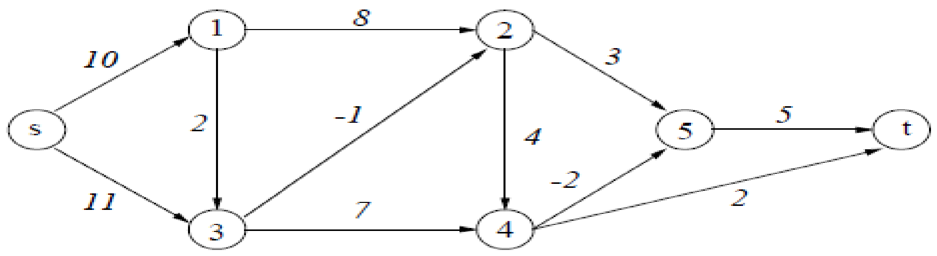
Lecture 3: Shortest Paths

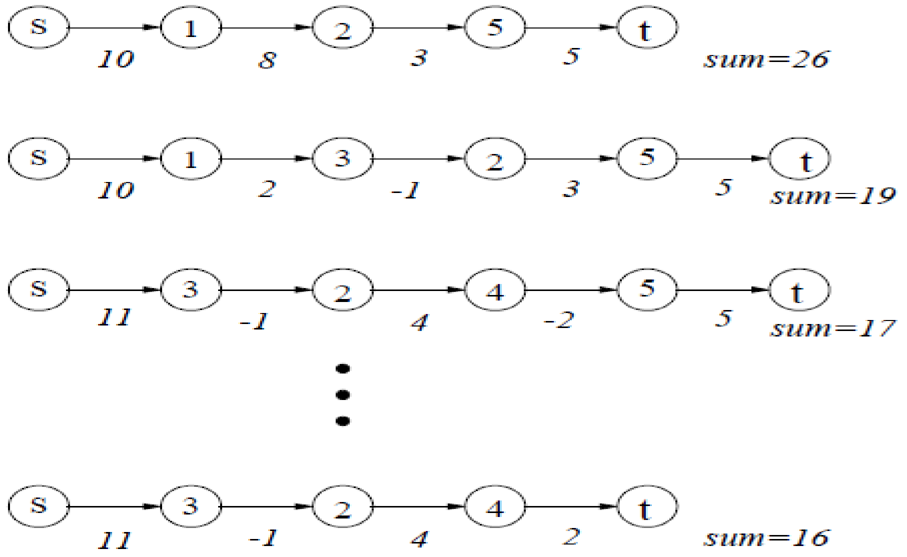
- Introduction
- Bellman’s Equation
- Dijkstra’s Algorithm
- Bellman Ford’s Method
- Floyd Warshall’s Method
- Dreyfus Method

Example

- Shortest Path Problem



To find a shortest directed path (with no repeated nodes)



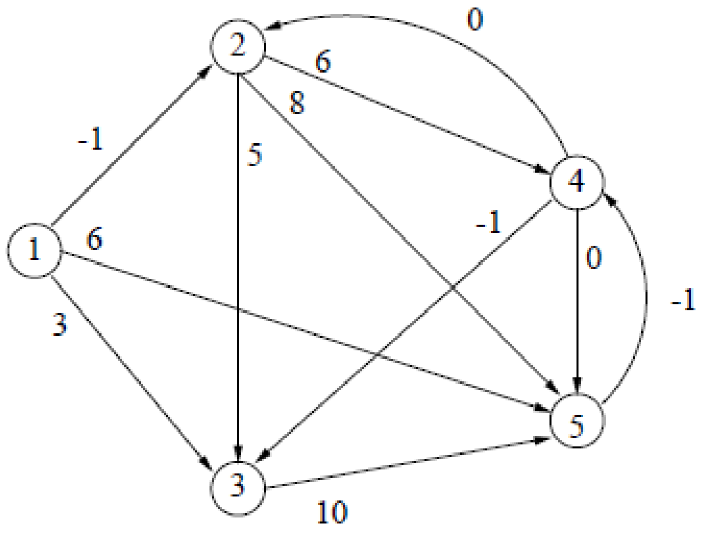
Shortest Path Problem

- In this chapter, we concern exclusively with shortest path problems in directed networks.
- If all arc lengths are positive, then an undirected network can be converted to a directed one with each arc replaced by a pair of arcs (i, j) and (j, i) with the original length.

- It is entirely feasible to compute shortest paths for undirected networks with positive and negative arc lengths, if such a network contains no cycles of negative length.

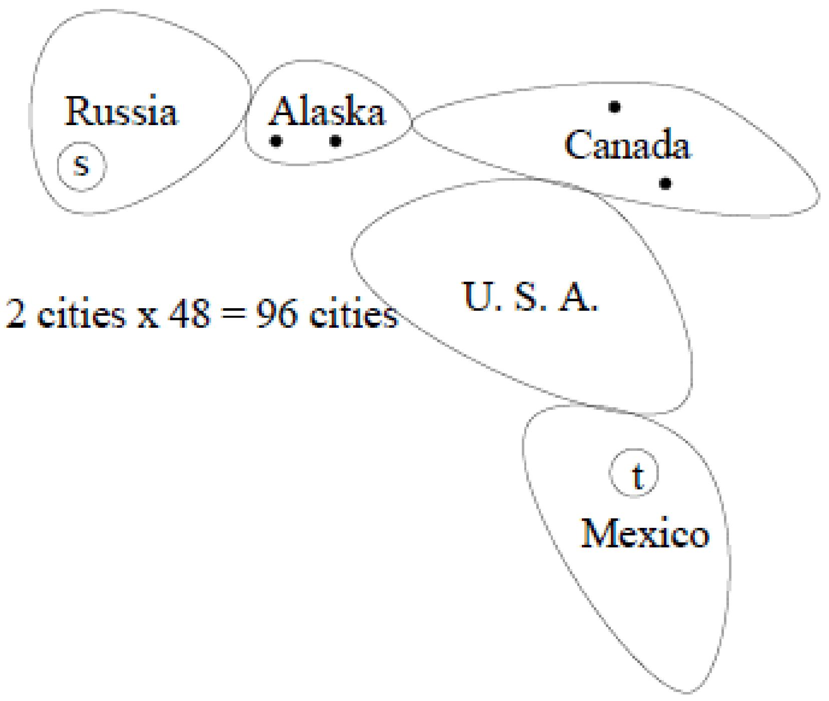
D. B. Johnson, *Algorithms for Shortest Paths*,
Ph.D.thesis, Cornell Univ., Ithaca,
New York, 1973.

Length Matrix



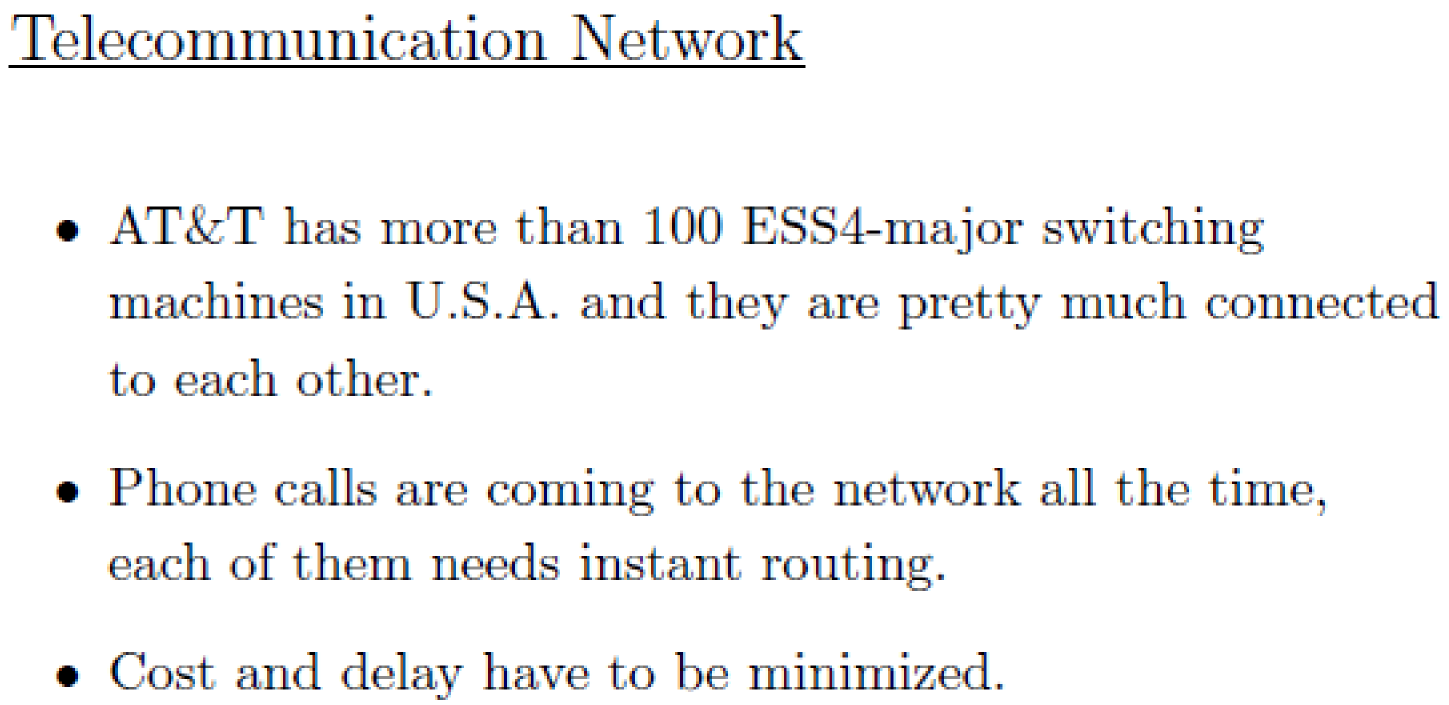
$$A = \begin{array}{c|ccccc} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & -1 & 3 & \infty & 6 \\ 2 & \infty & 0 & 5 & 6 & 8 \\ 3 & \infty & \infty & 0 & \infty & 10 \\ 4 & \infty & 0 & -1 & 0 & 0 \\ 5 & \infty & \infty & \infty & -1 & 0 \end{array}$$

Is the Problem Difficult?



- each pair of nodes are connected by a two-way arc.
- number of different paths (without repeated nodes) is $P(100) = 100! + 100(99!) + \binom{100}{2}(98!) + \cdots + \binom{100}{98}(2!) + 100 + 1.$

Is This a Real Problem?



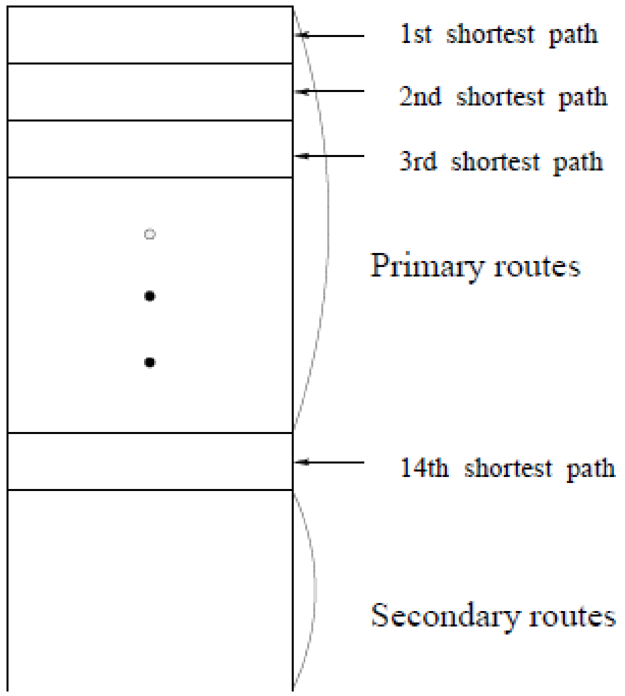
Telecommunication Network

- AT&T has more than 100 ESS4-major switching machines in U.S.A. and they are pretty much connected to each other.
- Phone calls are coming to the network all the time, each of them needs instant routing.
- Cost and delay have to be minimized.

Call Routing Problem

How are calls being handled ?

Routing Table: (Fixed routing directory)



Other Applications

- **Most Reliable Paths**

  In a communications network, the probability that link from i to j is operative is p_{ij}. Hence the probability that all the links in any given path are operative is the product of the link probabilities. What is the most reliable path from one designated node to another?

Most Reliable Paths

Reliability of a path P

$$\prod_{(i,j)\in P} p_{ij}$$

"length" $a_{ij} = -\log p_{ij}$

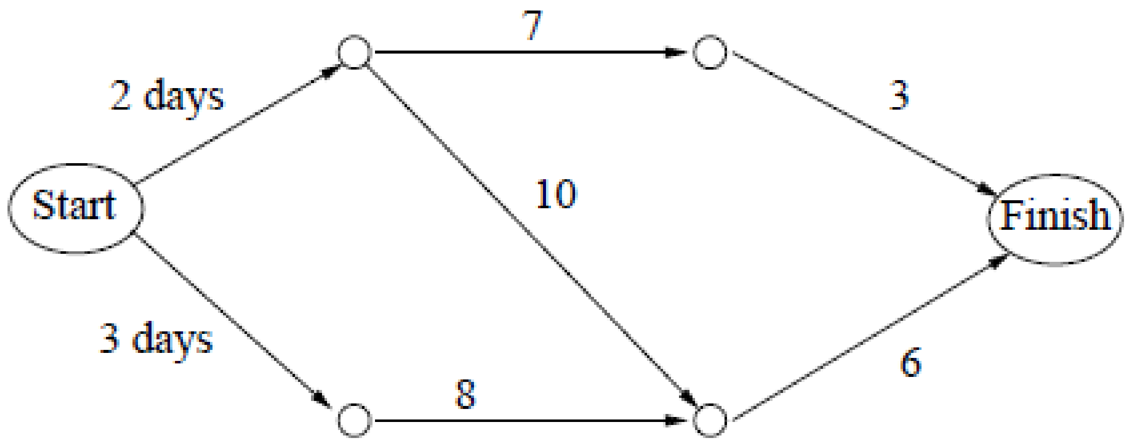
Other Applications

- **PERT Networks**
  A large project is divisible into many unit “tasks”. Each task requires a certain amount of time for its completion, and the tasks are partially ordered.

  One can form a network in which each arc (i, j) is identified with a task and the nodes are identified with “events,” i.e., the completion of various tasks. If (i, j) and (j, k) are arcs, then task (i, j) must be completed before task (j, k) is begun.

  This network is sometimes called a PERT (Project Evaluation and Review Technique) or CPM (Critical Path Method) network. Many types of analyses can be performed with such a network. For example, we may determine the shortest possible time in which the entire project can be completed.

Pert Network



Required time = longest path

"non-negative" time

"length" $a_{ij} = -t_{ij}$

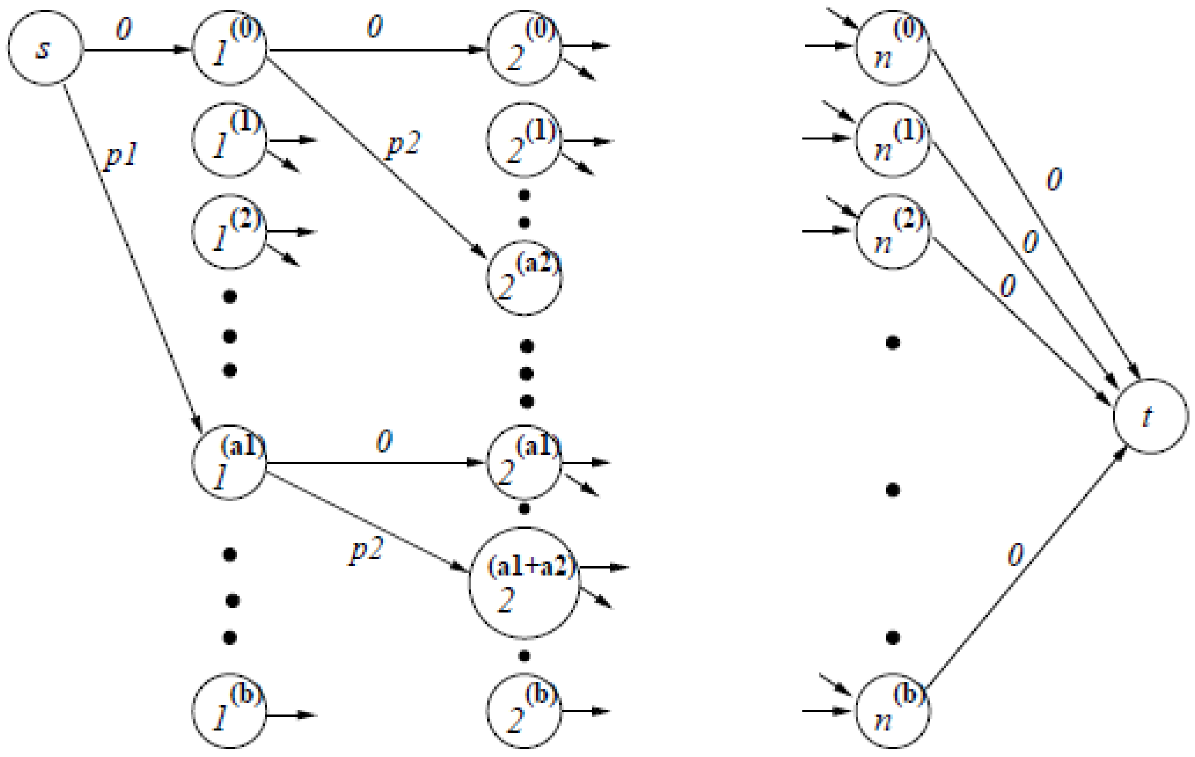
Other Applications

- **The Knapsack Problem**

  Suppose there are n objects, the ith object having a positive integer "weight" a_{ij} and "value" p_j. It is desired to find the most valuable subset of objects, subject to the restriction that their total weight does not exceed b, the capacity of a "knapsack."

$$\begin{array}{lll} \text{maximize} & & \sum_j p_j x_j \\ \text{subject to} & & \sum_j a_j x_j \leq b \\ \text{where} & x_j & = 1 \text{ if object } j \text{ is chosen} \\ & & = 0 \text{ otherwise} \end{array}$$

Knapsack Problem



$n(b+1)$ nodes between s, t

How to Approach This Problem?

- The dominant ideas in the solution of these shortest-path problems are those of dynamic programming.

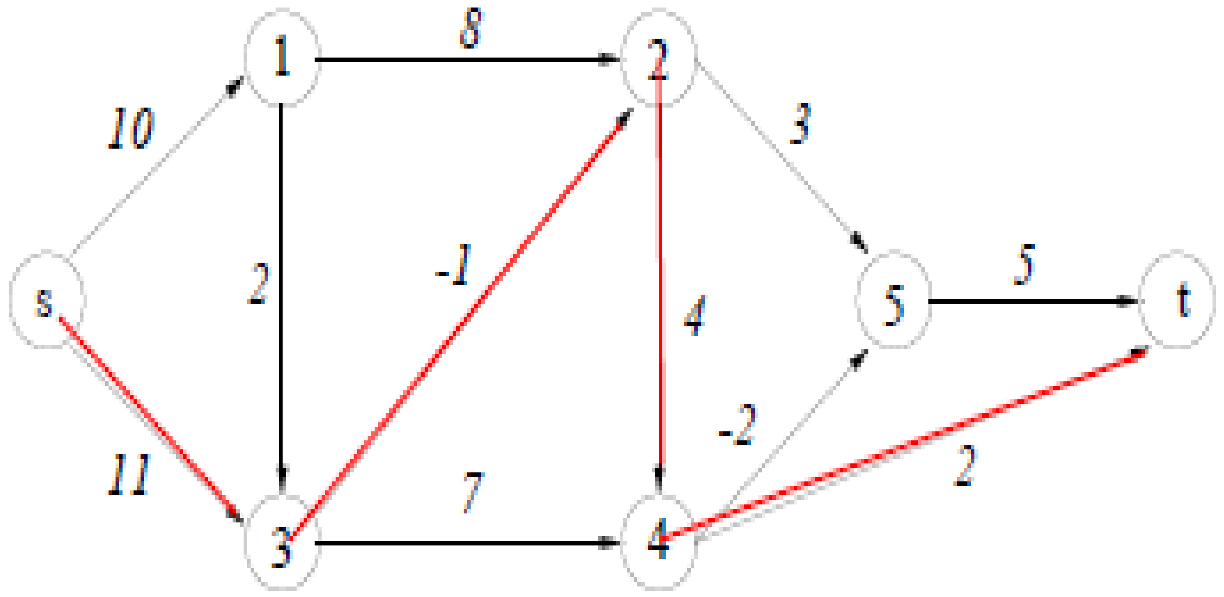
- Have you heard of “Bellman’s Equation” before?

  R. E. Bellman, *On a Routing Problem*",

  Quart. Appl. Math, 16 (1958) 87-90.

- How about “Principle of Optimality”?

Observations



Observations

- There seems to be no really good method for finding the length of a shortest path from a specific origin to a specific destination without, in effect, finding the lengths of shortest path from the origin to all other nodes.

- Any portion of a shortest path must be of shortest distance – Principle of Optimality!

- This is a necessary condition, but not sufficient.

Saying in A Mathematical Way

Let

$$\begin{aligned} a_{ij} &= \text{the (finite) length of arc } (i, j) \text{, if there is such an arc} \\ &= +\infty \text{, otherwise.} \\ u_j &= \text{the length of a shortest path from the the origin to node } j. \end{aligned}$$

Suppose the origin is numbered 1, and the other nodes are numbered $2, 3, , \ldots, n$. If there are no directed cycles with negative length (and, therefore, no negative closed paths), the shortest path lengths must satisfy Bellman's equations.

$$\left.\begin{aligned} u_1 &= 0, \\ u_j &= \min_{k \neq j}\{u_k + a_{kj}\} \quad (j = 2, 3, \ldots, n). \end{aligned}\right\} \text{(3.1)}$$

Bellman’s Equation

- It is a necessary condition.

- It holds under the assumption that “*no directed cycle with non-positive length*” - *(3.1)(a).*

- It applies the “*principle of optimality*” – *(3.1)(b).*

- It has *n* variables and *n* equations, but in nonlinear relations.

- Can be handled in an iterative fashion.

Questions

- When will Bellman's equation become a sufficient condition for shortest paths?

- (A1) There exists a finite-length path from the origin to each of the other nodes.
- (A2) All directed cycles are strictly positive in length.

  (No directed cycle with non-positive length.)

- Existence and uniqueness of a solution to Bellman’s equation?

Main Theorems (Under A1 and A2)

Theorem 3.1 If the network contains no nonpositive directed cycles, then there exists a tree rooted from the origin, such that the path in the tree from the origin to each of the other nodes is a shortest path. (We call such a tree a *tree of shortest paths.*)

Theorem 3.2 If the network contains no nonpositive cycles, and if there is a path from the origin to each of the other nodes, then there is a unique finite solution to the equation (3.1), where u_j is the length of shortest path from the origin to node j.

Proof of Existence

Let $u_1, u_2, \dots, u_n$ satisfy (3.1).

For any u_j, $\exists$ arc (k, j), such that $u_j = u_k + a_{kj}$

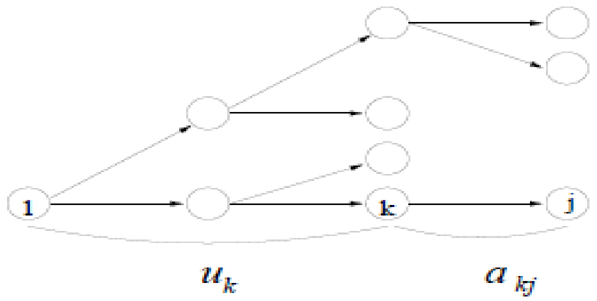
$\exists$ arc (l, k), such that $u_k = u_l + a_{lk}$

$\vdots$

until u_1 is reached

i.e. a directed $(1, j)$ path is found. $\underline{\text{(A1)+(A2)!}}$

Repeat this process for each j, we have a <u>tree rooted from the origin</u>.



This tree indeed contains shortest paths from the origin to each node and satisfy (3.1).
(This is Theorem 3.1)

Proof of Uniqueness

If (3.1) has two different solutions $u_1, u_2, \ldots, u_n$ and $\bar{u}_1, \bar{u}_2, \ldots, \bar{u}_n$, then $\exists\ j > 1$ such that $u_j \neq \bar{u}_j$.

Notice that $u_1 = \bar{u}_1 = 0$, we can choose j such that $\exists$ arc (k, j) in the tree of shortest paths and $u_k = \bar{u}_k$.

Now, if $u_j > \bar{u}_j$ then u_j violates (3.1)

if $\bar{u}_j > u_j$ then $\bar{u}_j$ violates (3.1)

Hence $u_j = \bar{u}_j$.

This contradicts the assumption that $u_j \neq \bar{u}_j$.

Hence Theorem 3.2 is true.

How to Solve Bellman’s Equation?

- Some simple cases:

  - Acyclic directed network
  - Directed network with positive arc lengths

Acyclic Network Case

- Acyclic Networks

  No directed cycle $\Rightarrow$ no directed cycle with nonpositive length

- Proposition 6.2

  A directed graph is acyclic if and only if its nodes can be numbered in such a way that for all arcs $(i, j), i < j$.

- (3.1) becomes

  $u_1 = 0$

  $u_j = \min_{k<j} \{u_k + a_{kj}\}, (j = 2, ..., n)$

Acyclic Network Case

- Implications

  $u_1 = 0$

  $u_2 = u_1 + a_{12}$

  $u_3 = min\{u_1 + a_{13}, u_2 + a_{23}\}$

  $\vdots$

  u_n

  sequencially determined.

- Complexity

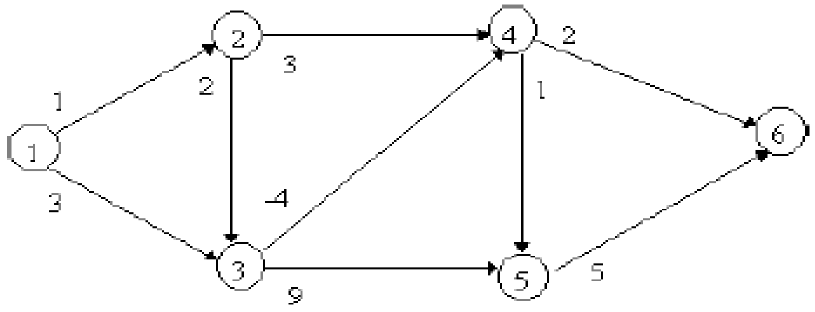
  Additions:
  $0 + 1 + 2 + ... + (n-1) = n(n-1)/2$

  Comparisons:
  $0 + 0 + 1 + 2 + ... + (n-2) = (n-1)(n-2)/2$

  Total= $O(n^2)$
  (Renumbering: $O(n^2)$, total= $O(n^2)$)

- Works for "*longest path problem*" too!

Acyclic Network Example



$$u_1 = 0$$

$$u_2 = 0 + 1 = 1$$

$$u_3 = \min\{0 + 3, 1 + 2\} = 3$$

$$u_4 = \min\{1 + 3, 3 - 4\} = -1$$

$$u_5 = \min\{3 + 9, -1 + 1\} = 0$$

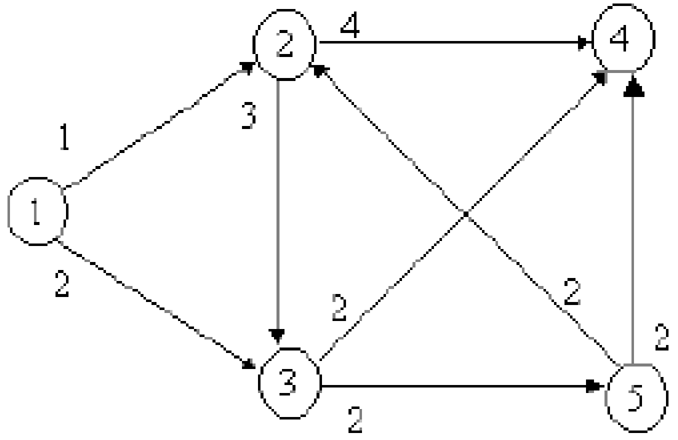
$$u_6 = \min\{-1 + 2, 0 + 5\} = 1$$

$$\begin{tikzcd} & & & & 6 \\ 1 \arrow[r] & 2 \arrow[r] & 3 \arrow[r] & 4 \arrow[ur] \arrow[r] & 5 \end{tikzcd}$$

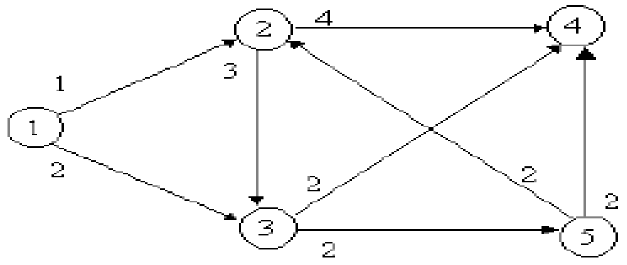
Case with Positive Arc Lengths

E. W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*," Numerische Mathematik, 1 (1950) 269-271.

- Networks with positive arcs:
  - positive arc lengths $\Rightarrow$ no directed cycle with nonpositive length
- Dijkstra's Method
  - A "labeling algorithm"



Dijkstra’s Algorithm



$u_1 = 0$, P = {1}, T = {2,3,4,5}

$u_2 = 1, u_3 = 2, u_4 = \infty, u_5 = \infty$

$u_2 = 1, P = \{1,2\}, T = \{3,4,5\}$

$u_3 = \min\{2, 1+3\} = 2$

$u_4 = \min\{\infty, 1+4\} = 5$

$u_5 = \infty$

$u_3 = 2, P = \{1,2,3\}, T = \{4,5\}$

$u_4 = \min\{5, 2+2\} = 4$

$u_5 = \min\{\infty, 2+2\} = 4$

$u_4 = 4, P = \{1,2,3,4\}, T = \{5\}$

$u_5 = \min\{4, 4+\infty\} = 4$

$u_5 = 4, P = \{1,2,3,4,5\}$

Dijkstra's Algorithm

Step 0 (Start)

Set u_1=0.

Set $u_j = a_{1j}$, for $j = 2, 3, \ldots, n$.

Set $P = \{1\}$, $T = \{2, 3, \ldots, n\}$.

Step1 (Designation of Permanent Label)

Find $k \in T$, where $u_k = \min_{j \in T}\{u_j\}$.

Set $T = T - k$, $P = P + k$.

If $T = \emptyset$, stop; the computation is completed.

Step 2 (Revision of Tentative Labels)

Set $u_j = \min\{u_j,\ u_k + a_{kj}\}$, for all $j \in T$.

Go to Step 1. //

Questions

- Why is Dijkstra's algorithm valid?

- How efficient is the algorithm?

Complexity Analysis

- Comparisons

  Step 1: $(n-2)+(n-3)+\ldots+1=(n-1)(n-2)/2$

  Step 2: $(n-2)+(n-3)+\ldots+1=(n-1)(n-2)/2$

- Additions:

  Step 2: $(n-2)+(n-3)+\ldots+1=(n-1)(n-2)/2$

Total: $(n-1)(n-2)$ comparisons

$(n-1)(n-2)/2$ additions

$O(n^2)$

Alternative Algorithm for Acyclic Network

- Dijkstra algorithm does suggest the following alternative computation for the acyclic shortest path problem.

$$\left.\begin{aligned}
&u_1^{(1)} = 0, \\
&u_j^{(2)} = a_{1j}, \qquad\qquad j = 2, 3, \ldots, n. \\
&u_j^{(k+1)} = \min\{u_j^{(k)},\ u_k^{(k)} + a_{kj}\}, \\
&j = 2, 3, \ldots, n-1;\ j \geq k+1.
\end{aligned}\right\} \quad (5.1)$$

We can view $u_j^{(1)} \geq u_j^{(2)} \geq \ldots \geq u_j^{(j)}$ as successive approximations of u_j, with $u_j^{(j)} = u_j$, for all j. Note that these equations imply exactly the same number of additions and comparisons as (4.1).

Alternative Algorithm for Acyclic Network

◇ Let

$$\bar{a}_{ij} = 1 \text{ if } a_{ij} < \infty, \text{ i.e., there is an arc } (i,j),$$
$$= 0, \text{ otherwise.}$$

Step 0 *(Start)*

```
Set $u_1$=0.
Set $u_j = a_{1j}$, for $j = 2, 3, \ldots, n$.
Set $P = \{1\}$, $T = \{2, 3, \ldots, n\}$.
Set $d_j = \sum_{i=2}^{n} \bar{a}_{ij}$, for $j = 2, 3, \ldots, n$.
```

Step1 *(Designation of Permanent Label)*

```
Find a $k \in T$, such that $d_k = 0$. If there is no such
$k$, stop; the network is not acyclic.
Set $T = T - k$, $P = P + k$.
If $T = \emptyset$, stop; the computation is completed.
```

Step 2 *(Revision of Tentative Labels)*

```
Set $u_j = \min\{u_j,\ u_k + a_{kj}\}$, for all $j \in T$.
Set $d_j = d_j - \bar{a}_{kj}$, for all $j \in T$.
Go to Step 1. //
```

No Non-positive Cycle Case

- Neither assume that the network is acyclic nor that all arc lengths are positive.

- Only assume that there are no non-positive cycles.

- Use the concept of *successive approximations.*

  - *Bellman-Ford Method*

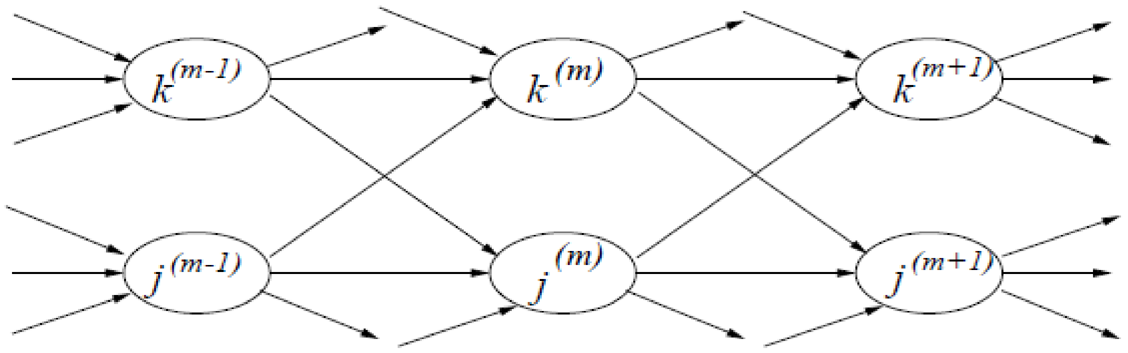
  L. R. Ford, Jr., *Network Flow Theory*,

  The Rand Corp., P-923, August 1956.

Bellman Ford Method – Successive Approximation

- Define

  $u_j^{(m)}$ = the length of a shortest path from the origin to *j*, subject to the condition that path contains no more than *m* arcs.



$$\begin{aligned} u_1^{(1)} &= 0, \\ u_j^{(1)} &= a_{1j}, \quad j \neq 1. \\ u_j^{(m+1)} &= \min\{u_j^{(m)},\ \min_{k \neq j}\{u_k^{(m)} + a_{kj}\}\}. \end{aligned} \tag{6.1}$$

Complexity Analysis

◇ It follows that approximately n^3 additions and n^3 comparisons are required overall, and the computation is clearly $O(n^3)$.

- $u_j^{(1)} \geq u_j^{(2)} \geq u_j^{(3)} \geq \dots$
- $u_j^{(n-1)} = u_j$
- Complexity

```
(6.1b), $m = 1, \dots, n-2$ iterations
    For each $m$, $j = 1, \dots, n$ equations to be solved
        For each equation
            $n - 1$ additions
            $n - 1$ comparisons
```

Total $\approx O(n^3)$

Improvement in Efficiency

- If not an order of reduction, how about a factor of reduction in complexity?

- Yen’s algorithm

  J. Y. Yen, *An Algorithm for Finding Shortest Routes from all Source Nodes to a Given Destination in General Network*," *Quart. Appl. Math.*, 27 (1970) 526-530.

Yen's Algorithm

- Observation
  - ◇ Suppose we call an arc (i, j) *upward* if $i < j$ and *downward* if $i > j$. A path is said to contain a *change in direction* whenever a downward arc is followed by an upward arc, or vice-versa. Note that because node 1 is the first node of any path, the first arc is upward and the first change in direction (if any) must be up to down.

Yen's Algorithm

Let

$u_j^{(m)} =$ the length of a shortest path from the origin to node j, subject to the condition that it contains no more than $m-1$ changes in direction.

$$\left.\begin{aligned}
&u_1^{(1)} = 0 \\
&u_j^{(1)} = a_{1j},\ \ j \neq 1 \text{ (by definition)}, \\
&u_j^{(m+1)} = \min\{u_j^{(m)}, \min_{k<j}\{u_k^{(m)} + a_{kj}\}\},\ \ m \text{ even} \\
&\text{and} \\
&u_j^{(m+1)} = \min\{u_j^{(m)}, \min_{k>j}\{u_k^{(m)} + a_{kj}\}\},\ \ m \text{ odd}
\end{aligned}\right\} \quad (7.1)$$

Complexity Reduction

- The work of the computation is reduced by a factor of approximately two.
- Yen has pointed out that the computation can be reduced even further to approximately $n^3/4.$

Linear Programming Relaxations

- Observations

  - ◇ Each of equations (3.1),

$$u_1 = 0,$$
$$u_j = \min_{k \neq j}\{u_k + a_{kj}\}, \quad (8.1)$$

implies a system of $n - 1$ inequalities, for fixed j and for $k = 1, 2, \ldots, j-1, j+1, \ldots, n$:

$$u_j \leq u_k + a_{kj}.$$

LP Problem

- This suggests the following linear programming problem: (n varibles, 1 equation, $(n-1)^2$ inequalities)

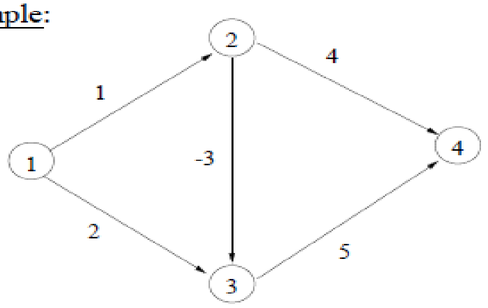
$$\left.\begin{array}{ll} \text{maximize} & u_1 + u_2 + \cdots + u_n \\ \text{subject to} & \\ & u_1 = 0 \\ \text{and, for} & i = 1, 2, \ldots, n;\ j = 2, 3, \ldots, n;\ i \neq j, \\ & u_j - u_i \leq a_{ij}. \end{array}\right\} \quad (8.3)$$

LP Dual Problem

- The dual of (8.3) is a minimum cost flow problem, and that the dual variable identified with the inequality constraints of (8.3) have a natural and intuitive interpretations in terms of arc flows.

We state, without proof or justification, that the basic variables of a basic feasible solution to the dual problem are identified with the arcs of a directed spanning tree rooted from the origin of the network.

Example:



LP Formulation:

$$\begin{array}{llllll}
\max & & u_2 & +u_3 & +u_4 & \text{(Shortest Paths Problem)} \\
\text{s.t.} & u_1 & & & & = 0 \\
(P) & -u_1 & +u_2 & & & \leq 1 \\
& -u_1 & & +u_3 & & \leq 2 \\
& & -u_2 & +u_3 & & \leq -3 \\
& & -u_2 & & +u_4 & \leq 4 \\
& & & -u_3 & +u_4 & \leq 5
\end{array}$$

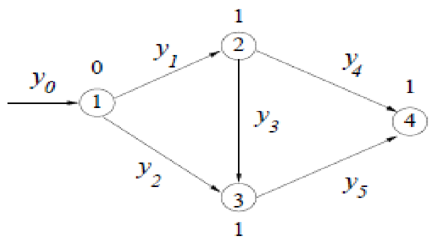
$$\begin{array}{llllllll}
\min & y_1 & +2y_2 & -3y_3 & +4y_4 & +5y_5 & \text{(Min} - \text{cost Flow Problem} \\
\text{s.t.} & y_0 - y_1 & -y_2 & & & & = 0 \\
(D) & y_1 & & -y_3 & -y_4 & & = 1 \\
& & y_2 & +y_3 & & -y_5 & = 1 \\
& & & & y_4 & +y_5 & = 1 \\
& y_1, & y_2, & y_3, & y_4, & y_5 & \geq 0
\end{array}$$

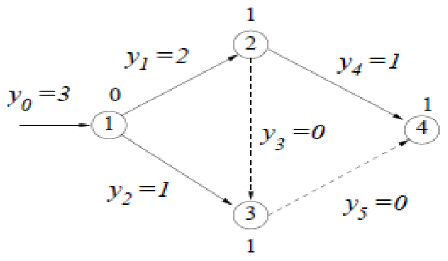
Optimality Conditions

(u_1, u_2, u_3, u_4)	primal feasible
$(y_1, y_2, y_3, y_4, y_5)$	dual feasible

Complementarities:

$$
\begin{aligned}
-(-u_1 + u_2 - 1)y_1 &= 0 \\
-(-u_1 + u_3 - 2)y_2 &= 0 \\
-(-u_2 + u_3 + 3)y_3 &= 0 \\
-(-u_2 + u_4 - 4)y_4 &= 0 \\
-(-u_3 + u_4 - 5)y_5 &= 0
\end{aligned}
$$





dual feasible ?

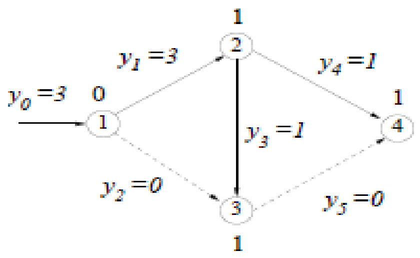
cost = 8

optimal ?

$$\begin{array}{rrrrl} -u_1 & +u_2 & & & = 1 \\ -u_1 & & +u_3 & & = 2 \\ & -u_2 & & +u_4 & = 4 \end{array} \quad \Rightarrow \quad \begin{array}{l} u_1 = 0 \leftarrow \text{known} \\ u_2 = 1 \\ u_3 = 2 \\ u_4 = 5 \end{array}$$

Primal feasible?

$$\begin{array}{lll} u_1 & = 0 & \leftarrow y_0 \\ -u_1 + u_2 & = 1 & \leftarrow y_1 \\ -u_1 + u_3 & = 2 & \leftarrow y_2 \\ -u_2 + u_3 & = 1 > -3 & \leftarrow y_3 \\ -u_2 + u_4 & = 4 & \leftarrow y_4 \\ -u_3 + u_4 & = 3 \leq 5 & \leftarrow y_5 \end{array}$$



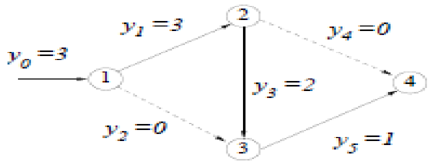
dual feasible ?

cost = 4

optimal ?

$$
\begin{array}{rrrrl}
-u_1 & +u_2 & & & = 1 \\
 & -u_2 & +u_3 & & = -3 \\
 & -u_2 & & +u_4 & = 4
\end{array}
\quad \Rightarrow \quad
\begin{array}{l}
u_1 = 0 \\
u_2 = 1 \\
u_3 = -2 \\
u_4 = 5 \\
\text{Primal feasible?}
\end{array}
$$

$$
\begin{array}{lllll}
u_1 & = 0 & & \leftarrow y_0 \\
-u_1 + u_2 & = 1 & & \leftarrow y_1 \\
-u_1 + u_3 & = -2 & \leq 2 & \leftarrow y_2 \\
-u_2 + u_3 & = -3 & & \leftarrow y_3 \\
-u_2 + u_4 & = 4 & & \leftarrow y_4 \\
-u_3 + u_4 & = 7 & > 5 & \leftarrow y_5
\end{array}
$$



dual feasible ?

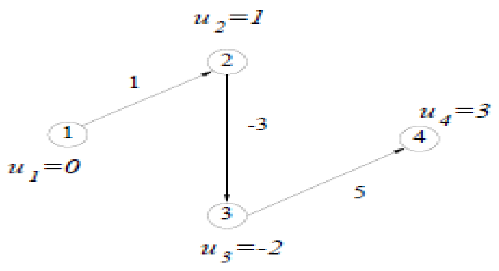
cost = 2

optimal ?

$$\begin{array}{rrrrl} -u_1 & +u_2 & & & = 1 \\ & -u_2 & +u_3 & & = -3 \\ & & -u_3 & +u_4 & = 5 \end{array} \Rightarrow \begin{array}{l} u_1 = 0 \\ u_2 = 1 \\ u_3 = -2 \\ u_4 = 3 \end{array}$$

Primal feasible?

$$\begin{array}{rcrlcl} u_1 & = & 0 & & & \\ -u_1 + u_2 & = & 1 & & & \\ -u_1 + u_3 & = & -2 & \le 2 & \Rightarrow & \max\; u_2 + u_3 + u_4 \\ -u_2 + u_3 & = & -3 & & & = 2 \\ -u_2 + u_4 & = & 2 & \le 4 & & \\ -u_3 + u_4 & = & 5 & & & \end{array}$$



Optimal !!

LP Relaxations Procedure

Step 0 *(Start)*

Set u_1=0. Set u_j, for $j = 2, 3, \ldots, n$, to any sufficiently large value (at least as large as the optimal value).

Step1 *(Test Inequalities)*

If all inequalities of (8.3) are satisfied, stop; the solution is optimal. Otherwise, find i, j such that

$$u_j - u_i > a_{ij}.$$

Step 2 *(Relaxation)*

Set $u_j = u_i + a_{ij}$.

Return to Step 1. //

Shortest Paths between All Node Pairs

- Instead of computing the shortest paths from an origin to each of the n - 1 nodes, we seek to compute shortest paths from each of the n nodes to each of the other n - 1 nodes, i.e., $n(n - 1)$ shortest paths in all.

- Assumption: No negative cycle in existence.

- Naïve approach:
  - apply Bellman-Ford method starting from each node
  - complexity no more tha $O(n^4)$.

Solution Method

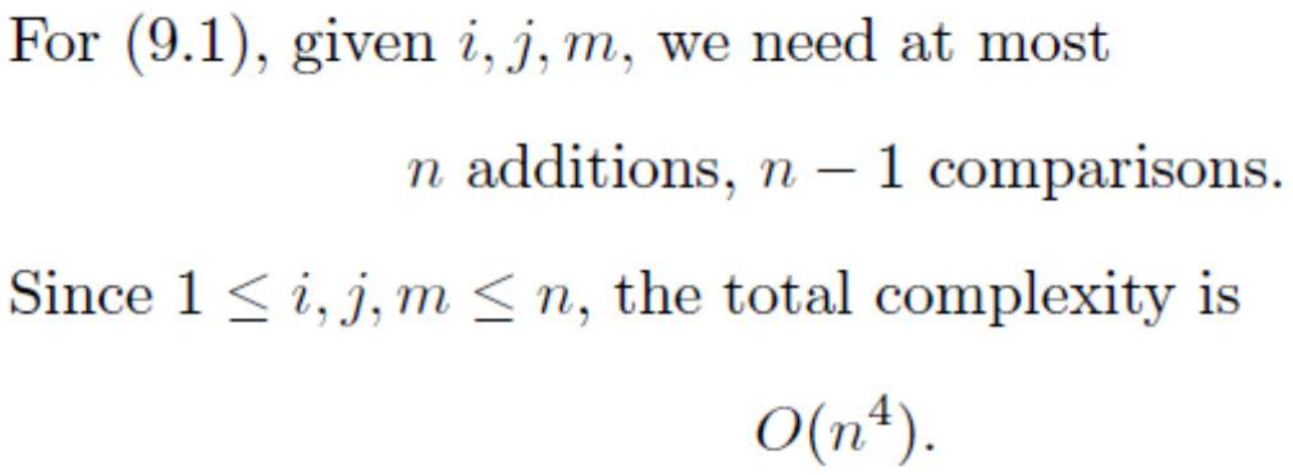
Let

$$
\begin{aligned}
u_{ij} &= \text{the length of a shortest path from } i \text{ to } j. \\
u_{ij}^{(m)} &= \text{the length of a shortest path from } i \text{ to } j, \\
&\quad \text{subject to the condition that path contains} \\
&\quad \text{no more than } m \text{ arcs.}
\end{aligned}
$$

If $a_{ii} = 0$,

$$
\left.
\begin{aligned}
u_{ii}^{(0)} &= 0, \\
u_{ij}^{(0)} &= +\infty, \ (i \neq j). \\
u_{ij}^{(m+1)} &= \min_k \{ u_{ik}^{(m)} + a_{kj} \}.
\end{aligned}
\right\} (9.1)
$$

Complexity Analysis



For (9.1), given i, j, m, we need at most

$$n \text{ additions}, \ n-1 \text{ comparisons.}$$

Since $1 \leq i, j, m \leq n$, the total complexity is

$$O(n^4).$$

- Can we do better?

Matrix Multiplication Method

- An efficient implementation of the solution method by using matrix multiplication-like mechanism.
- Recall the regular matrix multiplication

$$P = (p_{ij}) = AB$$

where

$$p_{ij} = \sum_k a_{ik} b_{kj}.$$

Define a new type of matrix multiplication "$\otimes$" as follows:

$$P = (p_{ij}) = A \otimes B$$

where

$$p_{ij} = \min_k \{a_{ik} + b_{kj}\}.$$

Let $A = (a_{ij})$ be the matrix of arc lengths.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}.$$

Now let $U^{(m)}$ be the matrix of mth order approximations, i.e., $U^{(m)} = (u_{ij}^{(m)})$. Note that

$$U^{(0)} = \begin{bmatrix} 0 & \infty & \cdots & \infty & \infty \\ \infty & 0 & \cdots & \infty & \infty \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \infty & \infty & \cdots & 0 & \infty \\ \infty & \infty & \cdots & \infty & 0 \end{bmatrix}$$

Matrix Multiplication Procedure

Then we see that $\otimes$ is associative, and

$$\begin{aligned}
U^{(1)} &= U^{(0)} \otimes A = A \\
U^{(2)} &= U^{(1)} \otimes A = (U^{(0)} \otimes A) \otimes A) = A^{(2)} \\
&\vdots \\
U^{(n-1)} &= U^{(n-2)} \otimes A = (((U^{(0)} \otimes A) \otimes A) \cdots \otimes A) = A^{(n-1)}
\end{aligned}$$

Together, these facts mean that we can write

$$U^{(n-1)} = A^{n-1},$$

where by A^{n-1} we mean the $(n-1)$st power of the A matrix.

Complexity Analysis

- Like regular matrix multiplication, each time takes $O(n^3)$ elementary operations.

  Note that

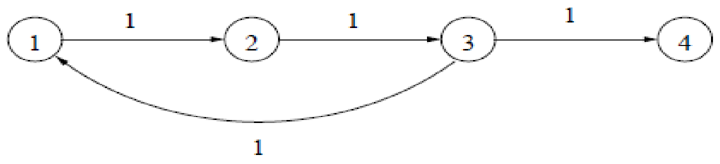
$$A^{2k} = A^{n-1} \text{ for any } 2^k \geq n-1.$$

  We need

$$\left.\begin{aligned} &A \\ &A^2 = A \otimes A \\ &A^4 = A^2 \otimes A^2 \\ &\quad\vdots \\ &A^{2k} = A^k \otimes A^k \end{aligned}\right\} \log_2 n \text{ matrix multiplications}$$

It follows that we have a computation which is $O(n^3 \log n)$ overall.

Example of Matrix Multiplication Method


$$\begin{tikzcd}
1 \arrow[r, "1"] & 2 \arrow[r, "1"] & 3 \arrow[r, "1"] \arrow[ll, bend left, "1"] & 4
\end{tikzcd}$$

$$A = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 1 & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}, \quad U^{(0)} = \begin{pmatrix} 0 & \infty & \infty & \infty \\ \infty & 0 & \infty & \infty \\ \infty & \infty & 0 & \infty \\ \infty & \infty & \infty & 0 \end{pmatrix},$$

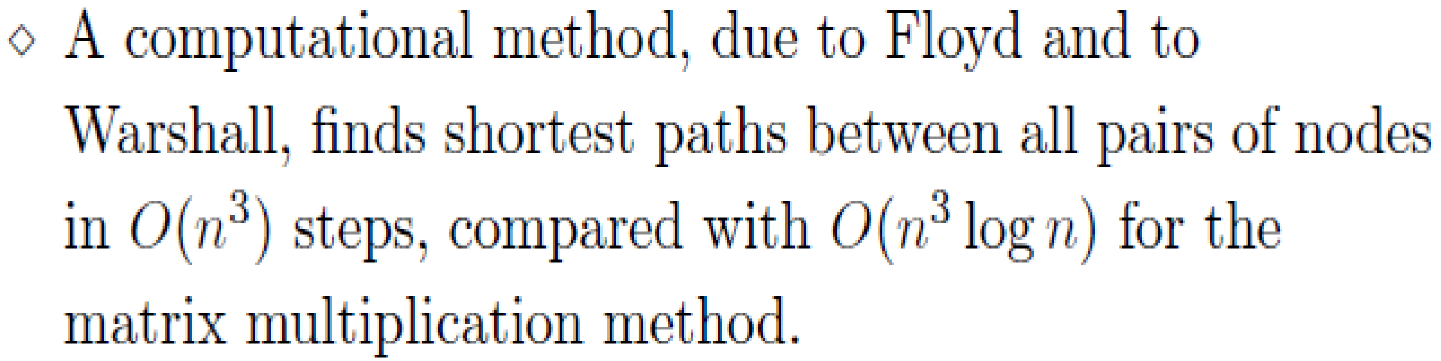
$$A^2 = \begin{pmatrix} 0 & 1 & 2 & \infty \\ 2 & 0 & 1 & 2 \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix},$$

$$A^4 = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 2 \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}.$$

Shortest Paths between All Node Pairs

- Can we do better than the Matrix Multiplication method?

- R. W. Floyd, *Algorithm 97, Shortest Path*, Comm. ACM, 5 (1962) 345.
- S. Warshall, *A Theorem on Boolean Matrices*, J. ACM, 9 (1962) 11-12.



◇ A computational method, due to Floyd and to Warshall, finds shortest paths between all pairs of nodes in $O(n^3)$ steps, compared with $O(n^3 \log n)$ for the matrix multiplication method.

Floyd-Warshall Method

Redefine

$u_{ij}^{(m)} =$ the length of a shortest path from i to j, subject to the condition that path does not pass through nodes $m, m+1, \ldots, n$ (i and j excepted).

A shortest path from node i to node j which does not pass through nodes $m+1, m+2, \ldots, n$ either (a) does not pass through node m, in which case

$$u_{ij}^{(m+1)} = u_{ij}^{(m)}$$

or (b) does pass through node m, in which case

$$u_{ij}^{(m+1)} = u_{im}^{(m)} + u_{mj}^{(m)}.$$

Floyd-Warshall Method

Thus we have

$$\left.\begin{aligned} u_{ij}^{(1)} &= a_{ij}, \\ \text{and} \qquad & \\ u_{ij}^{(m+1)} &= \min\{u_{ij}^{(m)},\ u_{im}^{(m)} + u_{mj}^{(m)}\}, \end{aligned}\right\} \quad (10.1)$$

and, clearly, $u_{ij}^{(n+1)} = u_{ij}$, the length of a shortest path from i to j.

Complexity Analysis

For (10.1), given any i, j, m, we need

$$\text{one addition, one comparison.}$$

Since $1 \leq i, j, m \leq n$, the total complexity is

$$O(n^3).$$

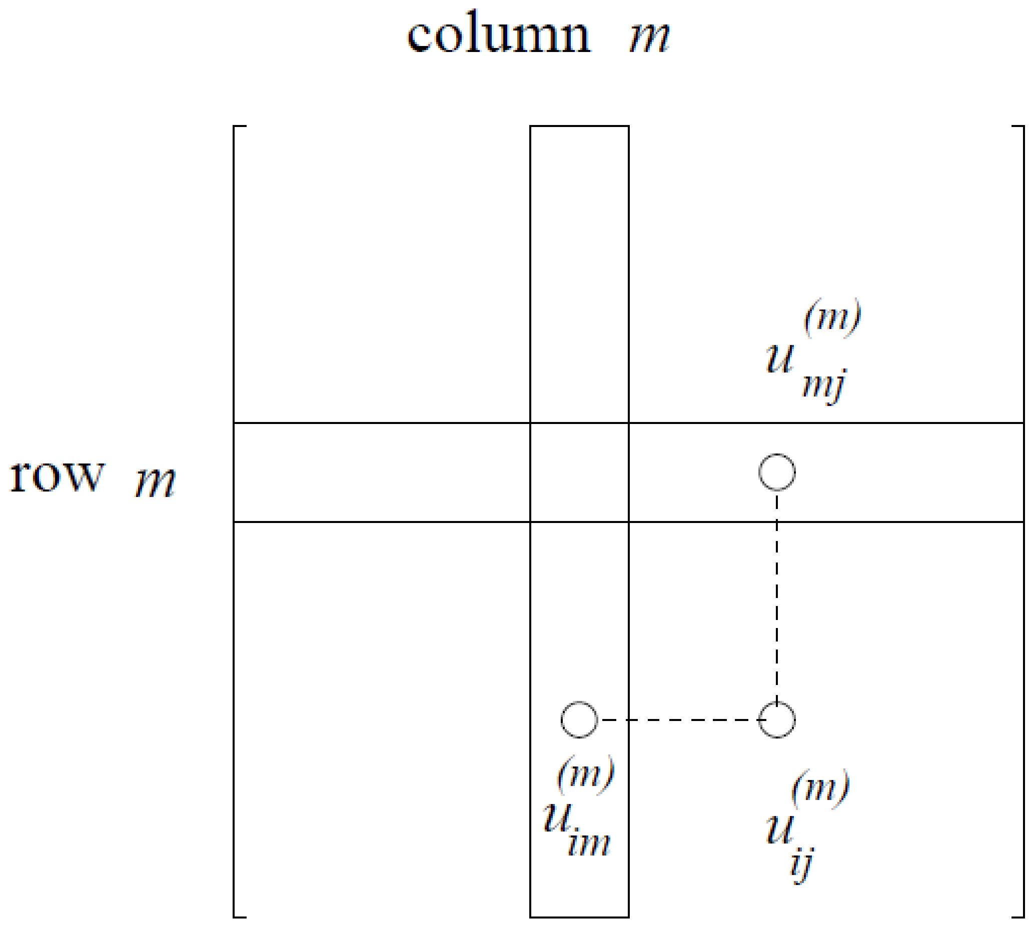
Floyd-Warshall Matrix Computation

$$U^{(1)} = A = \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix},$$

How to update?

$$U^{(m+1)} \longleftarrow U^{(m)}$$

Floyd-Warshall Matrix Computation



Example

$$U^{(1)} = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 1 & \infty & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix} = A,$$

$$U^{(2)} = \begin{pmatrix} 0 & 1 & \infty & \infty \\ \infty & 0 & 1 & \infty \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}, \quad U^{(3)} = \begin{pmatrix} 0 & 1 & 2 & \infty \\ \infty & 0 & 1 & \infty \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix},$$

$$U^{(4)} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 2 \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}, \quad U^{(5)} = \begin{pmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 2 \\ 1 & 2 & 0 & 1 \\ \infty & \infty & \infty & 0 \end{pmatrix}.$$

Beyond the Method

- We can derive some further insight into Floyd-Warshall method from the theorem below.

Theorem 10.1 An $n \times n$ matrix $U = (u_{ij})$ is a matrix of shortest path lengths if and only if

$$\left.\begin{aligned} &u_{ii} = 0, \\ &u_{ij} \le u_{ik} + u_{kj}, \text{ for all } i, j, k. \end{aligned}\right\} \quad (10.2)$$

Detection of Negative Cycles

Theorem 11.1 Let the network has a path from node 1 to each of the other nodes. Then the network contains a negative cycle if and only if, in (6.1) or (7.1), $u_j^{(n)} < u_j^{(n-1)}$, for at least one $j = 1, 2, \ldots, n$.

Theorem 11.2 The network contains a negative cycle if, in (6.1), $u_j^{(m+1)} < u_j^{(m)}$, for some $m = 1, 2, 3, \ldots, n-1$, and at least $n - m$ distinct nodes j.

Detection of Negative Cycles

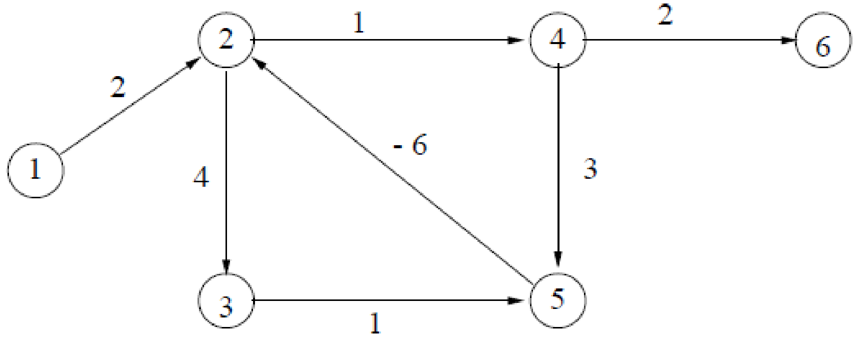
Theorem 11.3 The network contains a negative cycle if and only if, in (9.1) or (10.1), $u_{ii}^{(m)} < 0$, for some $i = 1, 2, 3, \ldots, n$, and some $m = 1, 2, 3, \ldots, n$.

Theorem 11.4 If the network does not contain a negative cycle, then the length of shortest cycle is given by

$$\min_i \{u_{ii}^{(n+1)}\},$$

where $u_{ii}^{(n+1)}$ is determined by (9.1) or (10.1), with $a_{ii} = +\infty$, for all i.

Example



$u_1^{(1)} = 0$	$u_1^{(2)} = 0$	$u_1^{(3)} = 0$	$u_1^{(4)} = 0$	$u_1^{(5)} = 0$	$u_1^{(6)} = 0$
$u_2^{(1)} = 2$	$u_2^{(2)} = 2$	$u_2^{(3)} = 2$	$u_2^{(4)} = 0$	$u_2^{(5)} = 0$	$u_2^{(6)} = 0$
$u_3^{(1)} = \infty$	$u_3^{(2)} = 6$	$u_3^{(3)} = 6$	$u_3^{(4)} = 6$	$u_3^{(5)} = 4$	$u_3^{(6)} = 4$
$u_4^{(1)} = \infty$	$u_4^{(2)} = 3$	$u_4^{(3)} = 3$	$u_4^{(4)} = 3$	$u_4^{(5)} = 1$	$u_4^{(6)} = 1$
$u_5^{(1)} = \infty$	$u_5^{(2)} = \infty$	$u_5^{(3)} = 6$	$u_5^{(4)} = 6$	$u_5^{(5)} = 6$	$u_5^{(6)} = 4$
$u_6^{(1)} = \infty$	$u_6^{(2)} = \infty$	$u_6^{(3)} = 5$	$u_6^{(4)} = 5$	$u_6^{(5)} = 5$	$u_6^{(6)} = 3$

M Shortest Paths

- Two paths are considered *distinct* if they do not visit precisely the same nodes in the same order.
- The method we shall describe, due to S. E. Dreyfus, computes the M shortest paths from an origin (node 1) to each of other $n-1$ nodes of the network, and does so in $O(Mn\log n)$ running time.

S. E. Dreyfus, “An Appraisal of Some Shortest-Path Algorithms,” *Operations Research*, **17** (1969) 395-412.

Dreyfus Method

Let

$$u_j^{[m]} = \text{ the length of the } m\text{th shortest path from the origin to } j,$$

and

$$\mu(k,j,m) = \text{ the number of paths in which } (k,j) \text{ is the final arc, in the set of 1st, 2nd,} \ldots, m\text{th shortest paths from node 1 to node } j.$$

By definition, $\mu(k,j,0) = 0$.

$$\mu(k,j,m+1) = \begin{cases} \mu(k,j,m)+1, \text{ if } (k,j) \text{ is the final arc in the} \\ (m+1)\text{st shortest path from the origin to } j \\ \mu(k,j,m), \text{ otherwise} \end{cases}$$

Dreyfus Method

- The $(m+1)$st shortest path from 1 to j has some final arc (k,j). The length of this path from the origin to k must be $u_k^{[\mu(k,j,m)+1]}$. Accordingly, by minimizing over all possible choices of k, we obtain

$$u_j^{[m+1]} = \min_k \{u_k^{[\mu(k,j,m)+1]} + a_{kj}\} \quad (14.1)$$

with the initial condition

$$u_1^{[1]} = 0.$$

- Equations (14.1) are clearly equivalent to Bellman's equations (3.1) for the case $m = 0$.

Question

- Is there any implicit functional relation in (14.1)?

(Such as $u_j^{[m+1]}$ depends on $u_i^{[m+1]}$

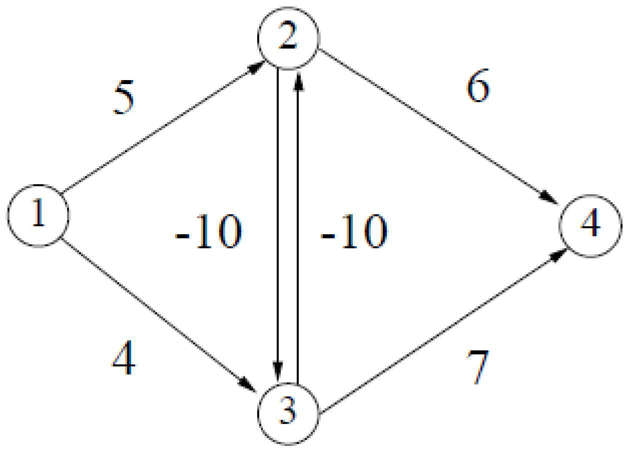
and $u_i^{[m+1]}$ depends on $u_j^{[m+1]}$.)

Other Shortest Paths of Interests

- M Shortest Paths without Repeated Nodes.
  - Every such shortest path has at most $n-1$ arcs (from node 1 to node j).
  - At least one arc in the ith shortest path (from node 1 to node j) will not appear in the $(i+1)$th shortest path (from node 1 to node j).
  - Cut one arc in the ith shortest path at a time, and then generate a shortest path from node 1 to node j.
  - Compare all such shortest paths and find the $(i+1)$th shortest path.
  - Complexity
$$O(Mn^3).$$

Other Shortest Paths of Interests

- Arc-disjoint paths



$1 \to 2 \to 3 \to 4$	2
$1 \to 2 \to 4$	11
$1 \to 3 \to 4$	11
$1 \to 3 \to 2 \to 4$	0

Other Shortest Paths of Interests

Problem: 1. Find a pair of arc-disjoint paths such that the total length is minimized.

2. Find k ($k > 2$) arc-disjoint paths such that the total length is minimized.

- Node-disjoint paths