 0; i < arr.size(); i++)
{
std::cout << arr[i] << " ";
}
}
```

Output:

```
Microsoft Visual Studio Debu
Original Min-Max Heap: 9 8 6 7 5 1 3
Element Deleted: 9
Remaining Min-Max Heap: 3 8 6 7 5 1
```

Q3)

```
#include <iostream>
#include <vector>


void heapify(std::vector<int>& arr, int size, int i)
{
int largest = i;
int left = 2 * i + 1;
int right = 2 * i + 2;
if (left < size && arr[left] > arr[largest])
{
largest = left;
}
if (right < size && arr[right] > arr[largest])
{
largest = right;
}
if (largest != i)
{
std::swap(arr[i], arr[largest]);
heapify(arr, size, largest);
}
}

void buildMaxHeap(std::vector<int>& arr)
{
for (int i = arr.size() / 2 - 1; i >= 0; i--)
{
heapify(arr, arr.size(), i);
}
}

void heapSort(std::vector<int>& arr)
{
buildMaxHeap(arr);
for (int i = arr.size() - 1; i > 0; i--)
{
std::swap(arr[0], arr[i]);
heapify(arr, i, 0);
}
}

void main()
{
std::vector<int> arr = { 12, 11, 13, 5, 6, 7 };
std::cout << "Original array: ";
for (int i = 0; i < arr.size(); i++)
{
std::cout << arr[i] << " ";
}
std::cout << std::endl;
heapSort(arr);
std::cout << "Sorted array: ";
for (int i = 0; i < arr.size(); i++)
{
std::cout << arr[i] << " ";
}
std::cout << std::endl;
}
```

Output:

```
Original array: 12 11 13 5 6 7
Sorted array: 5 6 7 11 12 13
```

Time Complexity: n log n