Computer Science 112
Data Structures

Lecture 20:
Graphs:
Representations
Depth First Search

Announcements

- **Midterm Exam 2**
  - **Sunday April 12**
  - **3:00 – 4:20 pm**
  - **See sakai announcements for rooms**

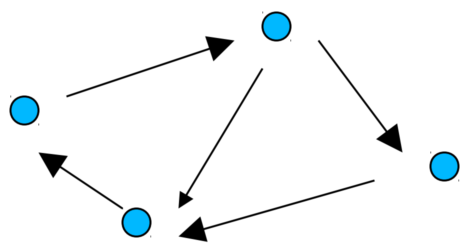
Announcements

- **You should already have watched videos on hashing and on graphs**
  - **see sakai announcements**
  - **Graph.java and data files website.txt and friendship.txt are on Sakai in Resources > Steinberg > Java > graph**

New: Graphs

Generalization of trees

- Digraph (Directed Graph)
  - Like a tree but any vertex can point to any other

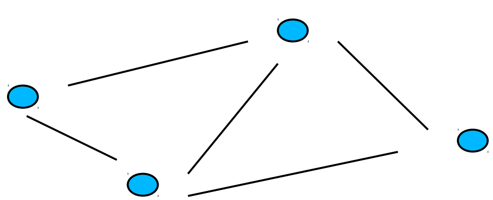


  - E.g., Twitter follows relationship

New: Graphs

Generalization of trees

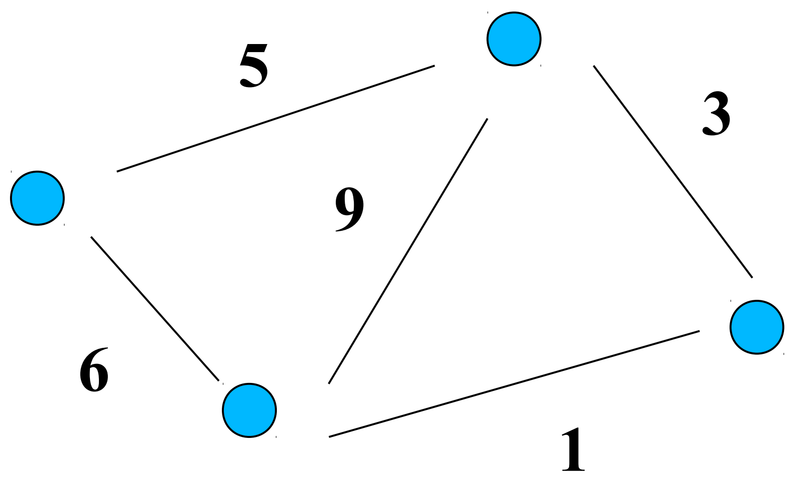
- **Graph**
  - **like digraph but arcs have no direction**



  - **E.g., Facebook friends relationship**

Graphs

- **Weighted Graph**
  - **Positive integer weights on each edge**

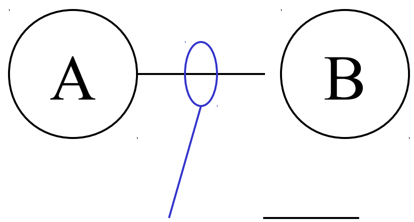


Applications

- **Paths**
  - **On streets (eg Google Maps)**
- **Electrical networks**
  - **Power lines**
- **Constraints**
  - **Ordering constraints on building steps eg counters before sinks**

Notation

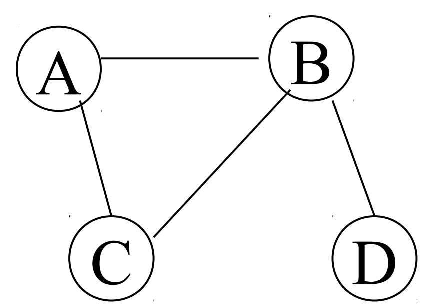
- **Arcs are named by the vertices they connect**



(B,A) or $\overline{\mathbf{BA}}$

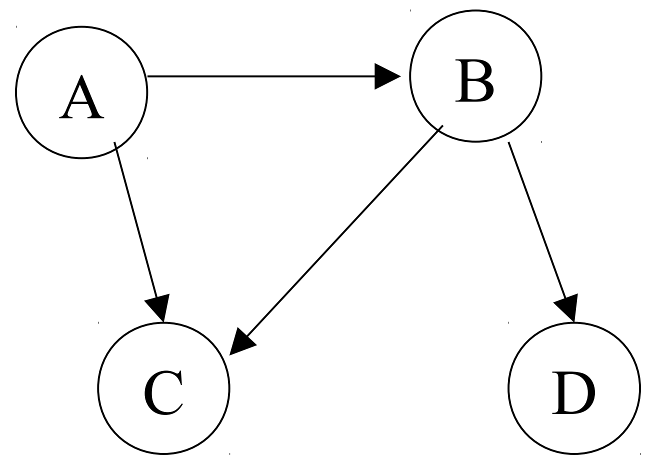
Graph Concepts

- **Neighbors of a vertex: vertices that it shares an arc with**
  - **Neighbors of A are B and C**
- **Degree of a vertex: number of neighbors**
  - **Degree of A is 2, degree of B is 3**



Graph Concepts

- **In degree (in a digraph): number of vertices that have arcs to this vertex**
  - **In degree of B is 1**
- **Out degree (in a digraph): number of vertices that have arcs from this vertex**
  - **Out degree of B is 2**



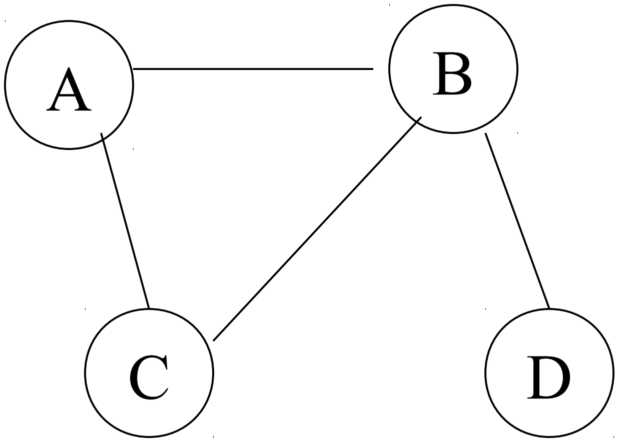
Graph Concepts

- **(Simple) Path**
  - **Sequence of arcs**

    (A,B),(B,C)
  - **May not revisit a vertex**

    ~~(B,A),(A,C),(C,B),(B,D)~~
  - **Except last vertex may = first**

    (B,A),(A,C),(C,B)
- **Vertex A is** reachable **from B if there is a path from B to A**

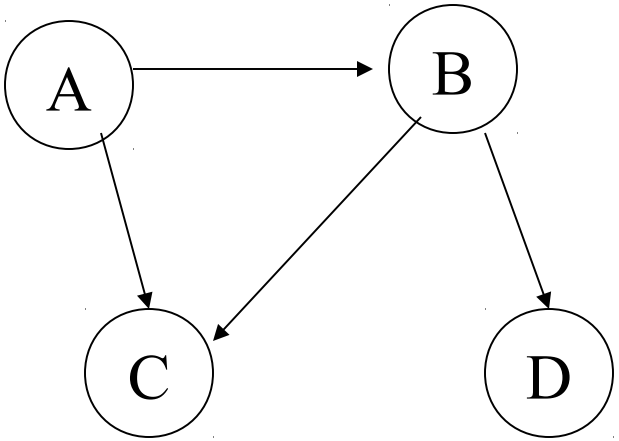


Graph Concepts

- **Path**
  - **On digraph must follow arc directions**

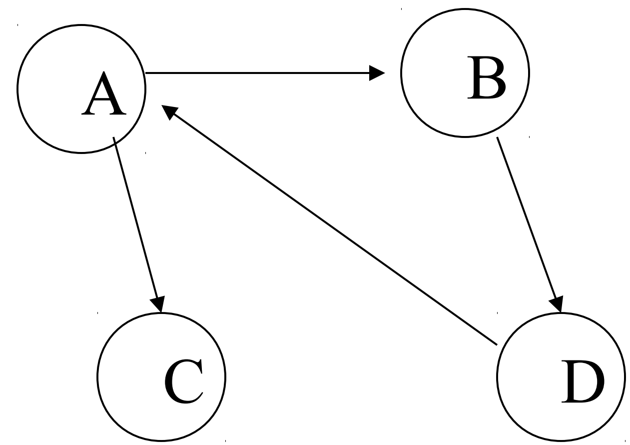
    (A,B),(B,D)

    ~~**(A,C),(C,B)**~~



Graph Concepts

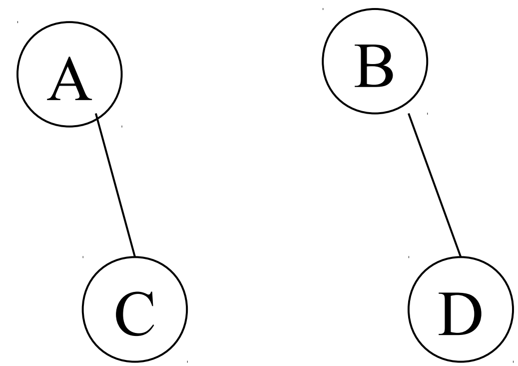
- **A cycle is a path from a node back to itself**
  - **(A, B)(B, D)(D, A)**
- **A graph with no cycles is called acyclic**



Graph Concepts

- **Connected Graph**

  For any two vertices X and Y there is a path from X to Y.

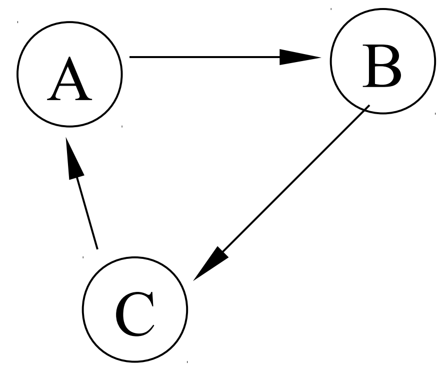


not connected

Graph Concepts

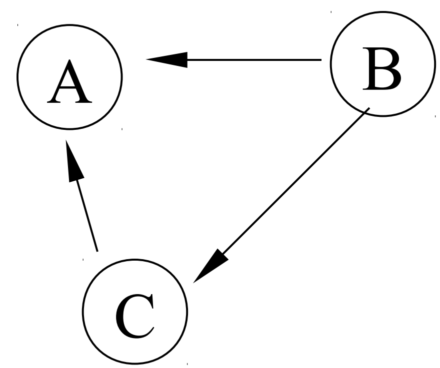
- **Strongly Connected Digraph**

  For any two vertices X and Y there is a path from X to Y. (Paths must follow arc directions)



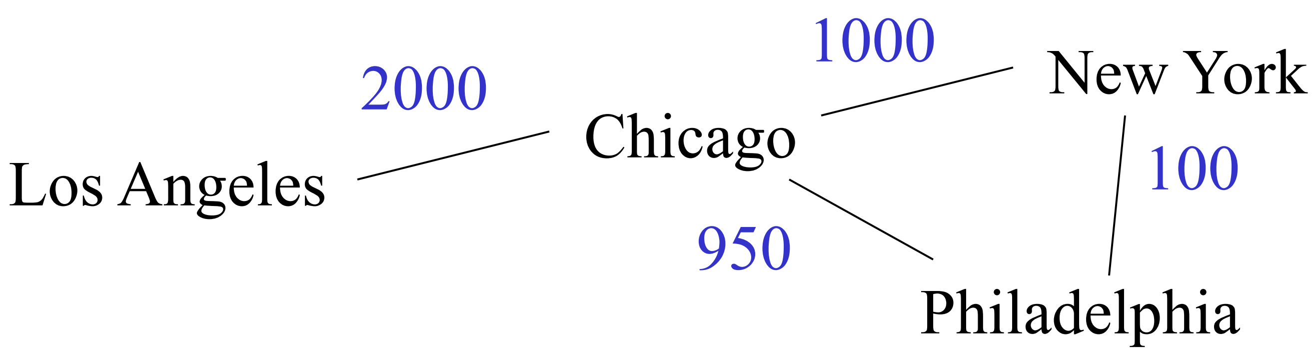
- **Weakly Connected Digraph**

  Corresponding graph is connected (i.e., ignoring arc direction)



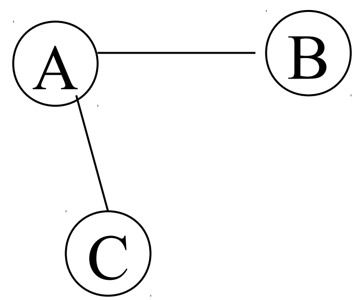
Graph Concepts

- **Weighted graph: each arc has a numerical weight**



Representing Graphs

- **Adjacency matrix**
  - **n x n boolean matrix: is there an arc?**

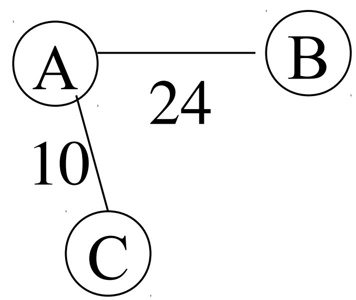


	name
0	A
1	B
2	C

	0	1	2
0		T	T
1	T		
2	T		

Representing Graphs

- **Adjacency matrix**
  - n x n boolean matrix: is there an arc?

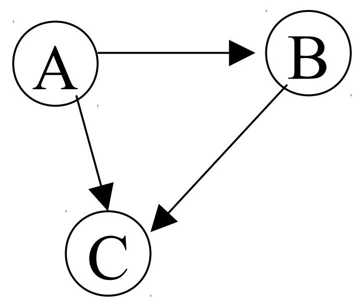


	name
0	A
1	B
2	C

	0	1	2
0	-1	24	10
1	24	-1	-1
2	10	-1	-1

Representing Graphs

- **Adjacency matrix**
  - **n x n boolean matrix: is there an arc?**



	name
0	**A**
1	**B**
2	**C**

	0	1	2
0		**T**	**T**
1			**T**
2			

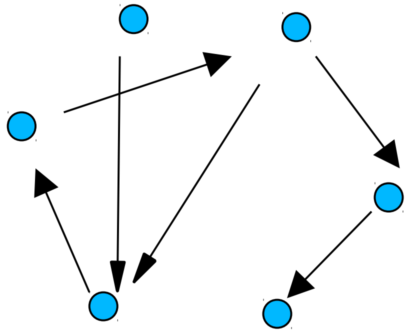
Adjacency Matrix

- **Space cost: v^2 booleans where v is number of vertices**
- **If v is large, v^2 is huge**
  - **Facebook: $v = 10^9$, $v^2 = 10^{18}$**

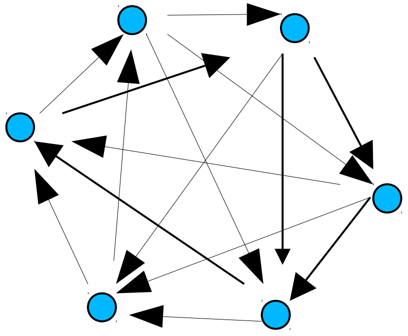
    1,000,000,000,000,000,000
  - **An average Facebook user has about 350 friends**
  - **if e is number of edges, $e = 10^9 * 175$**
  - **Fraction of Trues in matrix = $1.75 * 10^{-7}$**

    = 1 / 5,000,000

Sparse vs Dense Graphs



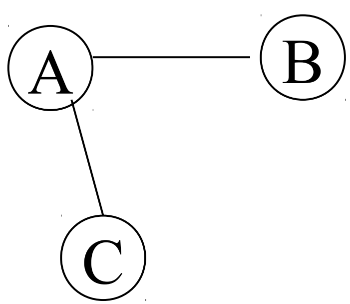
Sparse

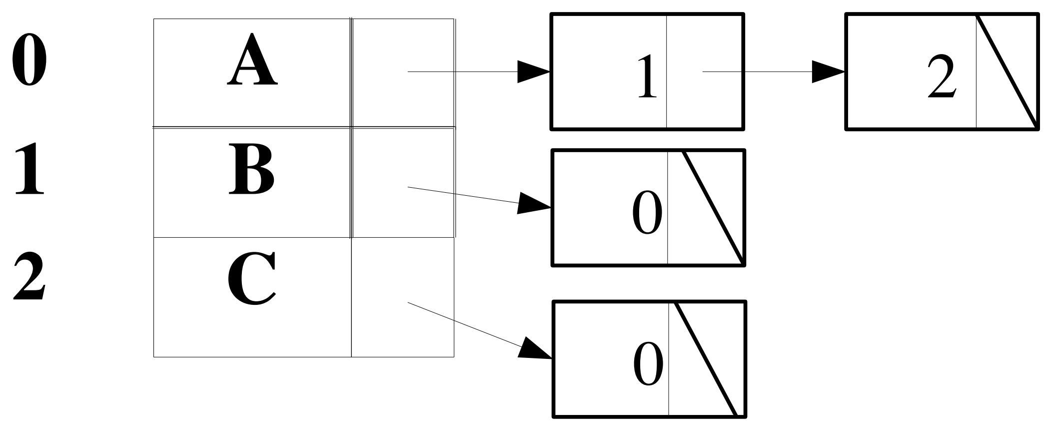


Dense

Representing Graphs

- **Adjacency list**
  - **for each node, a linked list of edges that touch it**



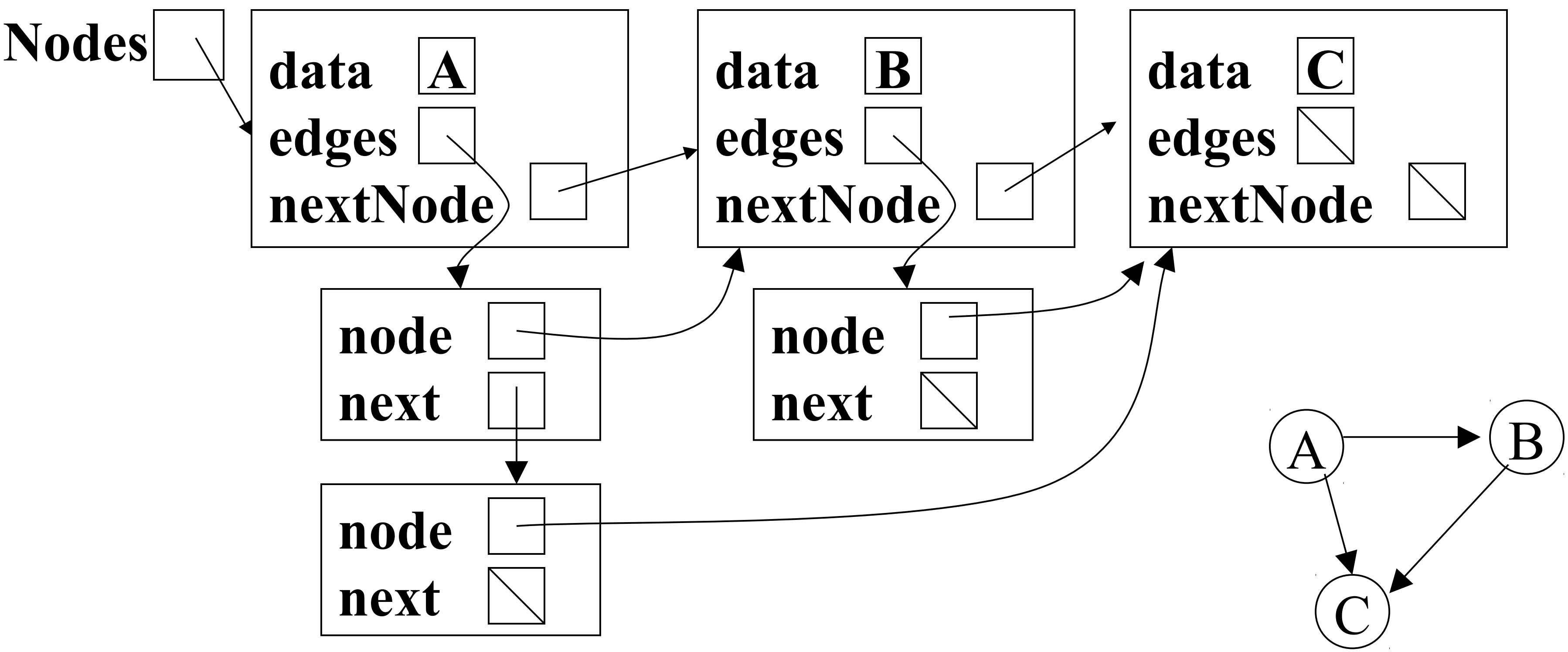


Representing Graphs

- **Adjacency list**
  - **for each node, a linked list of edges that touch it**
  - **Space cost: $v + 2 * 2 * e$**
  - **For Facebook: $10^9 + 4 * 175 * 10^9 = 700*10^9$**

Representing Graphs

- **Adjacency list**
  - for each node, linked list of edges



Time costs, Worst case

	Is there and edge from i to j	List the neighbors of i
Adjacency matrix	O(1)	O(v)
Adjacency list	O(d)	O(d)

d is degree of i, d<v

Exercise

- **You write countEdges( ) in class Graph**

Graph Traversals

- **Need to mark vertices as we see them to prevent infinite loops**
- **Need driver in case not connected**
- **Otherwise like tree traversals**
- **Depth first**

```
dfsG(v)
    if (marked(v)) return;
    visit v;
    mark v;
    for each vn in neighbors(v)
      dfsG(vn)
```

Graph Traversals

- **Need driver in case not connected**

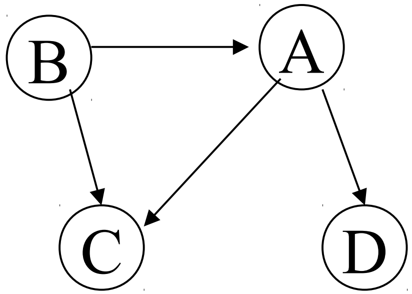
  For v in vertices

  dfsG(v)

DFS Graph Traversal

- Enters a vertex v
- Visits all vertices reachable from v (that have not yet been visited
- Leaves v

Graph Traversals



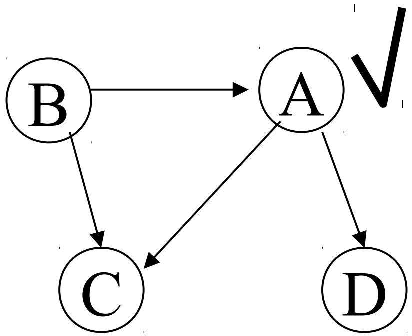
Driver

v = <A>

dfsG

v = <A>

Graph Traversals



Driver

v = <A>

dfsG

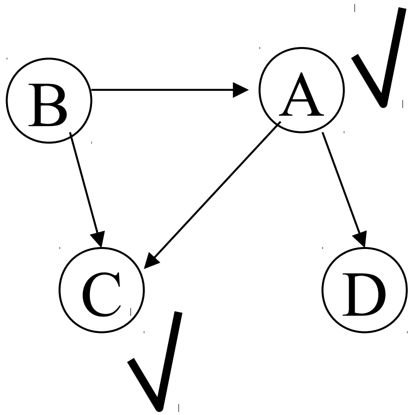
v = <A>

vn = <C>

dfsG

v = <C>

Graph Traversals



Driver

v = <A>

dfsG

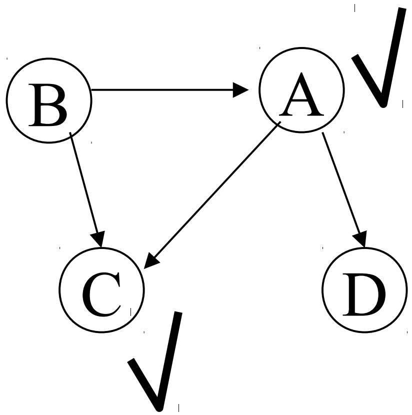
v = <A>

vn = <C>

dfsG

v = <C>

Graph Traversals



Driver

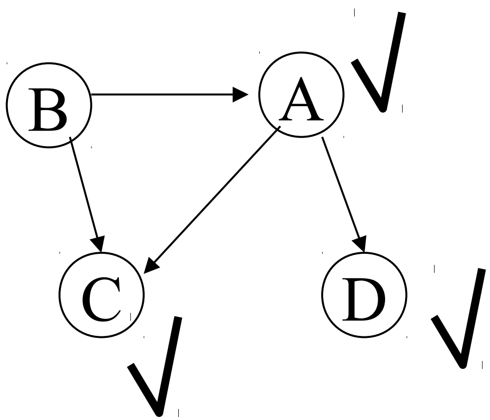
v = <A>

dfsG

v = <A>

vn = <D>

Graph Traversals



Driver

v = <A>

dfsG

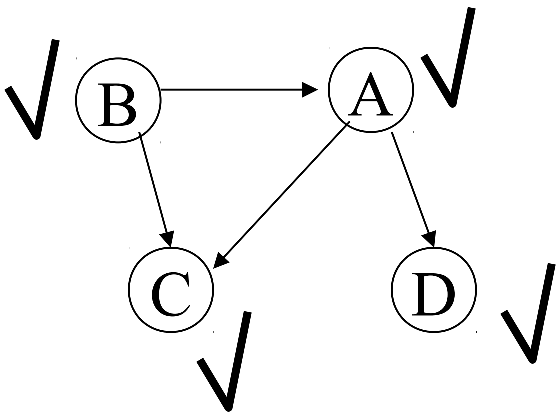
v = <A>

vn = <D>

dfsG

v = <D>

Graph Traversals



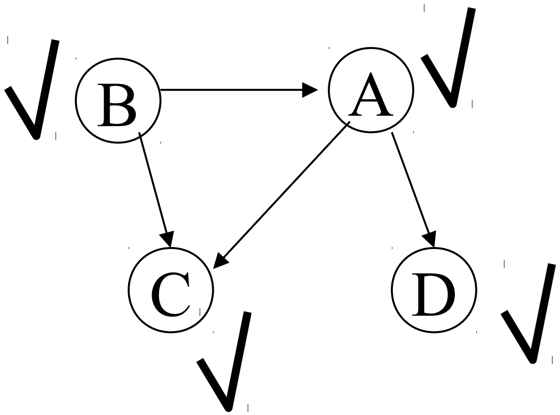
Driver

v = <B>

dfsG

v = <B>

Graph Traversals



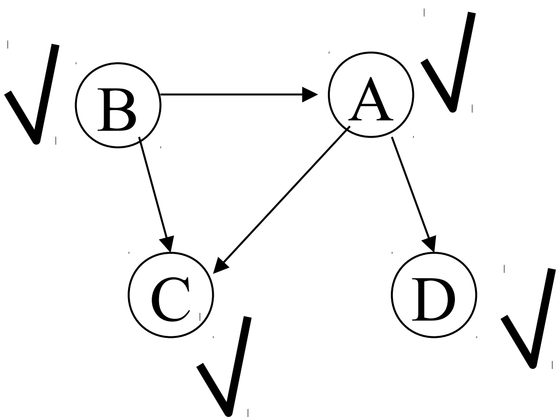
Driver

v = <C>

dfsG

v = <C>

Graph Traversals



Driver
v = <D>

dfsG
v = <D>

Graph Traversals

- **Time:**
  - **Visit each vertex**
  - **inspect each arc**
  - **driver**

  O(n + e) n vertices, e edges