COL 106 Lecture 34

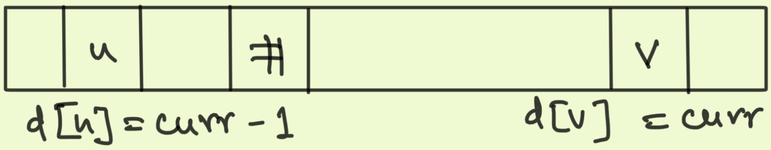
Topic: Breadth-First Traversal: Correctness

Announcement:

Pre-Major Quiz: Nov 8 11:00 – 12:00

Syllabus: Everything taught until today (inclusive).

BFT Loop Invariant:



If v lies before # in q, then d[v] = curr − 1

If v lies after # in q, then d[v] = curr.

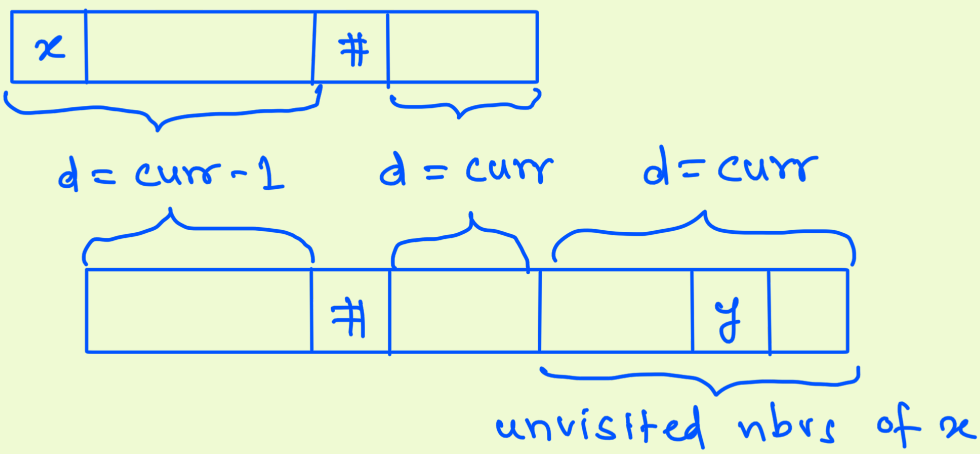
Proof: Invariant holds initially. Assume it holds at the beginning of an iteration.

Initially: $\boxed{s}\boxed{\#}$ curr = 1

Case 1: A vertex, say x, is dequeued.

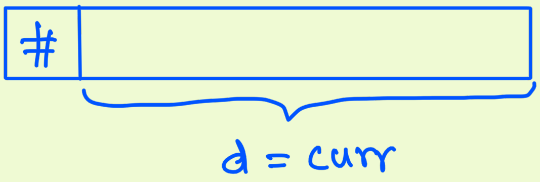
Before:

After:



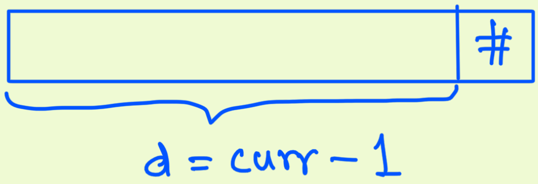
Case 2 : # is dequeued.

Before:



curr $\leftarrow$ curr + 1

After:



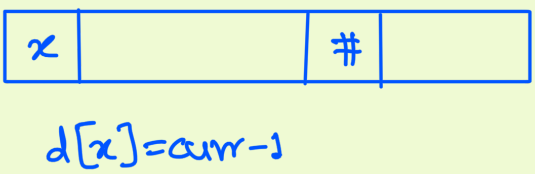
Claim: BFT(G, s) sets $d[u] = d$ iff the length of a shortest $s - u$ path is d.

Proof: By induction on d. Base case - $d=0$: easy to check. Assume claim is true for $\forall d' < d$.

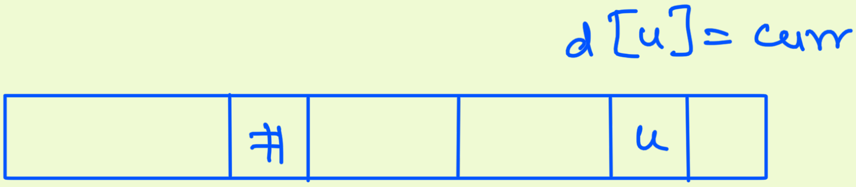
Part 1: $d[u] = d \Rightarrow$ distance between s and u is d.

Let x = prev[u]. $d[x]$ = curr-1

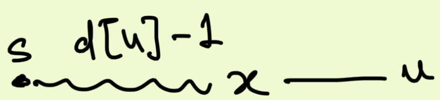
Before:



After:



I.H $\Rightarrow \exists$ $s-x$ path of length $d[x] = d[u] - 1 = d-1$



$\Rightarrow \exists$ $s-x$ walk of length d

$\Rightarrow \exists$ $s-x$ path of length $\leq d$

$\Rightarrow$ $s-x$ distance $\leq d$.

If $s-x$ distance $= d' < d$, then

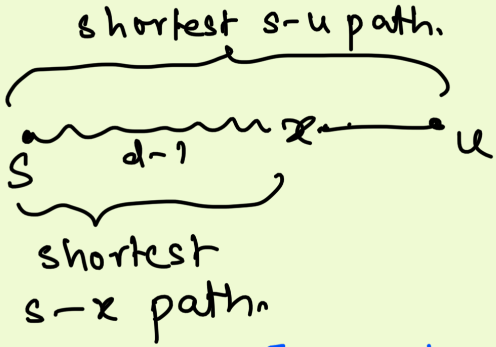
IH $\Rightarrow$ $d[x] = d'$ $\rightarrow$ contradiction to assumption $d[x] = d$.

$\therefore$ $s-x$ distance $= d$.

Part 2: If $s-u$ distance is d, then $d[u] = d$.

Consider an $s-u$ path of length d. Let x be the second-last vertex on this path.

By I.H. $d[x] = d-1$



Consider iteration in which x was dequeued.

If u was enqueued in this iteration, then $d[u] = d[x] + 1 = d$, as required

If u was not enqueued in this iteration: $d[u] \leq d$.

If $d[u]$ were $< d$, then

by I.H: $s-u$ distance $= d[u] < d$.

$\rightarrow$ contradiction to assumption: $s-u$ distance is d.

$\therefore$ $d[u] = d$, as required.

Listing connected components.

ccid[v] ← 0 for each vertex v

cccount ← 0; q ← Empty queue

For each vertex v: $\}$ ——— $O(n)$

    If ccid [v] = 0:

        cccount ← cccount + 1 ccid [v] ← cccount $\}$ $= \#$ conn. comps $\leq n$

        q. Enqueue (v)

        while q not empty:

For a given v: $= |\text{conn comp of } v| - 1$ — Overall: $O(n)$

            x ← q. Deque ()

            For each nbr y of x: $\sum_x d_x = 2m$

                If ccid [y] = 0: $\Big\}$ $O(1) \times d_x$

                    ccid [y] ← cccount

                    q. Enqueue (y)

Return (ccid, cccount).

Overall running time: $O(n+m)$.