Graphs

- "Graphs" are the mathematical and computer science abstraction that capture many shared and common concepts of real-life objects such as
  - networks, e.g.
    - water
    - electricity
    - internet
  - road maps
  - timetabling

Graphs

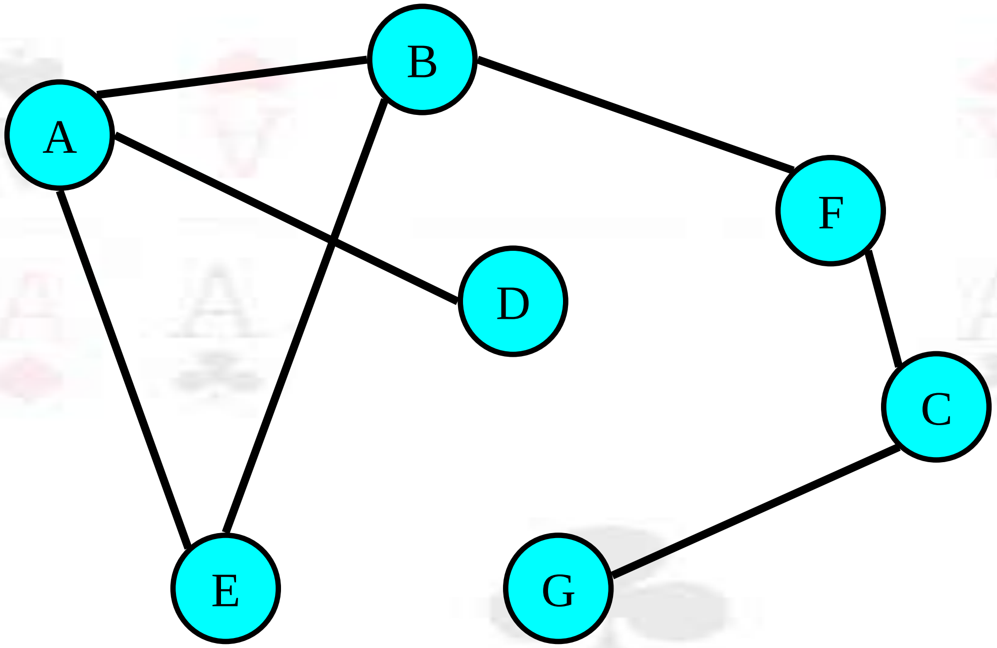
- In CS/Maths "graphs" has a specific meaning
  - e.g. is NOT the same meaning as in "plot a graph of sin(x) against x"
- Many problems in CS/AI can be reduced to problems in graph theory,
  - gives a good way to share results, methods & implementations across many areas

Terminology of Graphs

- A Graph consists of a set V of nodes, and a set E of edges
- Node
  - has a unique label for identification
- Edge
  - connects two nodes
  - any two nodes have at most one edge between them
  - usually forbid "self-edges" : edges from a node back to itself
  - can be a
    - "directed edge": e.g. "from A to B" – a "one-way street"
    - "undirected edge": e.g. "between A and B" – a "standard street"

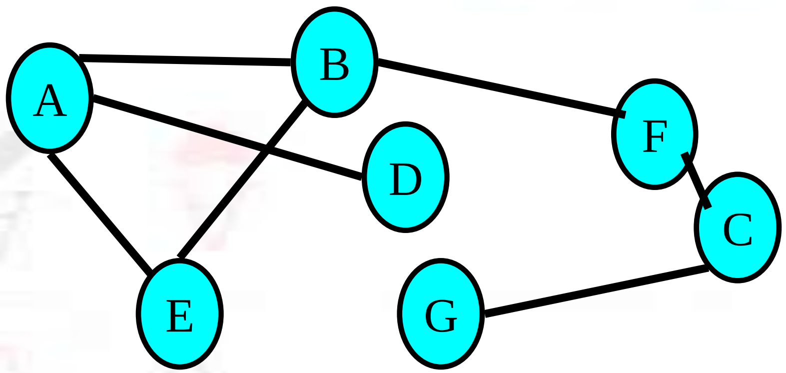
Graph Example

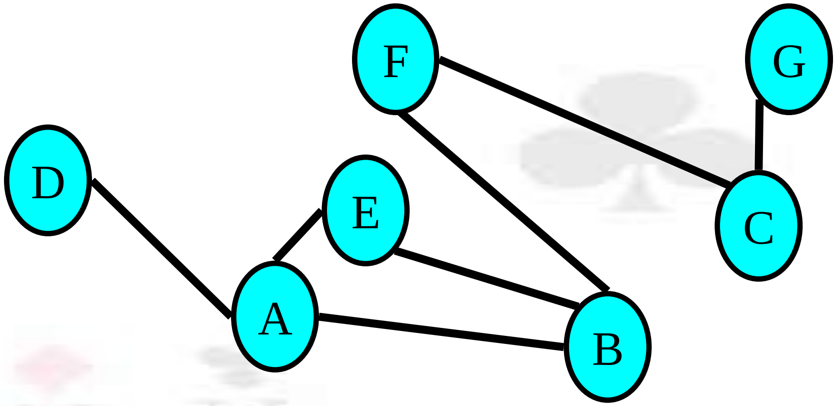
- This is not a map!
- The positions of the nodes do not matter, only their interconnections
- E.g. London Underground “Map” gives interconnections, but not true locations



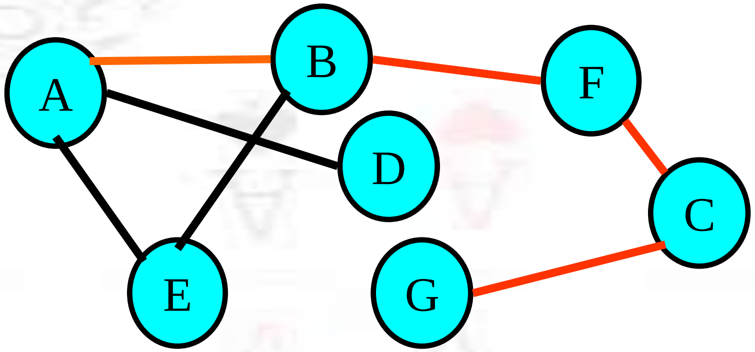
Graph Example

- Crossing of edges has no meaning
  - A-D and B-E cross in the picture but they do not interconnect
- These two pictures are the same graph





Terminology of Graphs



- Path
  - connected sequence of edges:
  - e.g. A to B to F to C to G
  - usually not allowed to use the same node (or edge) twice
  - if the edges are directed then the path has to follow the directions of the edges. E.g. follow the one-way streets on a map

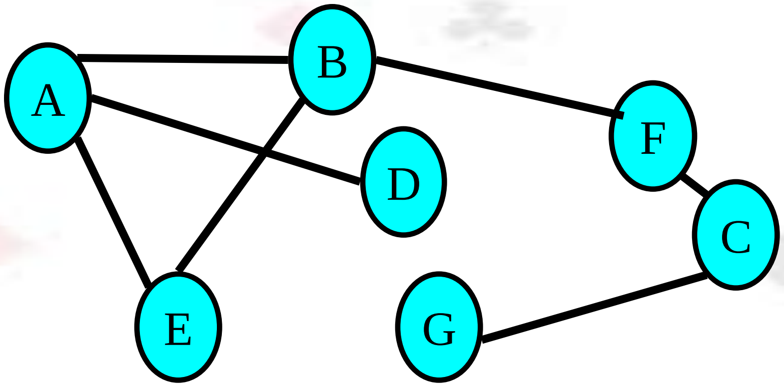
“Reachable”

Node B is said to be “reachable” from node A
if and only if
there is a path from A to B

- Relevance to real-world:
  - many search problems are questions about whether or not some “goal” is reachable from some “start” node
  - part of maintaining the internet topology is ensuring that any node (site) is reachable from any other site
  - a design factor in the internet was that nodes stay reachable even if some links are broken
    - e.g. in the event of nuclear war

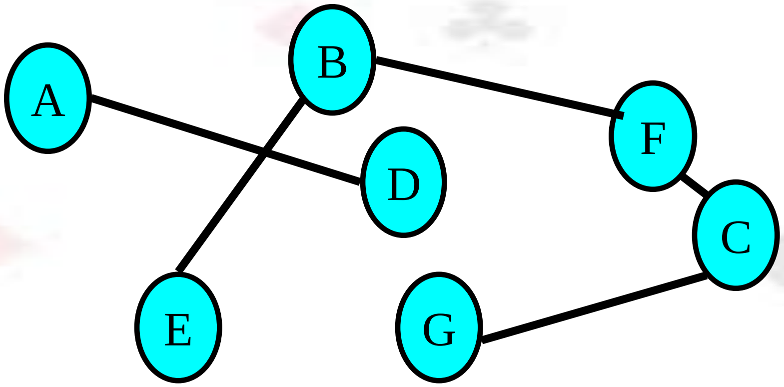
Connected Graphs

- Connected Graph. Definition:
  - for any two nodes there is a path between them
  - i.e. any node is reachable from any other node
- E.g. this graph is connected



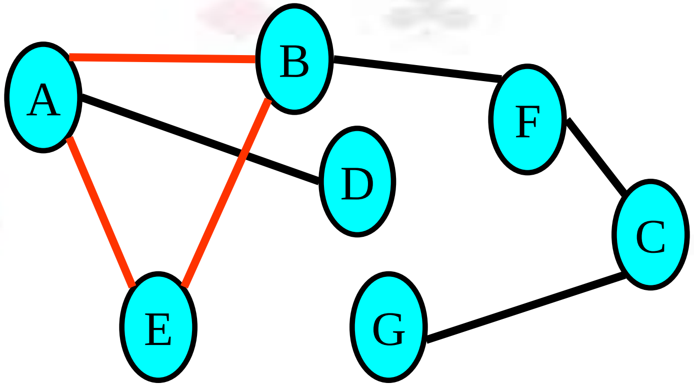
Connected Graphs

- E.g. this graph is disconnected
- There is no path from A to B



Cycles

- Definition: a cycle is a path that goes in a loop from a node back to itself, and without using the same node twice
- E.g. A-B-E-A in

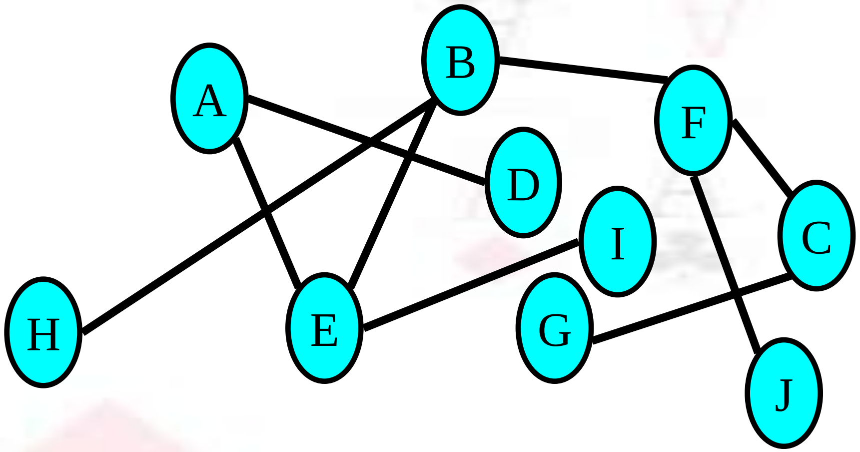


Tree

- A tree is a graph
  - that is connected
  - but becomes disconnected if you remove any edge (“cut the edge”)
- Matches “nature”
  - real trees are connected
  - cut any branch then the tree falls into two pieces

Tree: Example

- This is a tree



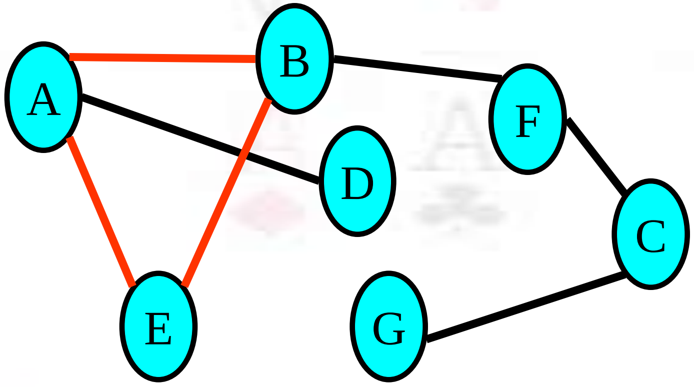
Acyclic Graphs

- Defn: An “acyclic graph” has no cycles

- Alternative definition:
- A tree is a graph that is
  - connected, and
  - acyclic

Trees

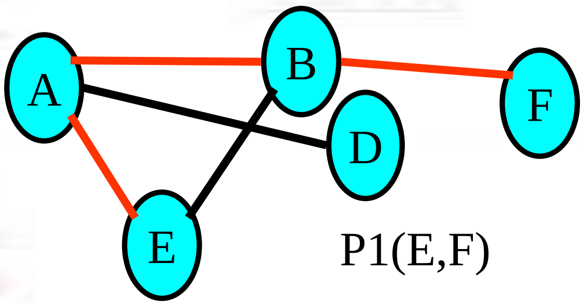
- Question: Why is a "acyclic" equivalent to "becomes disconnected on any cut"?



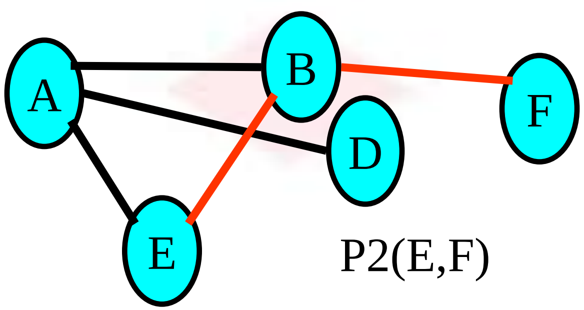
- Suppose we cut edge A-B
- Graph stays connected

Trees

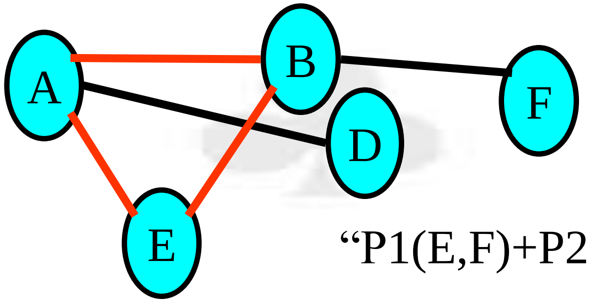
- A Tree is connected: given any two nodes A and B there exists at least one path between them
- Question: Given a tree, and any two nodes A and B, is the path between them unique? or might there be pairs of nodes that have more than one path between them?
- Suppose nodes E and F have two distinct paths P1 and P2 between them:
- Using P1 followed by reverse(P2), we would be able to construct a cycle.
- But trees are acyclic, hence, paths between nodes of a tree must be unique



P1(E,F)



P2(E,F)



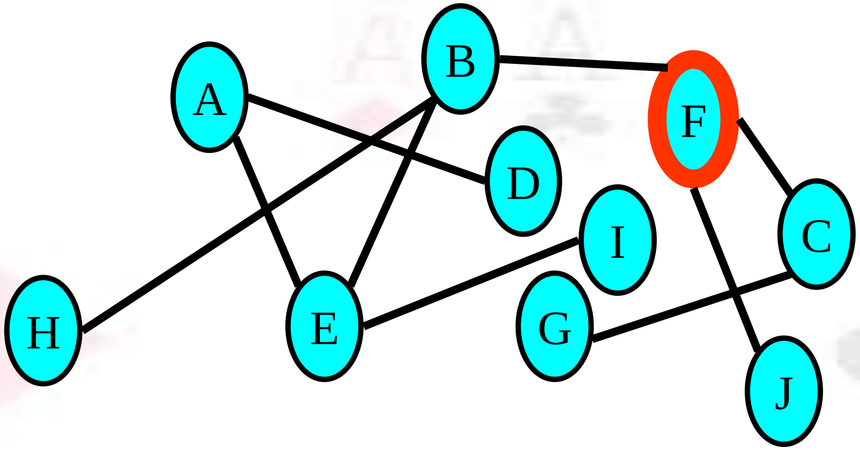
“P1(E,F)+P2(F,E)”

Trees

- A tree is a graph that:
  1. is connected but becomes disconnected on removing any edge
  2. is connected and acyclic
  3. has precisely one path between any two nodes
- The above 3 definitions are equivalent
- Suggestion: become comfortable enough with the definitions for this equivalence to be “self-evident”

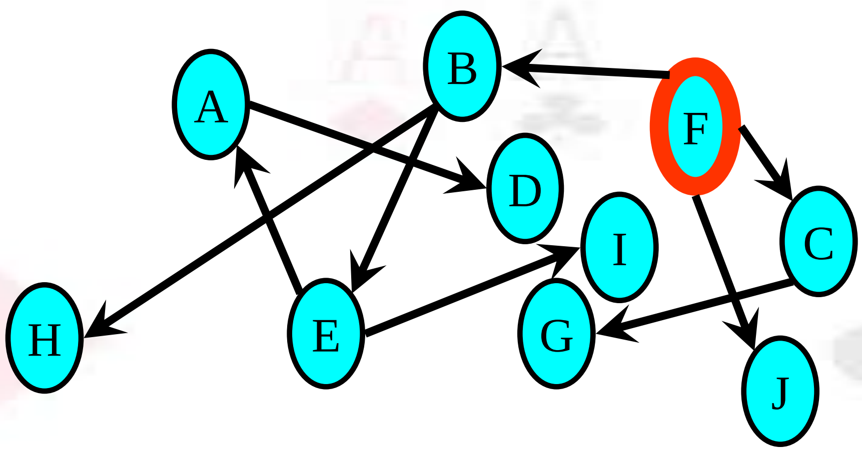
Rooted Trees

- Often in trees have a special node called the “root”
- E.g.



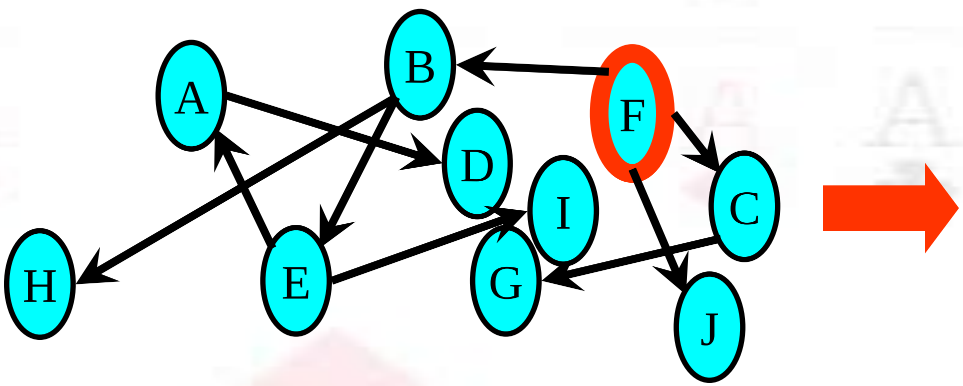
Rooted Trees

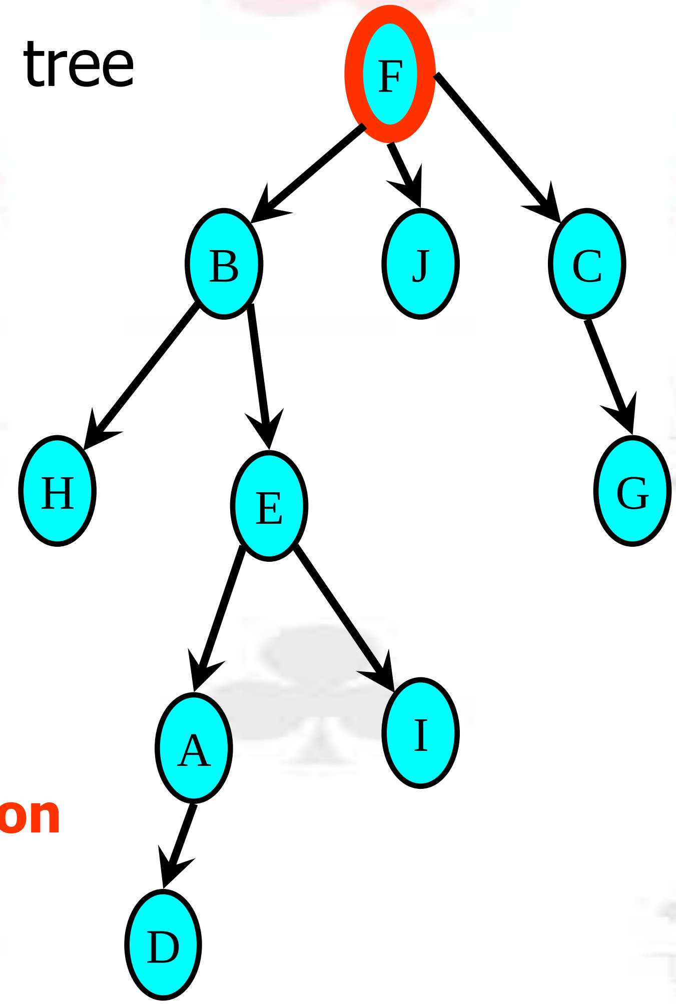
- Often in trees have a special node called the “root”
- Usually also give edges a direction away from the root
- e.g.



Rooted Trees

- Often, on making a picture of a tree
- Pick up the tree by the root
  - let it hang, and shake it a bit



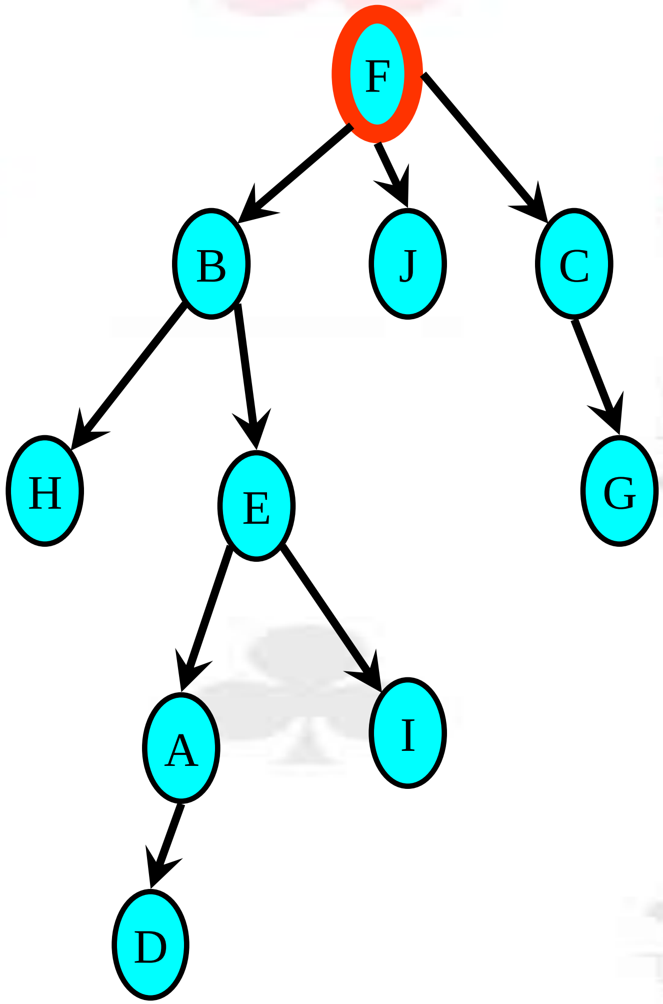


But in practice

- **you don't usually get the pretty version**
- **programs have to use "messy" representations**

Rooted Trees: Jargon

- The nodes directly below a node are called its children
  - e.g. H and E are children of B
  - B is the “parent” of H
  - H and E are “siblings”
- Children, children of children, etc are “descendents”
  - D is a descendent of B
- Parents,parents of parents, etc are “ancestors”
  - B is an ancestor of D
- A node without children is called a leaf node (or terminal node)
  - e.g. G is a leaf node



Rooted Tree : Example

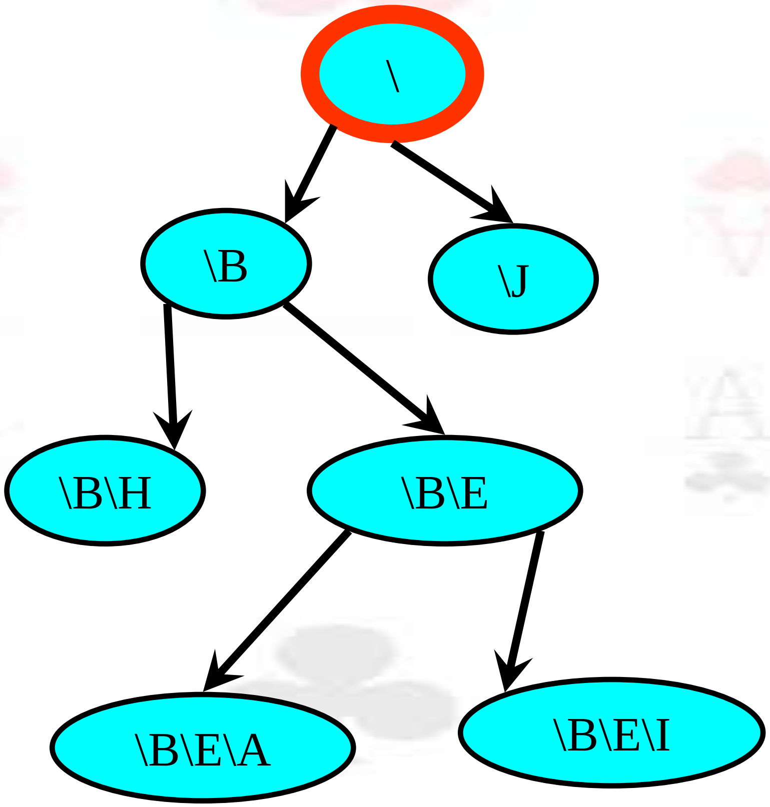
- File systems are rooted trees
- (Suggestion: When thinking/studying search and it becomes too abstract, then filesystems provide a concrete example that might make it easier to understand)

- DOS/Windows C:\ is the root (of the C drive)

- Unix "/" is the root
  - there is even a "root" user – the one that has enough privileges to modify the root directory (and all others)

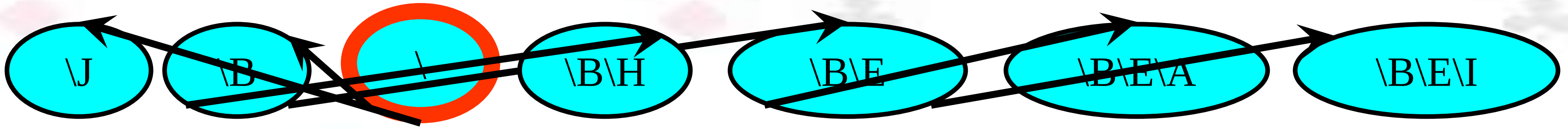
Rooted Tree : Example

- File systems are rooted trees
  - nodes = files or folders (a.k.a. directories)
  - edges = links from a folder to the files/folders that it contains
  - "links" are addresses on the hard drive



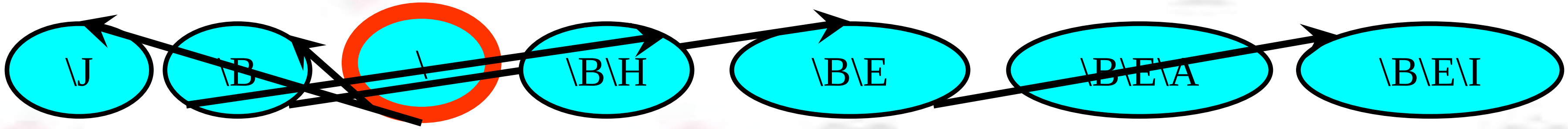
Rooted Tree : Example

- Actual layout on hard drive will be a mess



Rooted Tree : Example

- Note: there might be a lot of data on the hard drive that has no valid link to it
  - removing a file means just remove the link to it
  - can still be on the hard-drive but you cannot reach it by following links from the root directory
  - scanning the hard drive for a string does not tell you whether the string occurs in the filesystem
  - E.g. ‘del \B\E\A’ (or unix ‘rm /B/E/A’) gives

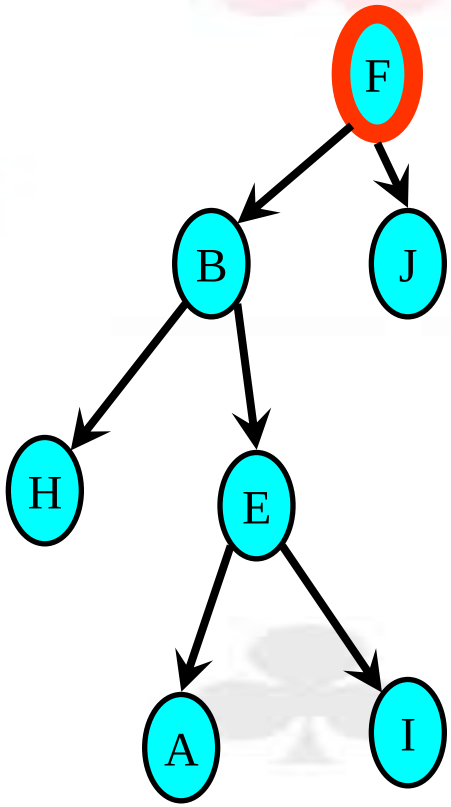


Trees: Branching factor

- The branching factor of a node is just the number of branches emerging from it – its number of children
- Branching factor of the tree itself is (usually): largest branching factor of any node
- (Might also read about an “effective branching factor” – some form of average over the nodes)

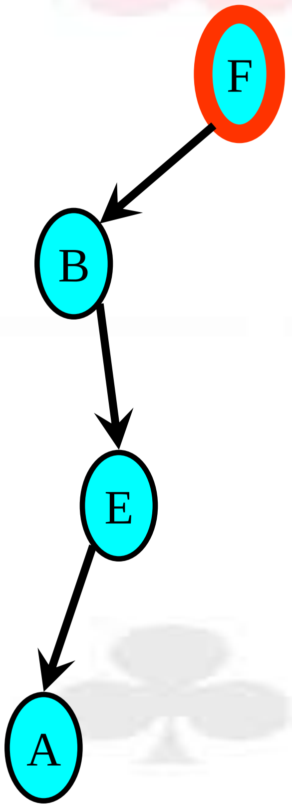
Trees: b=2

- If the largest branching factor is two then we have a “binary tree”



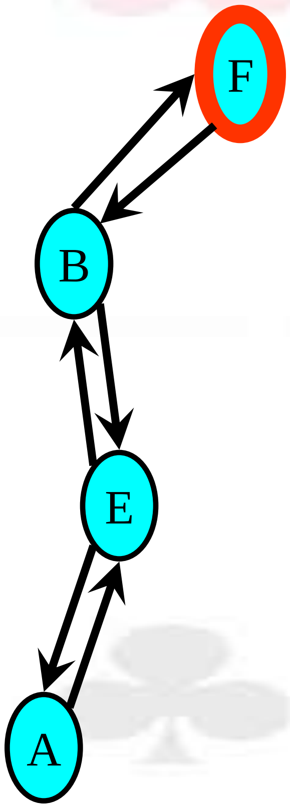
Trees: b=1

- If the largest branching factor is one then we have a “linked list”
- These are often used in programming as a “container”: a way to store a collection of objects



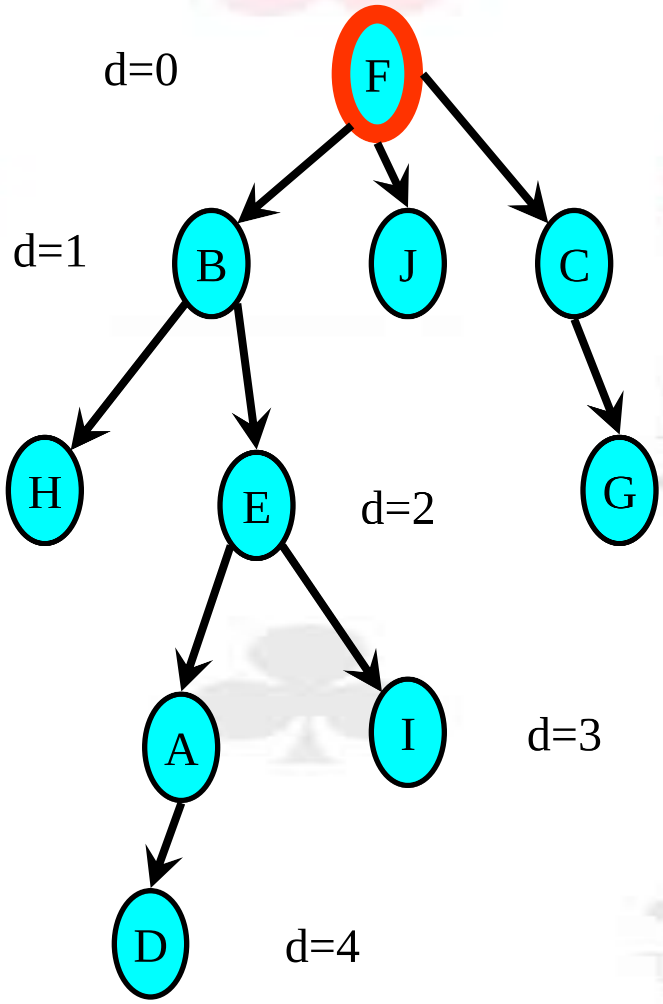
Aside: Doubly Linked Lists

- Take a linked list and also add edges from each (non-root) node to its parent
- Used in programming as a "container":
  - can easily "walk" in both directions



Trees: Depth

- The depth, d, of a node is just the number of edges it is away from the root node
- The depth of a tree is the depth of the deepest node
  - in this case, depth=4



Tree Sizes

- Suppose we have a tree with
  - branching factor b
  - depth d
- What is the maximum number of nodes it can have?

Tree Sizes

- Suppose branching factor b = 2

d	nodes at d, 2^d	nodes at d or less
0	1	1
1	2	3
2	4	7
3	8	15
4	16	31
5	32	63
6	64	127

Sizes of Trees

- Pattern seen in binary tree:
  "nodes at d or less" = "nodes at d+1" – 1
  $= 2^{d+1}-1 = O(2^d)$
- The number of nodes grows exponentially
- Another manifestation of the "Combinatorial Explosion"

- Similarly, for a general tree
  - number of nodes is $O(b^d)$
  - but, remember "big O" gives an upper bound,
  - real trees might have a lot fewer nodes

- Trees are generally very "leaf-heavy" – a large fraction of the nodes are leaves
  - e.g. on your file system you probably have far more files than folders

Sizes of Trees: Branching Factor

Increasing b rapidly increases the tree size:

d	nodes at d, b=2, 2^d	nodes at d, b=3, 3^d
0	1	1
1	2	3
2	4	9
3	8	27
4	16	81
5	32	243
6	64	729

Effects of Tree Sizes

- If possible try to work with trees with
  - smaller branching factor
  - smaller depth
- Be careful when programming that your memory requirements do not explode

Summary

- Definitions of
  - graphs
  - path
  - connected
  - cycles, and acyclic
  - tree, and rooted tree
- Trees grow exponentially with depth

Questions?