Dijkstra's algorithm (Single source shortest path)

Edsger W. Dijkstra EWD manuscripts

Graph with non-negative edge weights

Want shortest paths from 1 to every other j

Initialize Marked[i] = 0, Distance[i] = ∞ for all i

Distance[1] = 0

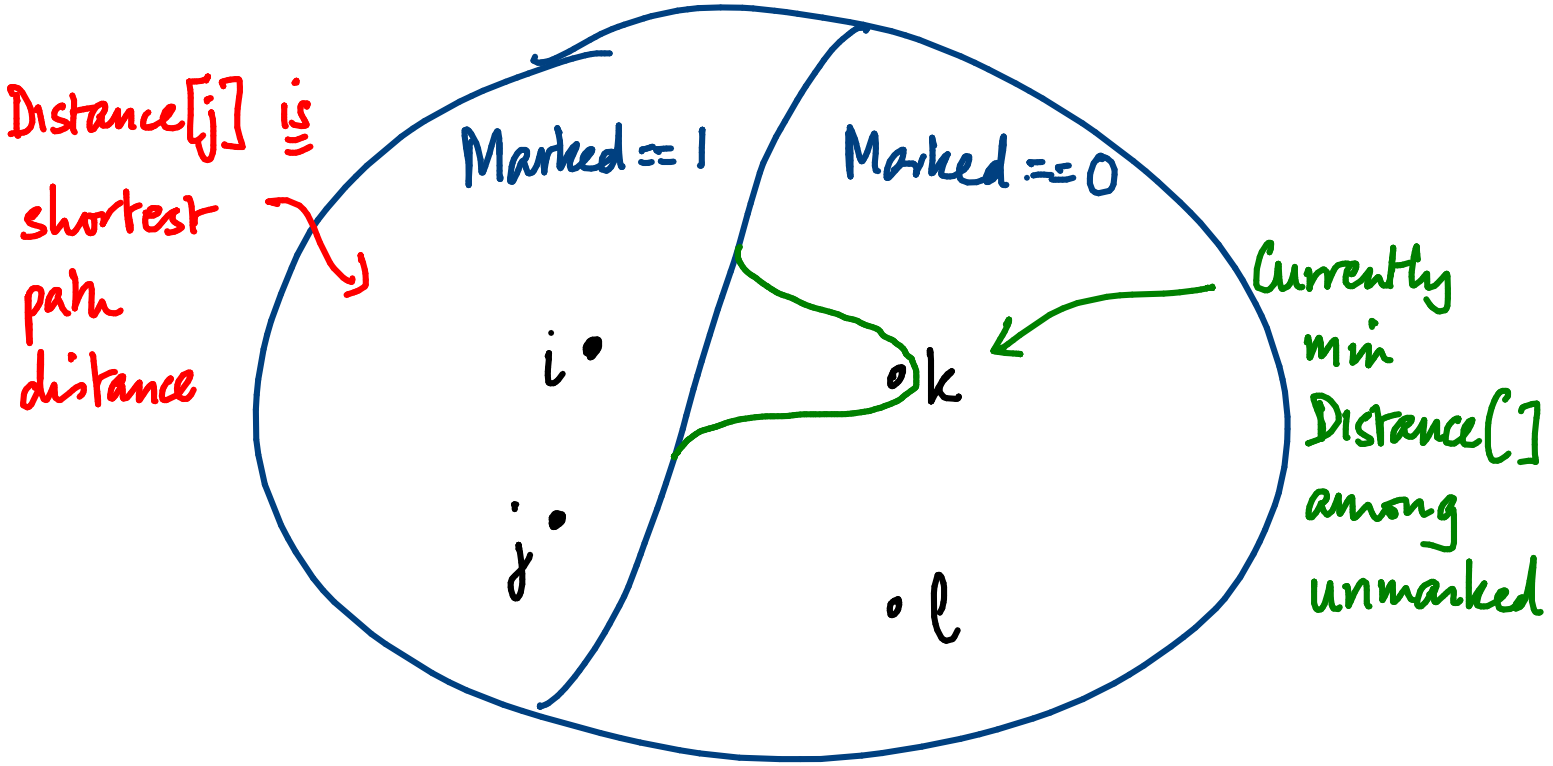
while there is j with Marked[j] == 0

choose such a j with min Distance[j]

for each edge (j,k), update Distance[k]

as min(Distance[k], Distance[j] + w(j,k))

Correctness



Suppose later we find a path $j \to l \to k$

Currently Distance[l] $\geq$ Distance[k]

$\therefore$ Distance via l to k $\geq$ Distance[l] + w(l,k)

Dijkstra = BFS with a priority queue

Priority queue :

Each element has a priority when it joins

Operations:

add((x, p)) x with priority p

remove-max-priority

update-priority — here priority is Distance[j]

How do we maintain a priority queue

(Ignore priority update for now, operations are insert(), delete_max())

Keep values as a sequence

	insert()	delete_max()
Sorted by Priority	O(n)	O(1)
Unsorted	O(1)	O(n)

Balanced binary search tree

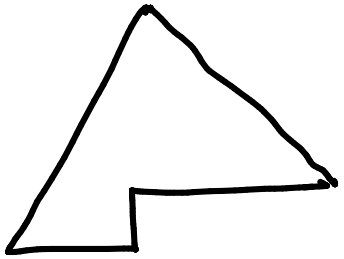
Duplicate values? — store (v, c)

Overkill?

Heap

Binary tree filled level by level, left to right

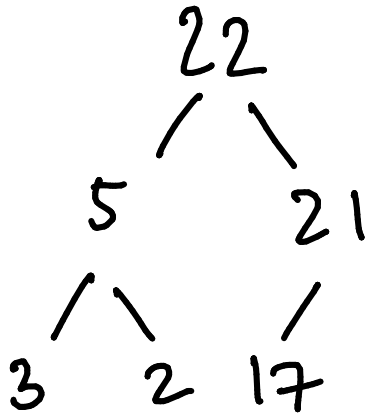
Fixes shape



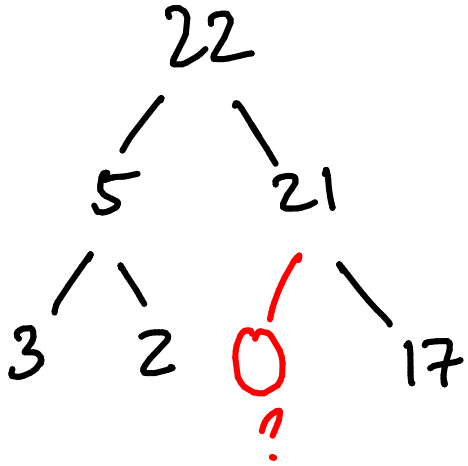
Constraint on values: every node is bigger than its children "Heap order" or "Heap property"

∴ max value at root

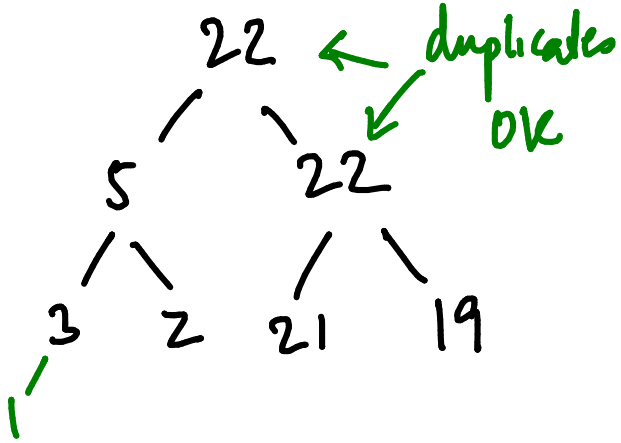
MAX-HEAP



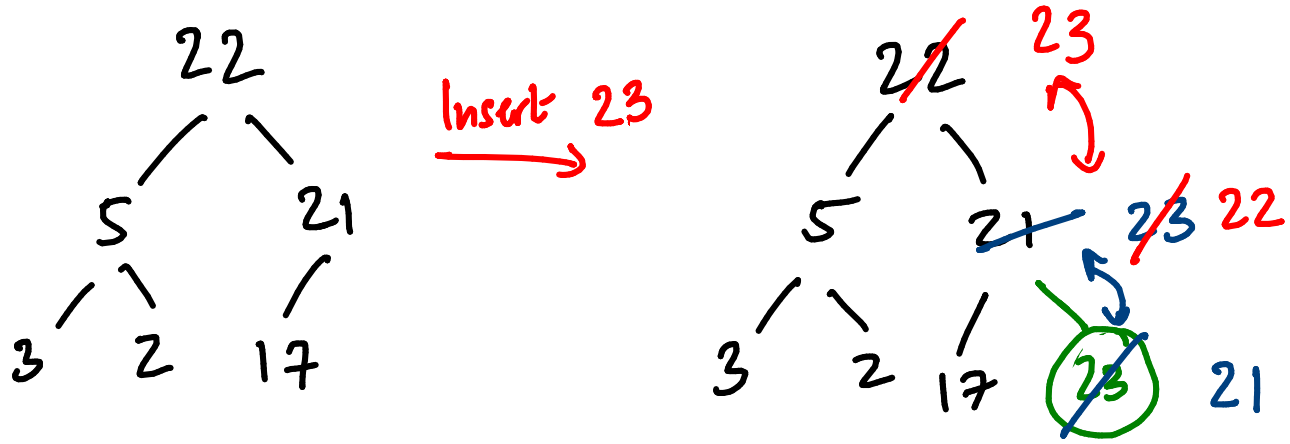
✓



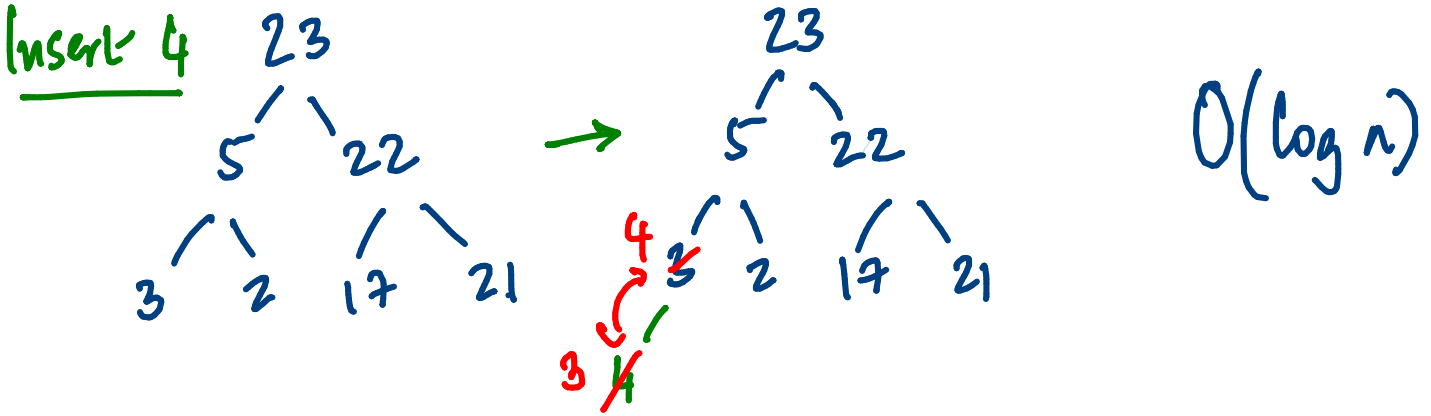
X Wrong shape



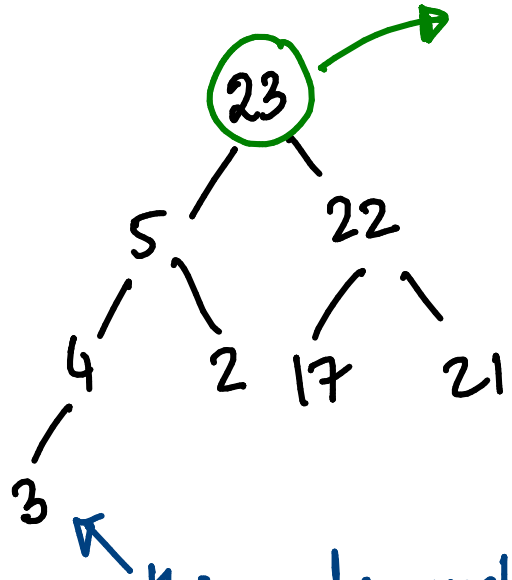
Operations: Insert (x)



Swap new value up, upto log n swaps



O(log n)

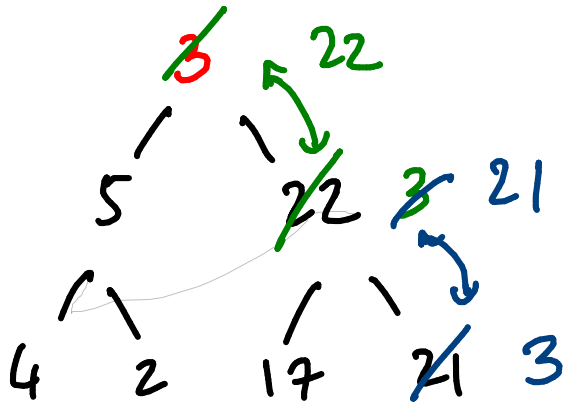


delete_max()

return value at root

this node must be removed

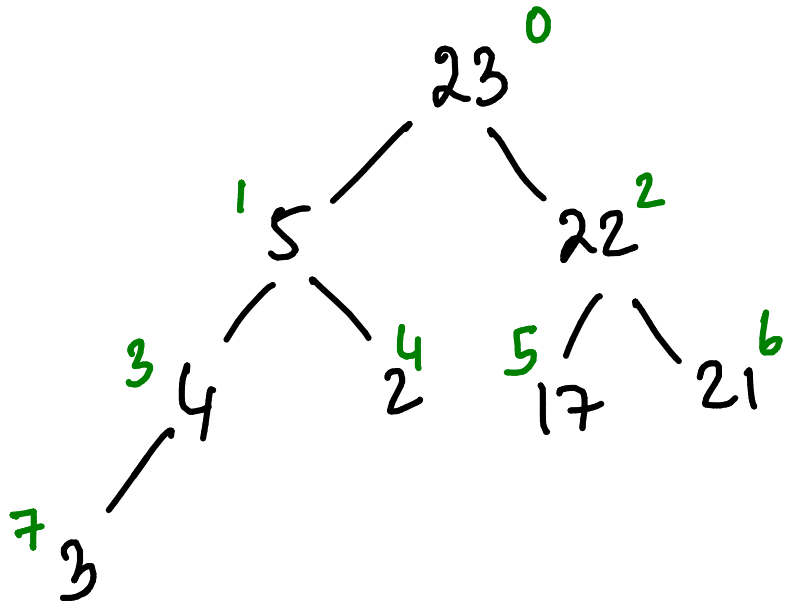
Move "homeless" 3 to "hole" at root!



Follows a single path down from root

$O(\log n)$

Manipulating a heap



Think of heap as a list

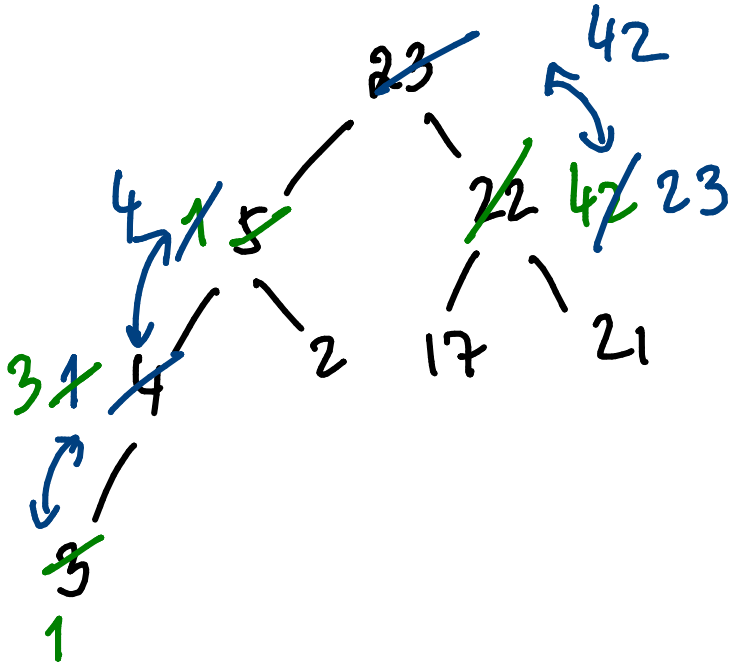
0	1	2	3	4	5	6	7
23	5	22	4	2	17	21	3

left child [i] = $2i+1$

right child [i] = $2i+2$

parent [j] = $\left\lfloor \frac{j-1}{2} \right\rfloor$

Update a priority



Update 5 to 1

Value reduced

No violation wrt parent

Propagate change down

like delete max

Update 22 to 42

Value increased

No violation wrt children

Propagate up like insert(x)

Dijkstra's algorithm

Priority(j) : Distance[j]

MIN HEAP Shape as before

Each value is smaller than children

Insert(), delete_min() can be adapted easily

So also update

Need a mapping

Vertices 1..n Heap Positions 0..n-1

VtoH[i] = position of vertex i in heap

HtoV[j] = vertex at position j in heap

Analysis

n times
↳ While there exists j s.t Marked[j] == 0
→ find such a j with min Distance[j]
O(log n)
update Distance[k] for edges (j,k)
O(log n)
O(m) overall with adj list

Overall $O(n \log n) + O(m \log n) \rightarrow O((m+n) \log n)$