Passing CIS 121

Emails

OH: Today 10pm-11pm ET

Topological Sort:

Input: Directed acyclic graph (DAG), G

Output: An ordering of the vertices of G in a line s.t. the edges go from left to right.

```
      b
     / \      
    /   \     
   a    c     d
     \   /    /
      o c o   o
     /   \    /
    a    b    d
  /     /   /
 e     e   e
```
Idea: vertices with no incoming edges go first.

Prove that a DAG always has a source & a sink: vertices with no incoming & outgoing edges.

P: maximal path in G.

Algorithm

\( TS(G) \)

\[ u \leftarrow \text{source vertex in } G. \quad O(n+m) \]

\[ G' \leftarrow G - u \quad O(1) \]

\[ L' \leftarrow TS(G') \quad T(n-1) \]

Output \( u \) followed by vertices in \( L' \). \( O(1) \)

Recurrence: \( T(n) = T(n-1) + n \)
An efficient implementation

1. Find all sources in $G$. $O(n+m)$
   Call this set $S$.

2. $u \leftarrow$ a vertex in $S$. $O(1)$

3. Output $u$ & remove it from $S$. $O(1)$

4. for all neighbors $v$ of $u$ in $V \setminus S$

   \[
   \text{ind}_y(v) \leftarrow \text{ind}_y(u) - 1
   \]

   \[
   \text{if } \text{ind}_y(v) = 0 \text{ then add } v \text{ to } S
   \]

5. Go to (2).

$S$
Time taken by steps during the alg.

1. \( O(n+m) \)

2. \( O(n) \)

3. \( O(n) \)

4. \( O\left(\sum_{u} d(y(u))\right) = O(m) \)

Total running time: \( O(n+m) \).

DFS

Krish's intuition: In DFS a vertex does not finish until all its neighbors finish.
Algorithm

1. Do DFS $G$ \hspace{2cm} \text{O}(n+m)

2. Sort vertices $m$ in order of $f(i)$.
   \hspace{2cm} \text{O}(n \log n)

\text{O}(m + n \log n).

We can do steps 1) & 2) together—
each time a vertex finishes, we put it at the beginning of the list.

\text{O}(n+m)

Correctness
What about back edges?

No back edges in a DAG.

$G$ is a DAG iff DFS$(G)$ yields no back edges.

Then: Our algo works.

Proof: Let $e = (u,v)$ be an edge in $G$.

We want to prove that in our output, $u$ is to the left of $v$. That is, we need to show that $fu > fv$.

\[ u \xrightarrow{e} v \]

Case I: $d[u] < d[v]$. (from $u$ to $v$ at time $d[v]$)

There is a white path $w$ in $G$. This path must be from $u$ to $v$. (because...
then in an edge \( e = (u, v) \). This implies \( v \) is a descendant of \( u \) (\( \text{DPS} \)).

By the parentheses then, \( f_u > f_v \).

\[ \text{Can II:} \quad d(v) < d(u) \]

\[ \text{(At } d[u] \text{) no path from } v \text{ to } u \text{ (otherwise } \text{A would have a cycle).} \]

At \( d[u] \) no WP from \( v \) to \( u \) in \( G \),

\( \Rightarrow \) \( u \) cannot be a descendant of \( v \) in the \( \text{DPS} \) first.

By the parentheses then we have
du \rightarrow fv \rightarrow du \rightarrow fu.

\therefore \text{ } fu > fu.

\underline{Shortest Paths}.

\underline{I/p: Directed graph } G = (V, E).

\underline{Output:} \text{ } \underline{Shortest path} tree rooted at \$ \text{. Shortest Paths from } \$ \text{ to every other vertex in } G.
pick the edge with the smallest cost to buy in the next unit.

Alg. (Dijkstra's algorithm).

for each \( u \in V \) do

\[ d[u] \leftarrow \infty \]

\[ d[s] \leftarrow 0 \]

\[ S \leftarrow \emptyset \]
// assume all vertices reachable from $S$
while $S \neq V$ do

$u \leftarrow$ vertex on $V \setminus S$ with the smallest $d[u]$

$S \leftarrow S \cup \{u\}$

for each $v \in N(u)$ s.t. $v \in V \setminus S$

\[ d[u] + w_{uv} \]

if $w_{uv} < d[v]$ then

\[ d[v] \leftarrow w_{uv} \]

$p[v] \leftarrow u$

paint $p[v]$. 

\[ \text{Fig. 4} \]