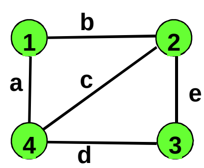
Computer Science

- Lecture notes
- CSAUTFALL2010

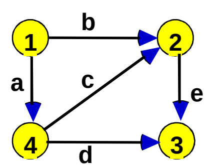
- Use by ONLY Group Members

Notation and Terminology

Network terminology as used in AMO.



An Undirected Graph or Undirected Network



A Directed Graph or Directed Network

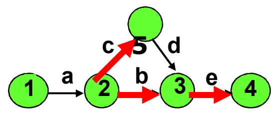
Network **G = (N, A)**

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

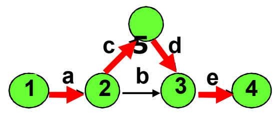
Arc Set **A = {(1,2), (1,3), (3,2), (3,4), (2,4)}**

In an undirected graph, (i,j) = (j,i)

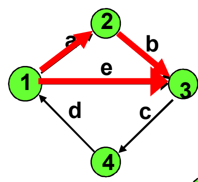
Path: Example: 5, 2, 3, 4.
(or 5, c, 2, b, 3, e, 4)
No node is repeated.•
Directions are ignored.•



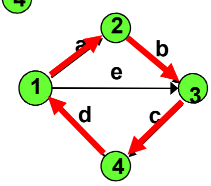
Directed Path . Example: 1, 2, 5, 3, 4
(or 1, a, 2, c, 5, d, 3, e, 4)
No node is repeated.•
Directions are important.•



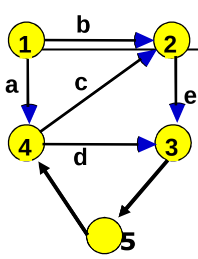
Cycle (or circuit or loop)
1, 2, 3, 1. (or 1, a, 2, b, 3, e)
A path with 2 or more nodes, except •
that the first node is the last node.
Directions are ignored.•

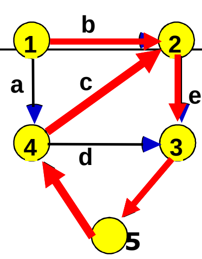


Directed Cycle: (1, 2, 3, 4, 1) or
1, a, 2, b, 3, c, 4, d, 1
No node is repeated.•
Directions are important.•



Walks





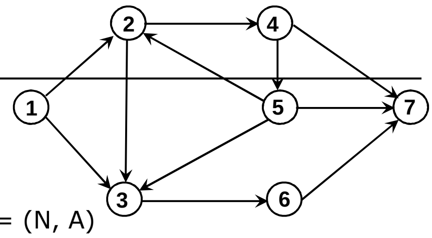
Walks are paths that can repeat nodes and arcs

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.

Network Notation

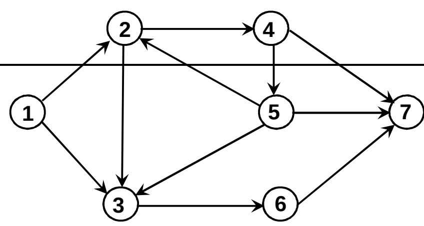


Network (or graph) G = (N, A)

N : Node Set = {1, 2, 3, 4, 5, 6, 7}

A : Arc Set = {(1, 2), (1, 3), (2, 3), (2, 4), (3, 6), (4, 5), (4, 7), (5, 2), (5, 3), (5, 7), (6, 7)}

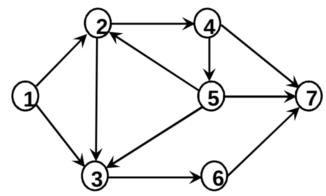
Tails and Head Nodes



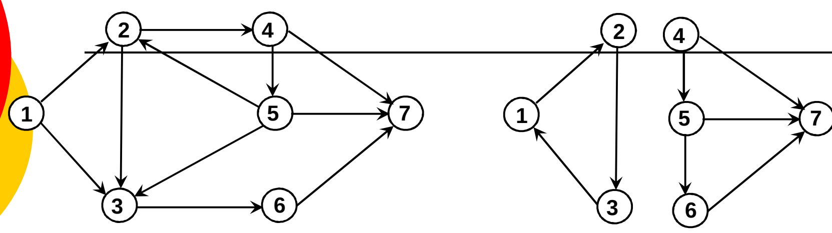
- Tail Node : Node i is the tail node of arc (i, j)
- Head Node : Node j is the head node of arc (i, j)
- Endpoints : Nodes i and j are endpoints of arc (i, j)
- Arc Adjacency List A(i) = {{(i, j) : (i, j) $\in$ A}
- Observe that $\sum_{i \in N} |A(i)| = m$

Subgraph

- Subgraph: G' = (N', A') is a subgraph of G = (N, A) if N' $\subseteq$ N and A' $\subseteq$ A.
- Spanning Subgraph: G' = (N', A') is a spanning subgraph of G = (N, A) if N' = N and A' $\subseteq$ A.
- Acyclic Network: A network is acyclic if it contains no directed cycle



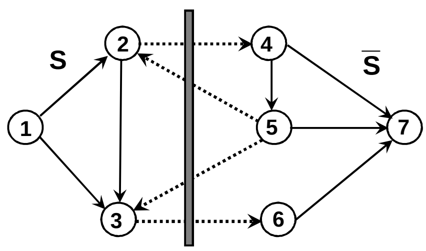
Network Connectivity



- Connected Graph: If every pair of nodes is connected by a path.
- Strongly Connected Graph: If every pair of nodes is connected by a *directed* path.
- Disconnected Graph: If some pair of nodes is not connected by a path. Maximally connected subgraphs of a disconnected graph are called components.

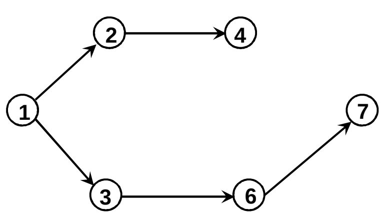
Cuts

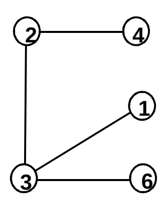
- Cut: A cut is a set of arcs whose deletion disconnects the network into two components S and $\bar{S}$ and no subset of it has this property. A cut is represented by [S, $\bar{S}$].



Trees

- Tree: A tree is a connected graph that contains no cycle.

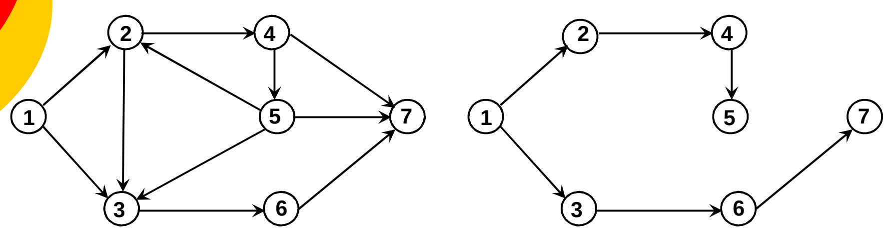




- A tree on n nodes contains exactly (n-1) arcs.
- A tree has at least two leaf nodes.
- Every two nodes of a tree are connected by a unique path.

Spanning Trees

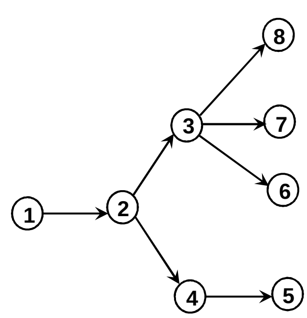
- Spanning Tree: A subgraph of a graph which is a tree and spans all nodes of the graph.

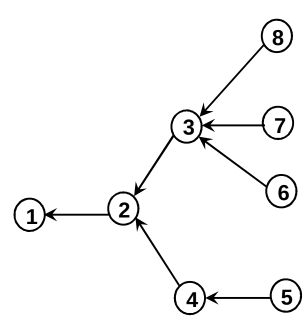


- Forest: A graph that contains no cycle is a forest.
- A forest on n nodes and k components has n-k arcs.

Directed Out-Trees and In-Trees

- Directed-out-tree: A tree in which the unique path in the tree from node s to every other node is a directed path.
- Directed-in-tree: A tree in which the unique path in the tree from any node to node s is a directed path.

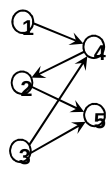


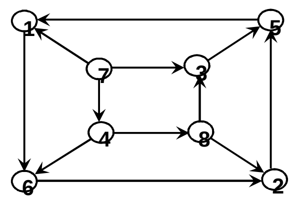


Bipartite Graph

- Bipartite Graph: A graph G = (N, A) is a bipartite graph if node set N can be partitioned into the sets N^1 and N^2 such that each arc (i, j) has either

(i) $i \in N^1$ and $j \in N^2$; or (ii) $i \in N^2$ and $j \in N^1$.





- A graph G is a bipartite graph if and only if every cycle in it has even number of arcs.

Network Representations

- Role of data structures is crucial in developing efficient algorithms. The manner in which networks are stored in a computer determines the performance of a network algorithm.

- We need to perform some of the following operations:

  - Scan through the node list
  - Scan through the arc list
  - Determine all arcs emanating from a specific node i
  - Determine all arcs entering a specific node i
  - Determine all arcs emanating from or entering a specific node i

Common Network Representations

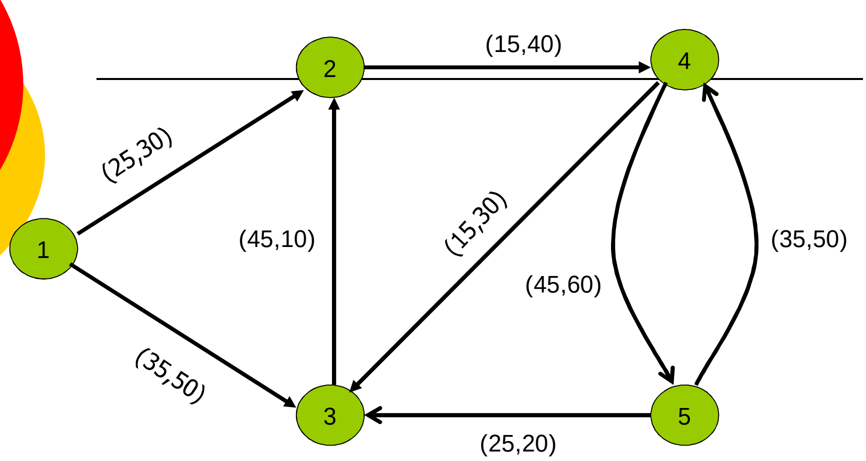
- Node-arc incidence matrix
- Node-node adjacency matrix
- Adjacency list
- Forward star representation
- Reverse star representation
- Forward and reverse star representations

Network representations

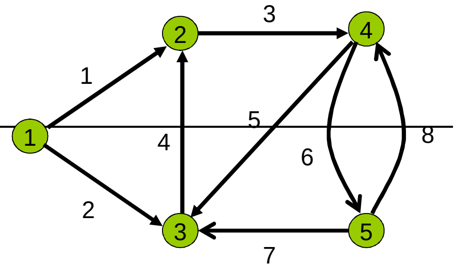
- We typically need to store two types of information:
  - Network topology
    - Nodes and arc structures
  - Data
    - Costs
    - Capacities
    - Supplies / demands

Incidence matrix representation

- *N* n x m matrix contains
  - One row for each node of the network
  - One column for each arc
  - Column *(i , j)* has only two nonzero elements
    - +1 in row *i*
    - -1 in row *j*

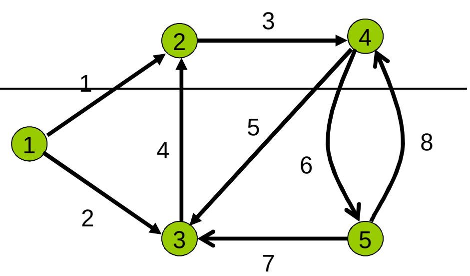


Node-Arc Incidence Matrix



$$\begin{array}{c|c}
 & \begin{array}{cccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & -1 & 0 & -1 & 0 \\
0 & 0 & -1 & 0 & 1 & 1 & 0 & -1 \\
0 & 0 & 0 & 0 & 0 & -1 & 1 & 1
\end{bmatrix}
\end{array}$$

Node-Node Adjacency Matrix

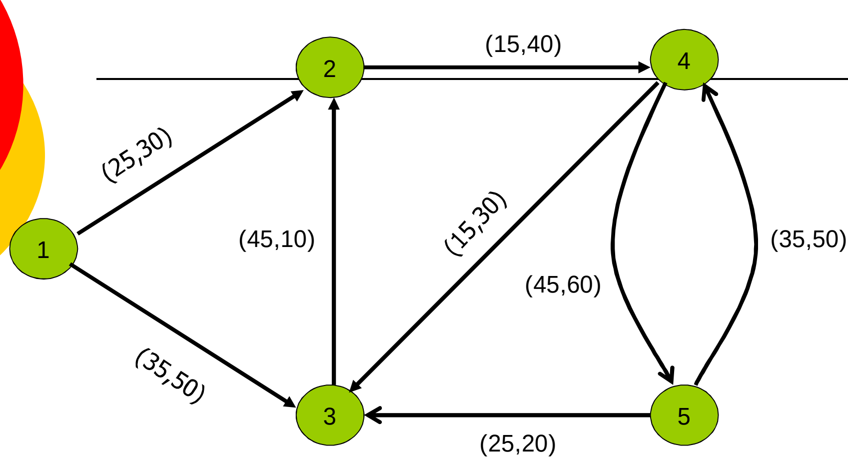


$$\begin{array}{c|c}
 & \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\
\begin{array}{c} 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} &
\begin{bmatrix}
0 & 1 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0
\end{bmatrix}
\end{array}$$

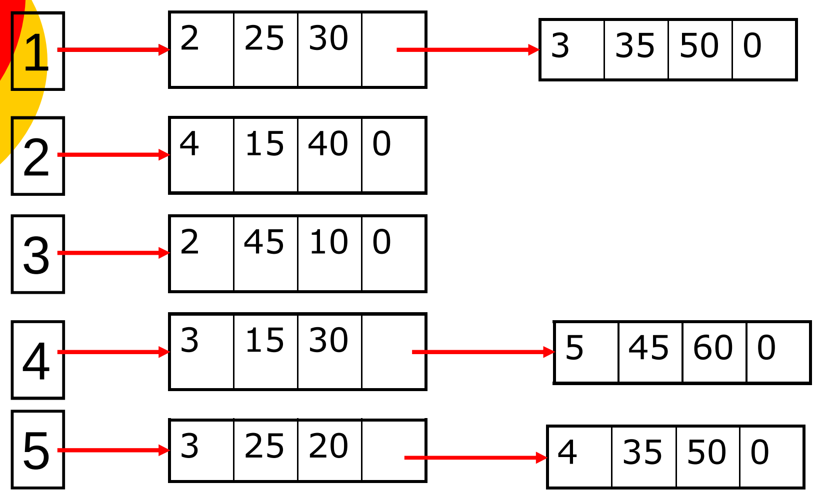
$$n \times n \quad \text{matrix } l_{ij} = \{^{1}_{0} \qquad (i, j) \in A$$

Adjacency list

- The adjacency list representation stores the node adjacency list of each node a singly linked list.



Adjacency Lists

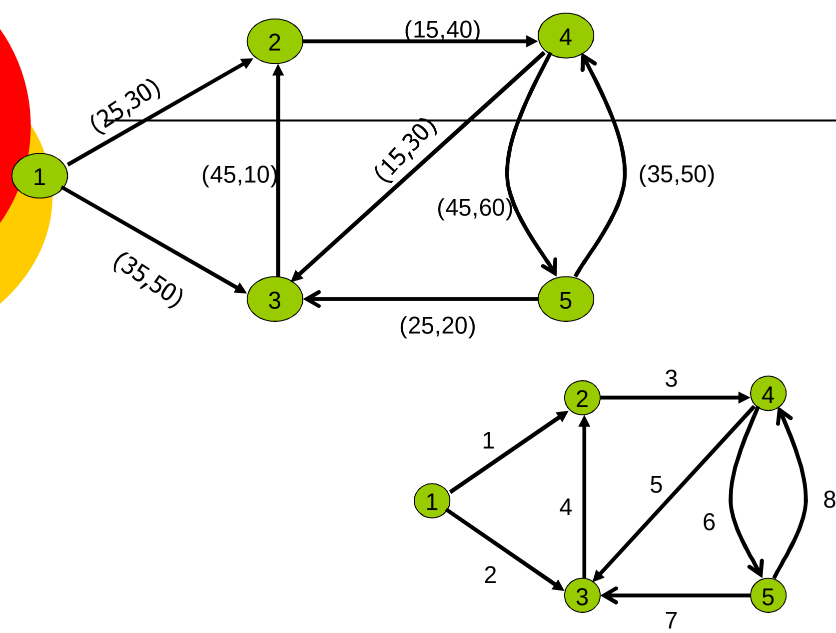


Forward Star Representation

- Similar to the adjacency list representation which stores the node adjacency list of each node
- It stores them in a single array
- We number the arcs in a specific order
- We store the tails, heads, costs, and capacities of the arcs in four arrays: tail, head, cost, and capacity

Forward Star Representation

- We maintain a pointer with each node i, denoted by Point (i),
- Point (i):= the smallest number arc in the arc list that emanates from node i
- Point (i):= Point (i+1)
  - if node i has no outgoing arcs
- Point (1)=1
- Point (n+1)=m+1



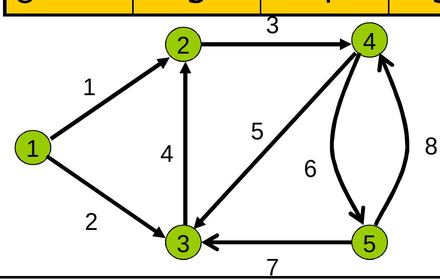
Forward Star Representation

	point
1	1
2	3
3	4
4	5
5	7
6	9

	tail	head	cost	capacity
1	1	2	25	30
2	1	3	35	50
3	2	4	15	40
4	3	2	45	10
5	4	3	15	30
6	4	5	45	60
7	5	3	25	20
8	5	4	35	50

	Point
1	1
2	3
3	4
4	5
5	7
6	9

	tail	head	cost	capacity
1	1	2	25	30
2	1	3	35	50
3	2	4	15	40
4	3	2	45	10
5	4	3	15	30
6	4	5	45	60
7	5	3	25	20
8	5	4	35	50



- Note: the FSR will store the outgoing arcs of node i at positions
  - Point (i) to point (i+1) – 1
  - In the arc list
  - If point (i) > point (i+1)-1 node i has no outgoing arc
  - FSR provides us with an efficient way for determining the set of outgoing arcs of any node

Exercises due date Sunday 5/7/89

- Exercises
- 2.4, 2.8, 2.10
- 2.12 , 2.13, 2.16
- 2.19, 2.20, 2.23 , 2.24
- 2.32, 2.33, 2.34, 2.35 2.36 , 2.37, 2.38