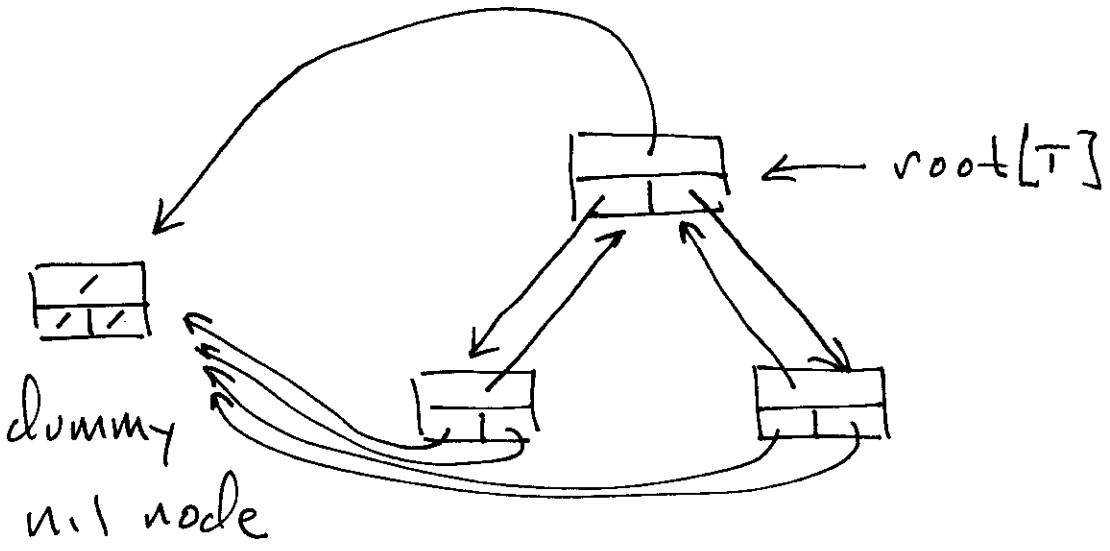
CSE 101 12-8-20

- quiz 5 today
- SETs is open. closes sunday Dec. 13 at 11:59 PM.
- Final exam: Wed. Dec 16.

Par stuff.



Continue SSSP in Weighted graph

Both Bellman-Ford & Dijkstra use vertex attributes

$\pi[x]$: Parent of x, encodes a shortest Paths tree.

$d[x]$: estimate of $\delta(s,x)$

Predecessor subgraph : $G_\pi = (V_\pi, E_\pi)$

$$V_\pi = \{ x \in V \mid \pi[x] \neq nil \text{ or } x = s \}$$

$$E_\pi = \{ (\pi[x], x) \mid \pi[x] \neq nil \}$$

To find shortest Paths use

PrintPath(G, s, x)

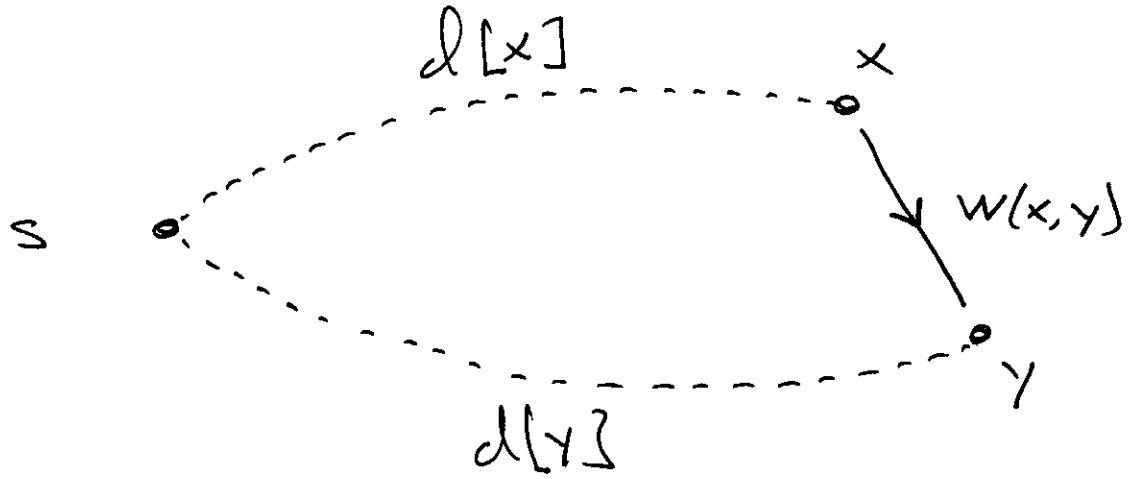
Sec. 22.2 P. 601 (3rd ed.).

Both use two subroutines:

Initialize(G, s)

and

Relax(x, y) Pre: $y \in adj[x]$



Note

- Relax(x,y) changes at most fields of y
- after Relax(x,y) then
  $d[y] \leq d[x] + w(x,y)$
  must be true.

Lemma 1

Let $x \in V(G)$. Suppose after Initialize(G, s), some sequence of calls to Relax$(\cdot,\cdot)$ results in $d[x]$ being finite. Then G contains an s-x path of weight $d[x]$.

Lemma 2

After Initialize(G, s), the inequality

$$\delta(s, x) \leq d[x] \quad (\text{for all } x \in V(G))$$

is maintained over any sequence of calls to Relax$(\cdot,\cdot)$

Lemma 3 (Path relaxation Property)

If $P : s = x_0, x_1, x_2, \ldots, x_k$ is min weight s-x_k path, and the edges of P are relaxed in order:

$$(x_0, x_1), (x_1, x_2), (x_2, x_3), \ldots, (x_{k-1}, x_k)$$

Then $d[x_k] = \delta(s, x_k)$. This is true regardless of any other relaxations that occur, even if interspersed with the above relaxations.

Pseudo-code: Bellman Ford

Runtime: $n = |V|$, $m = |E|$

- Initialize cost $\Theta(n)$
- Relax(·,·) costs $\Theta(1)$
  each edge is relaxed $n-1$ times,
  so loop 2-4 costs $\Theta(m(n-1))$
- loop 5-7 costs $\Theta(m)$.

Total cost $\Theta(n) + \Theta(1) + \Theta(m(n-1)) + \Theta(m)$

$= \Theta(mn)$.

Section 6.5 Priority Queues

A Priority Queue is an ADT that maintains a finite set S of elements (records) each with an attribute called key

$$x = (\underbrace{\cdot, \cdot, \cdot, \cdot, \cdot}_{\text{satellite data}}, key) \in S$$

$key[x]$ defines priority of x in S.

states : $\{ \ldots\ldots, S, \ldots\ldots \}$

Two kinds

- max priority queue ← concentrate
- min " "

Operations for a max P.Q.

- Insert(S, x):
  insert new x into S

- Max(S):
  returns element with largest key.

- IncreaseKey(S, x, k):
  changes key[x] to k if k > key[x],
  otherwise does nothing.

A min P.Q. is analogous.

Implementation: new data structure called Heap.