Lec 16 Graph Search Cont

# Parallel BFS cost analysis cont.

BFS cost

Let $\|F\| = \sum_{x \in F} (d^+(x) + 1)$ assume tree set

for iter $i$

| | W | S |
|---|---|---|
| $- X_i \cup F_i$ | $O(\|F_i\| \lg n)$ | $O(\lg n)$ |
| | $\|F\| \le \|X\| \Rightarrow O(\|F\| \lg(\frac{\|X\|}{\|F\|}+1)) \le \|F\| \lg n$ | |
| | $\|F\| > \|X\| \Rightarrow O(\|X\| \lg(\frac{\|F\|}{\|X\|}+1)) \le \|F\| \lg \|F\| \le \|F\| \lg n$ | |
| $- N_G^+(F_i)$ | $O(\|F_i\| \lg n)$ | $O(\lg^2 n)$ |
| | same analysis as above | |
| $- \setminus X_{i+1}$ | $O(\|F_i\| \lg n)$ | $O(\lg n)$ |
| | similar to union | |

Over all loops. Say max depth in search is $d$

$$W = O\left(\sum_{i=0}^{d} (\|F_i\| \lg n)\right)$$
$$= O\left(\lg n \sum_{i=0}^{d} \|F_i\|\right)$$
$$= O(\lg n \cdot (m+n))$$

using $\|F\| = \sum_{x \in F} d^+(x) + 1$
$= \sum_{x \in F} d^+(x) + \sum_{x \in F} 1$
$\vdots$
$= m + n$

$S = O(d \lg^2 n)$

* one $\lg n$ can be eliminated using some set data struct

# DFS

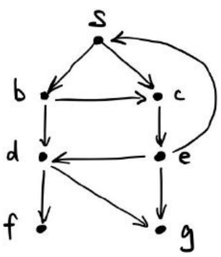
When $v$ = most recently seen vert in frontier

## Recursive impl

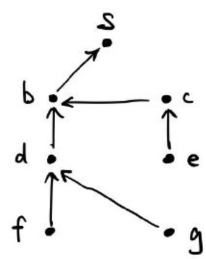
DFS G s =
  let
    DFS' (X, v) = if $\underbrace{v \in X}_{1}$ then X
                  else iterate DFS' $\underbrace{(X \cup \underbrace{\{v\}}_{2})\ N^{+}(v)}_{3}$
  in
    DFS' ({}, s)
  end

- inherently sequential!
- in fact DFS believed to be P-complete
- and believed that P-complete probs don't have polylog span sol

## Example

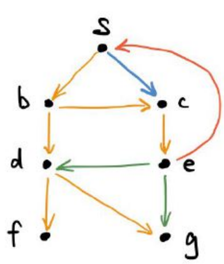



Possible order: s, b, d, f, g, c, e
↪ induced search tree:




## DFS edge types

- Tree edge — $u \to v$ if v visited from u in DFS viz. reversed edges in search tree
- Back edge — edge that go back to ancestor in DFS tree that's not tree edge
- Forward edge — edge that go to descendent in DFS tree that's not tree edge
- Cross edge — none of above, they cross btwn branches




Note these four partition the edges in G

## Generic DFS

Notice we may want to do some analysis while computing 1,2,3
We can let caller provide function for those computation

- application state $\Sigma$
- transition funcs $\Sigma \times V \to \Sigma$
  - visit — called when first visiting a vert
  - finished — called when done iterating over $N^+(v)$
  - revisit — if already visited

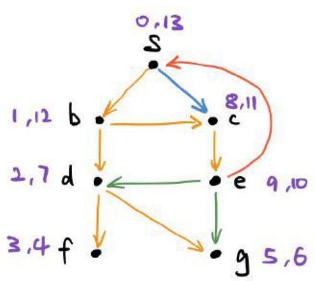
$$
\begin{aligned}
&\text{DFS } G\ ((\Sigma, X), v) = \\
&\quad \text{if } v \in X \text{ then } \underbrace{(\text{revisit}(\Sigma, v), X)}_{1} \\
&\quad \text{else let} \\
&\qquad \Sigma' = \underbrace{\text{visit}(\Sigma, v)}_{2} \\
&\qquad X' = X \cup \{v\} \\
&\qquad (\Sigma'', X'') = \text{iterate } (\text{DFS } G)\ (G', X')\ N^+(v) \\
&\quad \text{in} \\
&\qquad \underbrace{(\text{finish}(\Sigma'', v), X'')}_{3} \\
&\quad \text{end}
\end{aligned}
$$

Sometime we want

$$\text{DFSALL}(G = (V, E))\ \Sigma = \text{iterate } (\text{DFS } G)\ (\Sigma, \{\})\ V$$

Ex. application

→ DFS numbering : track visit / finish timestemp




Using our framework:

$$\Sigma : \underset{\uparrow \text{ curr time}}{\text{int}} \times \underset{\uparrow \text{ visit time}}{(V, \text{int})\ \text{table}} \times \underset{\uparrow \text{ finish time}}{(V, \text{int})\ \text{table}}$$

$$
\begin{aligned}
&\text{visit } ((t, V, F), v) \\
&\quad = (t+1,\ \text{insert } V\ (v, t),\ F)
\end{aligned}
$$

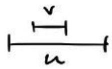
$$
\begin{aligned}
&\text{finish } ((t, V, F), v) \\
&\quad = (t+1,\ V,\ \text{insert } F\ (v, t))
\end{aligned}
$$

$$
\begin{aligned}
&\text{revisit } ((t, V, F), v) \\
&\quad = (t, V, F)
\end{aligned}
$$

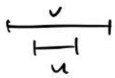
→ Determine edge types (reduced to start / finish time)

Keep set of tree edge in a set in $\Sigma$

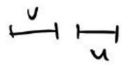
<u>Claims</u>

$e = (u, v)$ forward edge $\Leftrightarrow$  $\wedge$ e not tree edge



$e = (u, v)$ back edge $\Leftrightarrow$  $\wedge$ e not tree edge



$e = (u, v)$ cross edge $\Leftrightarrow$  viz. finished searching target of e before going to the source



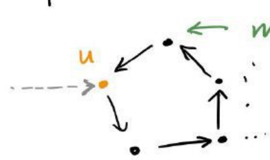
→ Cycle detection (reduced to edge type)

<u>Claim</u> G has cycle $\Leftrightarrow$ $\exists$ back edge

($\Leftarrow$) Trivial

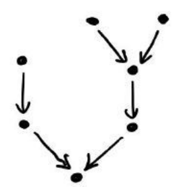
($\Rightarrow$) Fix $\underbrace{\text{first time encountering vert}}_{u}$ in some cycle, then at later point we visit an incoming edge to that vert

← must visit this at later point




→ Topological sort

Given DAG $G = (V, E)$

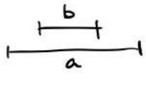


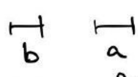
Observe it define partial order

$$a \leq_R b \Leftrightarrow b \text{ reachable from } a \wedge a \neq b$$

Want to sort V s.t. it respects $\leq_R$

<u>Lemma</u> DAG finish time:
if $a \leq_R b$, $\forall$ DFS, b finish finish before a
<u>C1</u> b visited before a <u>C2</u> a visited before b




<u>Alg</u> run DFSAll, return reverse finish time order