15.082J / 6.855J
February 27, 2003

The Label Correcting Algorithm

Overview of the Lecture

- A generic algorithm for solving shortest path problems
  - negative costs permitted
  - but no negative cost cycle (at least for now)
- The use of reduced costs
- All pair shortest path problem

- INPUT G = (N, A) with costs c
- Node 1 is the source node
- There is no negative cost cycle
  - We will relax that assumption later

Optimality Conditions

Lemma. Let d*(j) be the shortest path length from node 1 to node j, for each j. Let d( ) be node labels with the following properties:

$d(j) \leq d(i) + c_{ij}$ for $i \in N$ for $j \neq 1$. (1)

$d(1) = 0$. (2)

Then $d(j) \leq d^*(j)$ for each j.

Proof. Let P be any path from node 1 to node j, with length c(P), and suppose P has k arcs.

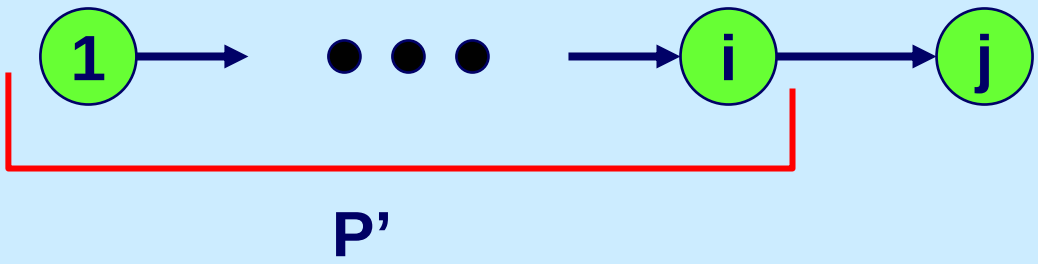
Claim: $d(j) \leq c(P)$.

Note: if P is the shortest path from 1 to j, then

$d(j) \leq c(P) = d^*(j)$, which is what we want to prove.

Completion of the proof.

Claim: $d(j) \leq c(P)$. Assume the claim is true for all paths with fewer than k arcs. We will show it is also true for path P. Suppose P = P', (i,j).



$d(i) \leq c(P')$ by inductive hypothesis.

$d(j) - d(i) \leq c_{ij}$ by assumption (1)

So, $d(j) \leq c(P') + c_{ij} = c(P)$, completing the proof.

Optimality Conditions

Theorem. Let $d(1), \ldots, d(n)$ satisfy the following properties for a directed graph $G = (N,A)$:

1. $d(1) = 0$.
2. $d(i)$ is the length of some path from node 1 to node i.
3. $d(j) \leq d(i) + c_{ij}$ for all $(i,j) \in A$.

Then $d(j) = d^*(j)$.

Proof. $d(j) \leq d^*(j)$. Also, $d(j) \geq d^*(j)$ as $d(j)$ is the length of some path from node 1 to node j. Thus $d(j) = d^*(j)$.

A Generic Shortest Path Algorithm

Notation.

d(j) = "temporary distance labels".

- **At each iteration, it is the length of a path (or walk) from 1 to j.**
- **At the end of the algorithm d(j) is the minimum length of a path from node 1 to node j.**

Pred(j) = Predecessor of j in the path of length d(j) from node 1 to node j.

c_{ij} = length of arc (i,j).

A Generic Shortest Path Algorithm

Algorithm LABEL CORRECTING;

begin

d(1) : = 0 and Pred(1) := ø;

d(j) : = ∞ for each j ∈ N – {1};

while some arc (i,j) satisfies d(j) > d(i) + c_{ij} do

begin

d(j) : = d(i) + c_{ij};

Pred(j) : = i;

end;

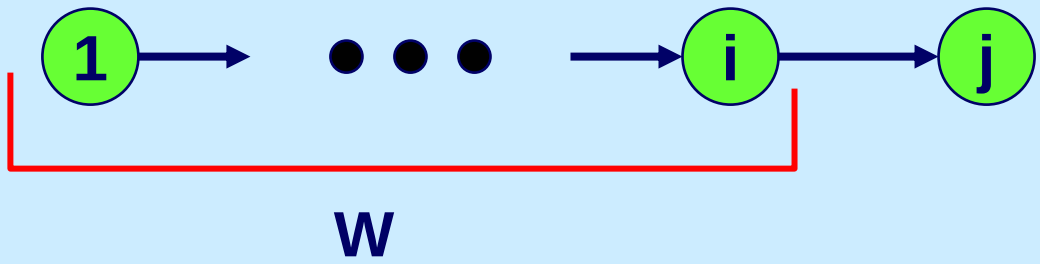
end;

Label correcting animation.

Theorem. Suppose all data are integral, and that there are no negative cost cycles in the network. Then the label correcting algorithm ends after a finite number of steps with the optimal solution.

Proof of Finiteness. Claim: d(j) is the length of some walk from node 1 to node j. (And thus $d(j) \geq d^*(j)$)

Assume the claim is true for all nodes before a relabel of node j. Then it is true when we let $d(j) = d(i) + c_{ij}$.



Finiteness continued

Proof of Finiteness. **At each iteration, d(j) decreases by at least one for some j.**

Also $d(j) \geq d^*(j) > -nC$, where $C = \max(|c_{ij}| : (i,j) \in A)$.

So, the number of iterations is $O(n^2C)$.

Claim: **at termination, the distances are all shortest path distances.**

Proof. **At end, $d(j) \leq d(i) + c_{ij}$ for all $(i,j) \in A$. By the previous theorem, $d(j) = d^*(j)$.**

More on Finiteness

What happens if data are not required to be integral?

The algorithm is still finite, but one needs to use a different proof.

What happens if there is a negative cost cycle?

The algorithm may no longer be finite.

Possibly, d(j) keeps decreasing to $-\infty$.

But we can stop when d(j) < -nC since this guarantees that there is a negative cost cycle.

On computational complexity

Proving finiteness is OK, but ….

Can we make the algorithm polynomial time?

If so, what is the best running time?

Can we implement it efficiently in practice?

Computational issues

Simple Polynomial Time Version (FIFO):

We define a *pass* to consist of scanning all arcs in A, updating distance labels when $d(j) > d(i) + c_{ij}$.

We refer to a "*pass*" as performing an update on each arc in A. The algorithm (in this simple version performs n passes or until no more updates take place, whichever comes first, at which point the algorithm terminates.

Theorem. The FIFO label correcting algorithm finds the minimum length path from 1 to j for all j in N in O(nm) steps, or else shows that there is a negative cost cycle.

Proof.

We want to show that no update takes place in pass n.

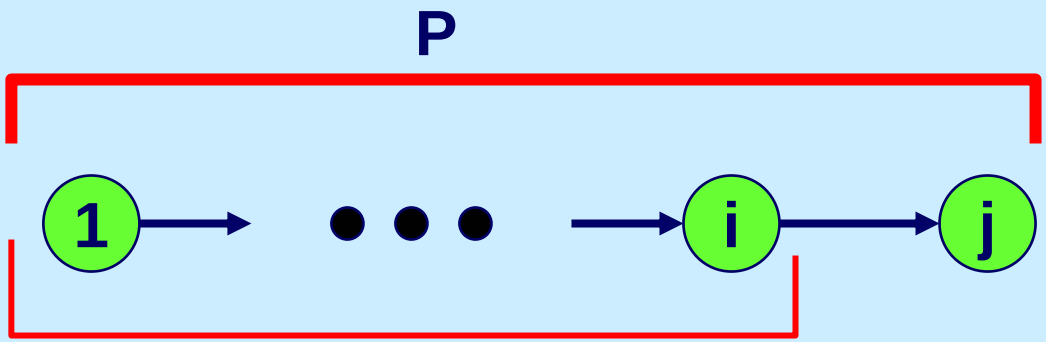
Let $d^k(j)$ be the value d(j) after k passes.

Claim $d^k(j) = d^*(j)$ whenever that shortest path from 1 to j has at most k arcs.

It is true for k = 1.

Proof. Claim $d^k(j) = d^*(j)$ whenever that shortest path from 1 to j has at most k arcs.

Assume that the claim is true for k-1.



Assume P is the shortest path from node 1 to node j. Then P' is the shortest path from node 1 to node i.

After pass k, $d^k(j) \leq d^{k-1}(i) + c_{ij} = d^*(j)$.

Completion of the proof.

Theorem. ***The FIFO label correcting algorithm finds the minimum length path from 1 to j for all j in N in O(nm) steps, or else shows that there is a negative cost cycle.***

Proof. If there is no negative cost cycle, the shortest walk from 1 to j has at most n-1 arcs, and so after n-1 passes, d(j) = d*(j) for all j.

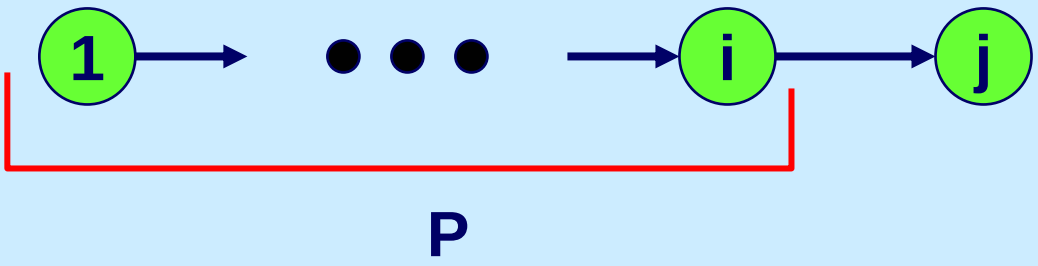
What if there is a negative cost cycle?

If in the nth pass, there is no update, then the algorithm has found the shortest path distances.

If in the nth pass, there is an update, then there must be a negative cost cycle.

Can we speed this up in practice?

Observation: if d(i) is not decreased in one pass, then there is no need to scan arcs out of i at the next pass.



d(j) = d(i) + c_{ij} at the end of pass k, and d(i) does not change during pass k.

Create a LIST of nodes j that need to be scanned.

Whenever d(j) is decreased, then add j to LIST

Major iteration: select a node i from LIST and

Procedure Update(i)

for each (i, j) $\in$ A(i) do

if d(j) > d(i) + c_{ij} then d(j) := d(i) + c_{ij} and pred(j) := i and LIST := LIST $\cup$ {j}.

Modified Label Correcting Algorithm

```
Algorithm  Modified Label Correcting;
begin
    d(1) : = 0 and  pred(1) : = ø;
    d(j) : = ∞  for each j ∈ N – {1};
    LIST : = {1};
    while LIST ≠ ø do
        begin
            delete an element  i from LIST;
            Update(i)
            for each j such that d(j) decreases, add j to LIST
        end;
end;
```

Modified Label Correcting Algorithm

FIFO Implementation

FIFO. Treat LIST as a Queue. Add nodes to the end of LIST and take nodes from the beginning.

LIFO. Treat LIST as a Stack. Add nodes to the "top" of LIST, and delete the top node of LIST as well. (Efficient in practice, but bad in the worst case.)

Theorem. *The FIFO modified label correcting algorithm finds the minimum length path from 1 to j for all j in N in O(nm) steps, or else shows that there is a negative cost cycle.*

Proof. Same as previous theorem.

Solving all pairs shortest problems

Note: Dijkstra's algorithm is much faster in the worst case than label correcting.

- **O(m + n log n) vs O(mn)**

- **To solve the all pairs shortest path problem we will solve it as**
  - **one shortest path problem using label correcting**
  - **n-1 shortest path problems using Dijkstra**
  - **Technique: transform the problem so that we transform negative arc costs into non-negative costs.**

Reduced Costs

Suppose that π is any vector of *node potentials*.

Let $c_{ij}^{\pi} = c_{ij} - \pi_i + \pi_j$ be the *reduced cost* of arc (i,j)

For a path P, let c(P) denote the cost (or length) of P.

Let $c^{\pi}(P)$ denote the reduced cost (or length) of P

$c(P) = \sum_{(i,j)\in P} c_{ij};$ $\qquad$ $c^{\pi}(P) = \sum_{(i,j)\in P} c_{ij}^{\pi}$;

Lemma. For any path P from node s to node t,

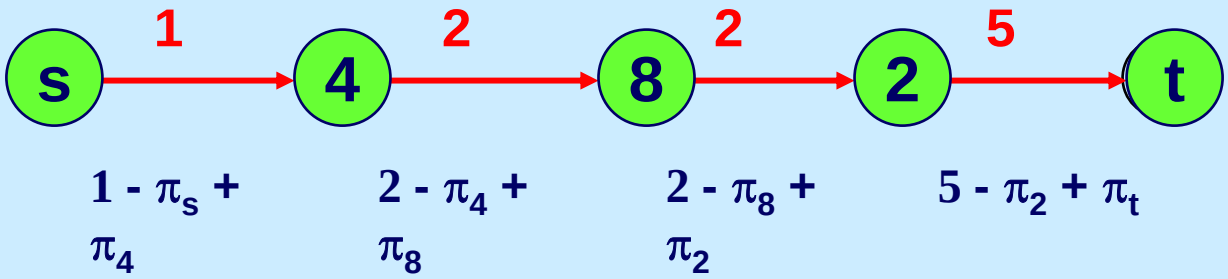
$c^{\pi}(P) = c(P) - \pi_s + \pi_t$.

For any path P from node s to node t, $c^{\pi}(P) = c(P) - \pi_s + \pi_t$.

Proof. When written as a summation, the terms in $c^{\pi}(P)$ involving π_i for some i all cancel, except for the term $-\pi_s$ and the term π_t.

Note: for fixed vector π of multipliers and for any pair of nodes s and t, $c^{\pi}(P) - c(P)$ is a constant for every path P from s to t.

Corollary. ***A shortest path P from s to t with respect c^{π} is also the shortest path with respect to c.***

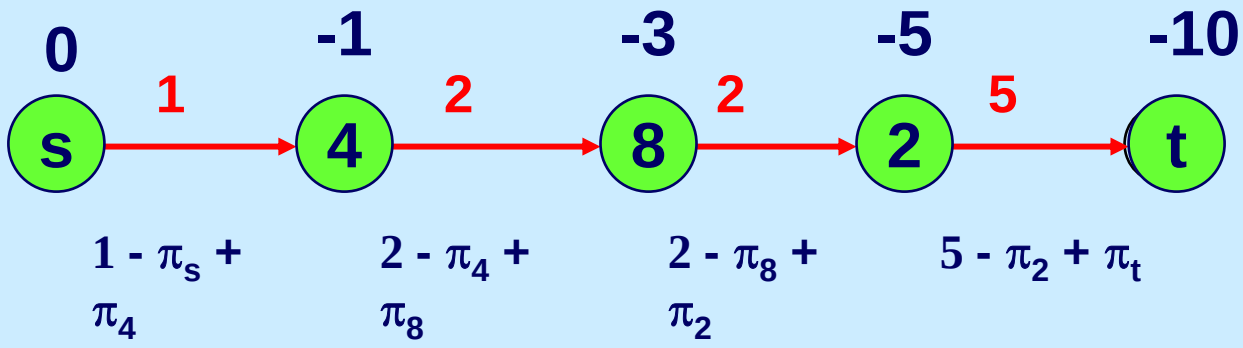


Using reduced costs

Lemma. ***Let d(j) denote the shortest path from node s to node j. Let $\pi_j = -d(j)$ for all j.***

Then $c_{ij}^{\pi} \geq 0$ for all (i,j) $\in$ A.

Proof. $d(j) \leq d(i) + c_{ij} \Rightarrow c_{ij} + d(i) - d(j) \geq 0 \Rightarrow c_{ij}^{\pi} \geq 0.$



Solving the all pair shortest path problem

Step 1. Find the shortest path from node 1 to all other nodes. Let d(j) denote the shortest path from 1 to j for all j.

Step 1B. Let $\pi_j = -d(j)$ for all j.

Step 2. For i = 2 to n, compute the shortest path from node i to all other nodes with respect to arc lengths c^{π}.

Running time using Radix Heaps.**

O(nm) for the first shortest path tree

O(m + n log C) for each other shortest path tree.

$O(nm + n^2 \log C)$ in total.

** One can choose a slightly faster approach.

Detecting Negative Cost Cycles

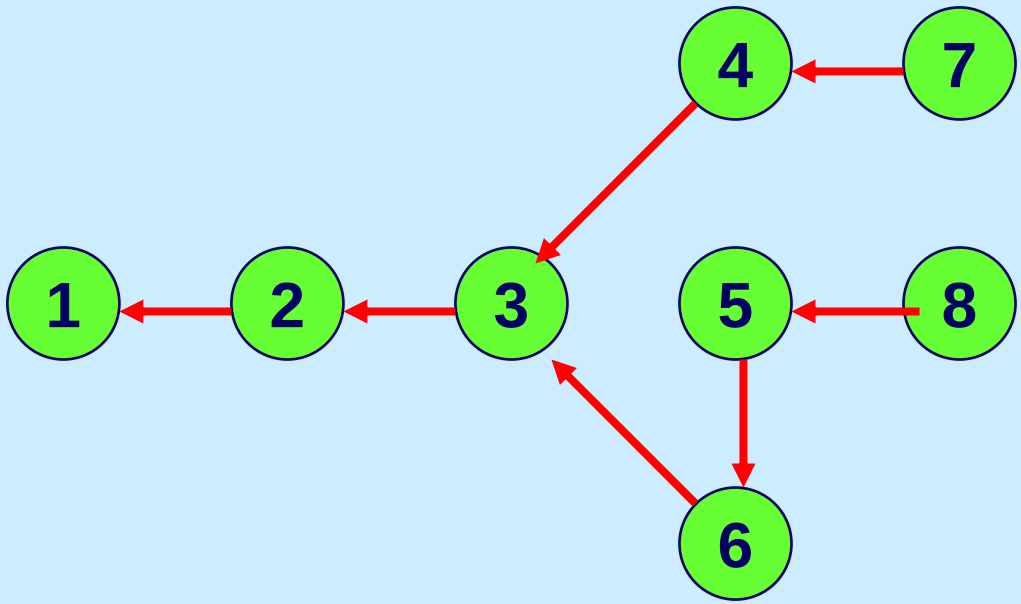
Approach 1. **Stop if d(j) is sufficiently small, say $d(j) \leq -nC$.**

Approach 2. **Run the FIFO label correcting algorithm of label correcting, and stop if you have scanned any node at least n times.**

Approach 3. **Run the FIFO label correcting algorithm, and keep track of the number of arcs on the "path" from s to j. If the number of arcs exceeds n-1, then quit.**

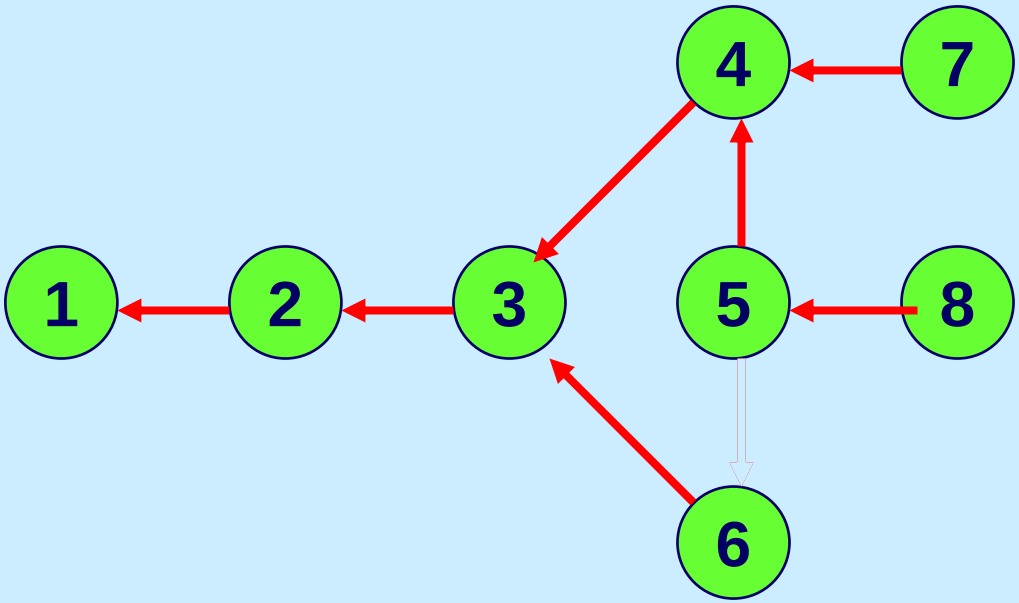
Approach 4. **At each iteration of the algorithm, each node j (except for the root) has a a temporary label d(j) and a predecessor pred(j). The *predecessor subgraph* consists of the n-1 arcs $\{(pred(j),j) : j \neq s\}$. It should be a tree. If it has a cycle, then the cost of the cycle will be negative, and the algorithm can terminate.**

A Predecessor Graph



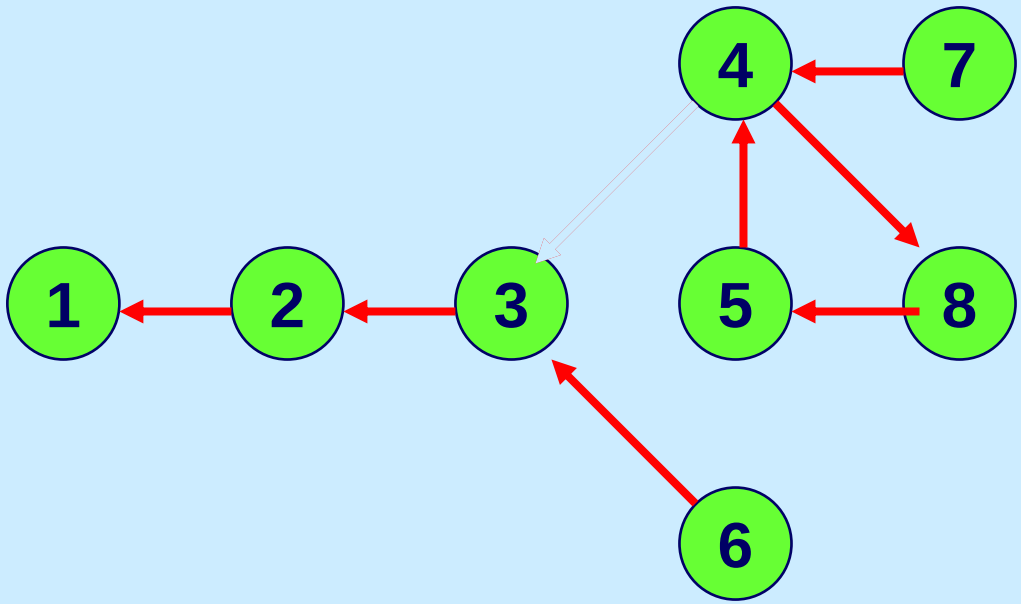
Each node except for node 1 has one predecessor. The graph either is an in-tree or it has a directed cycle.

A Predecessor Graph



Suppose we Update(5), and pred(5) := 4.

A Predecessor Graph



Suppose we Update(4) and pred(4) := 8.

Then 4-8-5-4 has negative cost.

Prior to Update(4), the following is true:

$d(5) = d(4) + c_{45}$

$d(8) = d(5) + c_{58}$

$d(4) > d(8) + c_{84}$

To find negative cost cycles, periodically check the predecessor subgraph to see if it contains a cycle.

Summary of Lecture

1. Optimality conditions for the shortest path algorithm.

2. The label correcting algorithm. Excellent in practice.

   O(nm) in theory, using a FIFO implementation of LIST.

3. All pairs shortest path problem

4. Detecting negative cost cycles.