Sarvajanik College of Engineering and Technology

Branch : Computer Engineering (Shift 1)

Subject : Analysis and Design of Algorithms (2150703)

Faculty : Prof. (Dr.) Mayuri Mehta

: Prof. Urvashi Mistry

Topic : Real time application of Heap-sort

Submitted by :

Name	Enrollment No.
Yash Patel	160420107043
Parth Roy	160420107046
Saurabh Maheshwari	160420107048
Jainam Shah	160420107051
Parth Shekhaliya	160420107053

Contents

Heap Sort

Heap is a specialized tree-based data structure that satisfies the heap property.

Heap properties:

A. It has to be a complete binary tree.

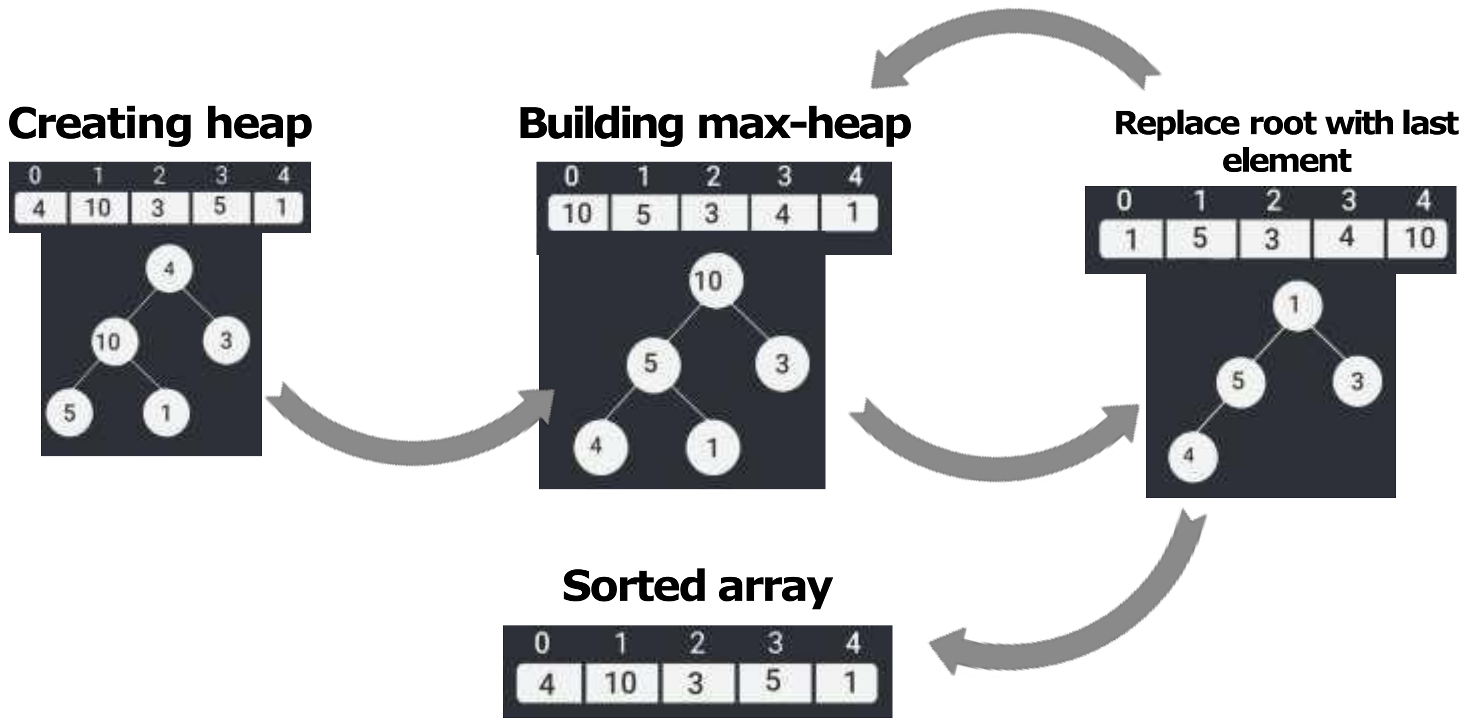
B. It has to have any of the following properties:

- Max Heap
  - Store data in ascending order
  - Has property of
    - A[Parent(i)] ≥ A[i]
- Min Heap
  - Store data in descending order
  - Has property of
    - A[Parent(i)] ≤ A[i]

Heapsort algorithm

1. Build a max heap from the input data.
2. Replace the root of the heap with the last item of the heap followed by reducing the size of heap.
3. Heapify the root of tree.
4. Repeat above steps until size of heap is greater than 1

Heapsort algorithm



Heapsort Algorithm

```
void heapSort(int arr[], int n)
{
   for(int i = n/2 - 1;i >= 0; i--)
   heapify(arr, n, i);
   for(int i = n - 1;i >= 0; i--)
   {
     swap(arr[0], arr[i]);
     heapify(arr,i,0);
   }
}
```

Creating heap

Swaps the first and last node

Creates max heap on reduced array

Merits and Demerits

Merits

- Time Complexity of heap sort is Ɵ(nlogn).
- It is in-place sorting algorithm so it doesn't require extra space for sorting elements.
- Space complexity of heap sort is Ɵ(1).
- It works better than selection and bubble sort.

Demerits

- For random input heap sort works slower than quick sort.
- It is not a stable sorting algorithm.

Best Case	Average Case	Worst case
O(nlogn)	*O*(nlogn)	*O*(nlogn)

Heap In Memory Management

Data structure "heap" might be used in various places.

Heap used for dynamic memory allocation wherever it is needed.

How to use heap in memory :

- The heap is a region of computer's memory that is not managed automatically for , and is not as tightly managed by the CPU.
- To allocate memory on the heap, we use malloc() or calloc(), which are built-in C functions.

Why we use heap in memory???

- Variables can be accessed globally
- No limit on memory size
- No guaranteed efficient use of space, memory may become fragmented over time as blocks of memory are allocated, then free'd
- User must manage memory
- Variables can be resized using realloc()

Priority queue

- A **priority queue** is an abstract data type which is like a regular queue or stack data structure, but where additionally each element has a "priority" associated with it. In a priority queue, an element with high priority is served before an element with low priority.
- Priority queues can be efficiently implemented using Binary Heap because it supports insert(), delete() and extractmax(), decreaseKey() operations in O(logn) time.

Priority Queue

- A typical Priority Queue requires following operations to be efficient.

1. Get Top Priority Element (Get minimum or maximum)
2. Insert an element
3. Remove top priority element
4. Decrease Key

- A Binary Heap supports above operations with following time complexities:

1. O(1)
2. O(Logn)
3. O(Logn)
4. O(Logn)

Priority Queue

- The major advantage of using the binary heap for priority queue is that you can add new values to it efficiently after initially constructing it.
- If all the values in the queue are known in advance, you can just sort the values, as you've mentioned. But what happens when you the want to add a new value to the priority queue?
- This might take time Θ(n) in the worst case because you'd have to shift down all the array elements to make space for the new element that you just added.
- On the other hand, insertion into a binary heap takes time O(log n), which is exponentially faster.

Graph Algorithms

- By using heaps as internal traversal data structures, run time will be reduced by polynomial order. Examples of such problems are Prim's minimal-spanning-tree algorithm and Dijkstra's shortest-path algorithm.
- For Prim’s Algorithm, using adjacency matrix and adjacency list it requires $O(|V^2|)$.
- *While using binary heap and adjacency list, time complexity is reduced to O*(|V|+|E|log|V|).

Dijkstra Algorithm

- A min-priority queue is an abstract data type that provides 3 basic operations : add_with_priority(), decrease_priority() and extract_min().
- As mentioned earlier, using such a data structure can lead to faster computing times than using a basic queue.
- Time Complexity of normal Dijkstra algorithm: $O(|E|+|V^2|)$
- Using Binary heap : $\Theta(|E|+|V|\log|V|)$

Embedded Systems

- Embedded systems are devices that perform a few simple functions.
- Embedded devices typically have limited power, memory and computational resources.
- Many embedded systems applications involve storing and querying large datasets.
- Sorting algorithms are commonly used in query processing.

Embedded Systems

- The optimal sorting algorithm for embedded systems has these characteristics:
- It must sort in place.
- The algorithm must not be recursive.
- Its best, average and worst case running times should be of similar magnitude.
- Its running time should increase linearly or logarithmically with the number of elements to be sorted.

Embedded Systems

- *Merge sort and quick sort are better than heap sort but they require extra space.*
- *In Embedded systems there is less space available, so quick sort and merge sort doesn't fit.*
- *So heap sort is better for Embedded systems*
- *Ex. Systems concerned with security and embedded system such as Linux Kernel uses Heap Sort because of the O(nlog(n)) which reduces the processing time of the system.*

K^{th} largest or smallest Element

- Selection algorithm is an algorithm for finding the kth smallest or largest number in a list or array.
- For finding kth smallest or largest number, in a list or array we require it to sort first.
- For this heap sort can be used because of its less time complexity than using other sorting techniques.
- For this we can use Max heap or Min heap to find kth smallest or largest number

K^{th} largest or smallest Element

- A heap allows access to the min or max element in constant time, and other selections i.e. kth-element can be done in sub-linear time on data that is in a heap.
- Time Complexity: O(n + klogn)

1) Build a Max Heap tree in O(n)
2) Use Extract Max k times to get k maximum elements from the Max Heap O(klogn)

References

- https://en.wikipedia.org/wiki/Heap_(data_structure)
- https://embeddedgurus.com/stack-overflow/2009/03/sorting-in-embedded-systems/
- https://www.geeksforgeeks.org
- https://people.ok.ubc.ca/rlawrenc/research/Students/TC_11_Presentation.pptx

“

Thank You !!!

”