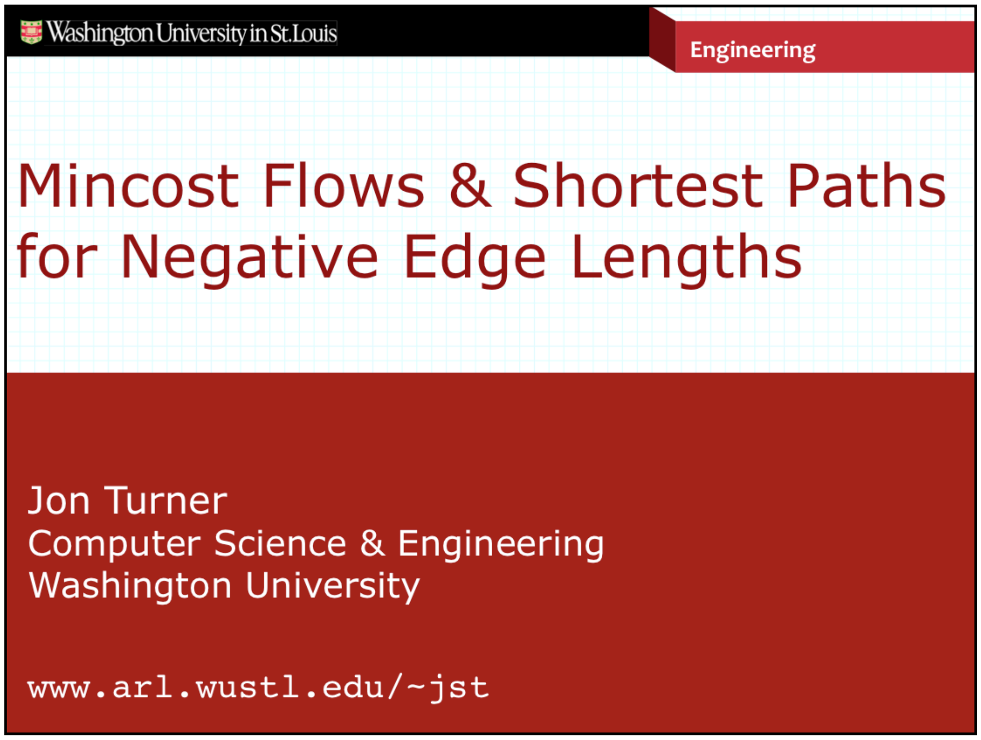
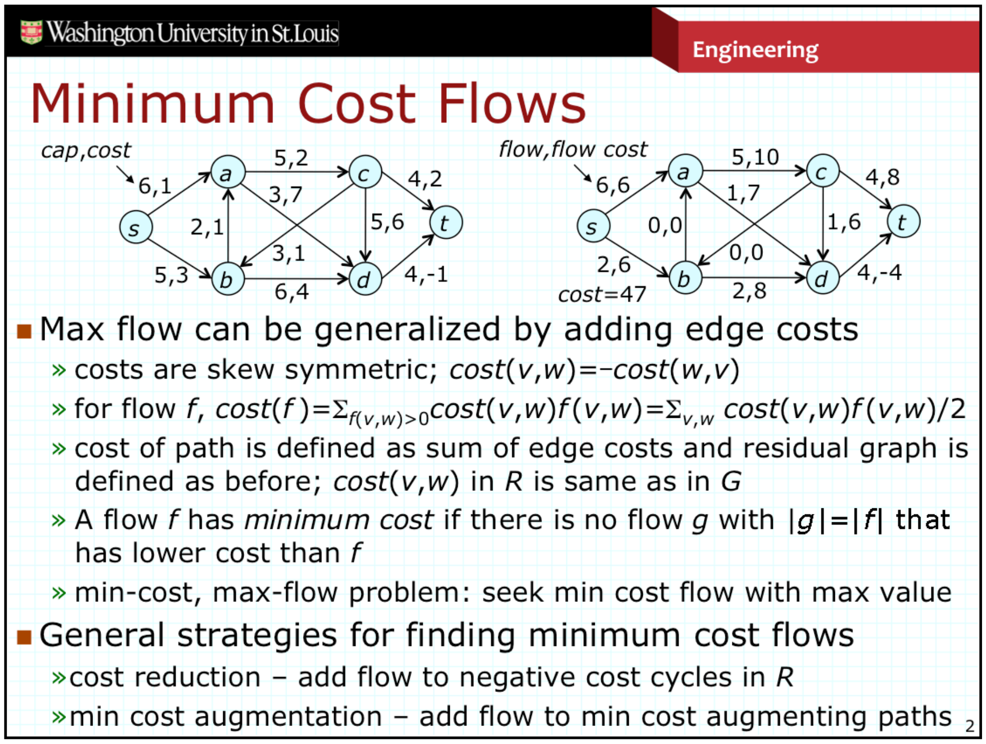


Mincost Flows & Shortest Paths for Negative Edge Lengths

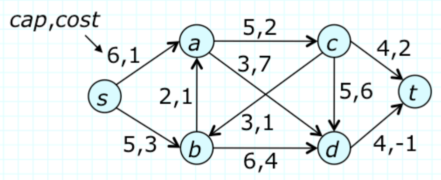
Jon Turner
Computer Science & Engineering
Washington University

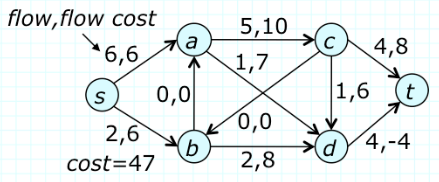
www.arl.wustl.edu/~jst



Washington University in St. Louis — Engineering

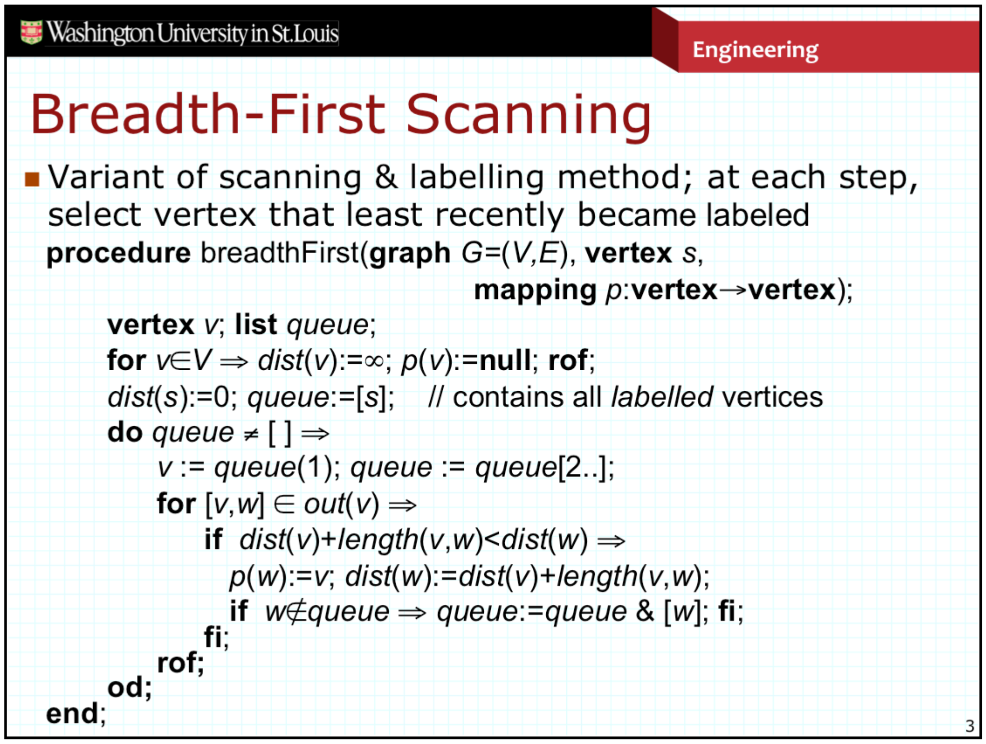
Minimum Cost Flows





- Max flow can be generalized by adding edge costs
  - » costs are skew symmetric; *cost*(v,w)=–*cost*(w,v)
  - » for flow *f*, $cost(f)=\Sigma_{f(v,w)>0}cost(v,w)f(v,w)=\Sigma_{v,w}\, cost(v,w)f(v,w)/2$
  - » cost of path is defined as sum of edge costs and residual graph is defined as before; *cost*(v,w) in *R* is same as in *G*
  - » A flow *f* has *minimum cost* if there is no flow *g* with $|g|=|f|$ that has lower cost than *f*
  - » min-cost, max-flow problem: seek min cost flow with max value
- General strategies for finding minimum cost flows
  - »cost reduction – add flow to negative cost cycles in *R*
  - »min cost augmentation – add flow to min cost augmenting paths

2

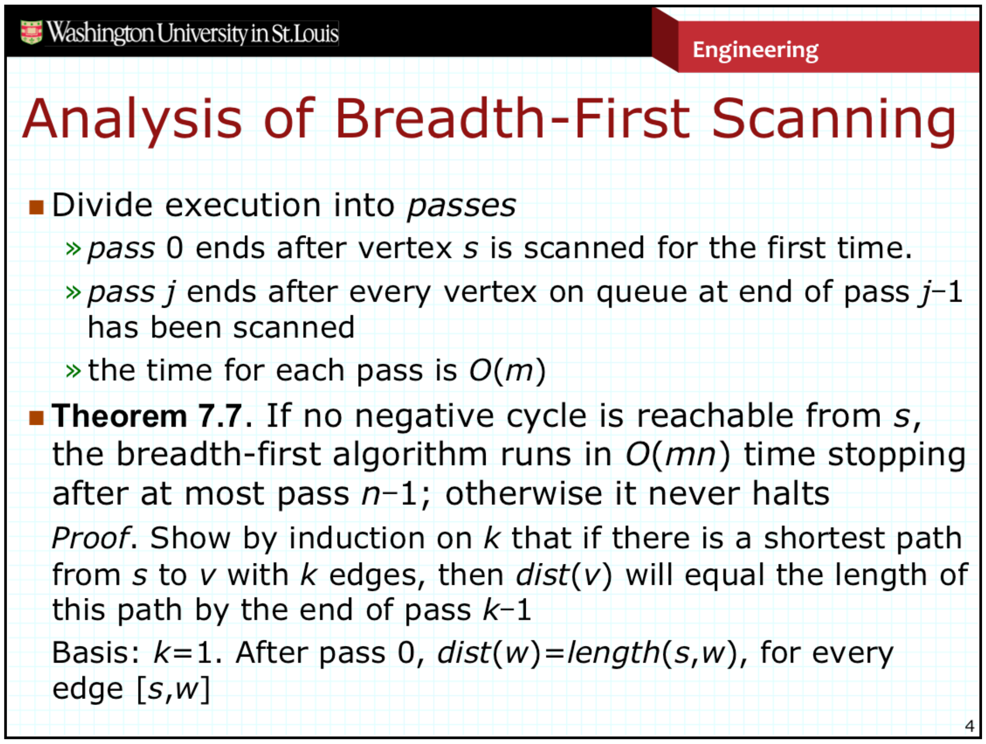

Washington University in St. Louis

Engineering

Breadth-First Scanning

- Variant of scanning & labelling method; at each step, select vertex that least recently became labeled

```
procedure breadthFirst(graph $G=(V,E)$, vertex $s$,
                                        mapping $p$:vertex→vertex);
     vertex $v$; list $queue$;
     for $v \in V \Rightarrow dist(v):=\infty; p(v):=$null; rof;
     $dist(s):=0; queue:=[s]$;    // contains all labelled vertices
     do $queue \neq [\,] \Rightarrow$
          $v := queue(1); queue := queue[2..]$;
          for $[v,w] \in out(v) \Rightarrow$
               if $dist(v)+length(v,w)<dist(w) \Rightarrow$
                  $p(w):=v; dist(w):=dist(v)+length(v,w)$;
                  if $w \notin queue \Rightarrow queue:=queue$ & $[w]$; fi;
               fi;
          rof;
     od;
end;
```



Analysis of Breadth-First Scanning

- Divide execution into *passes*
  - » *pass* 0 ends after vertex *s* is scanned for the first time.
  - » *pass j* ends after every vertex on queue at end of pass *j*–1 has been scanned
  - » the time for each pass is $O(m)$
- **Theorem 7.7**. If no negative cycle is reachable from *s*, the breadth-first algorithm runs in $O(mn)$ time stopping after at most pass n–1; otherwise it never halts

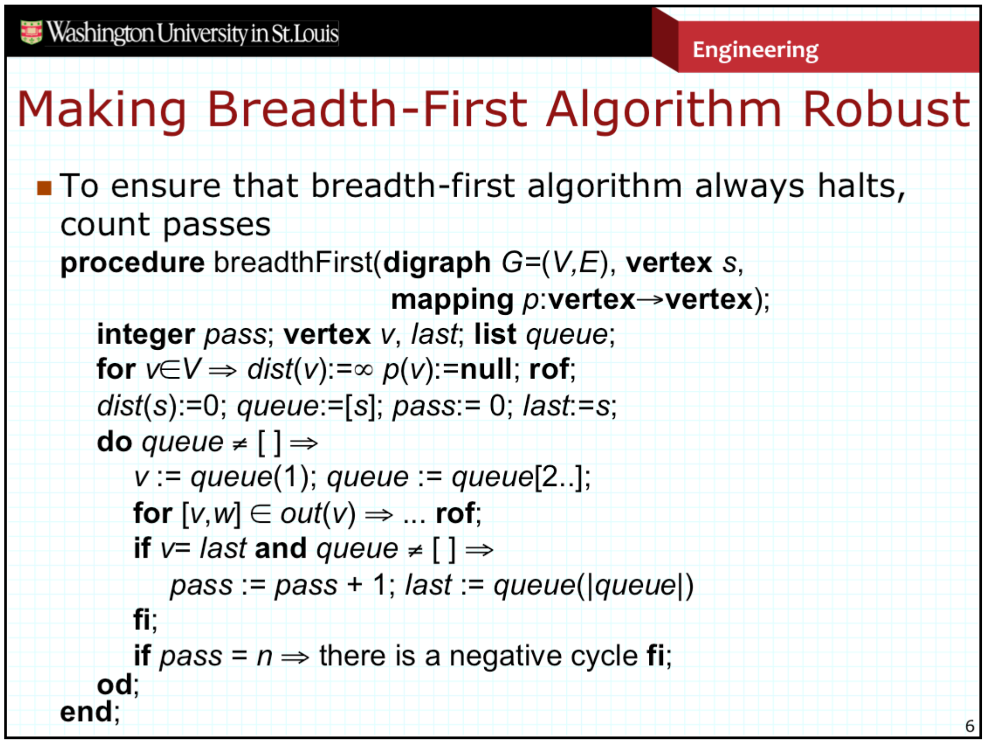
  Proof. Show by induction on *k* that if there is a shortest path from *s* to *v* with *k* edges, then *dist*(*v*) will equal the length of this path by the end of pass *k*–1

  Basis: *k*=1. After pass 0, *dist*(*w*)=*length*(*s*,*w*), for every edge [*s*,*w*]

Induction: assume that after pass k–1, for every vertex v with a shortest path from s with k edges, *dist*(v) is the length of this path

- » let w be any vertex that has a shortest path from s with k+1 edges and let v be its predecessor on this path
- » since *dist*(v) is length of a shortest path from s to v, *dist*(v) +*length*(v,w) is length of a shortest path from s to w
- » when *dist*(v) receives this value, v is placed on the queue
- » since this happens before end of pass k–1, v will be removed from queue before end of pass k
- » when v is removed from queue, *dist*(w) is set to *dist*(v)+*length*(v,w) and w is placed on queue
- » hence *dist*(w) equals the length of a shortest path from s by the end of pass k ■

- Theorem 7.7 implies Theorem 7.2



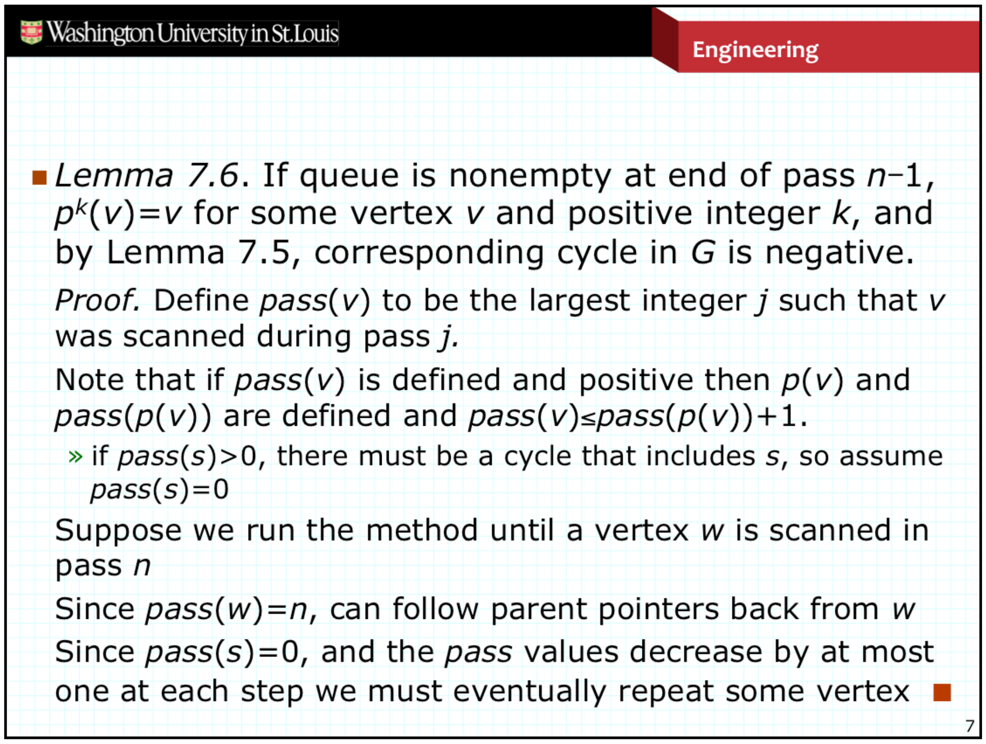
Washington University in St.Louis Engineering

Making Breadth-First Algorithm Robust

- To ensure that breadth-first algorithm always halts, count passes

```
procedure breadthFirst(digraph $G=(V,E)$, vertex $s$,
                            mapping $p$:vertex→vertex);
    integer $pass$; vertex $v$, $last$; list $queue$;
    for $v \in V \Rightarrow dist(v):=\infty$ $p(v)$:=null; rof;
    $dist(s):=0$; $queue:=[s]$; $pass:= 0$; $last:=s$;
    do $queue \neq [\,] \Rightarrow$
        $v := queue(1)$; $queue := queue[2..]$;
        for $[v,w] \in out(v) \Rightarrow$ ... rof;
        if $v= last$ and $queue \neq [\,] \Rightarrow$
            $pass := pass + 1$; $last := queue(|queue|)$
        fi;
        if $pass = n \Rightarrow$ there is a negative cycle fi;
    od;
end;
```

6



- *Lemma 7.6*. If queue is nonempty at end of pass $n-1$, $p^k(v)=v$ for some vertex v and positive integer k, and by Lemma 7.5, corresponding cycle in G is negative.

  Proof. Define $pass(v)$ to be the largest integer j such that v was scanned during pass j.

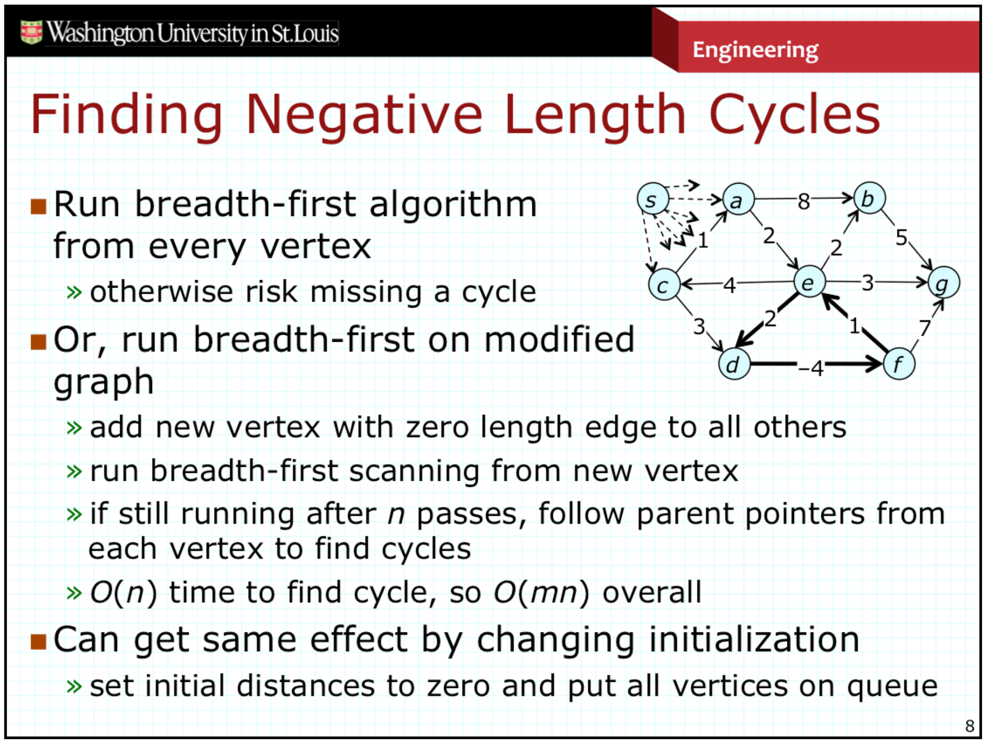
  Note that if $pass(v)$ is defined and positive then $p(v)$ and $pass(p(v))$ are defined and $pass(v) \leq pass(p(v))+1$.

  » if $pass(s)>0$, there must be a cycle that includes s, so assume $pass(s)=0$

  Suppose we run the method until a vertex w is scanned in pass n

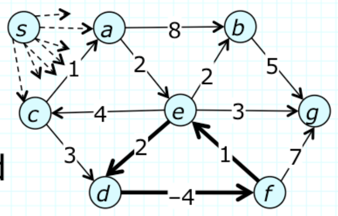
  Since $pass(w)=n$, can follow parent pointers back from w

  Since $pass(s)=0$, and the *pass* values decrease by at most one at each step we must eventually repeat some vertex ■



Finding Negative Length Cycles

- Run breadth-first algorithm from every vertex
  - » otherwise risk missing a cycle
- Or, run breadth-first on modified graph
  - » add new vertex with zero length edge to all others
  - » run breadth-first scanning from new vertex
  - » if still running after *n* passes, follow parent pointers from each vertex to find cycles
  - » $O(n)$ time to find cycle, so $O(mn)$ overall
- Can get same effect by changing initialization
  - » set initial distances to zero and put all vertices on queue



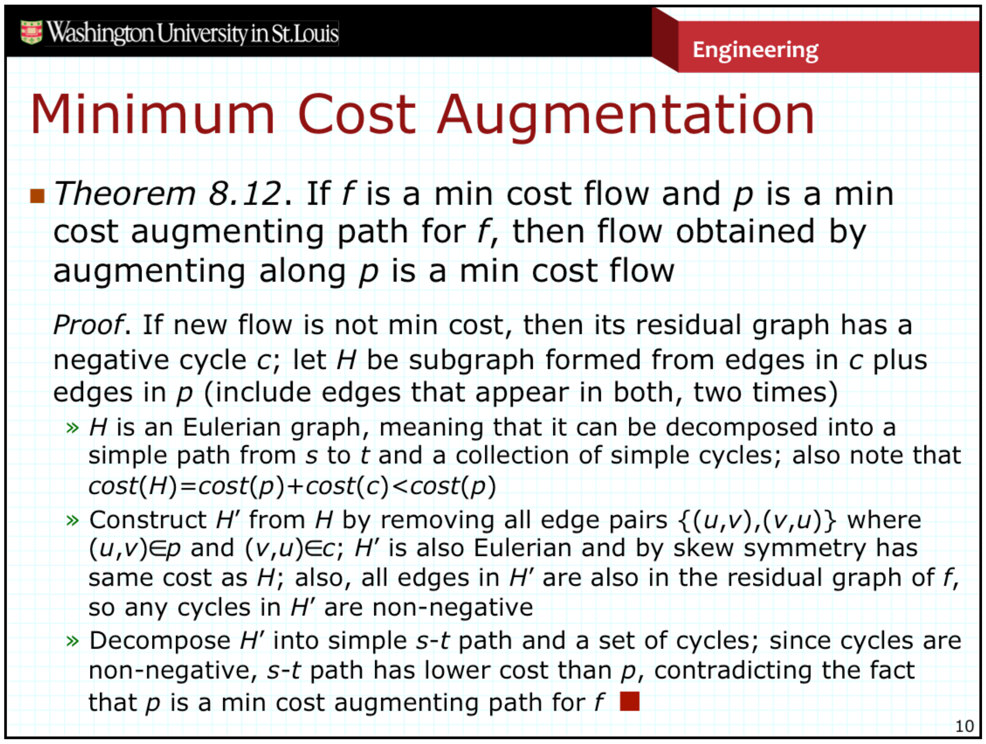
Cost Reduction Method

- *Theorem 8.11*. A flow f has min cost if and only if its residual graph R_f has no negative cost cycle

  Proof. If R_f contains a negative cost cycle, we can reduce cost of f without changing its value by pushing flow around the cycle

  If f does not have min cost and f^* is min cost flow with same value as f, then f^*-f is flow on R_f with zero value and negative cost

  By flow conservation, f^*-f can be partitioned into sum of flows on cycles; at least one of these cycles must have negative cost ■
- To find a max flow of min cost
  - » find any max flow then repeat following step as long as possible
    - *Cost reduction step:* find negative cost cycle in R; push as much flow as possible around cycle
  - » for integer costs and capacities, takes $\leq 2mc\gamma$ cost reduction steps, where c is max edge capacity and γ is the largest edge cost magnitude; each step can be done in $O(mn)$ time, using breadth-first scanning algorithm for shortest paths



Washington University in St. Louis — Engineering

Minimum Cost Augmentation

- *Theorem 8.12*. If *f* is a min cost flow and *p* is a min cost augmenting path for *f*, then flow obtained by augmenting along *p* is a min cost flow

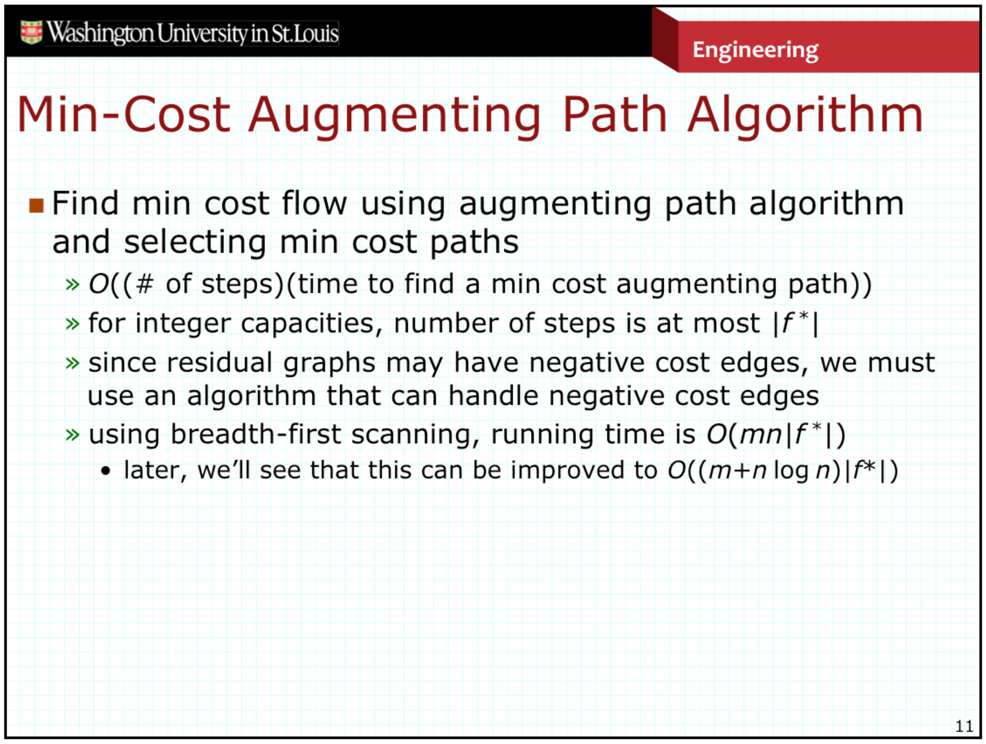
  Proof. If new flow is not min cost, then its residual graph has a negative cycle *c*; let *H* be subgraph formed from edges in *c* plus edges in *p* (include edges that appear in both, two times)

  » *H* is an Eulerian graph, meaning that it can be decomposed into a simple path from *s* to *t* and a collection of simple cycles; also note that $cost(H)=cost(p)+cost(c)<cost(p)$

  » Construct *H'* from *H* by removing all edge pairs $\{(u,v),(v,u)\}$ where $(u,v)\in p$ and $(v,u)\in c$; *H'* is also Eulerian and by skew symmetry has same cost as *H*; also, all edges in *H'* are also in the residual graph of *f*, so any cycles in *H'* are non-negative

  » Decompose *H'* into simple *s*-*t* path and a set of cycles; since cycles are non-negative, *s*-*t* path has lower cost than *p*, contradicting the fact that *p* is a min cost augmenting path for *f* ■

10



Washington University in St. Louis

Engineering

Min-Cost Augmenting Path Algorithm

- Find min cost flow using augmenting path algorithm and selecting min cost paths
  - » O((# of steps)(time to find a min cost augmenting path))
  - » for integer capacities, number of steps is at most $|f^*|$
  - » since residual graphs may have negative cost edges, we must use an algorithm that can handle negative cost edges
  - » using breadth-first scanning, running time is $O(mn|f^*|)$
    - later, we'll see that this can be improved to $O((m+n \log n)|f^*|)$

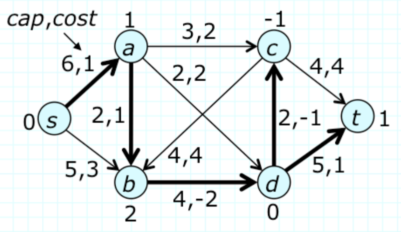
11

Washington University in St. Louis

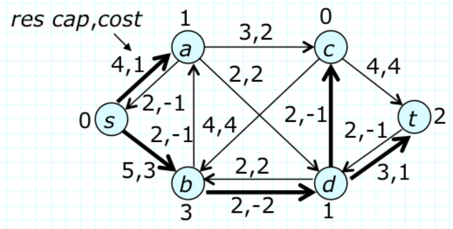
Engineering

Example

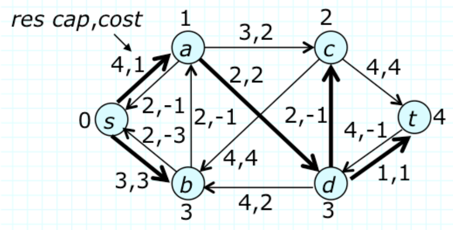
network with spt and distances from s



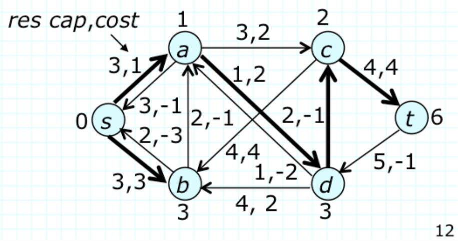
Step 2. showing residual graph and spt



Step 3.



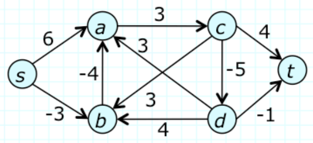
Step 4.



12

Exercises

1. The diagram below represents the residual graph for a flow. The numbers next to the edges are the edge costs. Is the flow corresponding to this residual graph a min-cost flow? Why or why not?



It is not a min cost flow because the cycle acdba has a cost of -2.

Explain how you can reduce the cost by 6 if all edges in the residual graph have capacity 3?

Since the edges in the residual graph all have capacity 3, we can push 3 units of flow around the cycle acdba. This reduces the cost by 6.

2. Let *G* be a directed graph in which each edge (*u*,*v*) has a length *length*(*u*,*v*) Describe (in words) an algorithm to determine if *G* has *k* edge disjoint paths between a given pair of vertices with total length no larger than a bound *B*.

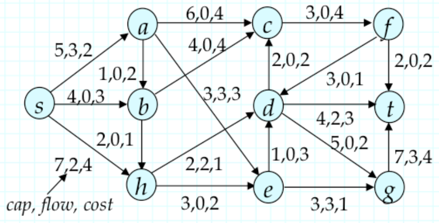
We can solve this using min-cost flows, by treating the given graph as a flow graph, with the two vertices as source and sink. All edges are assigned capacity 1 and the cost of each edge is equated to the given edge length.

If we do k steps of the min-cost augmenting path algorithm, the set of augmenting paths will all be edge-disjoint. If the sum of their costs is less than or equal the bound B, then the algorithm succeeds. If the algorithm terminates before finding k paths, or if the total length of the paths found is too large, then the original graph does not have k edge disjoint paths with the desired total length.

Washington University in St.Louis

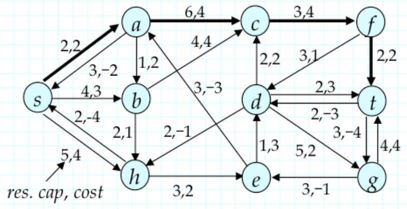
Engineering

3. The figure below represents an instance of the minimum cost, maximum flow problem. What is the cost of the flow that is shown?

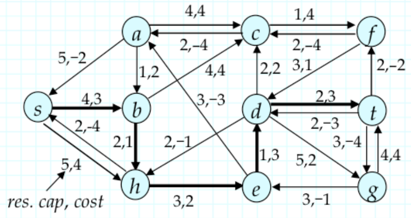


The cost of the original flow is 46.

Show the residual graph corresponding to this flow (including the edge costs) and show the minimum cost augmenting path in the residual graph.



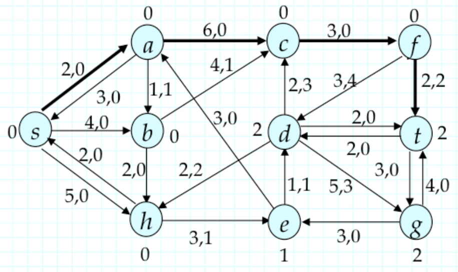
Show how the residual graph changes when the minimum cost augmenting path is saturated and show the new minimum cost augmenting path.



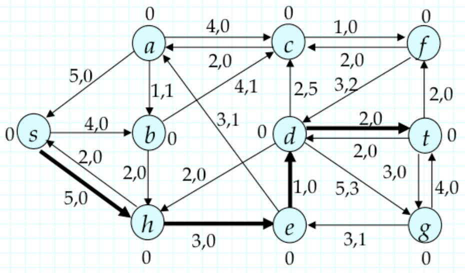
14

4. Take the original residual graph from the previous problem and transform the edge costs to make them non-negative. Also, show the distances from *s* using the transformed costs.

Find the shortest augmenting path using the transformed costs, add flow to that path and show the new residual graph with the new transformed edge costs and new distance values. Identify the shortest augmenting path in this residual graph.



s
a
b
c
d
e
f
g
h
t
0
2
1
2,0
3,0
4,0
2,0
5,0
6,0
1,1
4,1
3,0
2,3
3,4
2,2
2,0
2,2
3,0
1,1
5,3
4,0
3,1



s
a
b
c
d
e
f
g
h
t
0
4,0
2,0
1,0
5,0
1,1
4,1
2,5
3,2
3,1
2,0
2,0
3,0
5,3
4,0
3,1