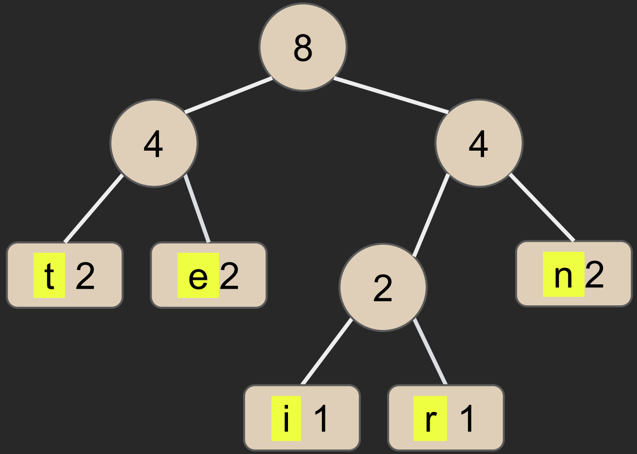
Algorithms Analysis and Design from scratch

تحليل وتصميم Algorithms من تحت

إحمد متولي

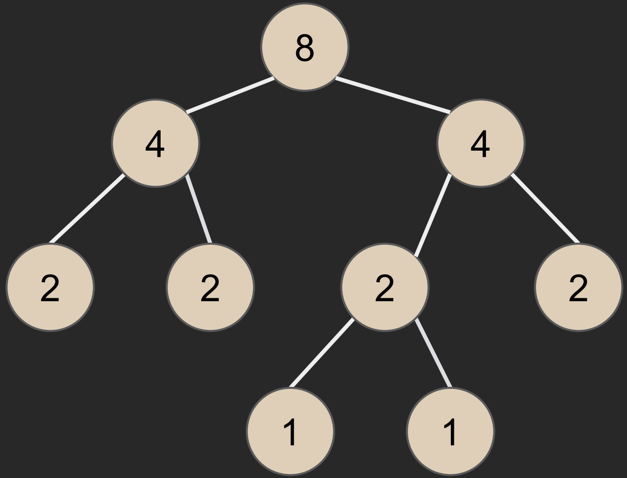
METWALLY LABS

Binary Tree

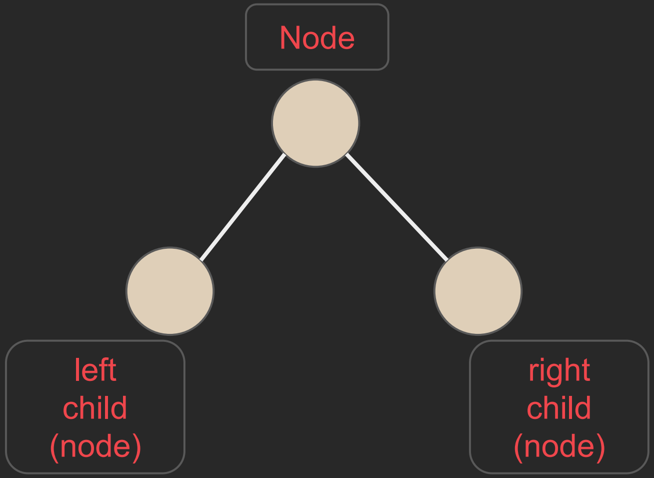


Data Structures

Binary Tree

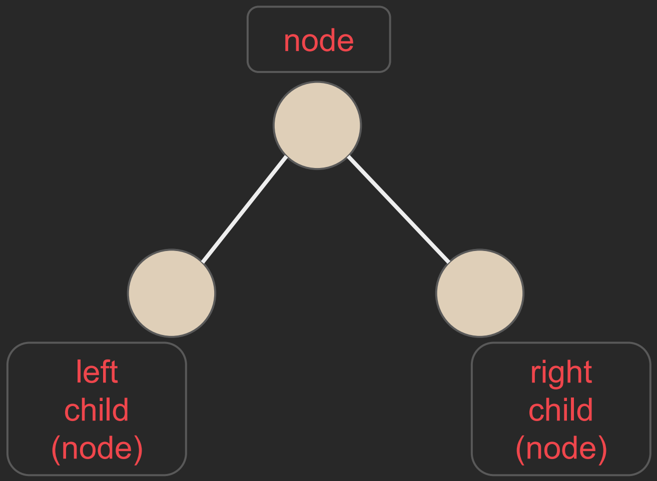


Binary Tree



Data Structures

Binary Tree



node0

data: 8
right: *node1
left: *node2

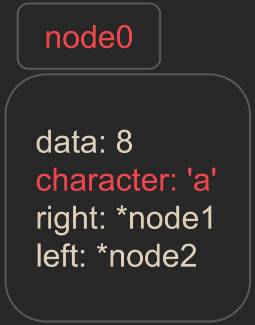
node1

data: 4
right: null
left: null

node2

data: 4
right: null
left: null

Binary Tree

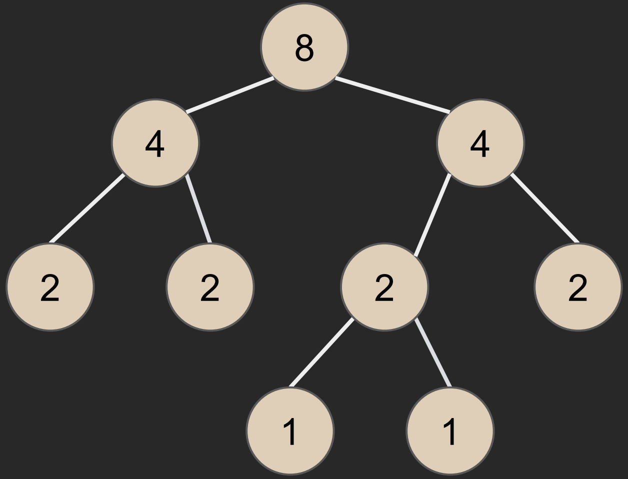


node0

data: 8
character: 'a'
right: *node1
left: *node2

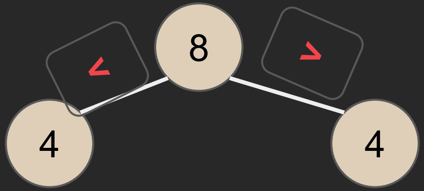
Heap

It's a complete binary tree.



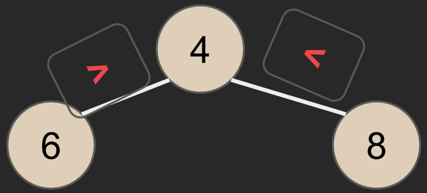
Heap

Max Heap



Heap

Min Heap



Data Structures

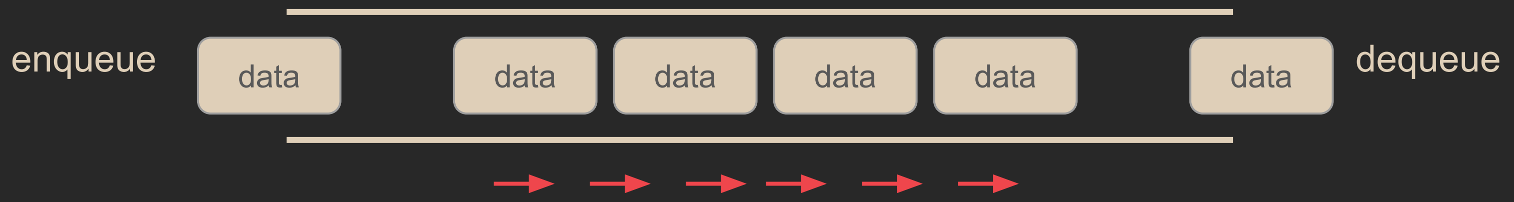
Heap Operations

- Heapify:
  Convert any data structure to Heap data structure (e.g. convert array to heap)

- Find-max (or Find-min)
- Insertion
- Deletion
- Extract Min-Max: Returning and deleting the maximum or minimum element.

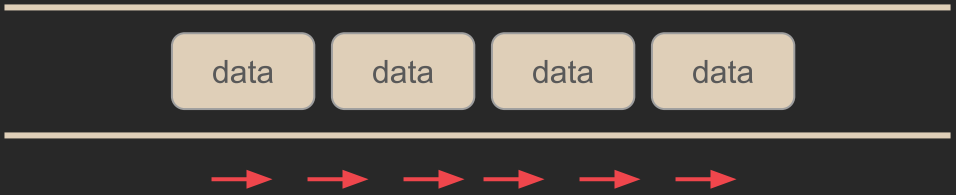
Data Structures

Queue



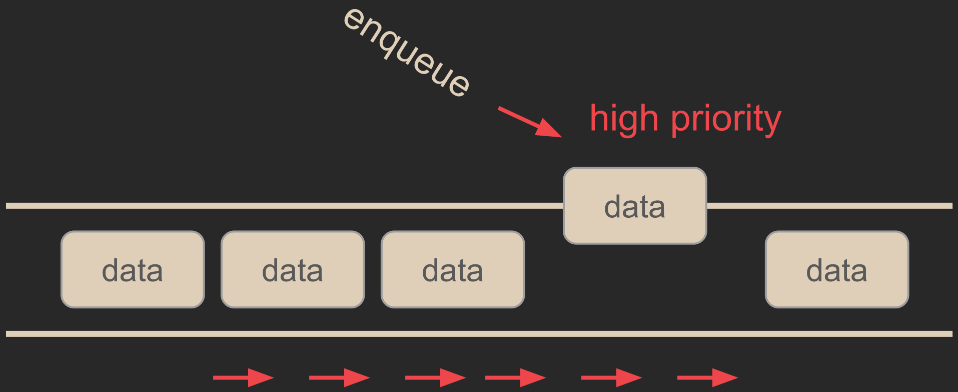
Data Structures

Priority Queue



Data Structures

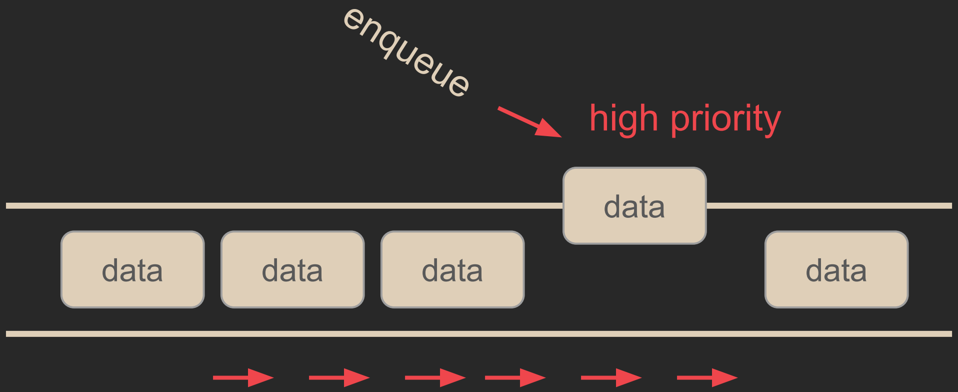
Priority Queue



sorted by priority

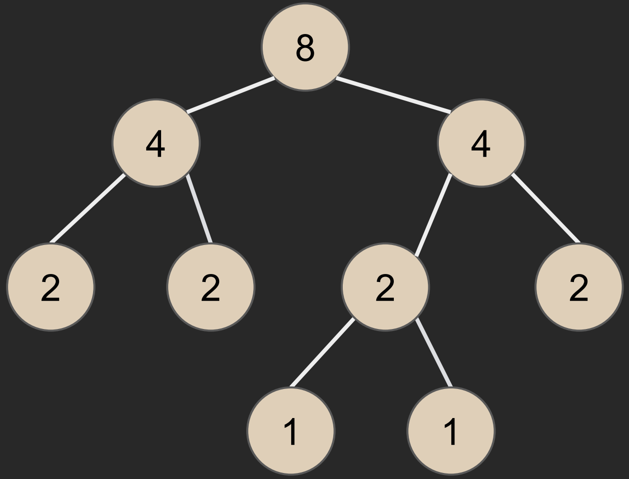
Data Structures

Priority Queue



sorted by priority

It's a Min or Max Heap.



Data Structures

- Priority Queue is Min-Heap or Max-Heap

- Heap is a complete binary tree

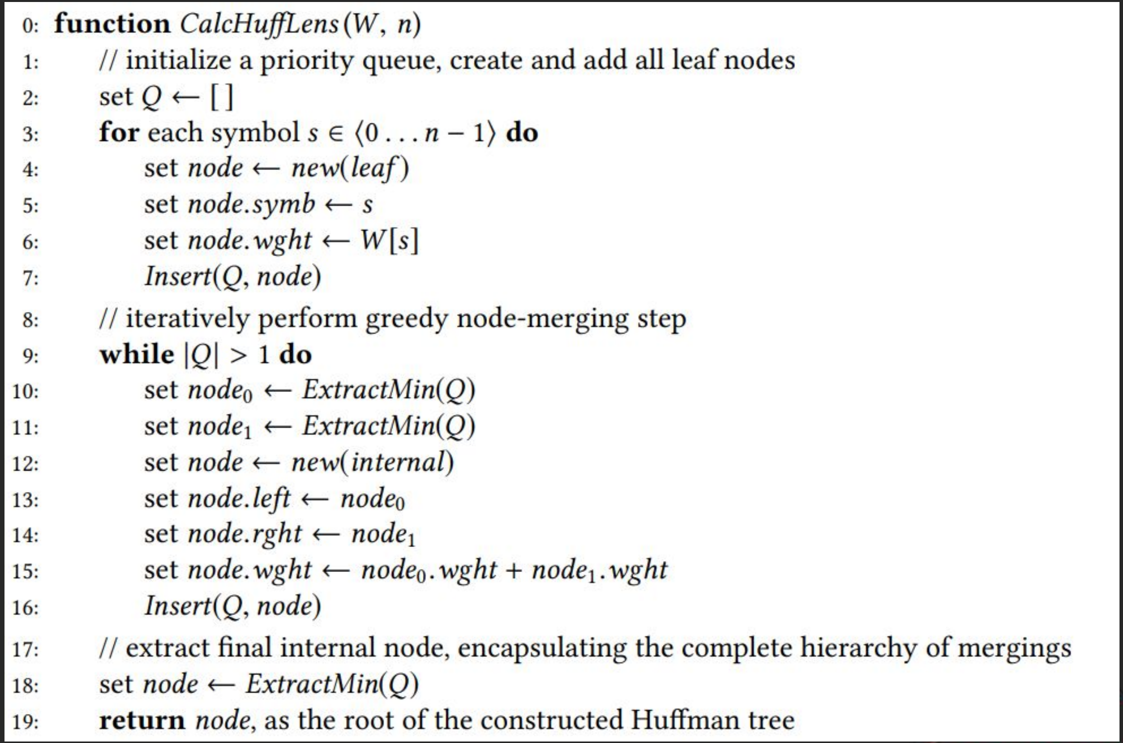
- Binary tree is a collection of related nodes

- Node is an object has data and references to another nodes

Greedy Algorithm

Huffman coding

```
HUFFMAN(C)
n |C|
Q C
for i 1 to n - 1
    do z ALLOCATE-NODE()
        x left[z] EXTRACT-MIN(Q)
        y right[z] EXTRACT-MIN(Q)
        f[z] f[x] + f[y]
        INSERT(Q, z)
return EXTRACT-MIN(Q)
```



```
function CalcHuffLens(W, n)
    // initialize a priority queue, create and add all leaf nodes
    set $Q \leftarrow [\,]$
    for each symbol $s \in \langle 0 \ldots n-1 \rangle$ do
        set $node \leftarrow new(leaf)$
        set $node.symb \leftarrow s$
        set $node.wght \leftarrow W[s]$
        Insert(Q, node)
    // iteratively perform greedy node-merging step
    while $|Q| > 1$ do
        set $node_0 \leftarrow ExtractMin(Q)$
        set $node_1 \leftarrow ExtractMin(Q)$
        set $node \leftarrow new(internal)$
        set $node.left \leftarrow node_0$
        set $node.rght \leftarrow node_1$
        set $node.wght \leftarrow node_0.wght + node_1.wght$
        Insert(Q, node)
    // extract final internal node, encapsulating the complete hierarchy of mergings
    set $node \leftarrow ExtractMin(Q)$
    return node, as the root of the constructed Huffman tree
```

Complexity Classes

Notation	Pronunciation	Name in Math	rate
$O(1)$	Oh one	Constant	Excellent
$O(\log_2 n)$ or $O(\log n)$	Oh log n	Logarithmic	Good
$O(n)$	Oh n	Linear	Fair
$O(n \log n)$	Oh n log n		Bad
$O(n^2)$ & $O(n^3)$	Oh n square & Oh n cube	Quadratic & Cubic	Worst
$O(2^n)$	Oh two power n	Exponential	Worst
$O(n!)$	Oh n factorial		Horrible