CS 91 USACO

Bronze Division

Unit 4: Basic Tree and Graphs

LECTURE 20: REVIEW OF GRAPH PROBLEMS

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- Review of Graph problems – Wormholes
- Programming Paradigms

Practice: Wormholes Problem

SECTION 1

Problem Statement

- Farmer John's hobby of conducting high-energy physics experiments on weekends has backfired, causing N wormholes (2 <= N <= 12, N even) to materialize on his farm, each located at a distinct point on the 2D map of his farm (the x,y coordinates are both integers).
- According to his calculations, Farmer John knows that his wormholes will form N/2 connected pairs. For example, if wormholes A and B are connected as a pair, then any object entering wormhole A will exit wormhole B moving in the same direction, and any object entering wormhole B will similarly exit from wormhole A moving in the same direction. This can have rather unpleasant consequences.

Problem Statement

- For example, suppose there are two paired wormholes A at (1,1) and B at (3,1), and that Bessie the cow starts from position (2,1) moving in the +x direction. Bessie will enter wormhole B [at (3,1)], exit from A [at (1,1)], then enter B again, and so on, getting trapped in an infinite cycle!

```
| . . . .
| A > B .       Bessie will travel to B then
+ . . . .       A then across to B again
```

Problem Statement

- Farmer John knows the exact location of each wormhole on his farm. He knows that Bessie the cow always walks in the +x direction, although he does not remember where Bessie is currently located.
- Please help Farmer John count the number of distinct pairings of the wormholes such that Bessie could possibly get trapped in an infinite cycle if she starts from an unlucky position. FJ doesn't know which wormhole pairs with any other wormhole, so find all the possibilities (i.e., all the different ways that N wormholes could be paired such that Bessie can, in some way, get in a cycle). Note that a loop with a smaller number of wormholes might contribute a number of different sets of pairings to the total count as those wormholes that are not in the loop are paired in many different ways.

INPUT FORMAT (wormhole.in):

- Line 1: The number of wormholes, N.
- Lines 2..1+N: Each line contains two space-separated integers describing the (x,y) coordinates of a single wormhole. Each coordinate is in the range 0..1,000,000,000..

SAMPLE INPUT:

```
4
0 0
1 0
1 1
0 1
```

There are 4 wormholes, forming the corners of a square.

OUTPUT FORMAT (wormhole.out):

- Line 1: The number of distinct pairings of wormholes such that Bessie could conceivably get stuck in a cycle walking from some starting point in the +x direction.

SAMPLE OUTPUT:

```
2
```

Output Details

- If we number the wormholes 1..4 as we read them from the input, then if wormhole 1 pairs with wormhole 2 and wormhole 3 pairs with wormhole 4, Bessie can get stuck if she starts anywhere between (0,0) and (1,0) or between (0,1) and (1,1).

```
|  .  .
d  c  .
a  b  .
```

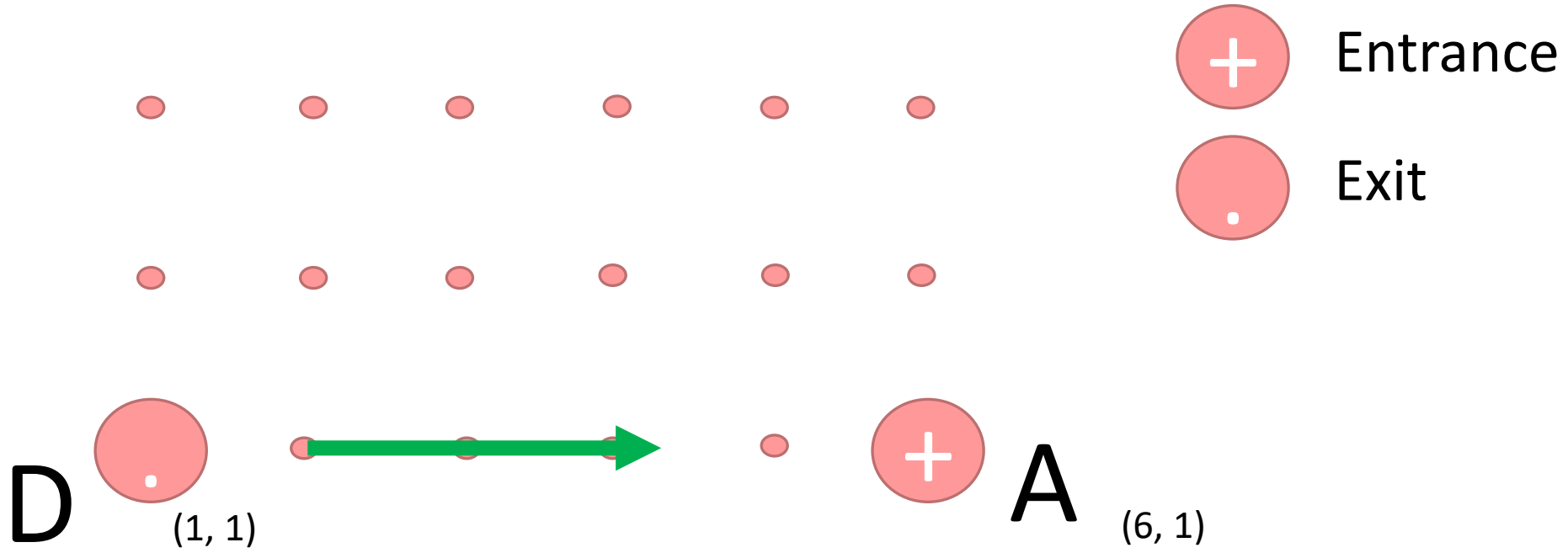
Output Details

- Here is a list of all the pairings, annotated with their wormhole results:
  - (a b) (c d) -- Bessie loops (a b) if she is on that line
  - (a c) (b d) -- Bessie loops b->d->c->a->b ... if she gets caught
  - (a d) (b c)
- Only the pairings a-d and b-c allow Bessie to walk in the +x direction from any point in the 2D plane with no danger of cycling.

Worm Hole Problem

A wormhole is a theoretical **passage** through **space-time** that could create shortcuts for long journeys across the universe. **Wormholes** are predicted by the theory of general **relativity**. But be wary: wormholes bring with them the dangers of sudden collapse, high radiation and dangerous contact with exotic matter.

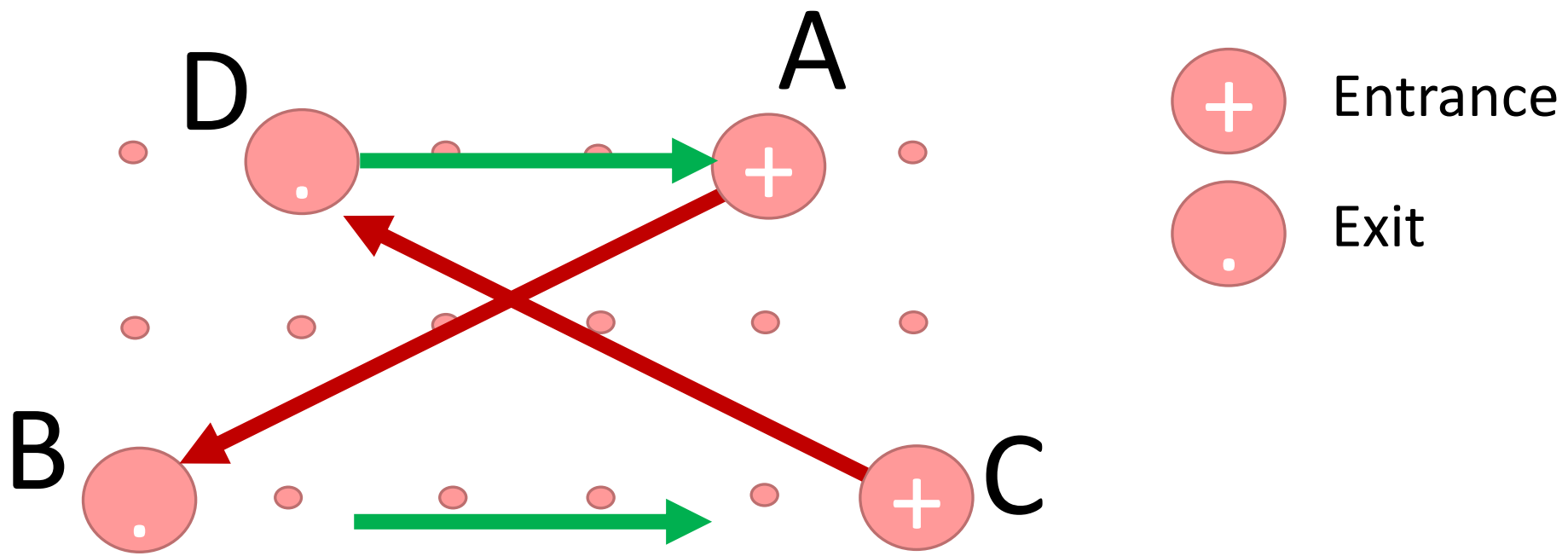
Worm Hole Analysis



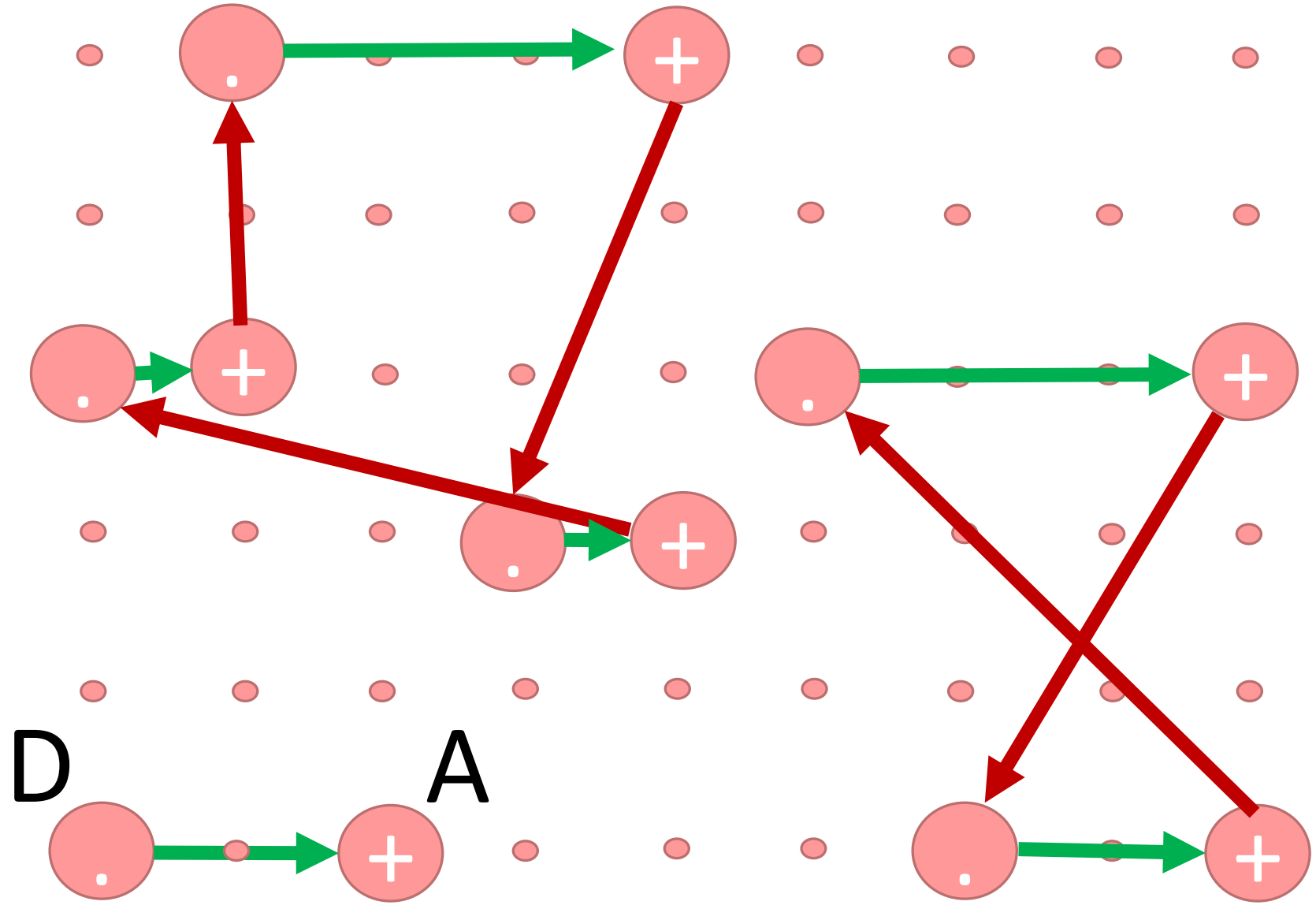
If a pairing contains pair [A, D], then there is a cycle, there is a trap.

Worm Hole Analysis

Next Neighbor on the Right



If we start from A
If a pairing contains pair [A, B] and A is the next on the right for C point, while [C, D] paired, and
[C, D] is the last pair we can find in this looping, and D has a point as next neighbor on the right. Then, there is a cycle.



- If we tried all of the wormholes as starting points, and
- there exists a certain starting point that loops around partner and next neighbor to the right and end at a point which has a next neighbor to the right, then there is a cycle. Otherwise there is no a cycle.

Nature of the Problem

- Starts from the recursive pairing problem.
- Finding a of the possible pairings for a group of wormholes.
- If a pairings has cycle, then we increase the trapped case by one.
- Then, at each printing of pairing location, we should check each pairing if there exists a cycle in the pairing. An isCycle() need to be checked.

Nature of the isCycle() Problem

- Repeating the steps on finding partner, then find the partner's next neighbor on the right as a new point to find the next partner.
- If the last neighbor on the right exists, then, there is a cycle.
- If the last next neighbor to the right does not exists, then, there is no cycle.

isCycle() Function

```
public static boolean isCycle() {
    for(int start = 0; start <N; start++) {
      int pos = start;
      for(int count = 0; count < N; count++) {
         if (pos != -1) pos = next_on_right[partners[pos]];
      }
      if(pos != -1) return true;
    }
    return false;
}
```

Programming Paradigms

SECTION 2

Search vs Traversal

- Search: Look for a given node
  - stop when node found, even if not all nodes were visited
- Traversal: Always visit all nodes
- Query: Searching with given criteria

Depth-first Search

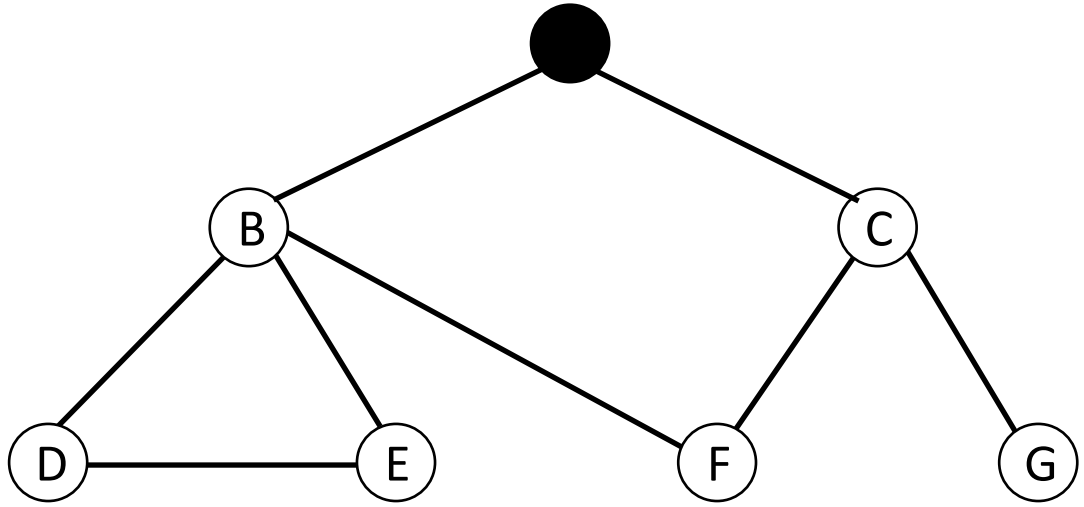
- Similar to Depth-first Traversal of a Binary Tree
- Choose a starting vertex
- Do a depth-first search on each adjacent vertex

Pseudo-Code for Depth-First Search

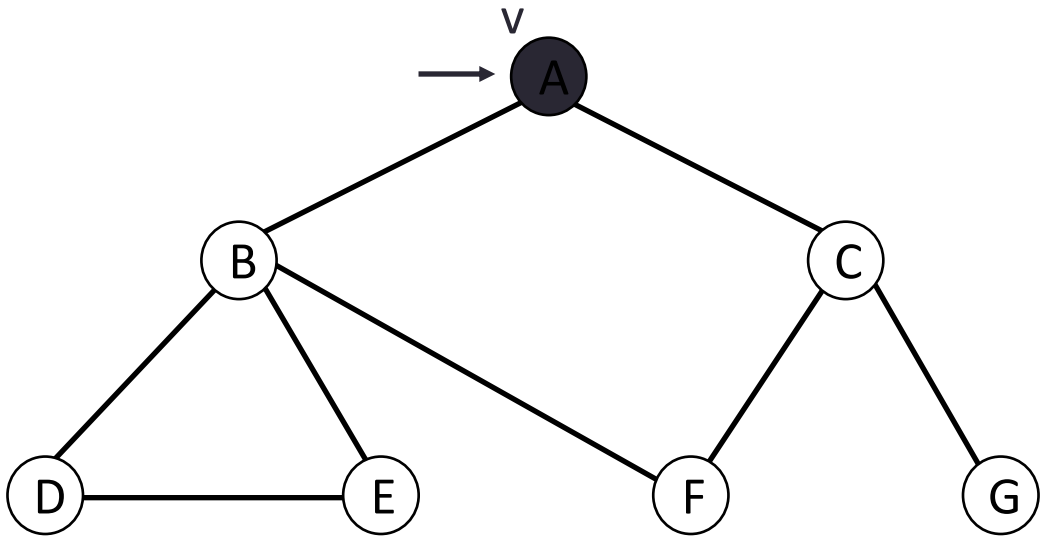
Depth-First-Search

```
Mark vertex as visited
For each adjacent vertex
  If unvisited
    Do a depth-first search on adjacent vertex
```

Depth-First Search

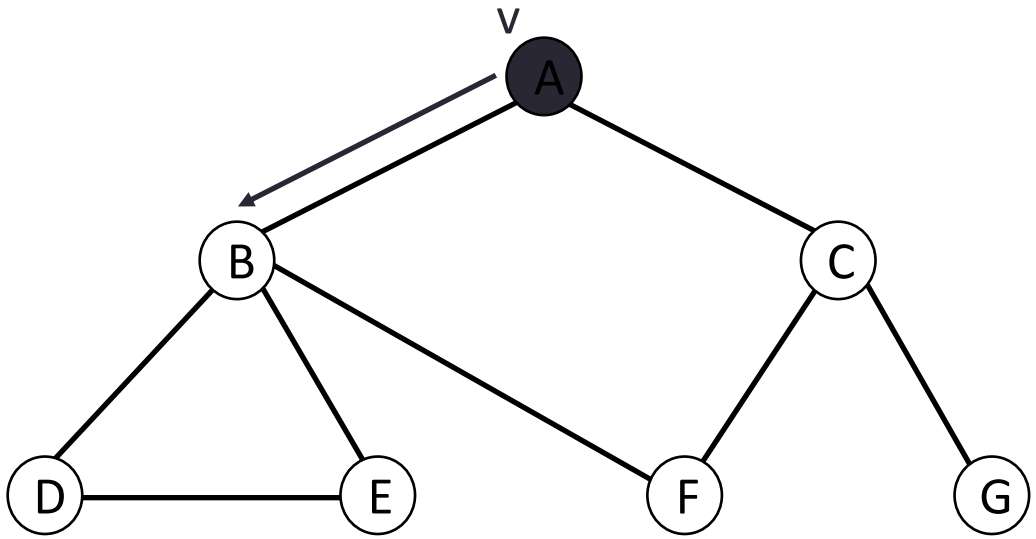


Depth-First Search



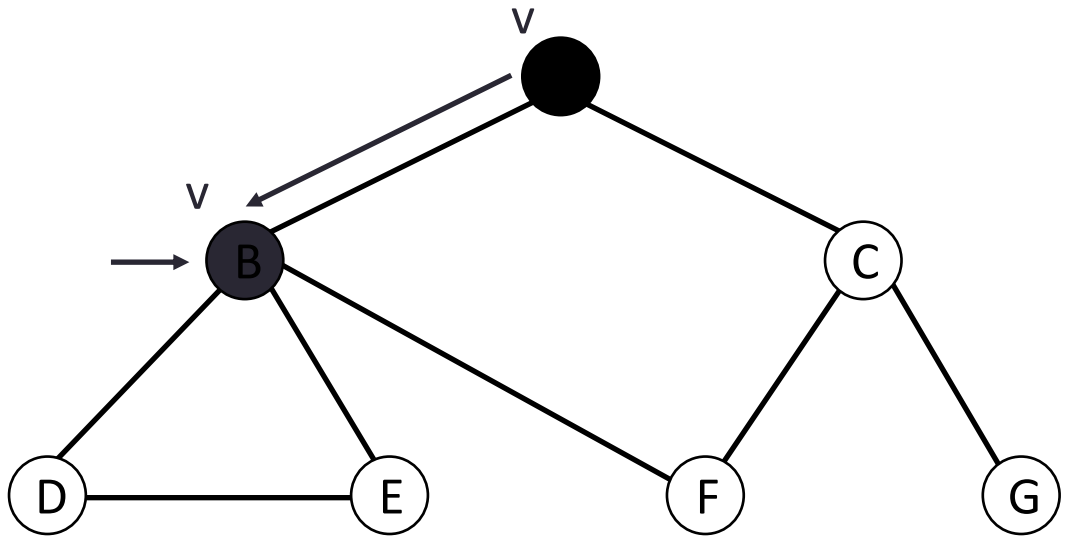
A

Depth-First Search



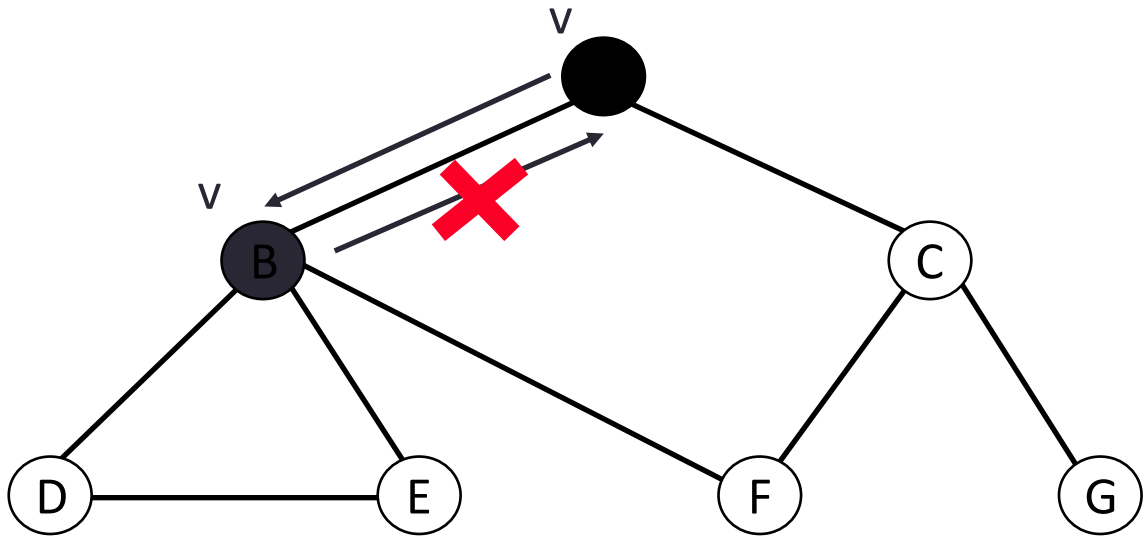
A

Depth-First Search



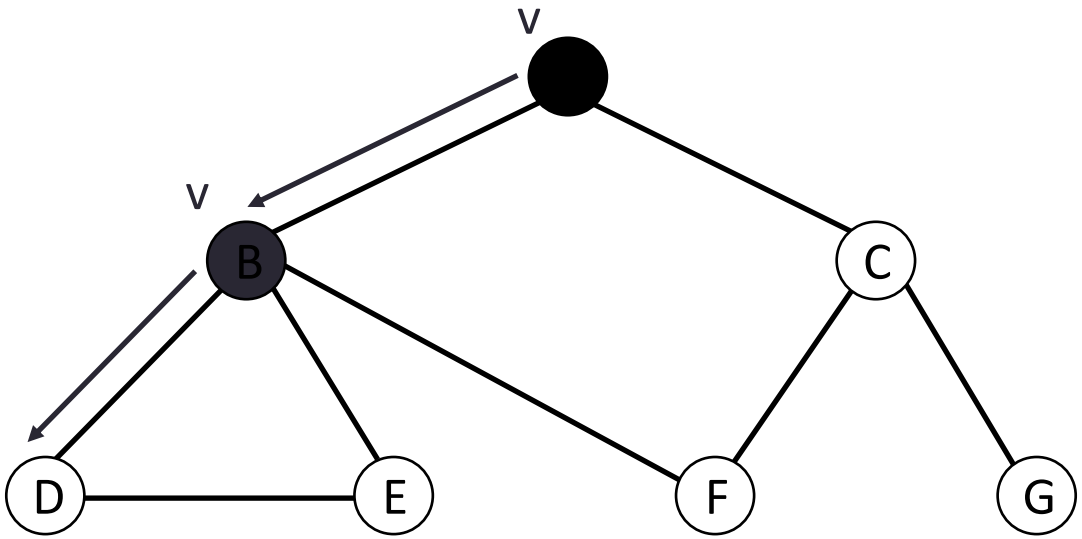
A B

Depth-First Search



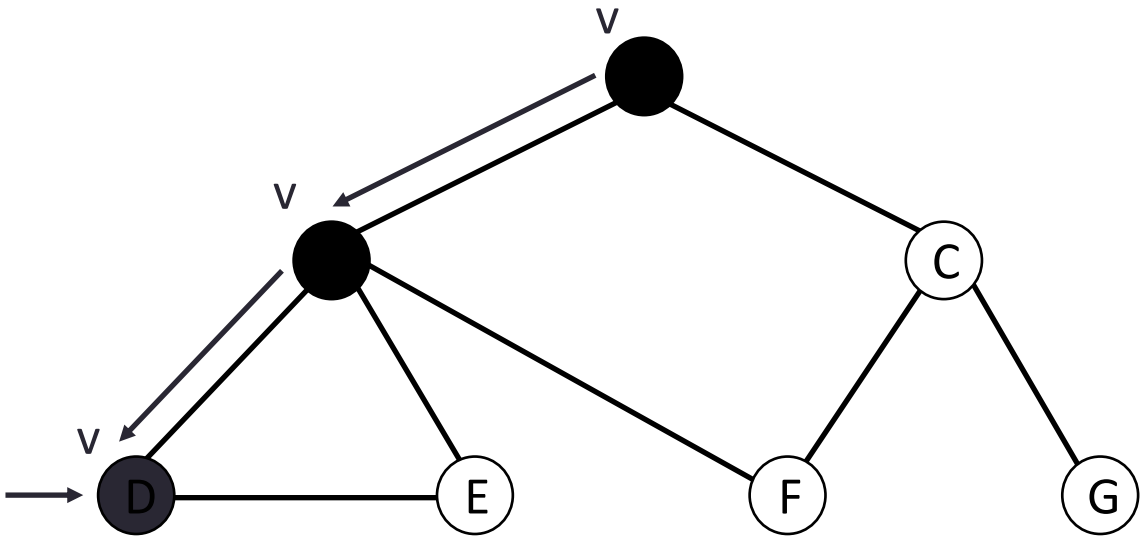
A B

Depth-First Search



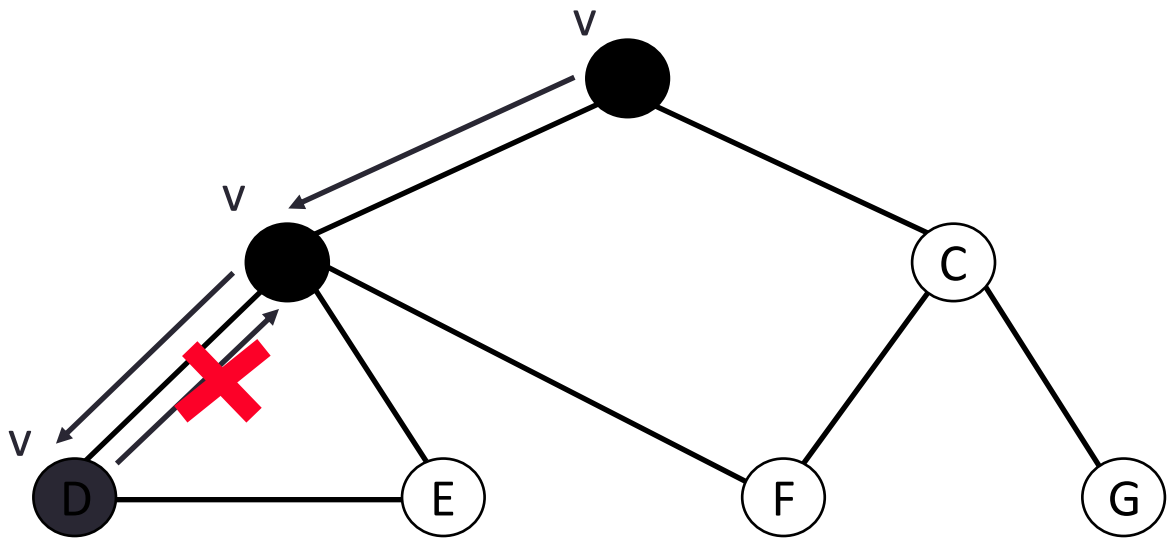
A B

Depth-First Search



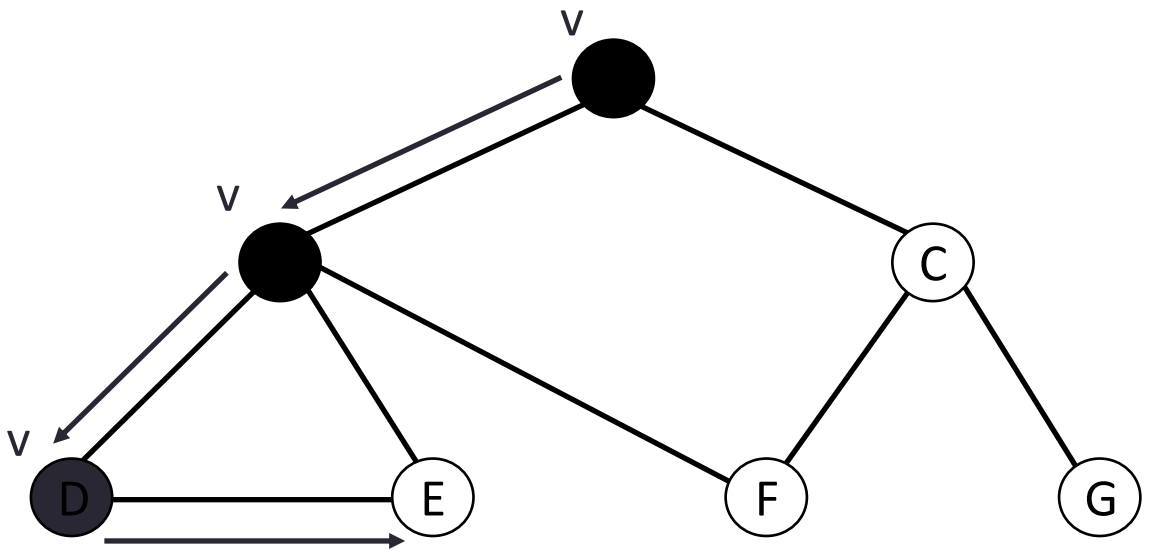
A B D

Depth-First Search



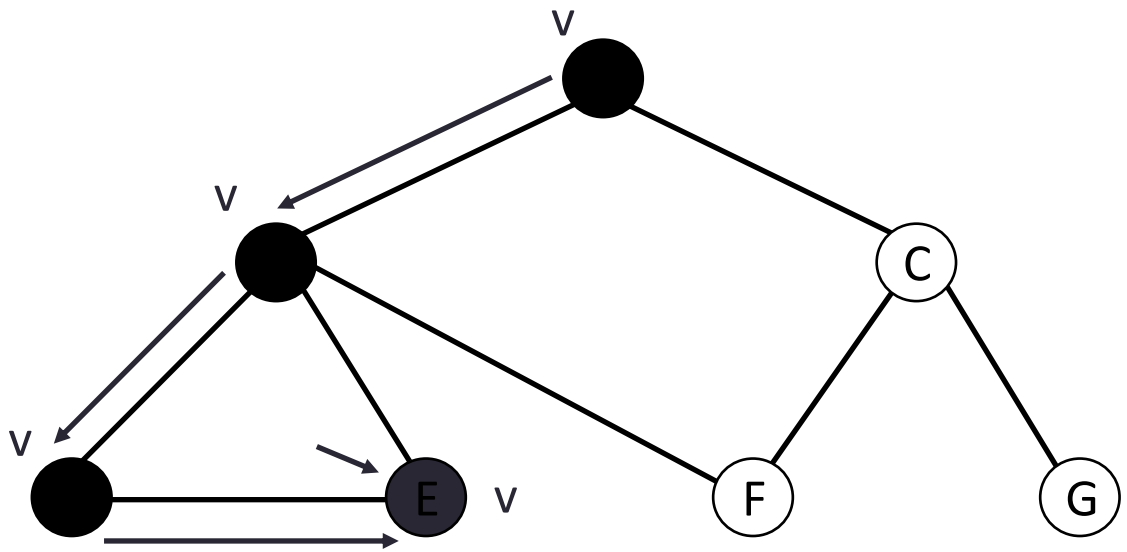
A B D

Depth-First Search



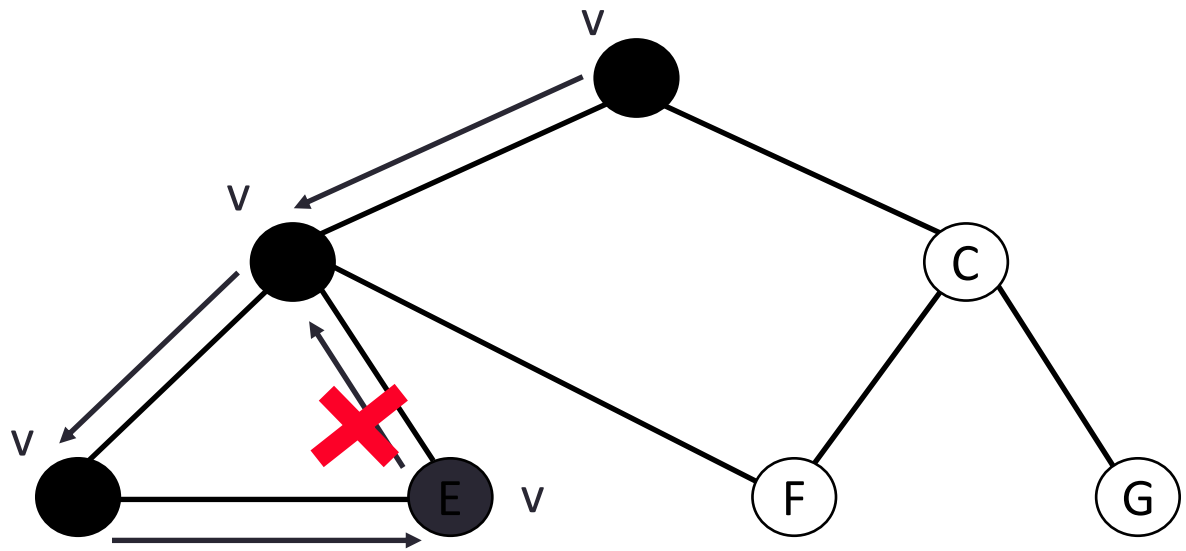
A B D

Depth-First Search



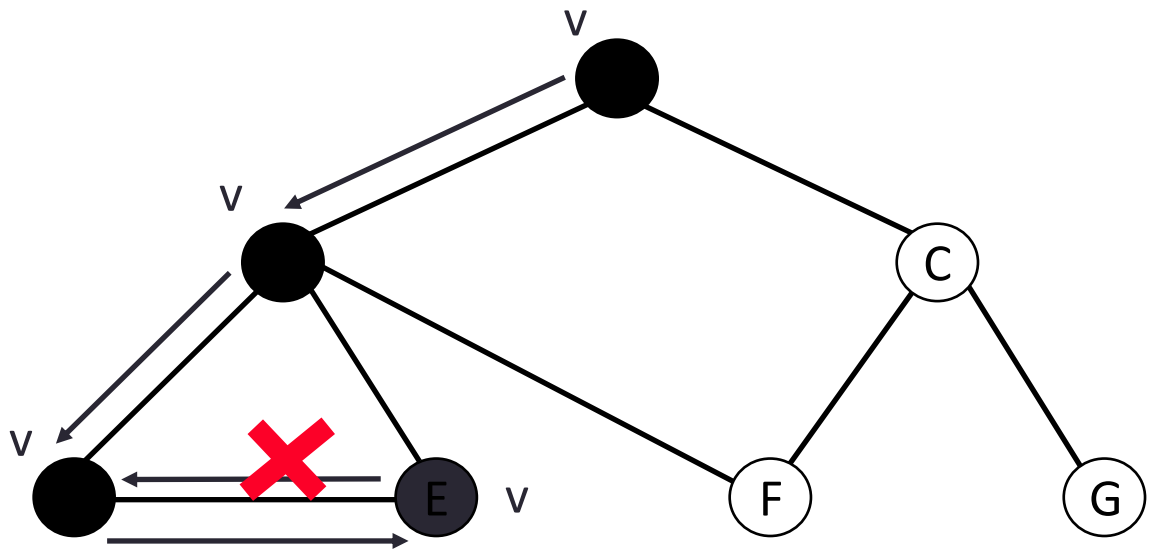
A B D E

Depth-First Search



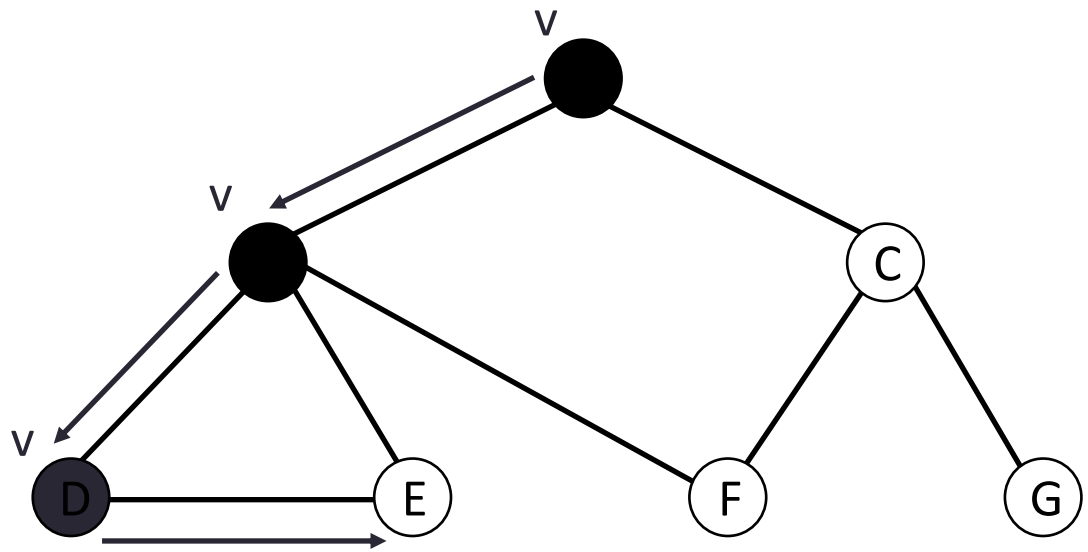
A B D E

Depth-First Search



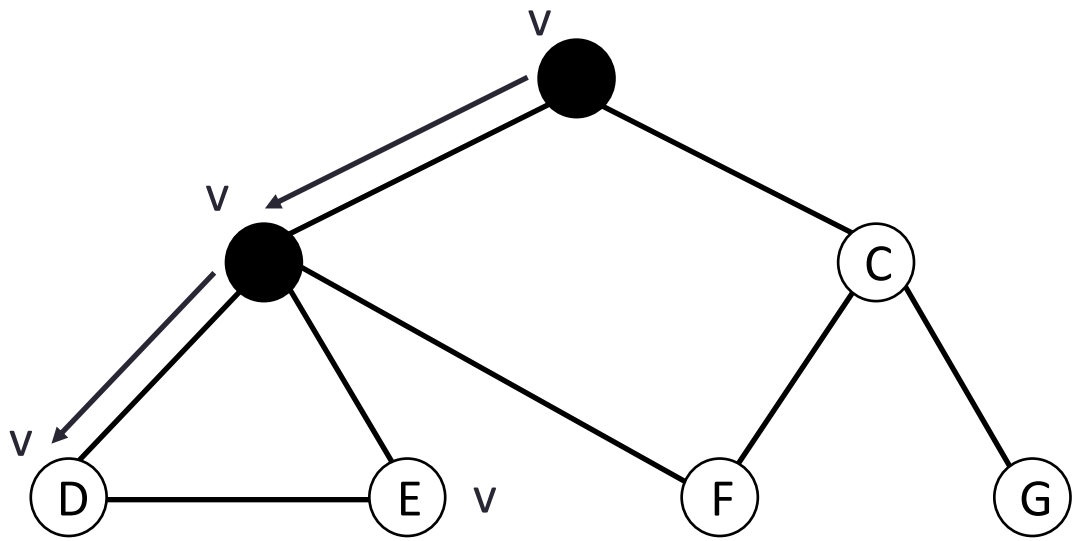
A B D E

Depth-First Search



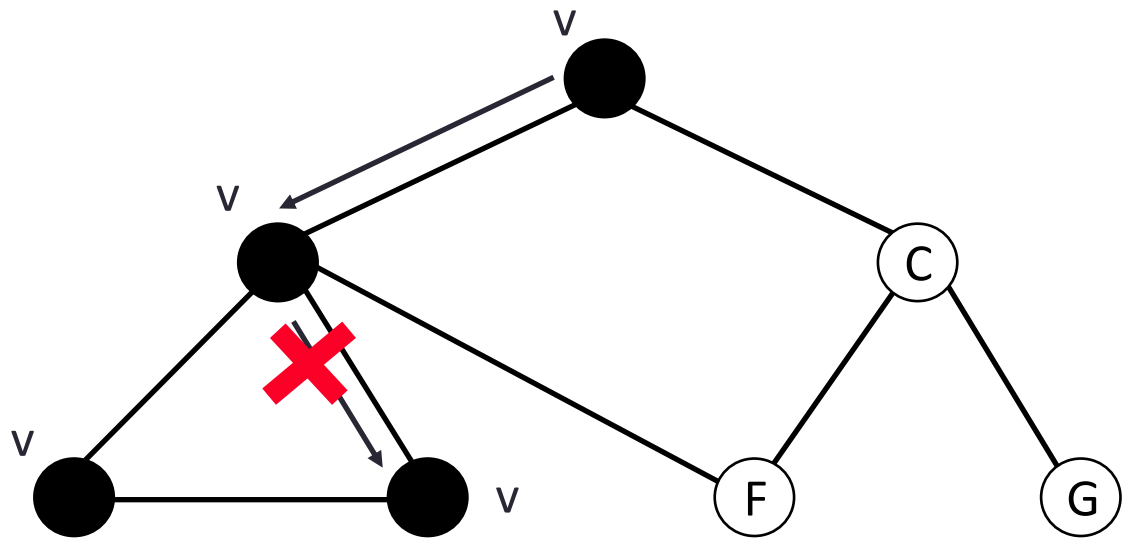
A B D E

Depth-First Search



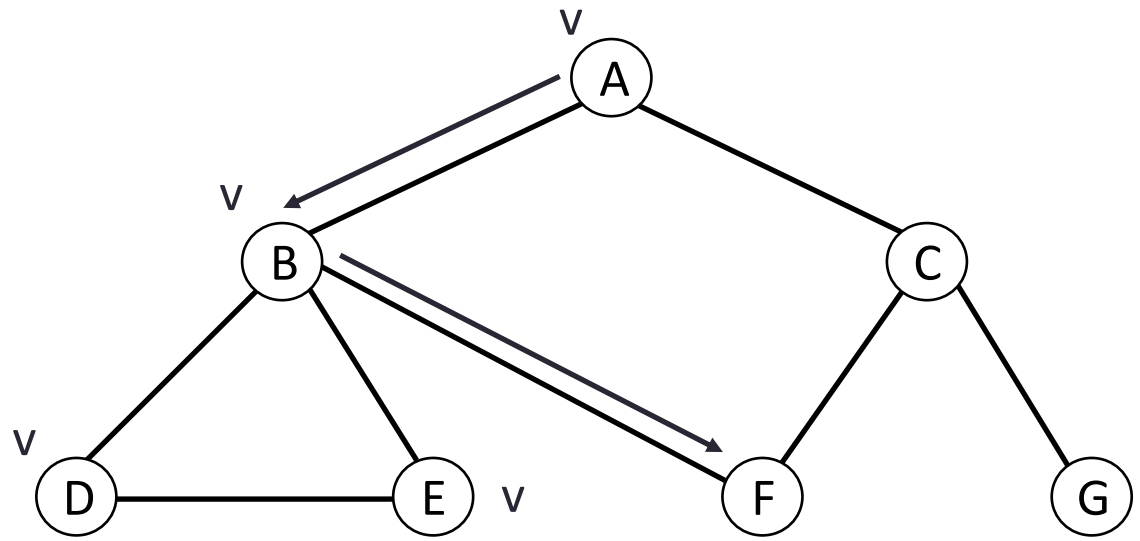
A B D E

Depth-First Search



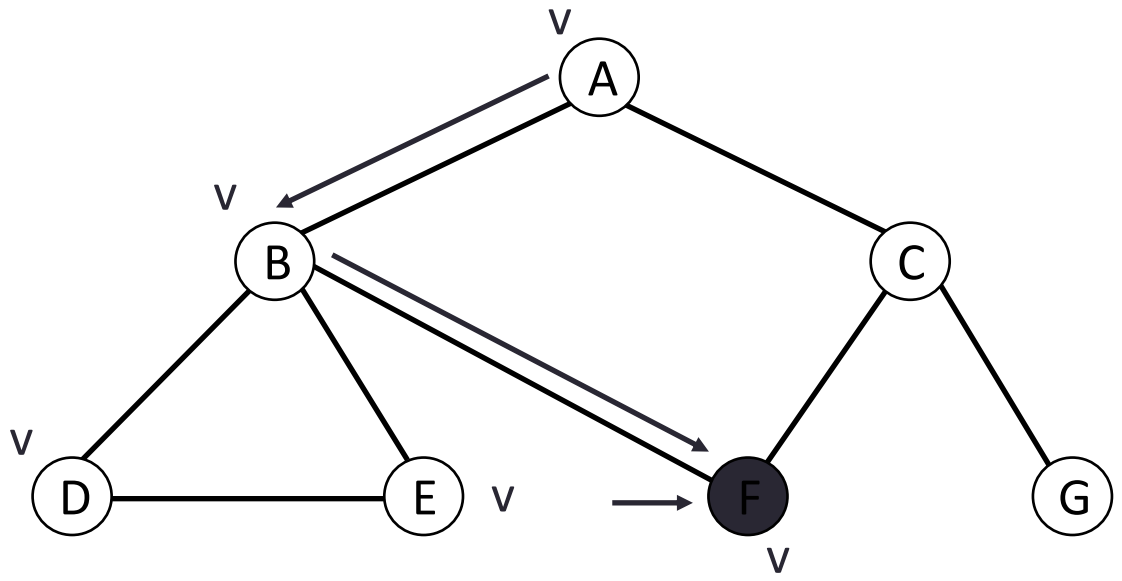
A B D E

Depth-First Search



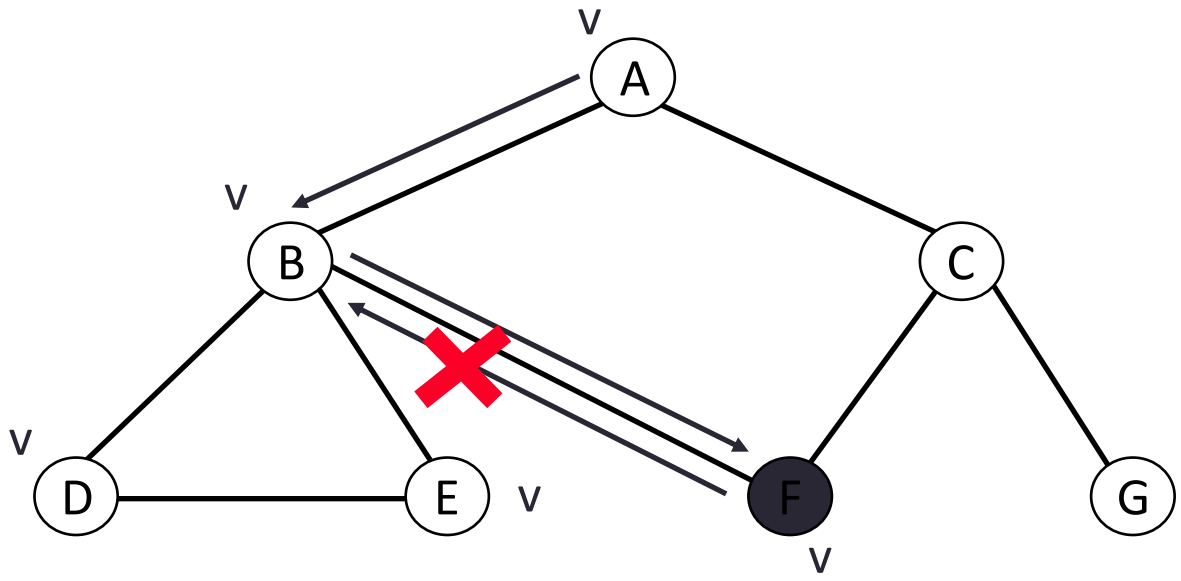
A B D E

Depth-First Search



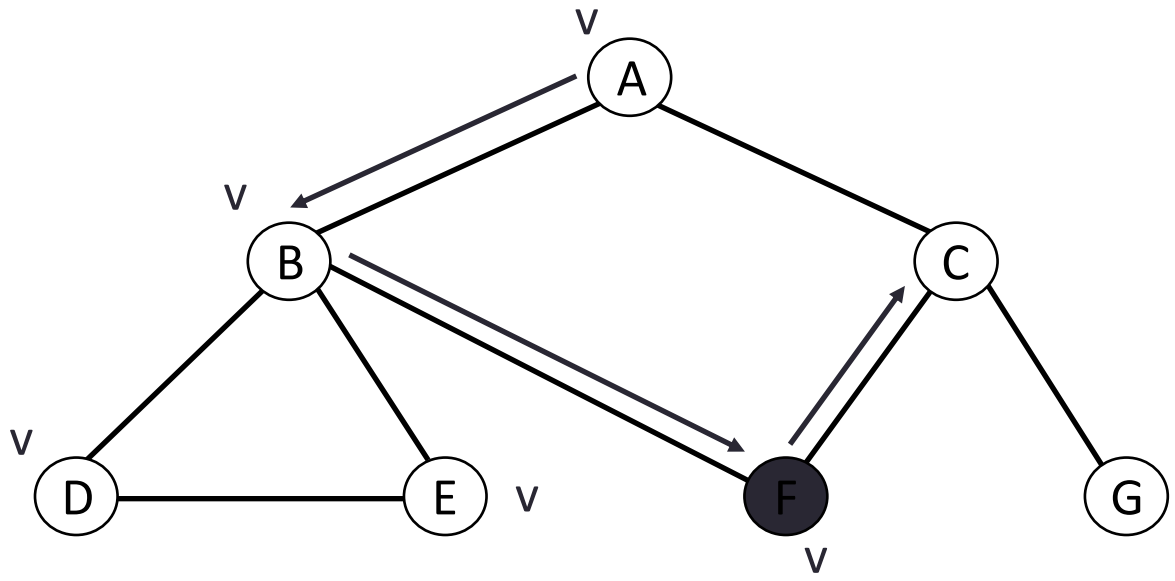
A B D E F

Depth-First Search



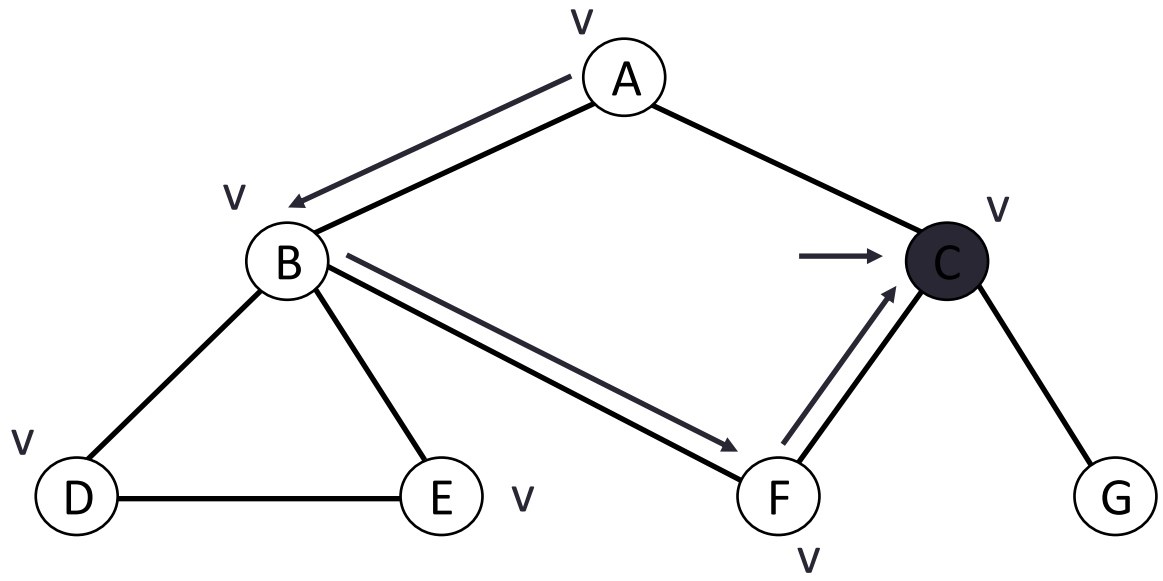
A B D E F

Depth-First Search



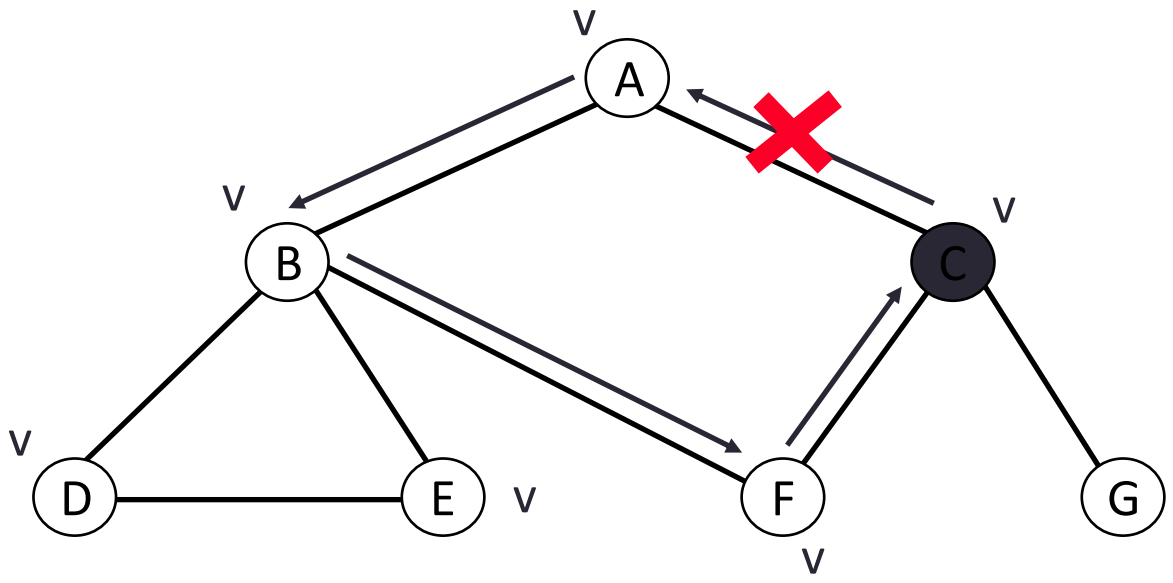
A B D E F

Depth-First Search



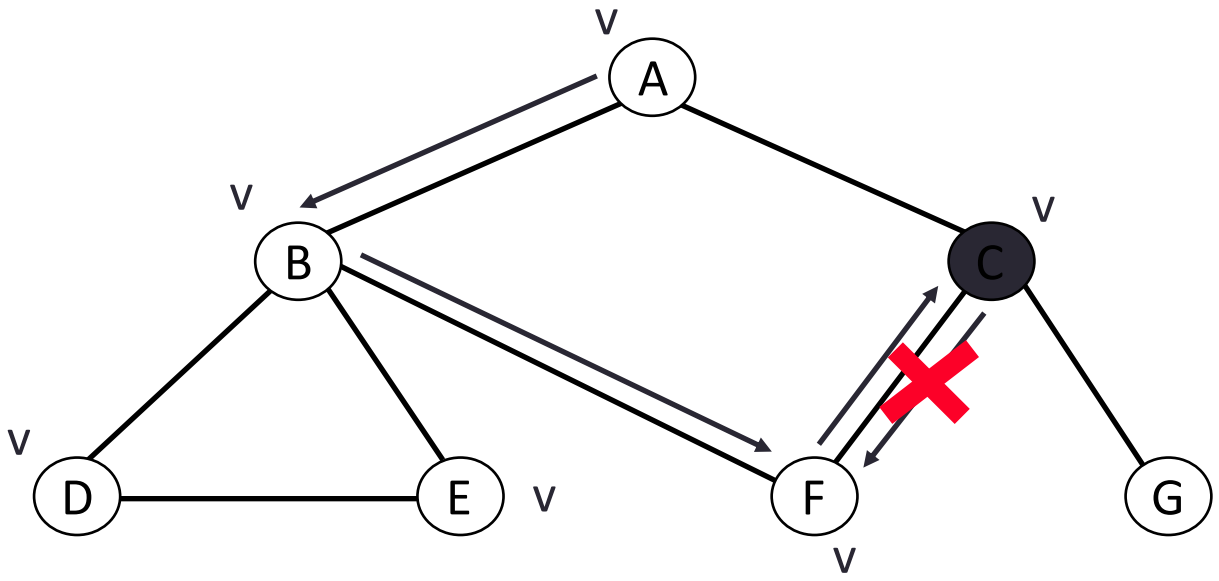
A B D E F C

Depth-First Search



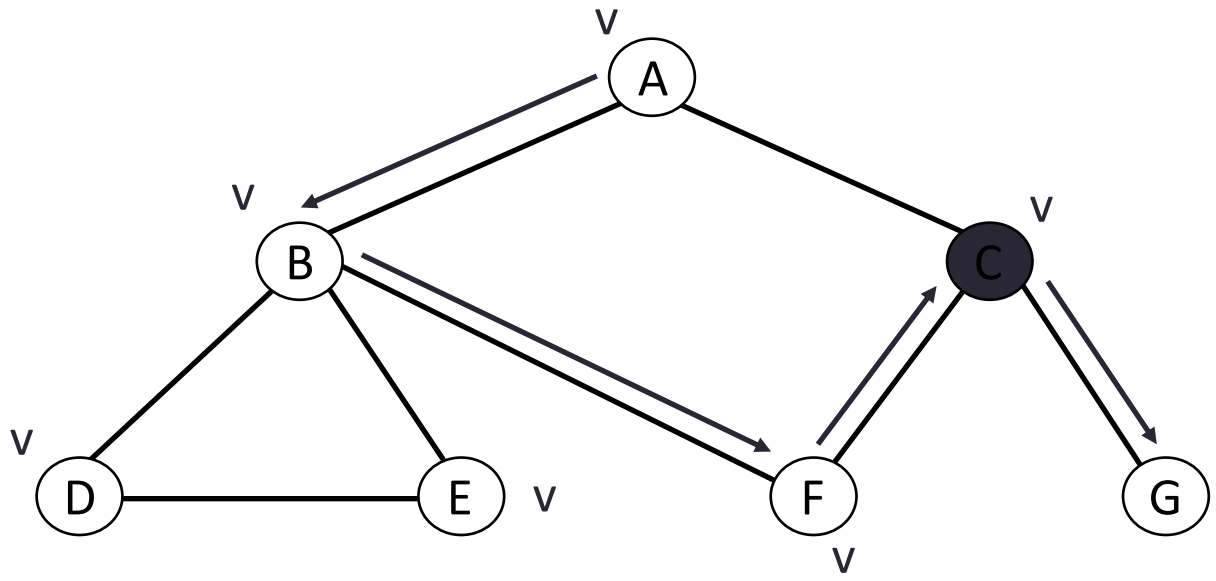
A B D E F C

Depth-First Search



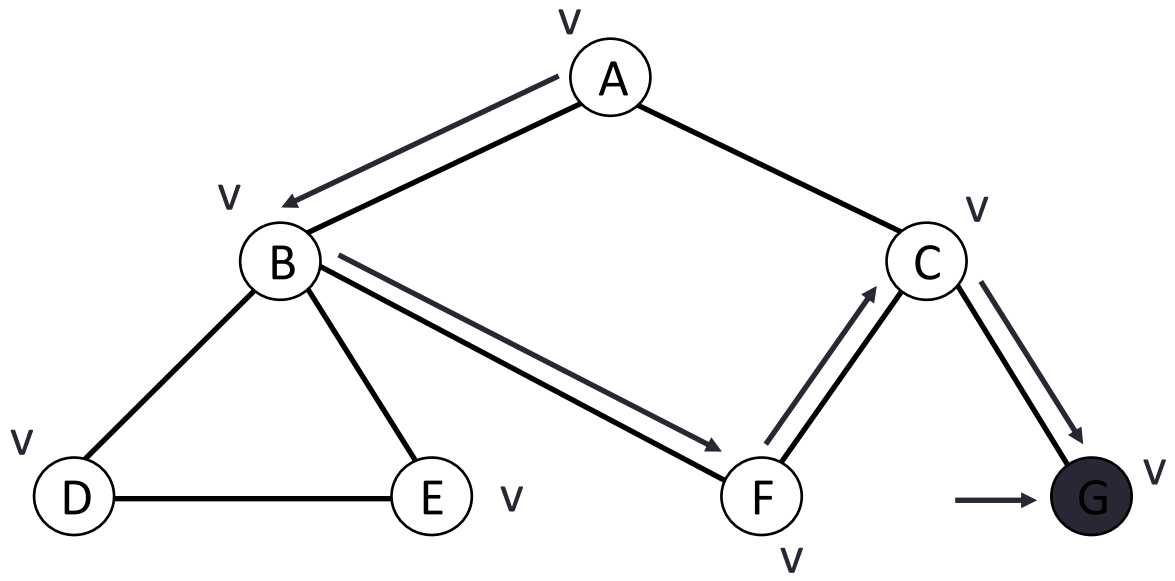
A B D E F C

Depth-First Search



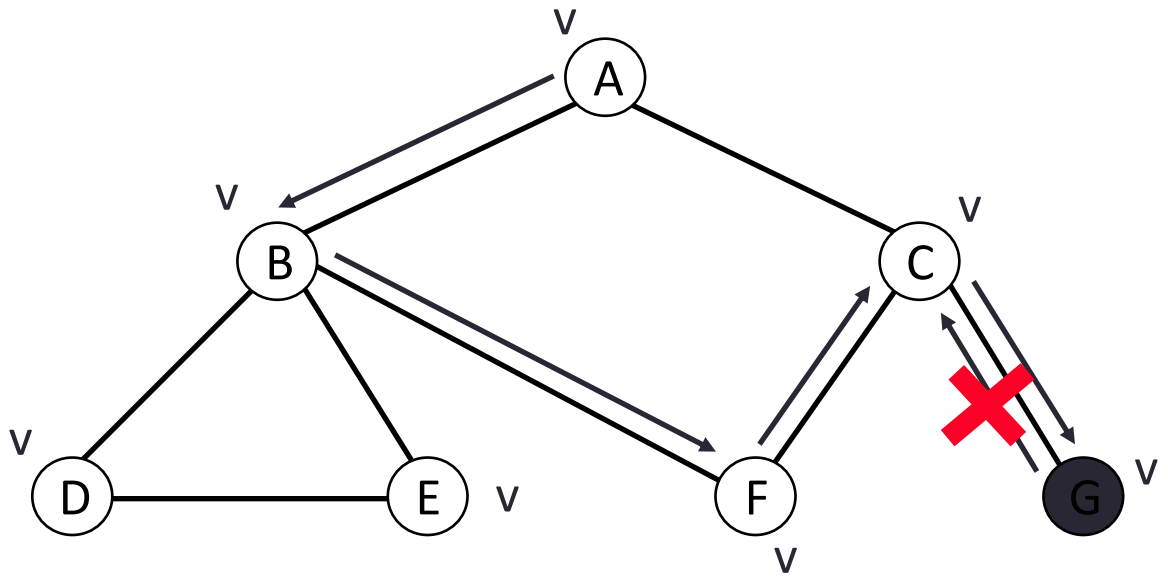
A B D E F C

Depth-First Search



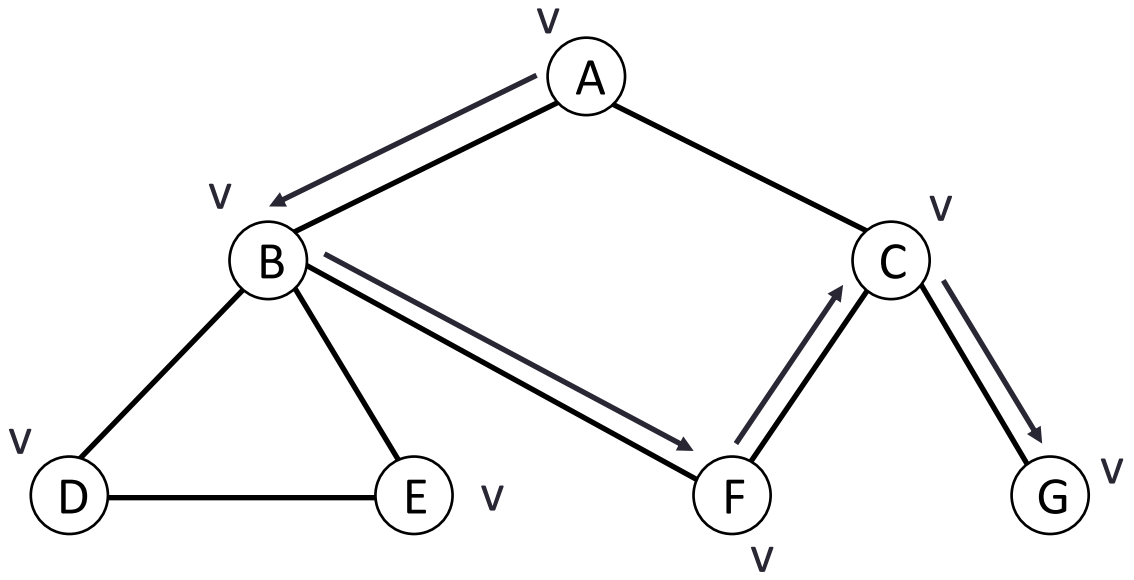
A B D E F C G

Depth-First Search



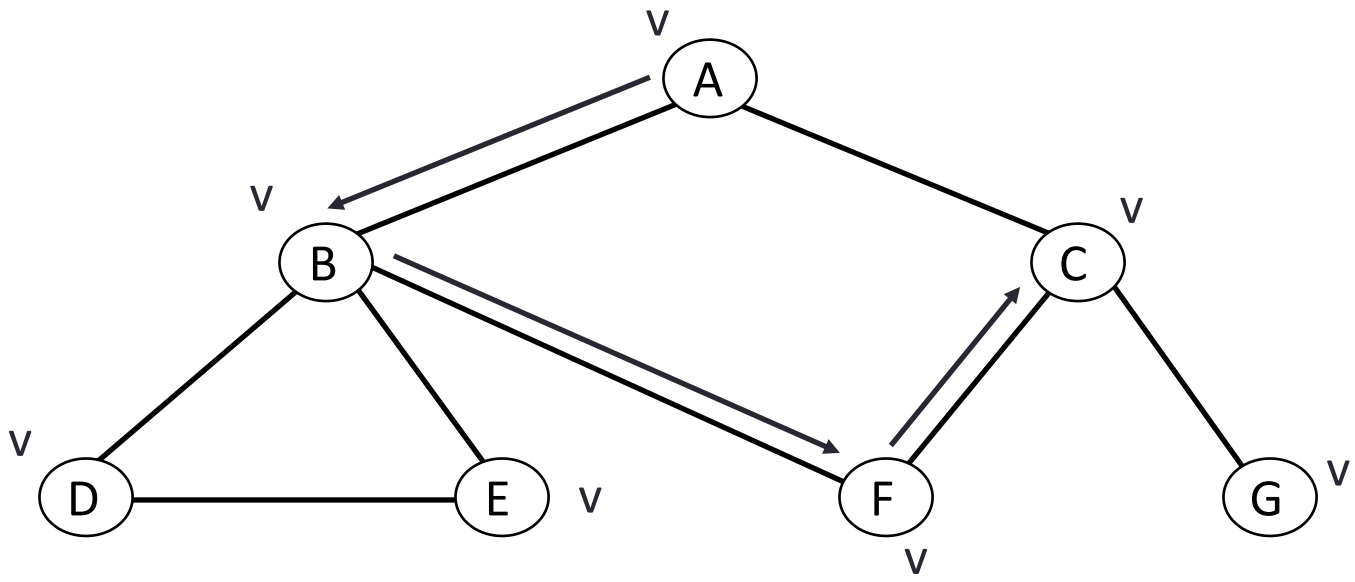
A B D E F C G

Depth-First Search



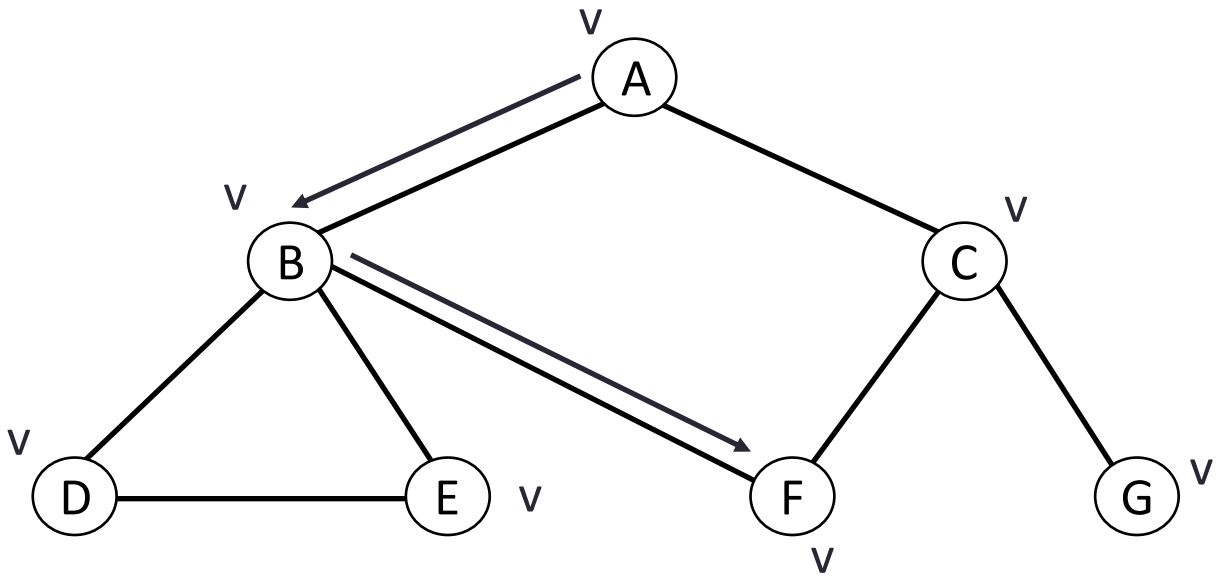
A B D E F C G

Depth-First Search



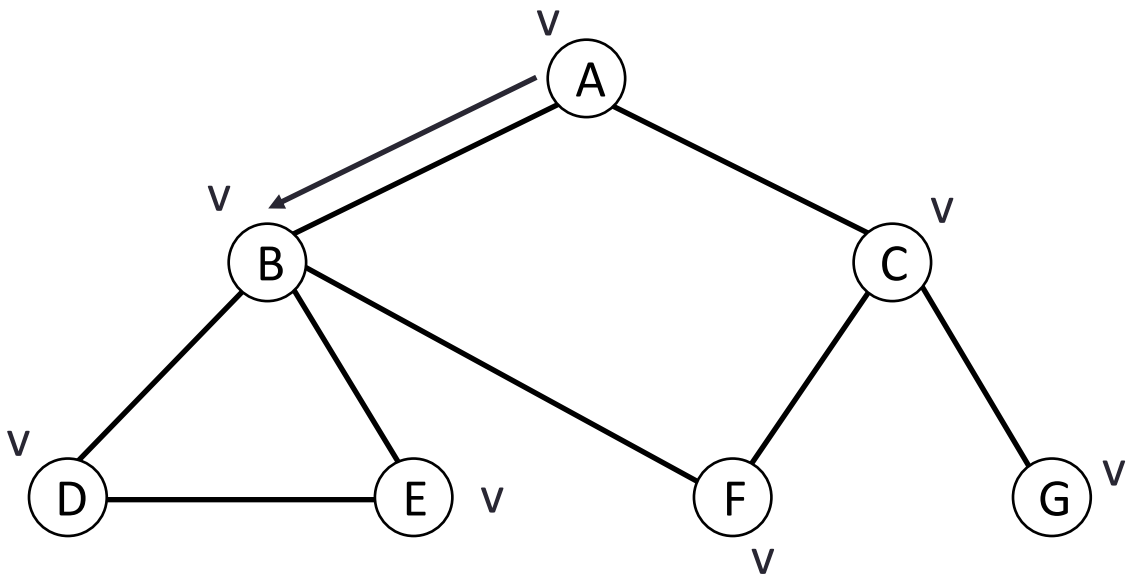
A B D E F C G

Depth-First Search



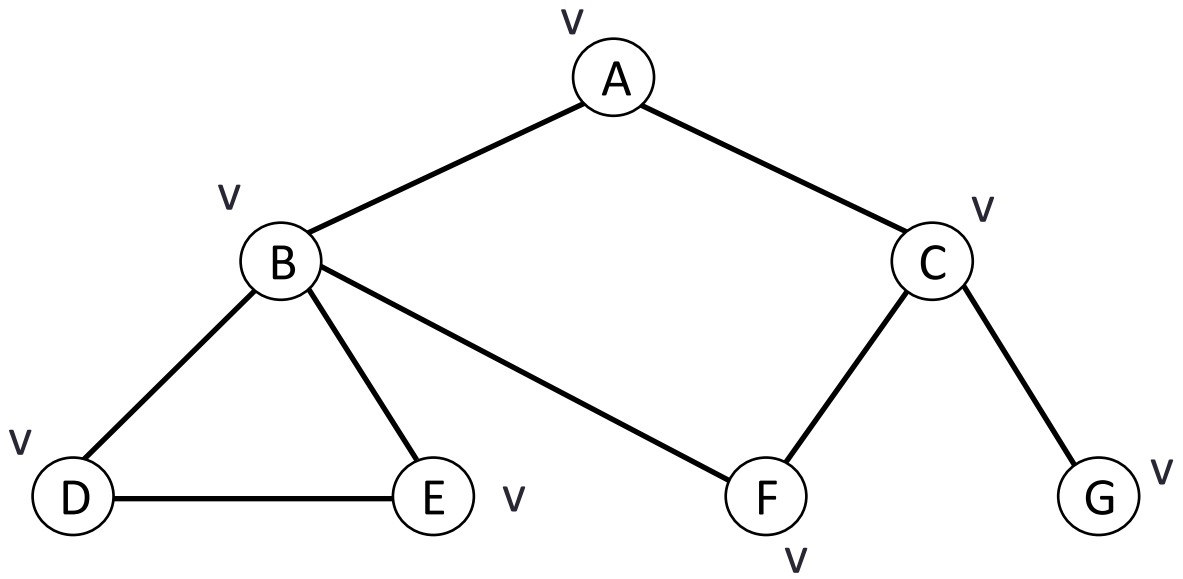
A B D E F C G

Depth-First Search



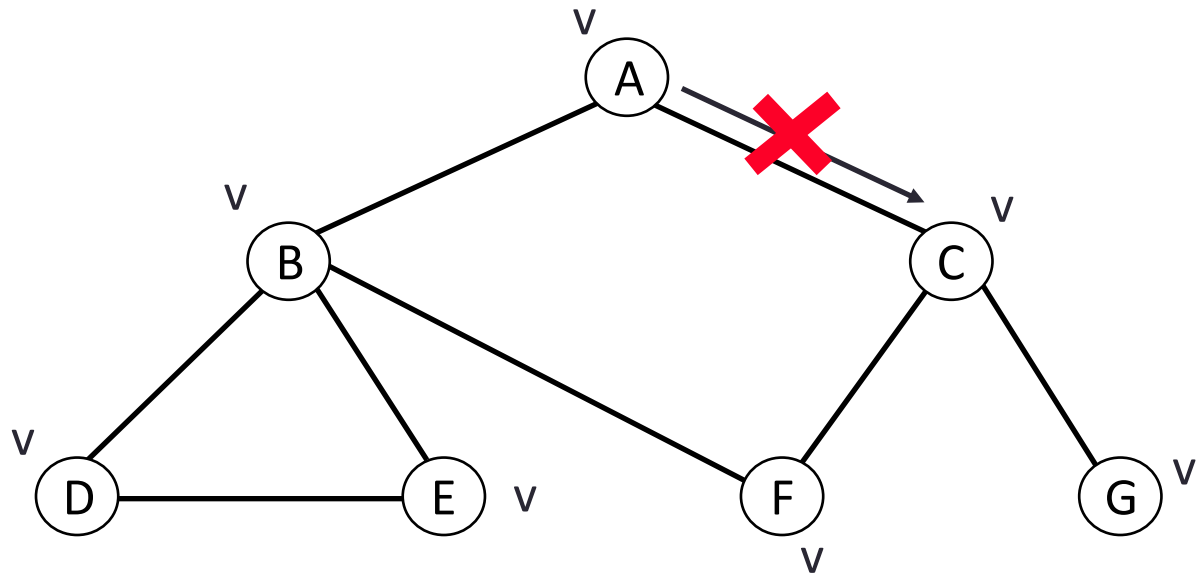
A B D E F C G

Depth-First Search



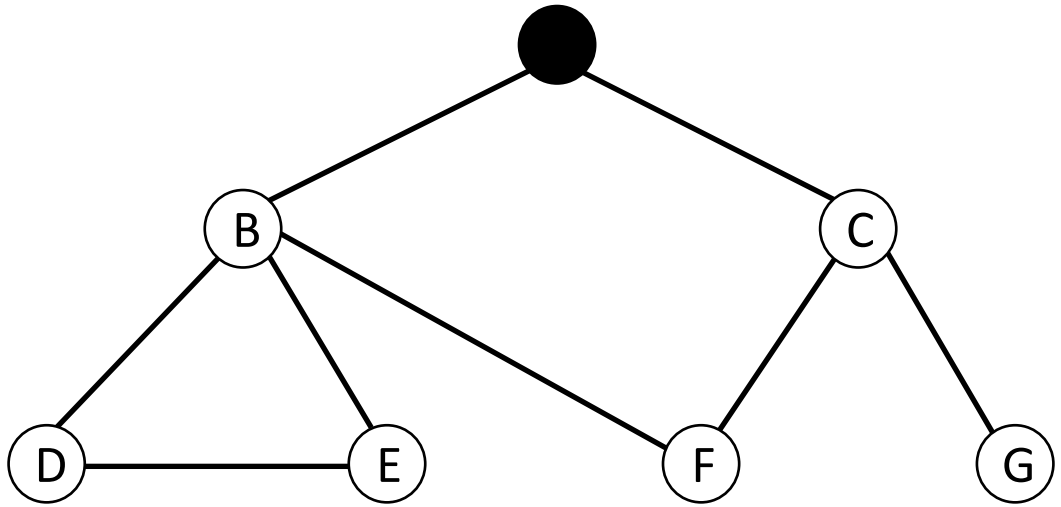
A B D E F C G

Depth-First Search



A B D E F C G

Depth-First Search



A B D E F C G

Search vs Traversal

Was this a true search?

- How would we make it a true search?

Was this a true traversal?

- How would we make it a true traversal?

Time and Space Complexity for Depth-First Search

Time Complexity

- Adjacency Lists
  - Each node is marked visited once
  - Each node is checked for each incoming edge
  - O (v + e)
- Adjacency Matrix
  - Have to check all entries in matrix: $O(n^2)$

Time and Space Complexity for Depth-First Search

Space Complexity

- Stack to handle nodes as they are explored
  - Worst case: all nodes put on stack (if graph is linear)
  - *O(n)*