MATH 566: Lecture 19 (10/22/2024)

Today: * Shortest augmenting path algorithm

Recall Distance labels $d(\cdot)$ are valid if $d(i) \le d(j)+1 \;\forall (i,j) \in G(\bar{x})$, $d(t)=0$.

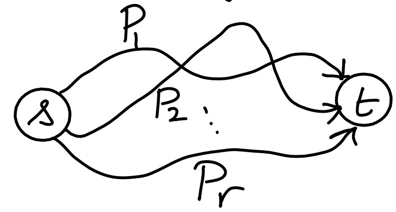
We now look for candidate arcs in $G(\bar{x})$ that could be part of augmenting path(s). Recall the notion of admissible arcs in search (BFS/DFS) — we redefine admissibility here.

Def An arc $(i,j) \in G(\bar{x})$ is **admissible** if $d(i) = d(j)+1$.
A path from s to t consisting of only admissible arcs is an **admissible path**.

We formalize the notion that admissible paths are precisely augmenting paths.

Property 7.3 An admissible path P is a shortest augmenting path (from s to t).

Proof Add admissibility conditions over all arcs in P. We get $d(s)=k$, where P has k arcs. Since $d(s)$ provides a lower bound on the length of arcs (in terms of # arcs) by Property 7.1, equality here implies optimality. $\square$

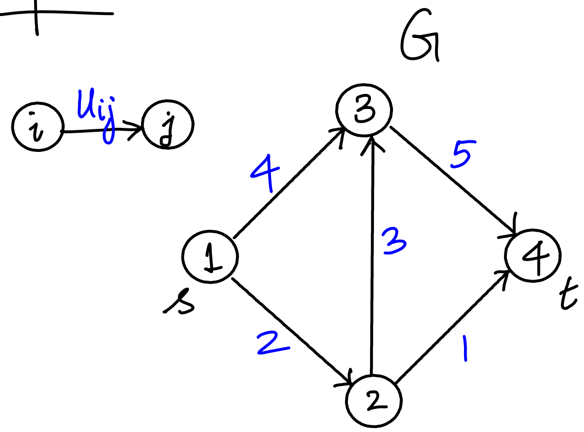


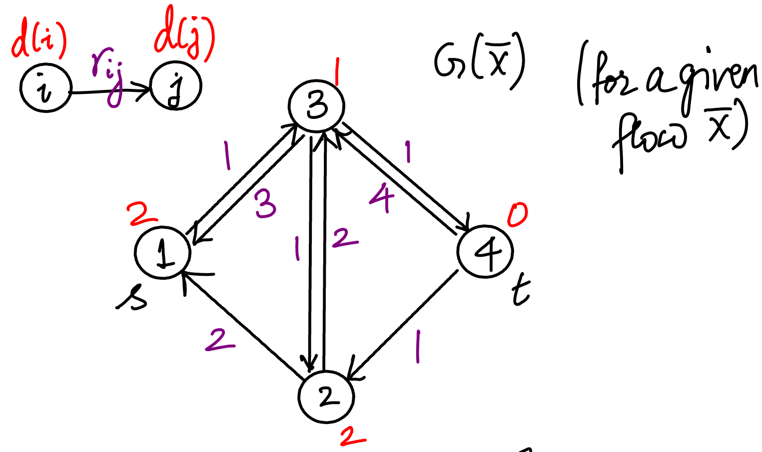
$$\left.\begin{array}{l} d(s) \le k_1 \\ d(s) \le k_2 \\ \vdots \end{array}\right\} \Rightarrow d(s) \le k = \min_{i=1\ldots r}\{k_i\}$$

We need one more definition before looking at some examples.

Def A set of distance labels $d(i)$ is **exact** if $d(i)$ is the length of an SP from i to t (in terms of # arcs) $\forall\, i \in N$.

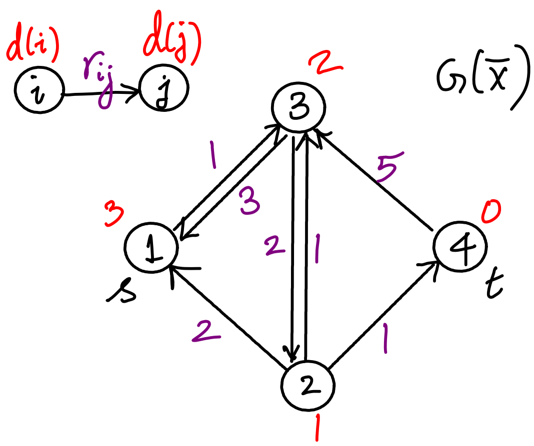
Examples



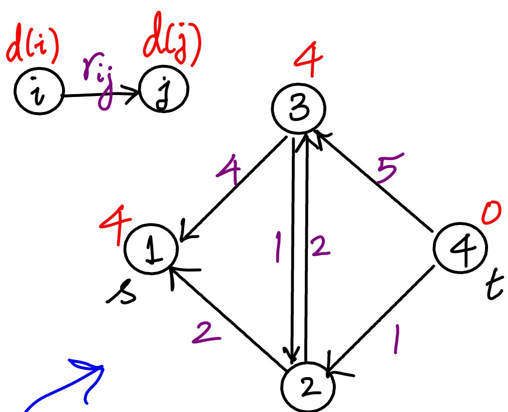


Note that $\bar{d} = \begin{array}{c} 1\\2\\3\\4 \end{array}\begin{bmatrix} 0\\0\\0\\0 \end{bmatrix}$ is valid, but is not exact, while $\bar{d} = \begin{array}{c} 1\\2\\3\\4 \end{array}\begin{bmatrix} 2\\2\\1\\0 \end{bmatrix}$ is exact.

Let's consider a couple more flows (on the same network):



$\bar{d} = \begin{array}{c} 1\\2\\3\\4 \end{array}\begin{bmatrix} 3\\1\\2\\0 \end{bmatrix}$ is exact.



This is $G(\bar{x})$ for the max flow.

Here, there is no directed path from any $i \neq t$ to t in $G(\bar{x})$. Setting $d(i) = n$ is sufficient (rather than setting $d(i) = \infty$), as we know the largest meaningful value for $d(i)$ is $n-1$.

If $d(i)$ goes above n at any point of the SAP algorithm, we can ignore node i from further consideration.

Recall! Intuition for $d(i)$: Think of $d(i)$ as the height above ground level that node i has to be raised for flow to happen "freely". Node t is at ground level ($d(t)=0$) and s need not be raised above level $n-1$ from the ground.

Shortest Augmenting Path Algorithm

algorithm *shortest augmenting path*;
begin
    $x := 0$; $\Rightarrow G(\bar{x}) = G$ at start
    obtain the exact distance labels $d(i)$; $\rightarrow$ reverse BFS to t
    $i := s$;
    while $d(s) < n$ **do**
    begin
        if i has an admissible arc **then**
            begin
                advance(i);
                if $i = t$ **then** *augment* and set $i = s$
            end
        else *retreat(i)*
    end;
end;

Figure 7.5 Shortest augmenting path algorithm.

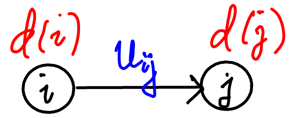
procedure *advance(i)*;
begin
    let (i, j) be an admissible arc in $A(i)$; $\rightarrow$ look at $A(i)$ in $G(\bar{x})$ $\rightarrow d(i) = d(j)+1$ for $(i,j) \in G(\bar{x})$
    pred$(j) := i$ and $i := j$;
end;

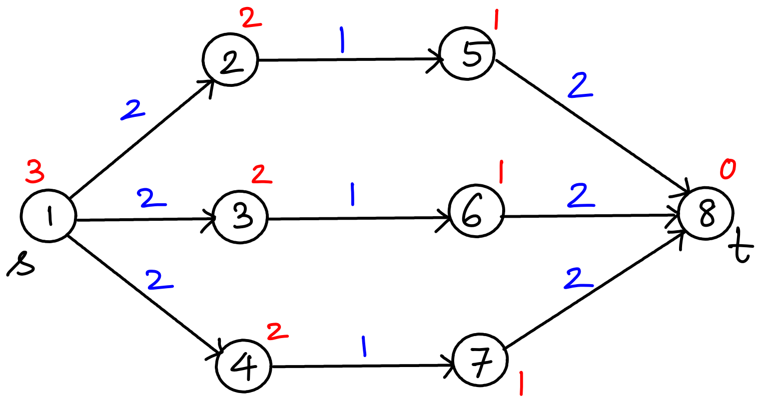
procedure *retreat(i)*; $\rightarrow d(i) < d(j)+1 \;\forall (i,j) \in G(\bar{x})$
(as $d(i) \le d(j)+1$ by validity, and no admissible arc $\Rightarrow d(i) = d(j)+1$ does not hold)
begin
    [$d(i) := \min\{d(j) + 1 : (i, j) \in A(i)$ and $r_{ij} > 0\}$;] $\rightarrow$ in $G(\bar{x})$ $\rightarrow$ relabel operation
    if $i \neq s$ **then** $i :=$ pred(i);
end;

This is the one step where we update $d(i)$. The $d(i)$ value strictly increases by a relabel.

procedure *augment*;
begin
    using the predecessor indices identify an augmenting path P from the source to the sink;
    $\delta := \min\{r_{ij} : (i, j) \in P\}$;
    augment δ units of flow along path P; $\rightarrow$ also update $G(\bar{x})$
end;

Illustration

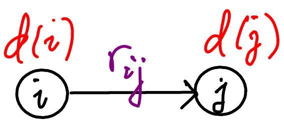


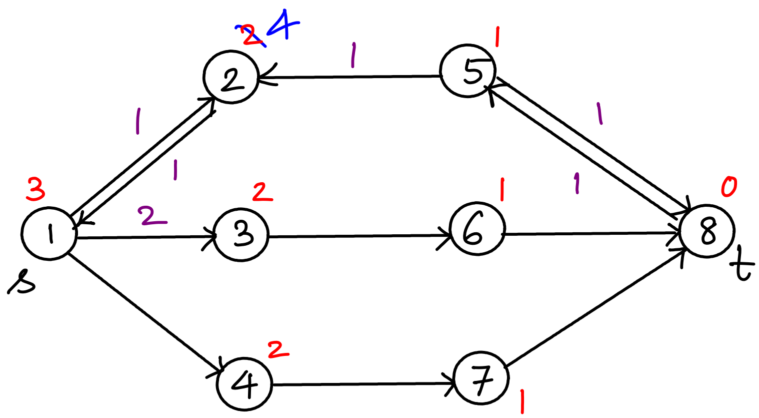


We examine outarcs ($A(i)$ in $G(\bar{x})$) in lexicographic order, by default.

Iteration 1

$P_1 = 1\text{-}2\text{-}5\text{-}8$, augment $\delta(P_1) = 1$ unit. In detail, we advance from $s=1$ to 2 to 5 to $8=t$, identifying the augmenting path P_1.





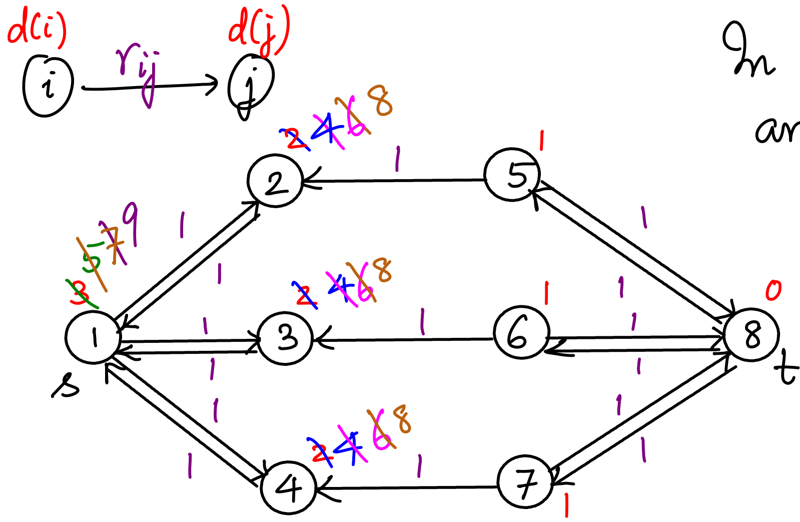
Iteration 2

Advance from $s=1$ to 2. But there are no admissible arcs out of 2. So we relabel $d(2)$ to $d(1)+1 = 4$, and retreat back to $s=1$.

Iteration 3 (1,2) is no longer admissible, but (1,3) is. We advance to t via $P_2 = 1\text{-}3\text{-}6\text{-}8$, augment $\delta(P_2)=1$, as we did for P_1. In Iteration 4, we advance along (1,3), and then relabel 3 before retreating back to $s=1$.

In Iteration 5, we repeat steps similar to Iteration 3 to augment along $P_3 = 1\text{-}4\text{-}7\text{-}8$ $(\delta(P_3)=1)$.

In Iteration 6, we advance on (1,4), relabel 4, and retreat to $s=1$.



In Iteration 7, there are no admissible arcs out of $s=1$. So, relabel node $s=1$:

$$d(1) = \min_{j=2,3,4} \{d(j)+1\} = 5.$$

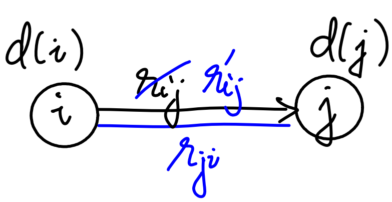
We could have stopped after Iteration 6, but that's just because of the special structure of the network. The algorithm relabels $s=1$ and 2, 3, 4 nodes using a series of retreat operations until $d(s) = 9 > n = 8$, when it terminates.

Proof of Correctness

We show that distance labels remain valid after each augmentation and each relabeling.

Augmentation: Bottleneck arc(s) disappear from $G(\bar{x})$, we decrease some r_{ij}, and we add (j,i) to $G(\bar{x})$ (with $r_{ji} > 0$).

Consider the more general situation where we push some flow forward along arc (i,j), but not saturate it. Thus, r_{ij} is decreased to $r_{ij}' < r_{ij}$, and (j,i) is added to $G(\bar{x})$.



$d(i) = d(j)+1$, as $(i,j) \in G(\bar{x})$ is an admissible arc.

For (i,j), validity is maintained $(r_{ij}' < r_{ij})$.

For (j,i), we need $d(j) \le d(i)+1$. But this condition holds, as $d(j) = d(i)-1$, and hence satisfies the validity condition.