IE 400
Principles of Engineering Management

Network Optimization

Lecture Notes 8

Network Models

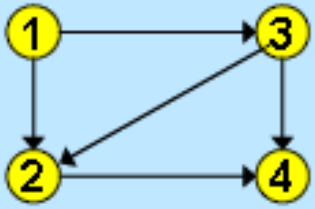
- Optimization models that exhibit a very special structure.
- For special cases, this structure to dramatically reduce computational complexity (running time).
- First widespread application of LP to problems of industrial logistics
- Addresses huge number of diverse applications

Notation and Terminology

Note: Network terminology is not (and never will be) standardized. The same concept may be denoted in many different ways.

Called:

- NETWORK
- directed graph
- digraph
- graph



Network **G = (N, A)**

Node set **N = {1, 2, 3, 4}**

Arc Set **{(1,2), (1,3), (3,2), (3,4), (2,4)}**

Also Seen

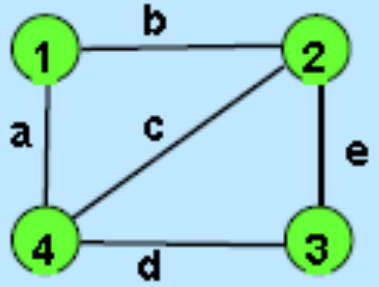
Graph **G = (V,E)**

Vertex set **V = {1,2,3,4}**

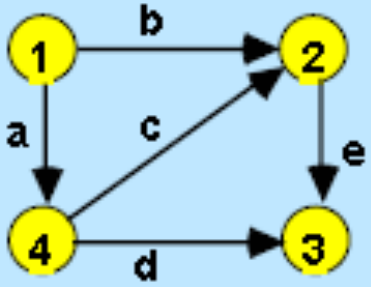
Edge set:

E={1-2,1-3,3-2,3-4,2-4}

Directed and Undirected Networks



An Undirected Graph



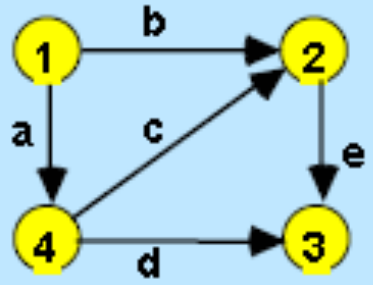
A Directed Graph

- **Networks are used to transport commodities**
  - physical goods (products, liquids)
  - communication
  - electricity, etc.
- **The field of Network Optimization concerns optimization problems on networks**

Networks are Everywhere

- Physical Networks
  - Road Networks
  - Railway Networks
  - Airline traffic Networks
  - Electrical networks, e.g., the power grid
- Abstract networks
  - organizational charts
  - precedence relationships in projects
- Others?

The Adjacency Matrix (for directed graphs)



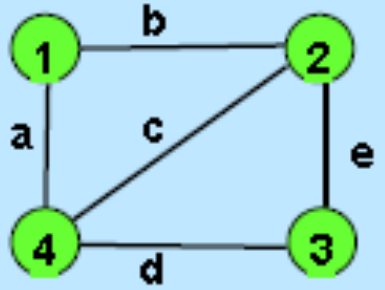
A Directed Graph

	1	2	3	4
1	0	1	0	1
2	0	0	1	0
3	0	0	0	0
4	0	1	1	0

- Have a row for each node
- Have a column for each node
- Put a 1 in row i- column j if (i, j) is an arc

What would happen if (4, 2) became (2, 4)?

The Adjacency Matrix (for undirected graphs)



An Undirected Graph

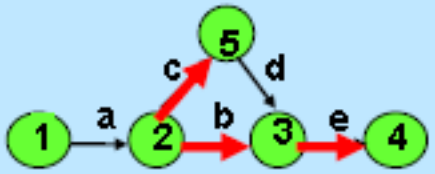
	1	2	3	4	degree
1	0	1	0	1	2
2	1	0	1	1	3
3	0	1	0	1	2
4	1	1	1	0	3

- Have a row for each node
- Have a column for each node
- Put a 1 in row i- column j if (i, j) is an arc

The ***degree*** of a node is the number of incident arcs

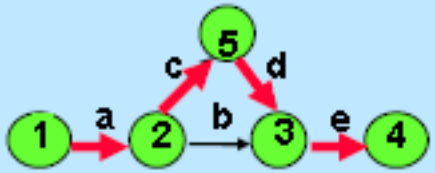
Path: **Example: 5, 2, 3, 4.**
(or 5, c, 2, b, 3, e, 4)

- No node is repeated.
- Directions are ignored.



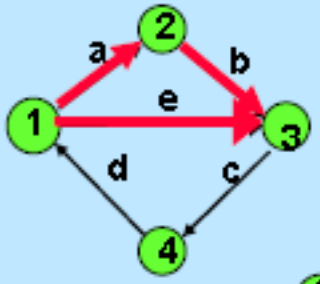
Directed Path **. Example: 1, 2, 5, 3, 4**
(or 1, a, 2, c, 5, d, 3, e, 4)

- No node is repeated.
- Directions are important.



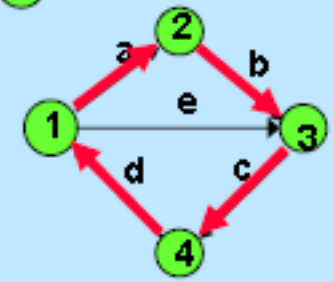
Cycle (or circuit or loop)
1, 2, 3, 1. (or 1, a, 2, b, 3, e)

- A path with 2 or more nodes, except that the first node is the last node.
- Directions are ignored.

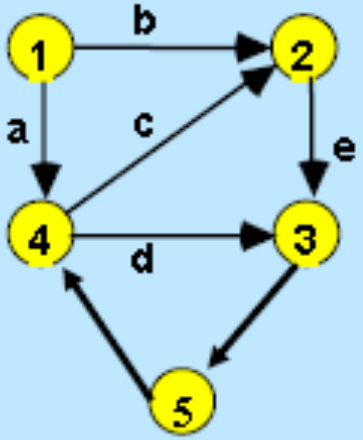


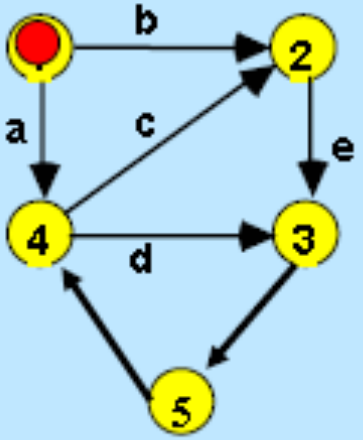
Directed Cycle: (1, 2, 3, 4, 1) or
1, a, 2, b, 3, c, 4, d, 1

- No node is repeated, except that the first node is the last node.
- Directions are important.



Walks





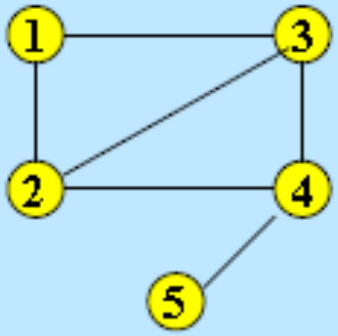
Walks are paths that can repeat nodes and arcs

Example of a ***directed walk***: 1-2-3-5-4-2-3-5

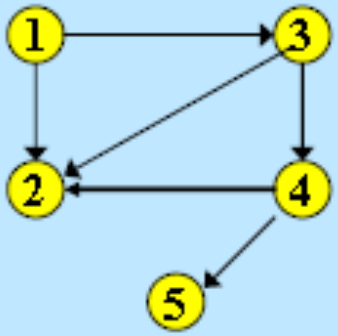
A walk is ***closed*** if its first and last nodes are the same

A closed walk is a cycle except that it can repeat nodes and arcs.

More terminology



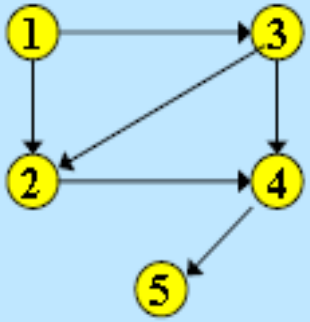
An undirected network is ***connected*** if every node can be reached from every other node by a path



A directed network is ***connected*** if it's undirected version is connected.

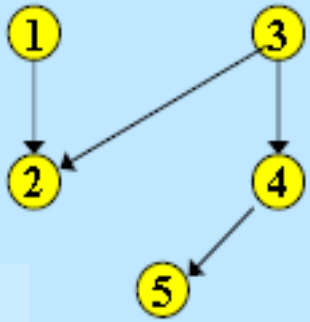
This directed graph is connected, even though there is no directed path between 2 and 5.

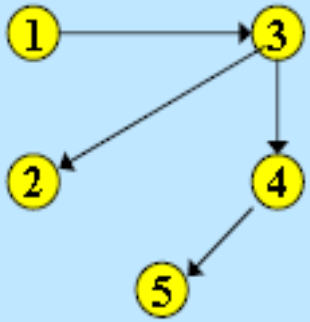
More Definitions

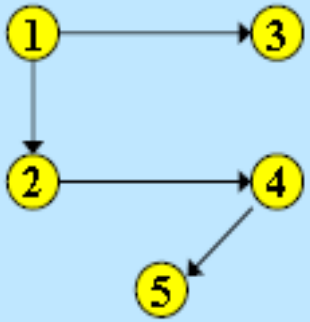


A network is ***connected*** if every node can be reached from every other node by a path

A ***spanning tree*** is a connected subset of a network including all nodes, but containing no cycles.



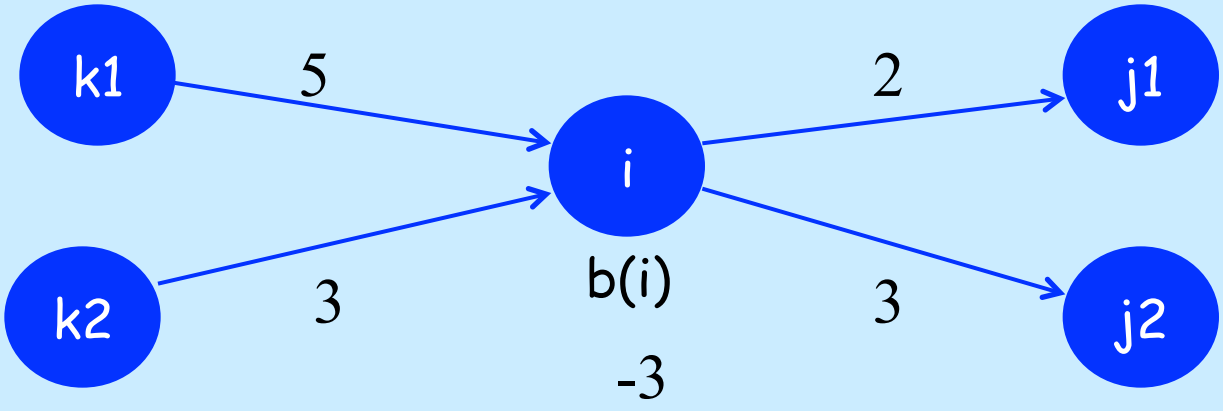




Flows and Paths

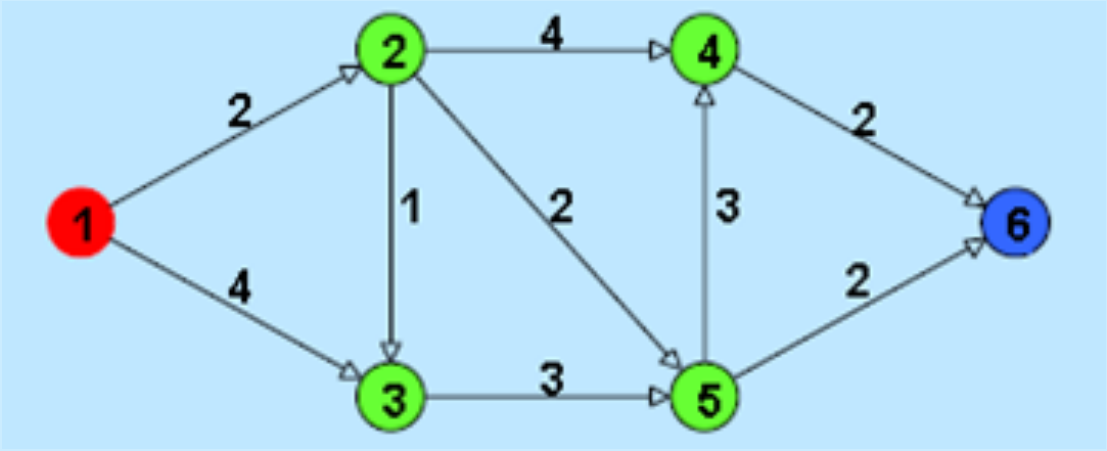
- Arc flows: an arc flow x is a vector x satisfying
  - $b(i) = \Sigma_j \; x_{ij} \; - \Sigma_k \; x_{ki}$
  - x_{ij} is the amount of flow from node i to node j

$$x_{i,j1}+x_{i,j2}-x_{k1,i}-x_{k2,i}=b(i)$$



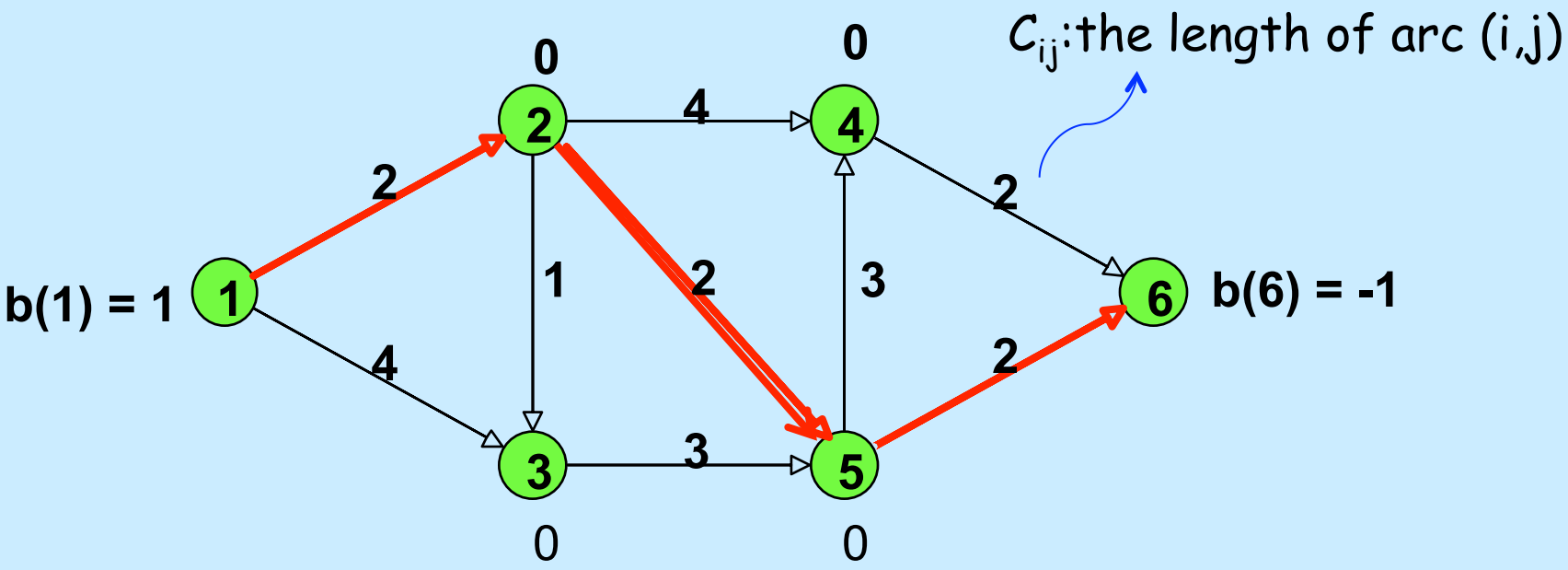
Shortest Paths

- Single Pair Shortest Path Problem
- Single Source Shortest Path Problem
- All-Pairs Shortest Path Problem



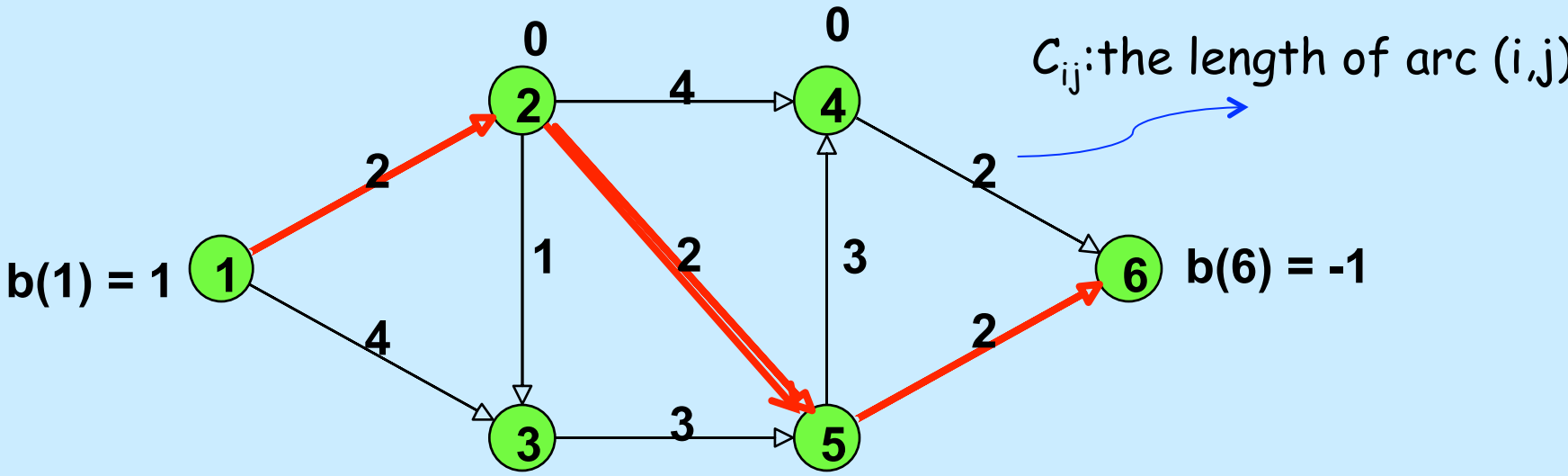
Length of (directed) path=total cost of arcs on this path

Find the shortest path from source node 1 to sink node 6 (single-pair shortesth path problem)



- Think of sending one unit of flow from 1 to 6
- This transformation works so long as there are no negative cost cycles in G.
- What if there are negative cost cycles?

LP Formulation of the Shortest Path Problem



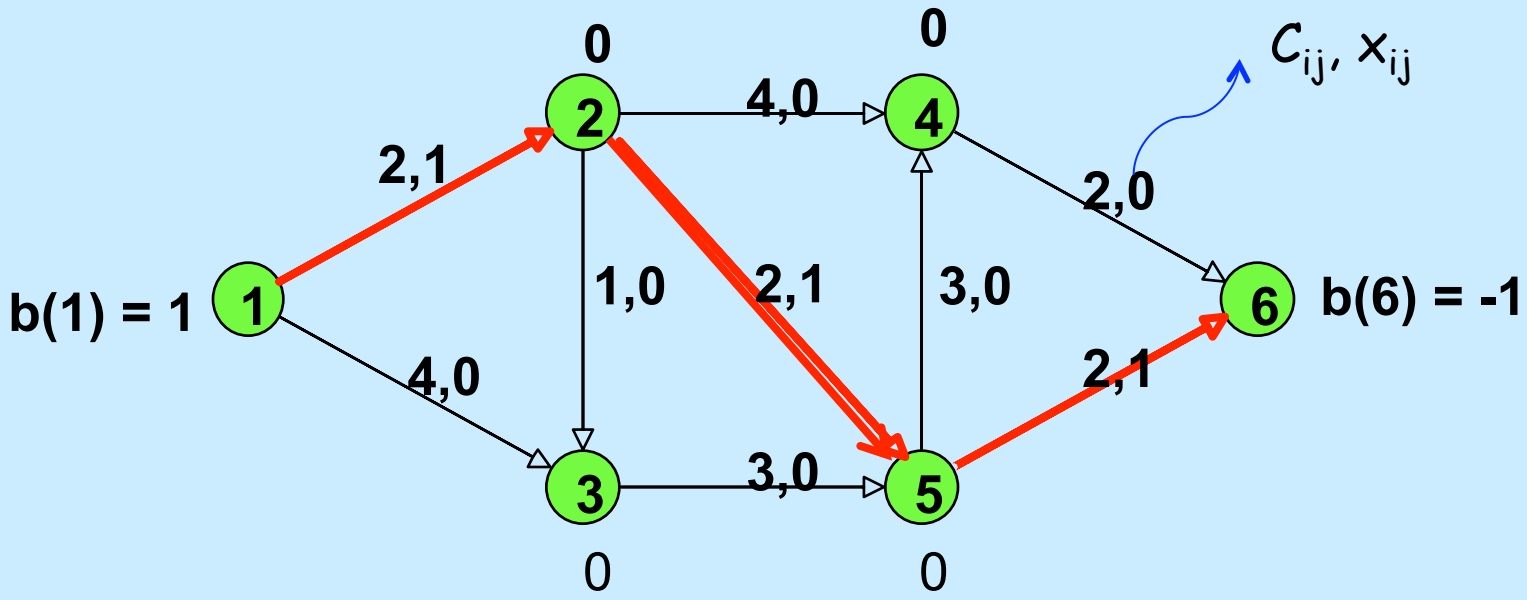
Find the shortest path from node 1 to node 6

$$Min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} 1 & i = s \\ 0 & i \neq s,t \\ -1 & i = t \end{cases}$$

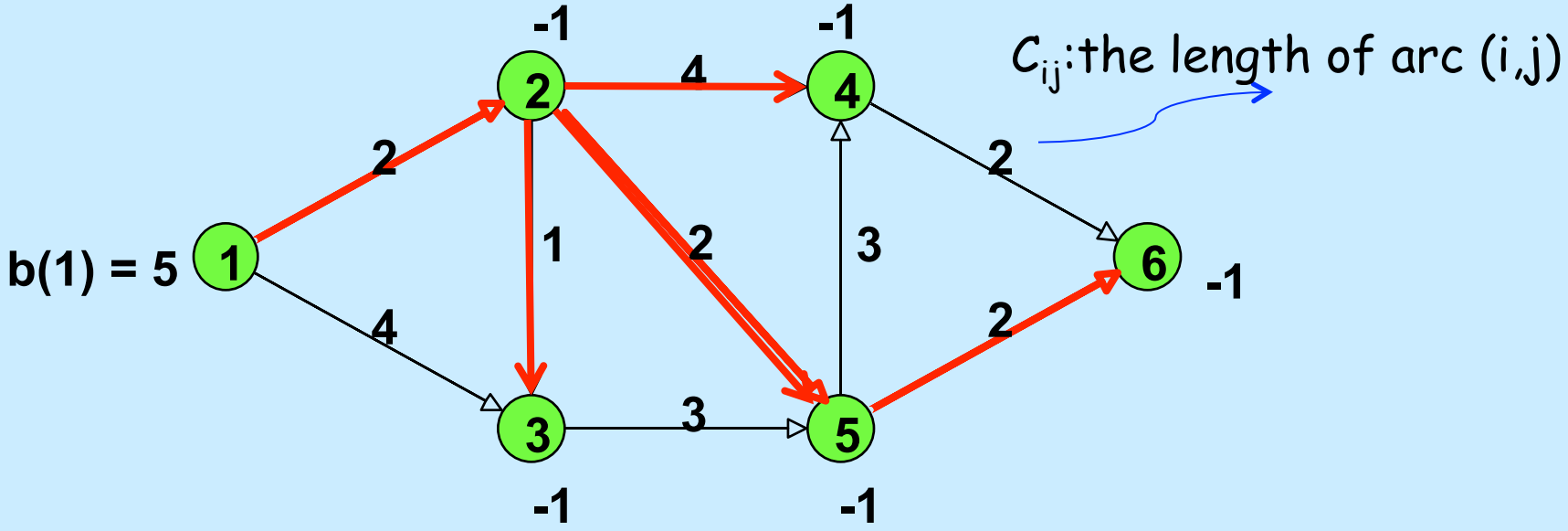
$$x_{ij} \geq 0, \ integer \qquad (i,j) \in A$$

Find the shortest path from source node 1 to sink node 6 (single-pair shortesth path problem)



- Optimal flow is to send one unit of flow along 1-2-5-6.
  - X12=1;X25=1, X56=1, X13=X23=X24=X35=X45=X46=0
  - Obj.fcn. Value=2.1+2.1+2.1=6
- C_{ij}: the length of arc (i,j)
  x_{ij}: the amount of flow on arc (i,j)

Find the shortest path from source node 1 to all other nodes (single-source shortesth path problem)



- Each node except the source node is treated as a sink node
- Assume that a source of n-1 units is at node 1 and a demand of 1 unit at all other nodes

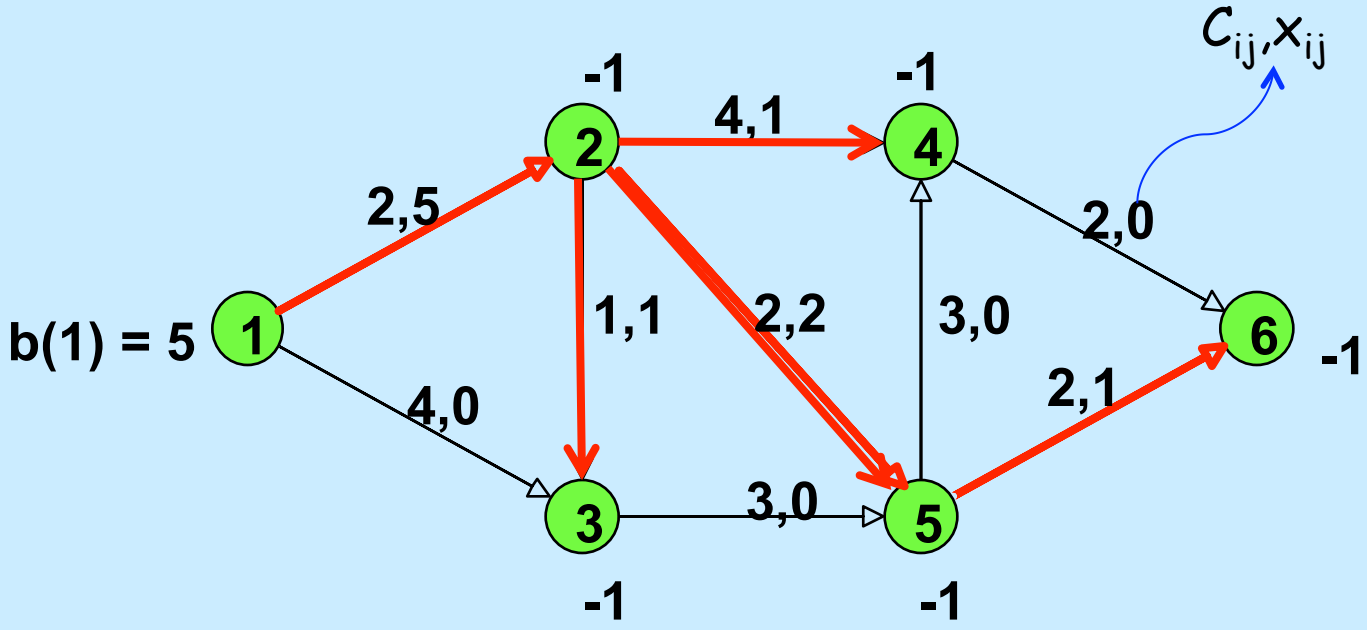
LP Formulation of the Shortest Path Problem

$$Min \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{j:(i,j)\in A} x_{ij} - \sum_{j:(j,i)\in A} x_{ji} = \begin{cases} n-1 & i = s \\ -1 & i \neq s \end{cases}$$

$$x_{ij} \geq 0, integer \qquad (i,j) \in A$$

Find the shortest path from source node 1 to all other nodes (single-source shortesth path problem)



Obj.fcn value=2.5+2.2+1.1+4.1+2.1=21

Integrality Property

- If we relax the integrality requirement and solve as an LP, we get an integral solution and the arcs with positive x_{ij} values give a shortest path tree rooted at s.
- The constraint matrix has a special property called "total unimodularity". Any extreme point is an integer vector and hence the optimal solution is integral.
- In general, network flow problem have this special "integrality property".

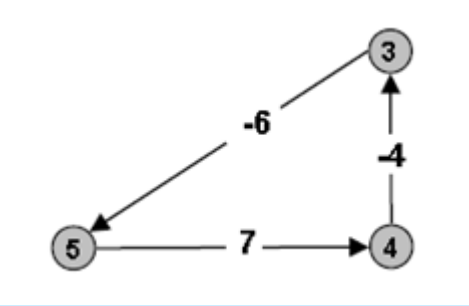
Total Unimodularity

- A matrix A is called *totally unimodular* if each of its subdeterminants equals 0, 1, or -1. (In particular, each entry of A is 0, 1, or -1.)

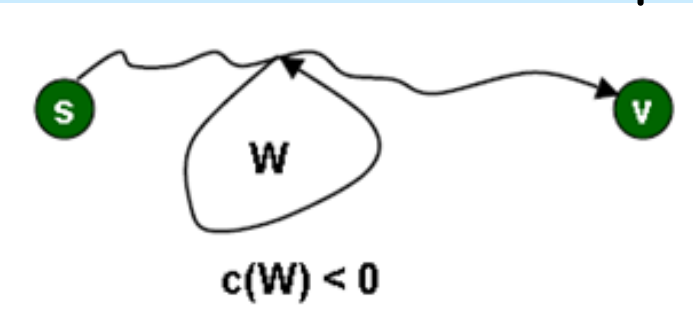
- If A is totally unimodular and b is integer valued, then the polyhedron $P=\{x \mid Ax \leq b\}$ is integral.

Shortest Paths with Negative Directed Cost Cycles

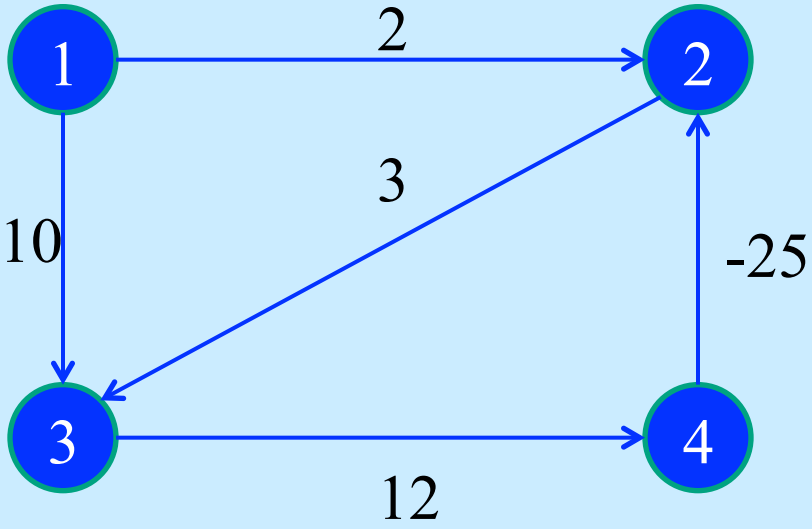
- A directed cycle is a path that begins and ends at the same node and a negative directed cycle is a directed cycle of negative total length



- If some path from s to v contains a negative directed cost cycle, there does not exist a shortest s-v path, otherwise, there exists one that is simple



Shortest Paths with Negative Cost Cycles

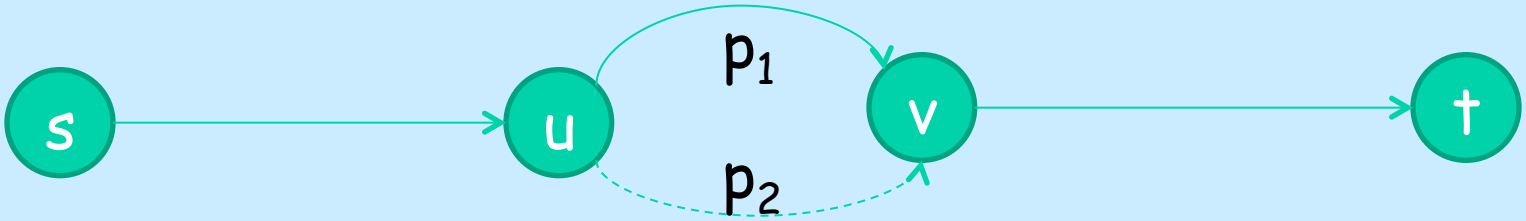


- Shortest path from 1 to 3: 1-2-3 with length 5
- Shortest path from 1 to 2: 1-3-4-2 with length -3
- The shortest path from 1 to 3 does not use the shortest subpath from node 1 to node 2 (it should have been 1-3-4-2-3, which is not a path)

Shortest path problems in the presence of negative directed cycles are much less tractable!

Principle of Optimality

In a network with no negative directed cycles, if P is a shortest path from node s to node t and if it goes through nodes u and v, then the subpath of P from u to v is a shortest path from u to v.



If the subpath p_2 is shorter than the subpath p_1 from u to v, then the shortest path from s to v cannot use the subpath p_1 from u to v.

Dynamic Programming Approach to Shortest Path Problems

- Divide and conquer method
- Dynamic programming exploits the fact that it is sometimes easier to solve one optimization problem by taking on an entire family
- If strong relationships can be found among optimal solutions to the various members of the family, attacking all together can be the most efficient way of solving the one(s) we really care about

Dynamic Programming Approach to Shortest Path Problems

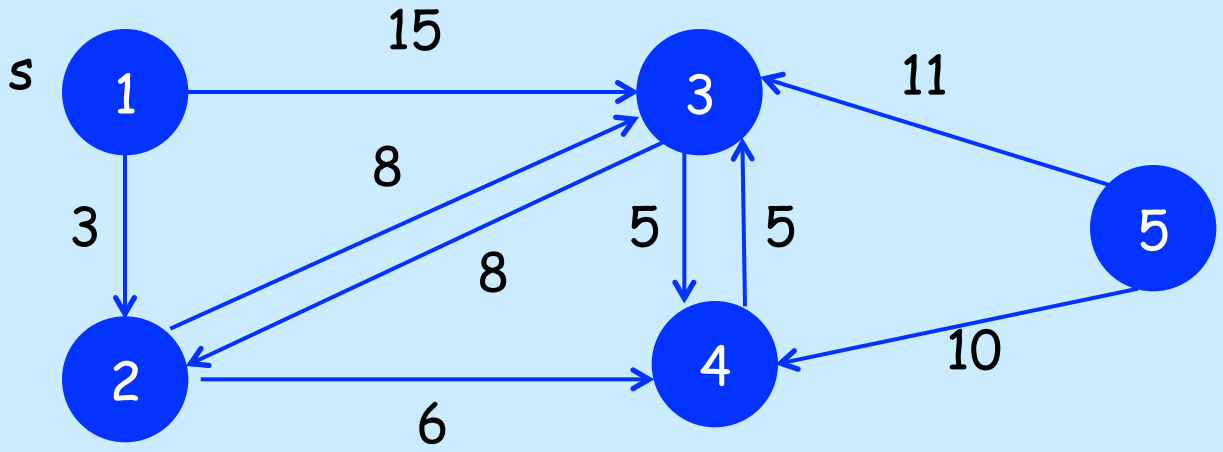
- Shortest path algorithms exploit relationships among optimal paths (and path lengths) for diferent pairs of nodes

Functional Notation
For Single Source Shortest Path Problem

- v[k] : the length of a shortest path from the source node to node k
  - If no path exists, v[k]=∞
- $X_{ij}[k] = \begin{cases} 1 & \text{if arc/edge (i,j) is part of the optimal path from the source node to node k} \\ 0 & \text{otherwise} \end{cases}$

Functional Notation

Example



Optimal paths from node 1 to all other nodes

$$
\begin{array}{ll}
v[1] = 0 & x_{1,2}[1] = x_{1,3}[1] = x_{2,3}[1] = x_{2,4}[1] = x_{3,4}[1] = 0 \\
v[2] = 3 & x_{1,2}[2] = 1, x_{1,3}[2] = x_{2,3}[2] = x_{2,4}[2] = x_{3,4}[2] = 0 \\
v[3] = 11 & x_{1,2}[3] = x_{2,3}[3] = 1, x_{1,3}[3] = x_{2,4}[3] = x_{3,4}[3] = 0 \\
v[4] = 9 & x_{1,2}[4] = x_{2,4}[4] = 1, x_{1,3}[4] = x_{2,3}[4] = x_{3,4}[4] = 0 \\
v[5] = +\infty & \text{(no path)}
\end{array}
$$

Functional Equations

- If optimal paths must have optimal subpaths, it follows that a shortest path can be constructed by extending known shortest paths
- Functional equations of a dynamic program encode the recursive connections among optimal solution values that are implied by a principle of optimality
  - In a graph with no negative directed cycles, optimal paths must have optimal subpaths

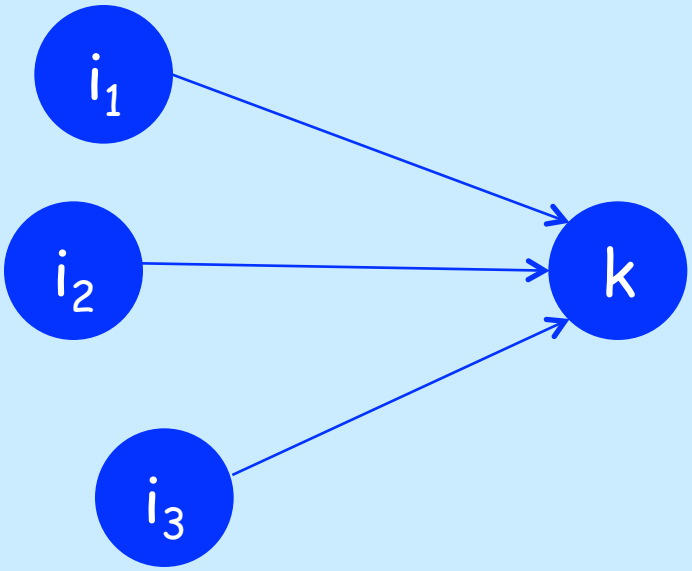
Functional Equations

- For single-source shortest path problem with s being the source

$v[s]=0$

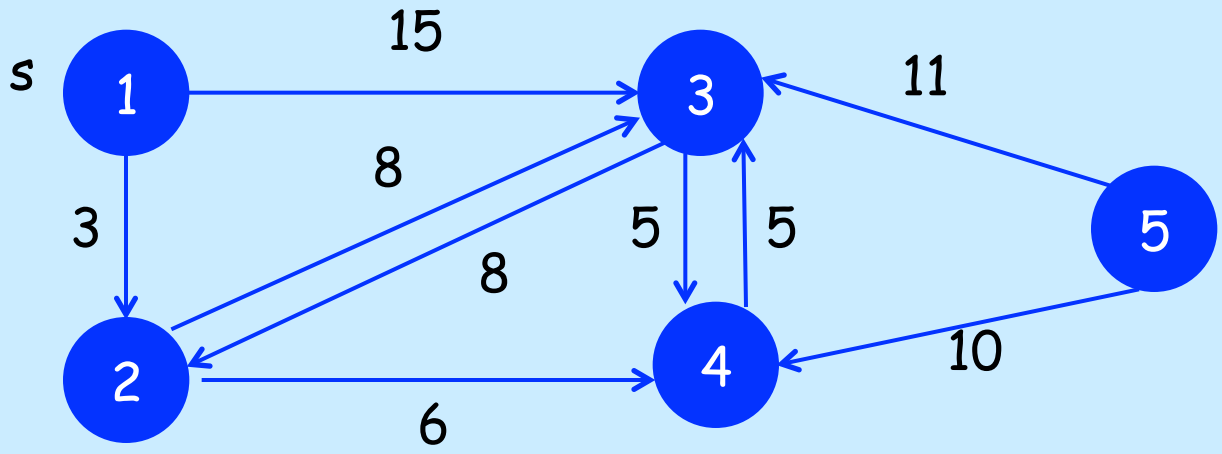
$v[k]=\min_{i:(i,k)\in A} \{v[i]+c_{ik}\}$ $\quad \forall k \in N\setminus\{s\}$

If there is no neighbor i leading to k, then the minimum is ∞



Functional Equations

Example



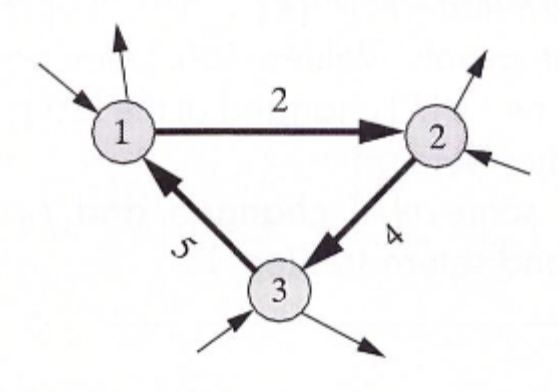
v[1]=0
v[2]=min {v[1]+3,v[3]+8}=3
v[3]=min{v[1]+15,v[2]+8,v[4]+5,v[4]+5,v[5]+11}=11
v[4]=min{v[2]+6,v[3]+5,v[5]+10]}=9
V[5]=∞

Optimal paths must extend optimal subpaths

Functional Equations

How to solve functional equations

- Nonlinear due to min operator
- Because each equation shows an expression for a single value v[k], possible just to evaluate those expressions (one-pass evaluation).
- However, directed cycles introduce circular dependencies in functional equations that preclude their solution by one-pass evaluations



$$v[1] = \min\{v[3] + 5, \ldots\}$$
$$v[2] = \min\{v[1] + 2, \ldots\}$$
$$v[3] = \min\{v[2] + 4, \ldots\}$$

Evaluating the expression for $v[2]$ requires $v[1]$; evaluating the expression for $v[3]$ requires $v[2]$; and evaluating the expression for $v[1]$ requires $v[3]$.

Bellman-Ford Algorithm

Repeated Evaluation Algorithm for Single Source Shortest Path Problem

- Each major iteration of the search evaluates the functional equations for each v[k] using results from the preceding iteration. The algorithm stops, when the results do not change.
  - $v^{(t)}[k]$: value of v[k] obtained on the t^{th} iteration
- To be able to know the shortest path at the termination of the algorithm, labels d[k] keep notes to allow us to recover the optimal paths
  - d[k] : node preceding k in the best known path from s to k

Bellman-Ford Algorithm

Step 0: Initialization. With s the source node, initialize optimal path lengths

$$v^{(0)}[k] \leftarrow \begin{cases} 0 & \text{if } k = s \\ +\infty & \text{otherwise} \end{cases}$$

and set iteration counter $t \leftarrow 1$.

Step 1: Evaluation. For each k evaluate

$$v^{(t)}[k] \leftarrow \min\{v^{(t-1)}[i] + c_{i,k} : (i, k) \text{ exists}\}$$

If $v^{(t)}[k] < v^{(t-1)}[k]$, also set $d[k] \leftarrow$ the number of a neighboring node i achieving the minimum $v^{(t)}[k]$.

Step 2: Stopping. Terminate if $v^{(t)}[k] = v^{(t-1)}[k]$ for every k, or if $t =$ the number of nodes in the graph. Values $v^{(t)}[k]$ then equal the required shortest path lengths unless some $v^{(t)}[k]$ changed at the last t, in which case the graph contains a negative dicycle.

Step 3: Advance. If some $v[k]$ changed and $t <$ the number of nodes, increment $t \leftarrow t + 1$ and return to Step 1.

The algorithm works on any graph with no negative directed cycles

Bellman-Ford Algorithm

Why does the algorithm work?

- Remark that in a graph with n nodes, a path can contain at most n-1 arcs
- At the t^{th} iteration, the length of optimal paths of at most t arcs should be correct
- Moreover, if we have a whole iteration without changing any node's value, no more changes can occur
- An example will be solved in the class