What was my problem while understanding
0-1 BFS queue.

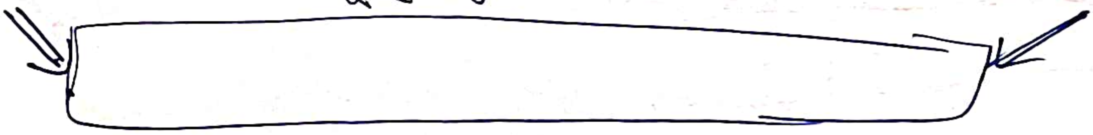
I had the doubt.

Why for
edge weight 0
→ we push to the front of the queue

→ And for the weight 1 we push it to the end of the queue

My doubt was,

~~[illegible] edge weight~~



queue. Q) How are distances maintained in order, like in dijkstra priority queue internally takes care of doing it.

But how in this deque is that happening?

The reason why it works is that.

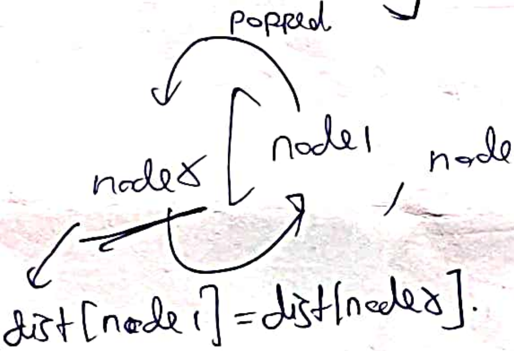
[0-1] <u>bfs</u>..

[node1, node2, - - - - - -, nodep].

For 0 edge weight am satisfied, that.

$[node1 \overset{0}{\longleftrightarrow} nodex]$ {dist[nodex] = dist[node1].}.

So ~~th~~



For edge weight =1, am having second thoughts.

node1 $\xrightarrow{1}$ nodex.

Somenode here

[node1, node2, —[ ] nodep, nodex].

My concern, what if nodex will be ~~inside nodep~~ havig distance less that some node which is already in a higher depth.

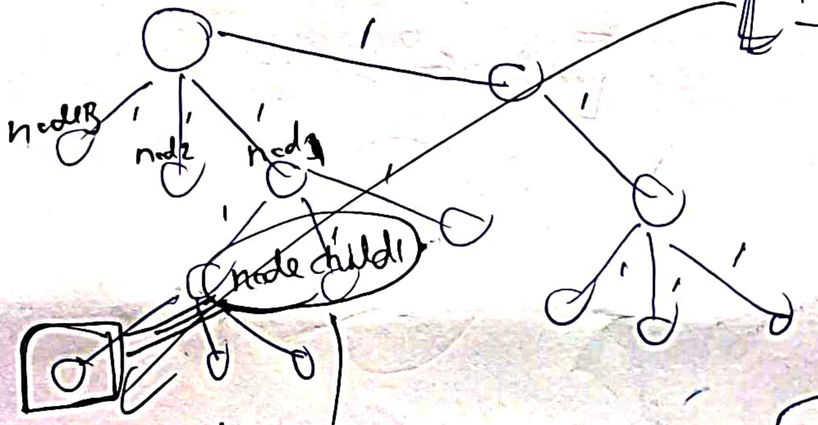
But never such a scenario will come that

[node1, node2, ------ , nodex]

$\text{distance}[\text{node}x] - \text{distance}[1] \leq 1$ → this will always hold true

nodechild1of1 is child of node1

Lets say each of the edge weights is 1,

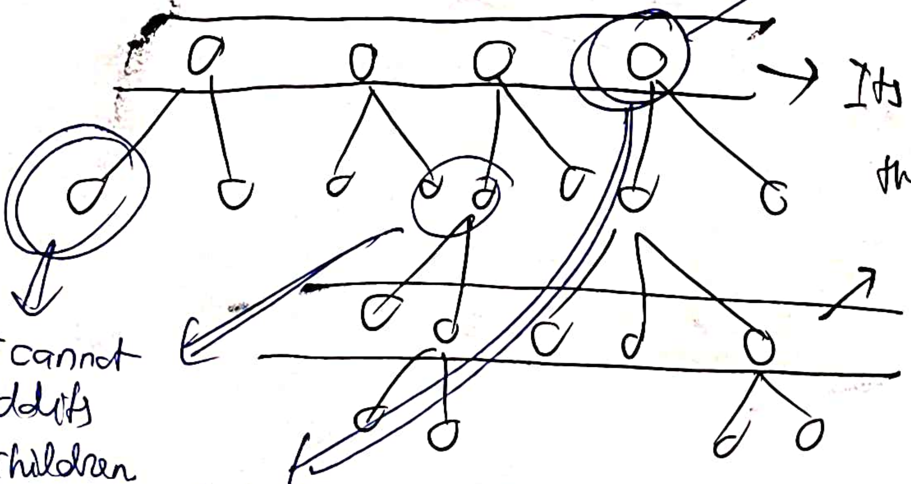


[node1, node2, node3, nodechild1of1, ---, nodex] → To have a

Property of BFS

To have this node we need to have this node in the front of the queue.

Since BFS is level order traversal



→ unless and untill this is popped from the queue

→ Its Impossible to have these two layers together in the queue

this cannot add its children

So these need to be popped in order to make way for other