CS2001
Foundations of Computation
Lecture 11

Trees

Jon Lewis (JC 0.26)
School of Computer Science
University of St Andrews

Topics

- The tree data structure
- Implementation
- Complexity

Trees

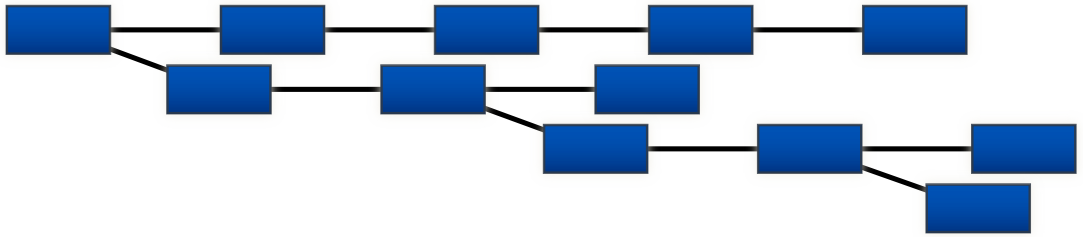
- Generalisation of a linked list
  - each node can have multiple successors

Trees

- Generalisation of a linked list
  - each node can have multiple successors

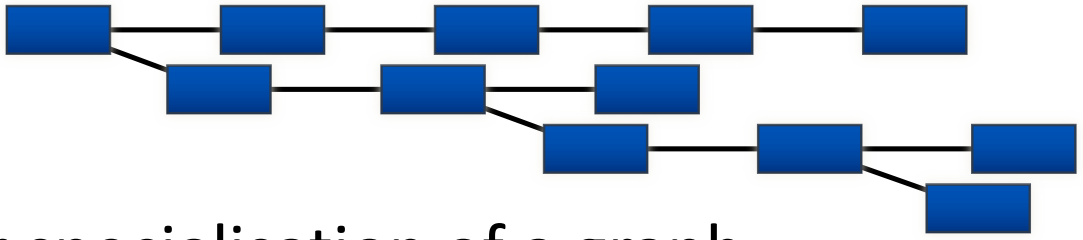
Trees

- Generalisation of a linked list
  - each node can have multiple successors



Trees

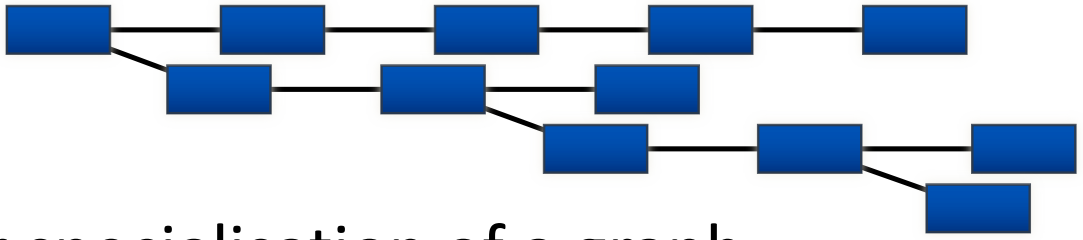
- Generalisation of a linked list
  - each node can have multiple successors



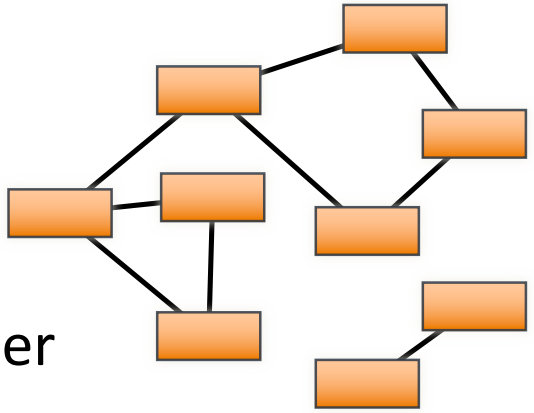
- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

Trees

- Generalisation of a linked list
  - each node can have multiple successors

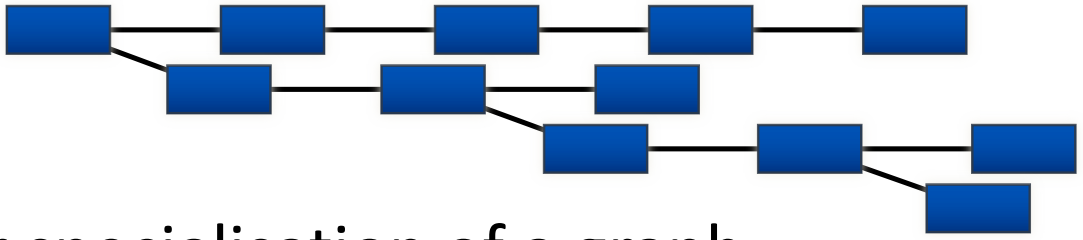


- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

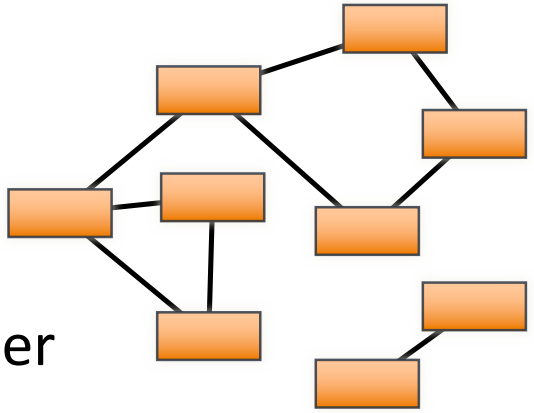


Trees

- Generalisation of a linked list
  - each node can have multiple successors

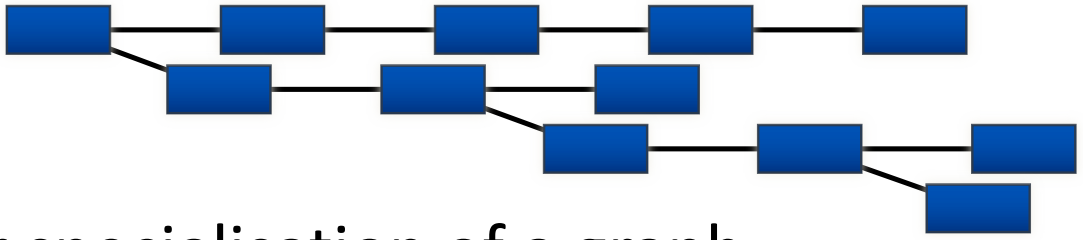


- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

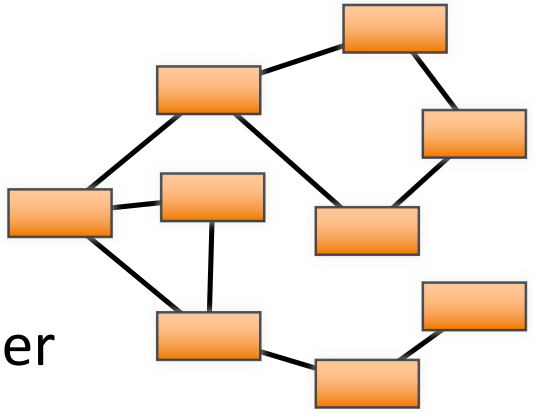


Trees

- Generalisation of a linked list
  - each node can have multiple successors

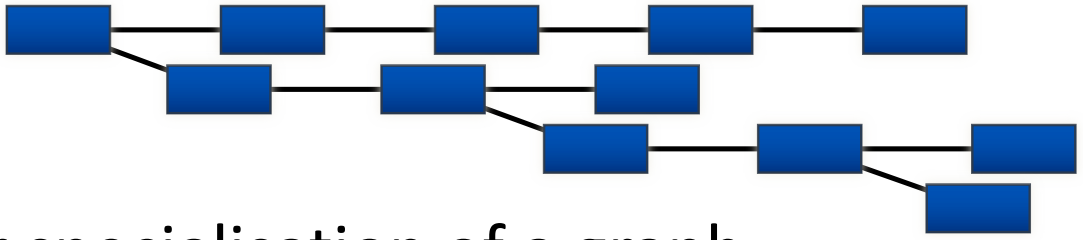


- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

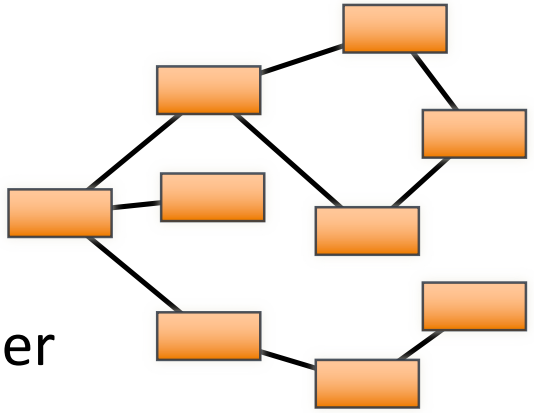


Trees

- Generalisation of a linked list
  - each node can have multiple successors

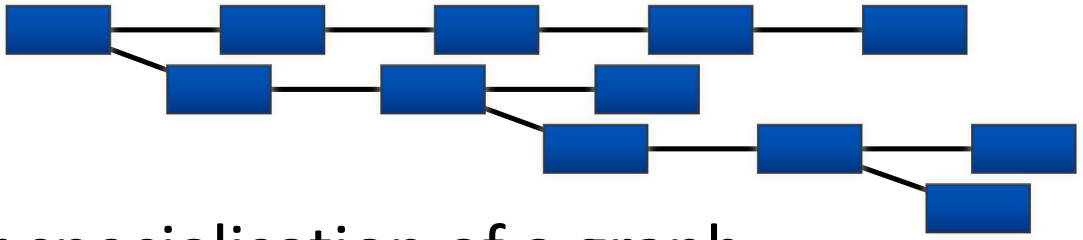


- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

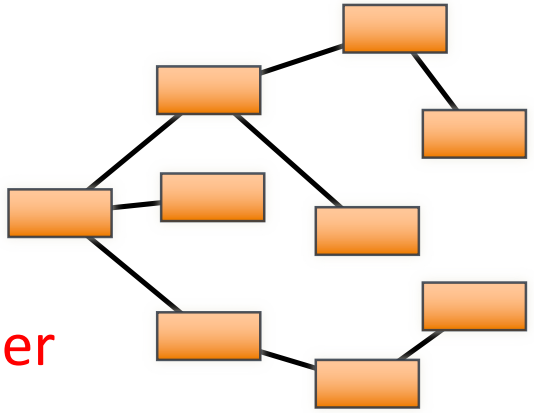


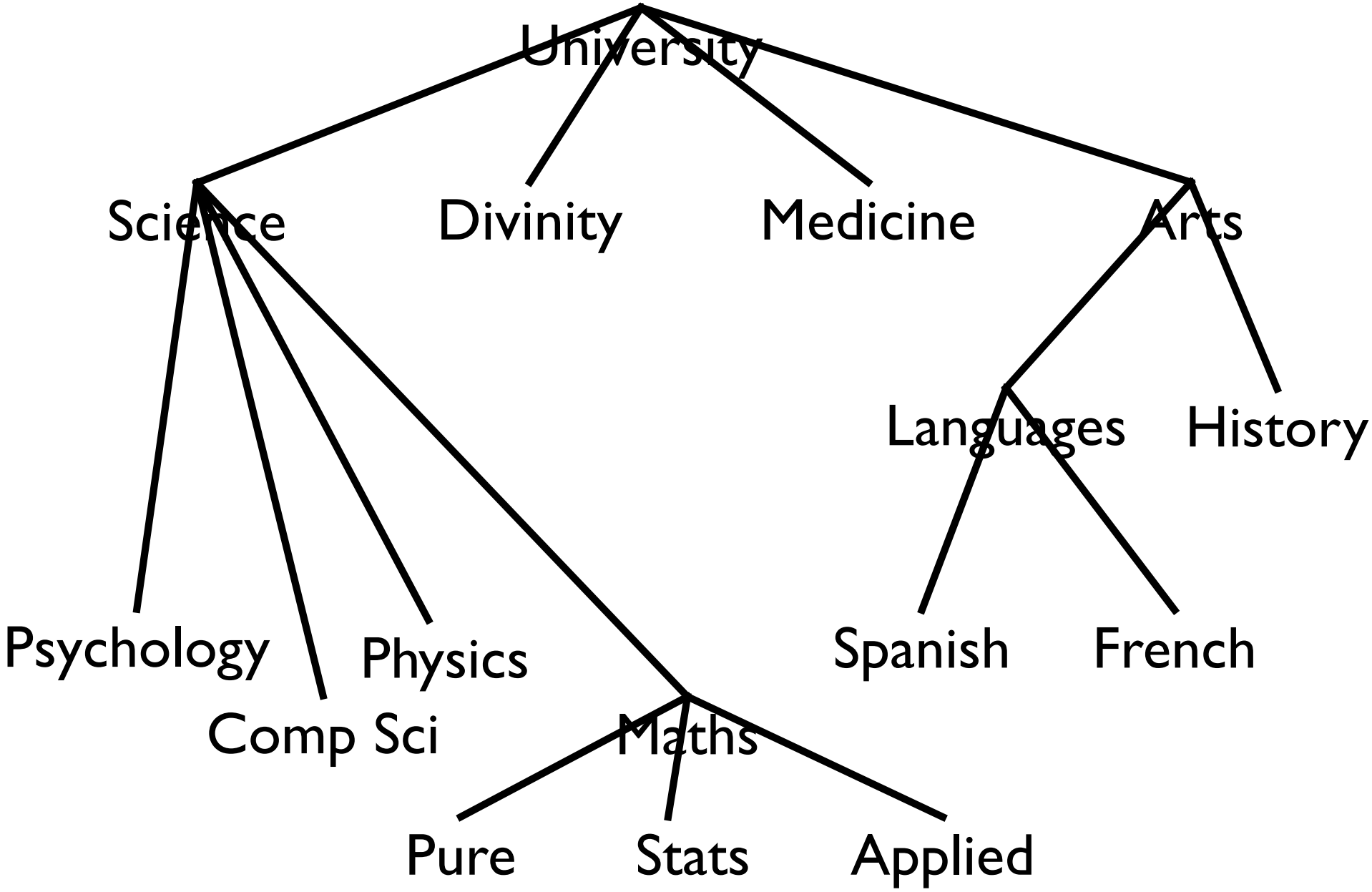
Trees

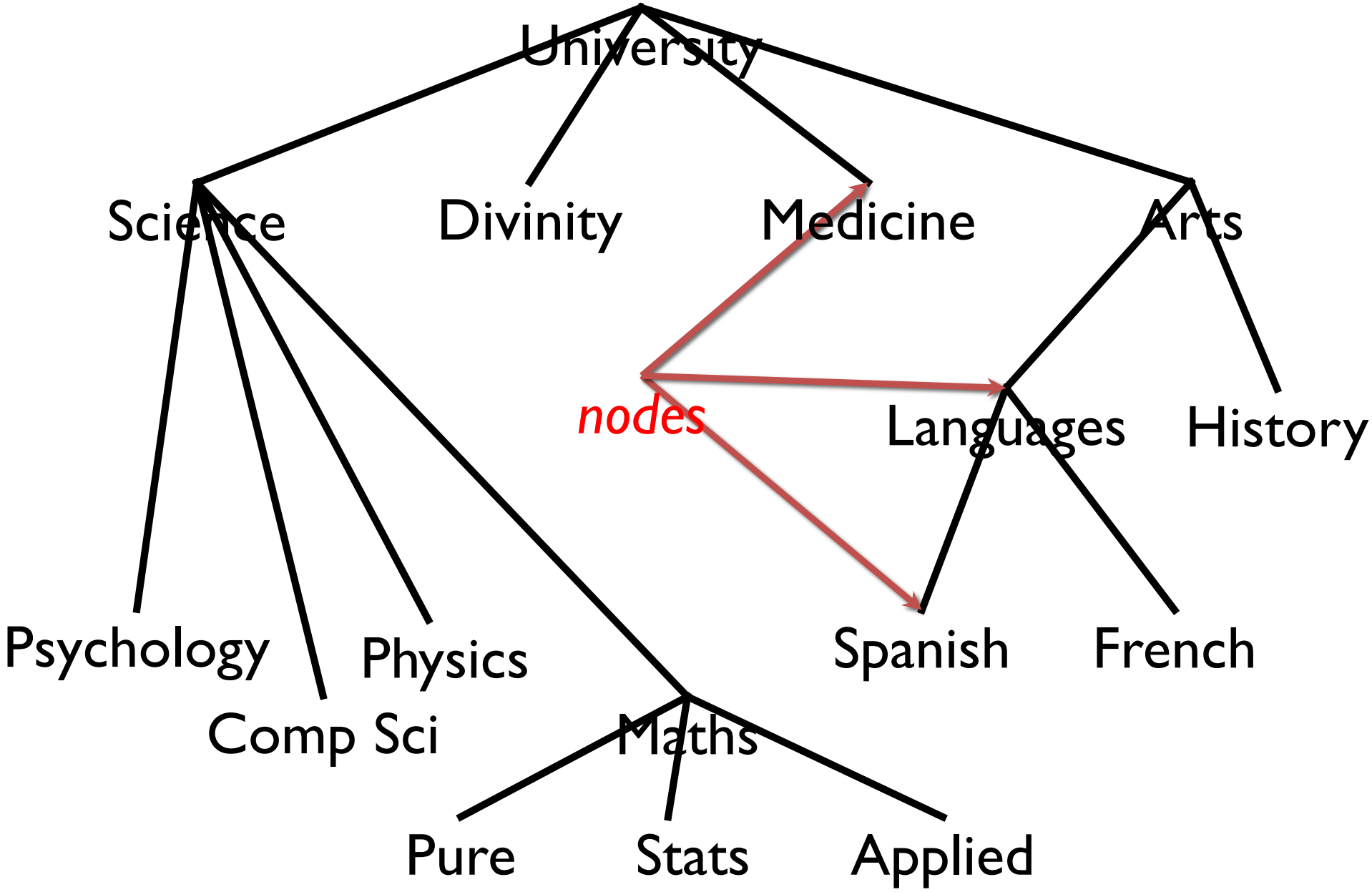
- Generalisation of a linked list
  - each node can have multiple successors

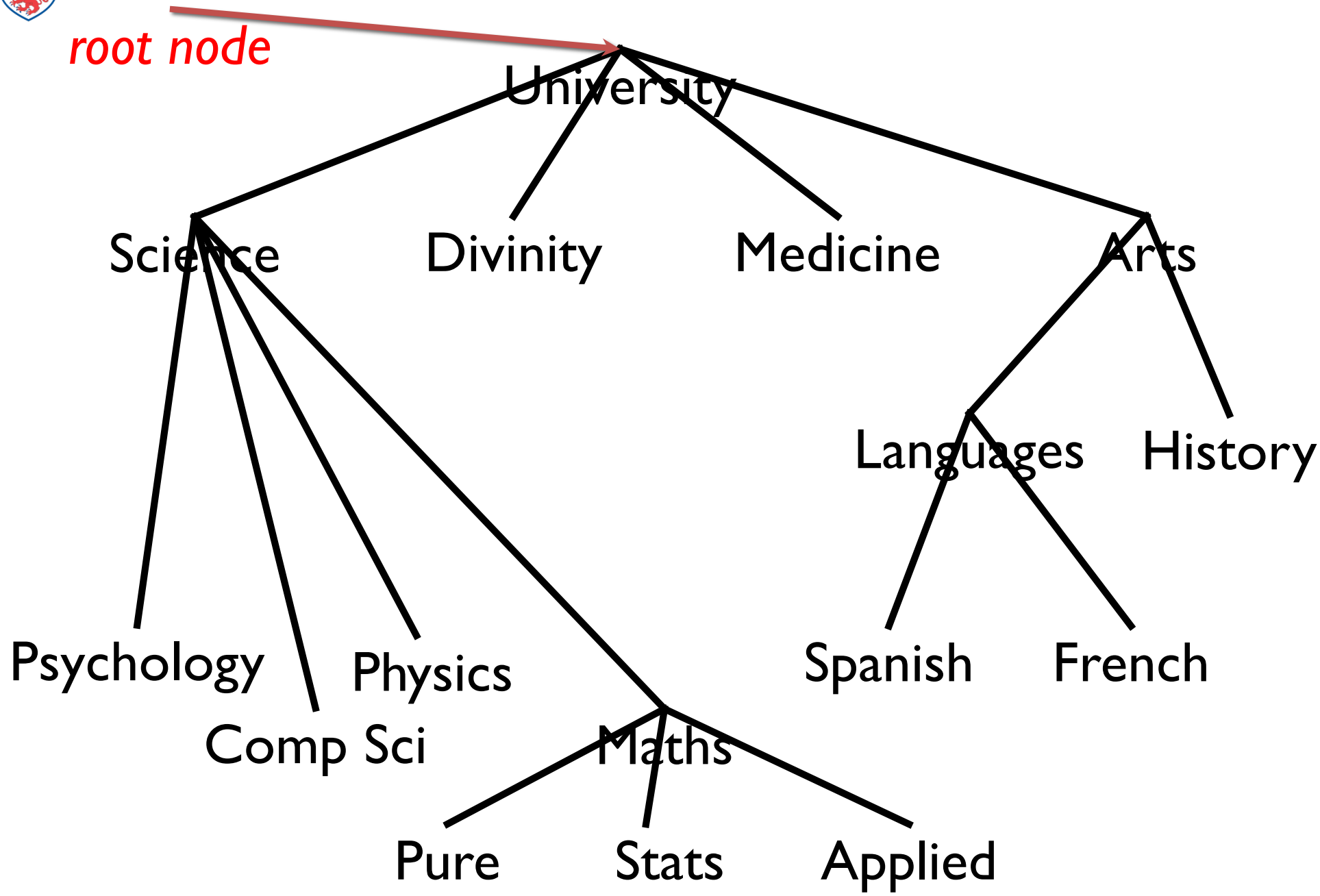


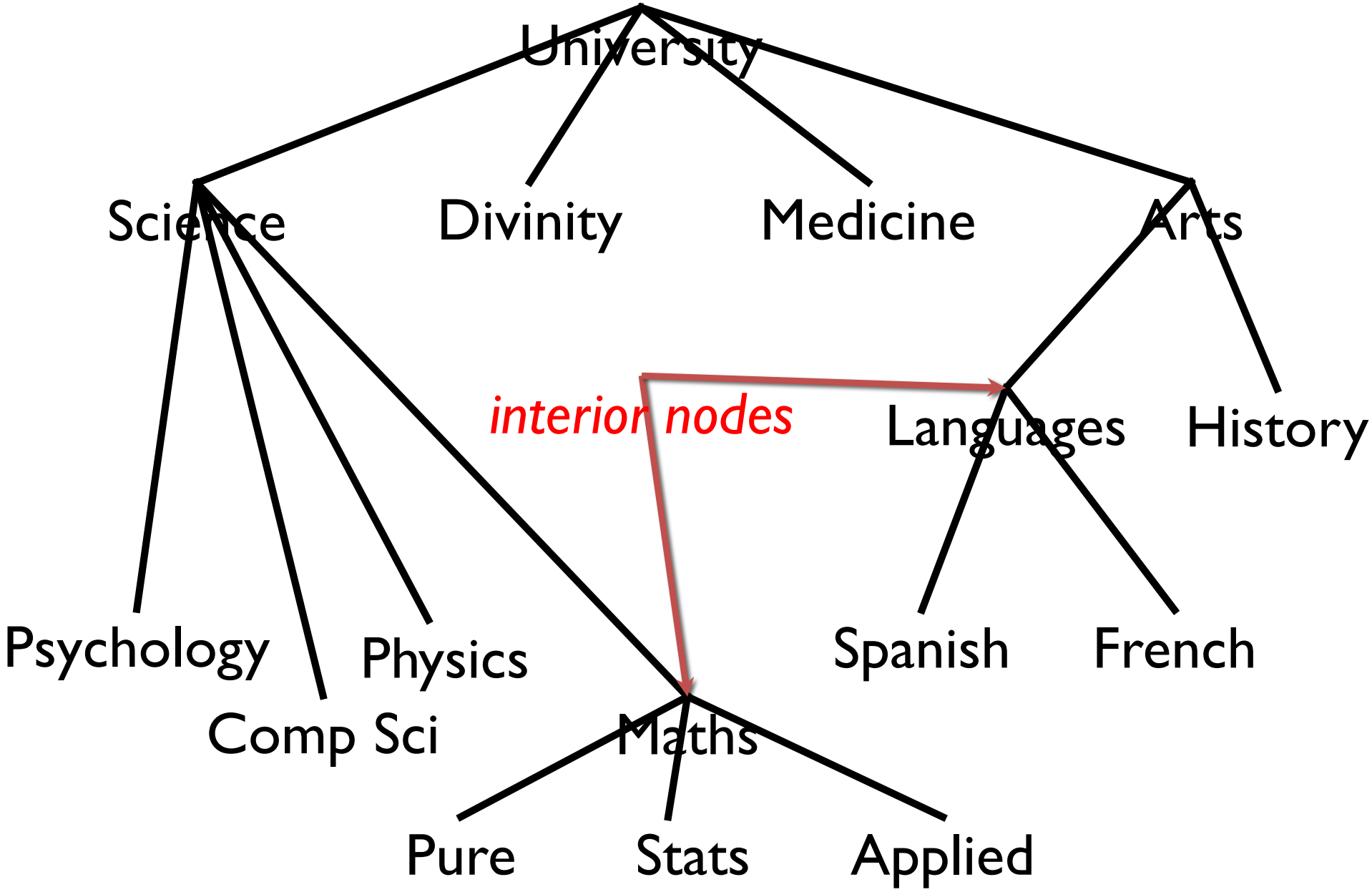
- Or specialisation of a graph
  - directed, connected
  - no cycles
  - each node pointed at by at most one other

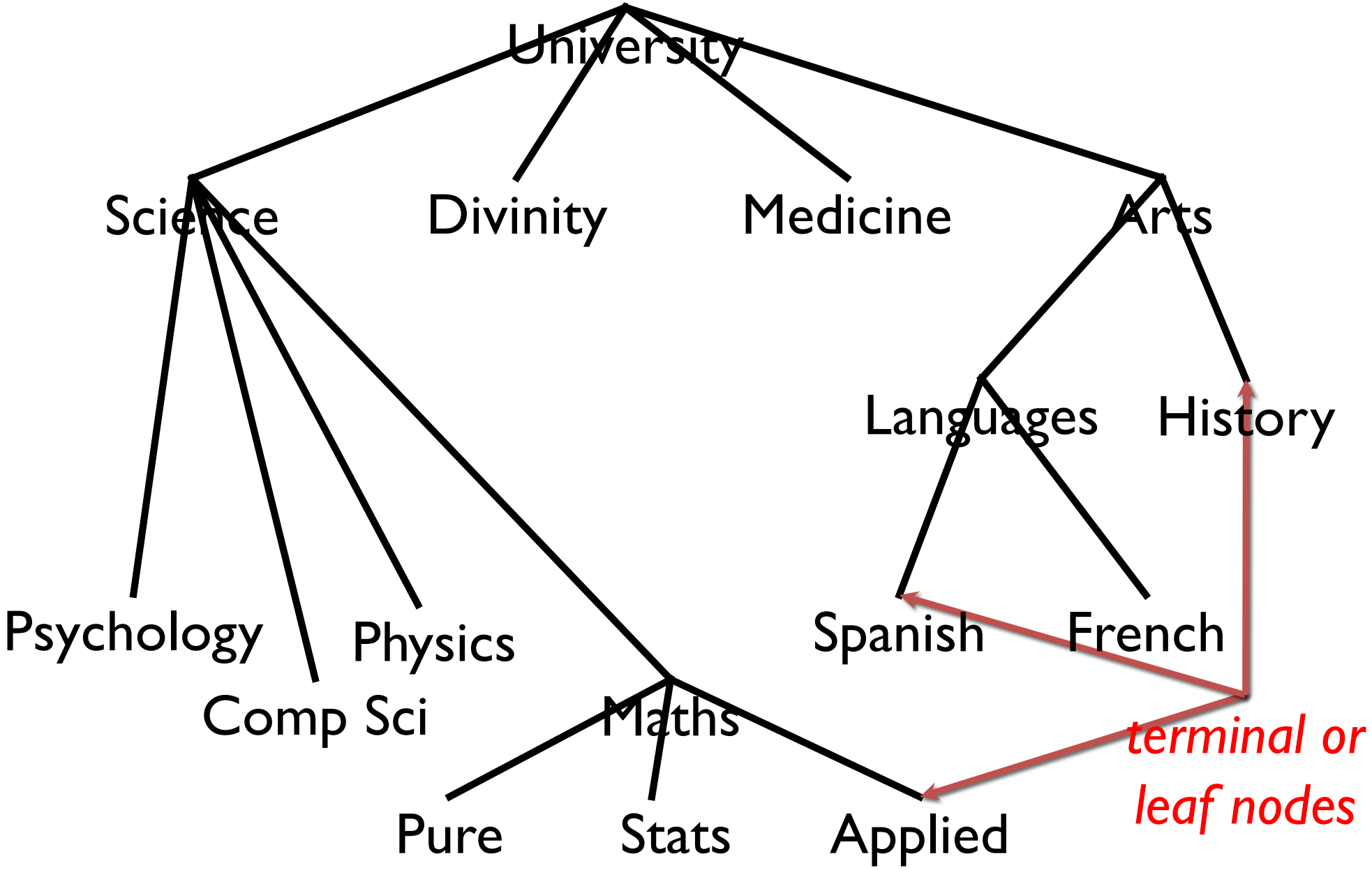


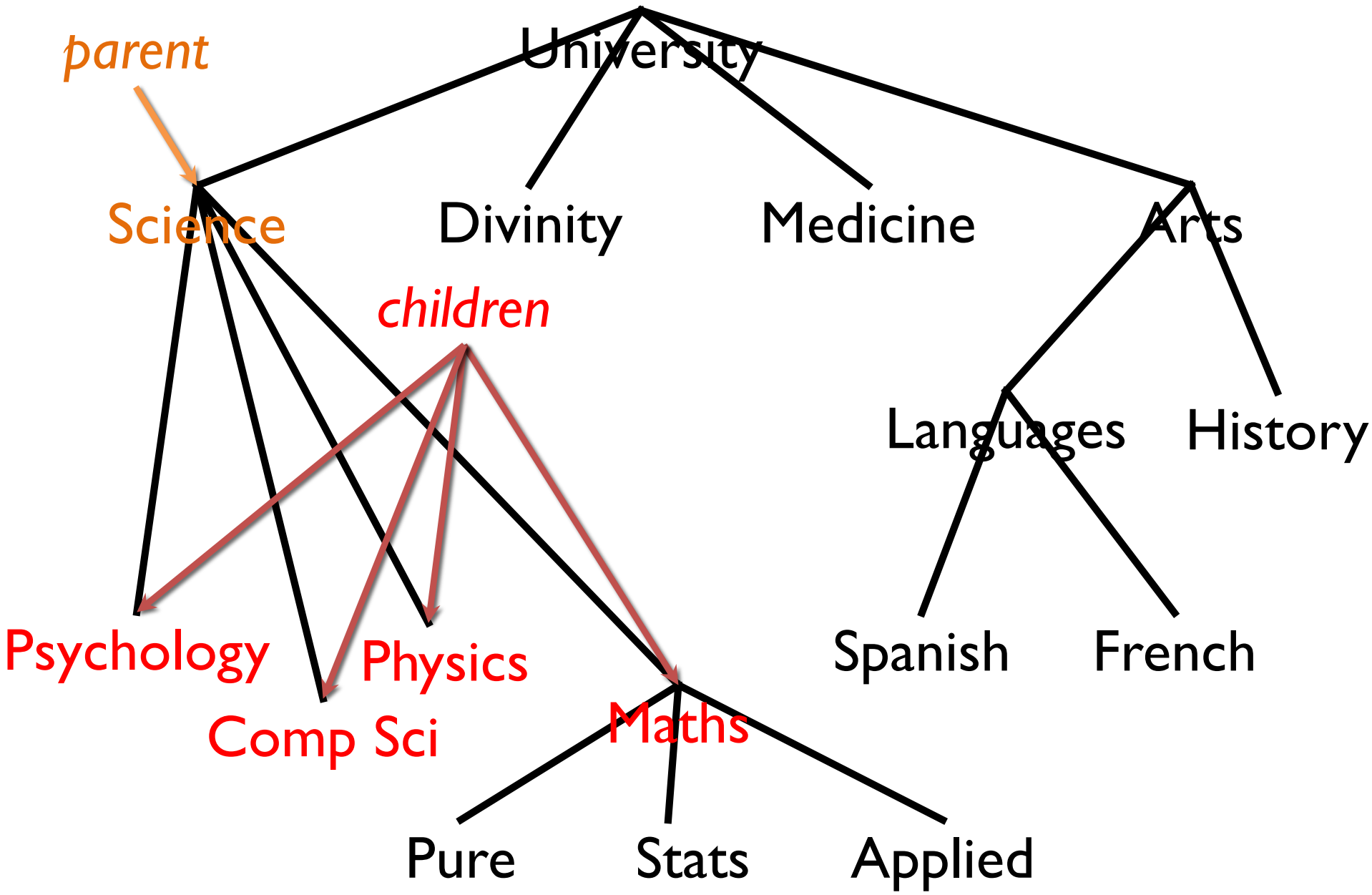

University
Science
Divinity
Medicine
Arts
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied

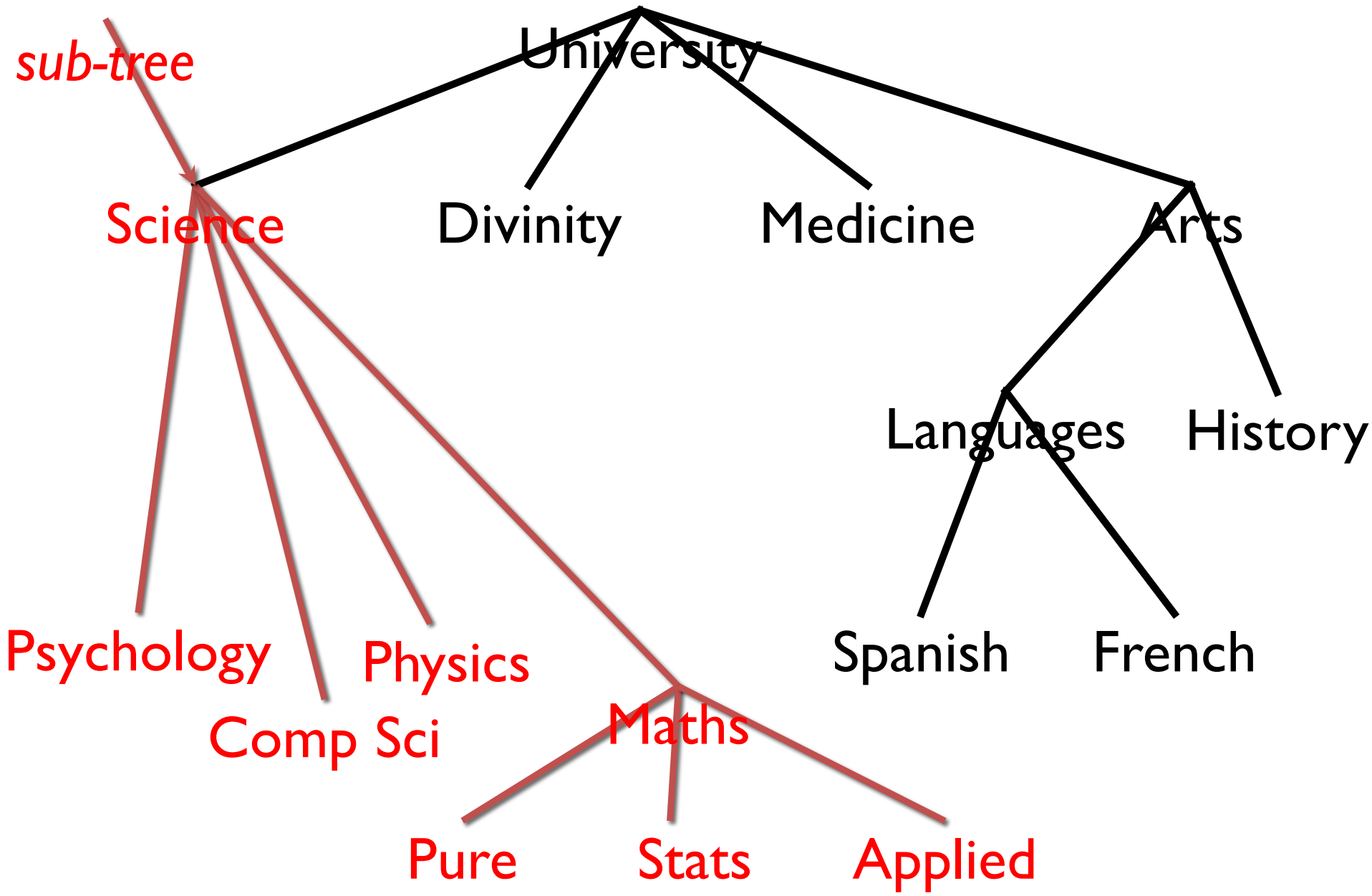

University
Science
Divinity
Medicine
Arts
nodes
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied

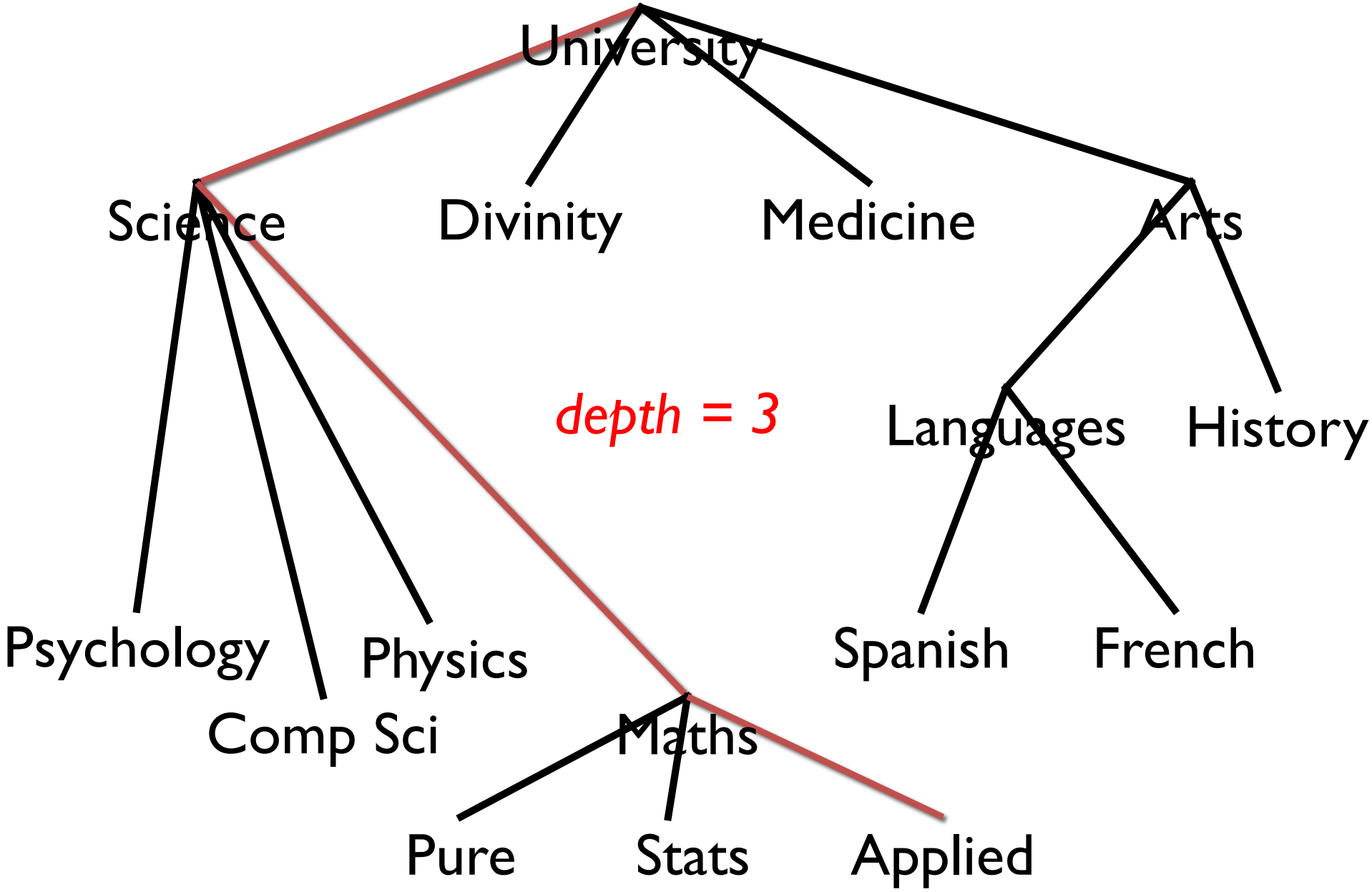

root node
University
Science
Divinity
Medicine
Arts
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied

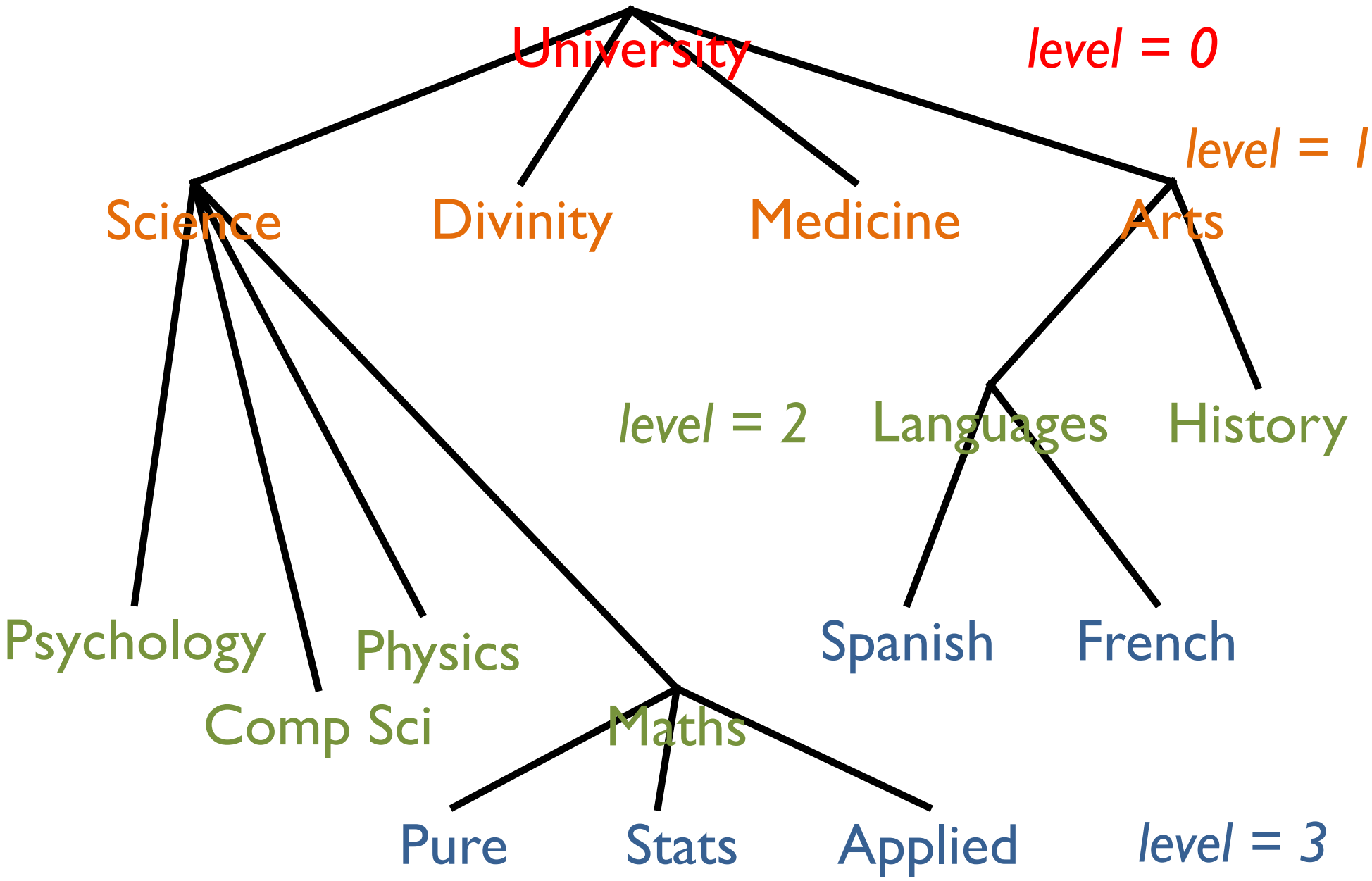

University
Science
Divinity
Medicine
Arts
interior nodes
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied


University
Science
Divinity
Medicine
Arts
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied
terminal or
leaf nodes


parent
University
Science
Divinity
Medicine
Arts
children
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied

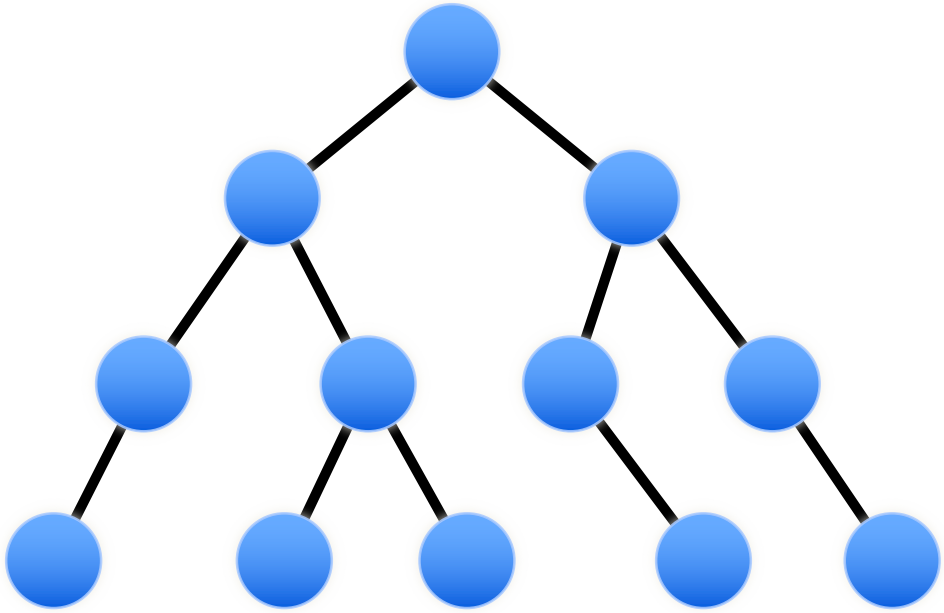

sub-tree
University
Science
Divinity
Medicine
Arts
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied


University
Science
Divinity
Medicine
Arts
depth = 3
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied


University
level = 0
level = 1
Science
Divinity
Medicine
Arts
level = 2
Languages
History
Psychology
Physics
Comp Sci
Maths
Spanish
French
Pure
Stats
Applied
level = 3

Binary Trees

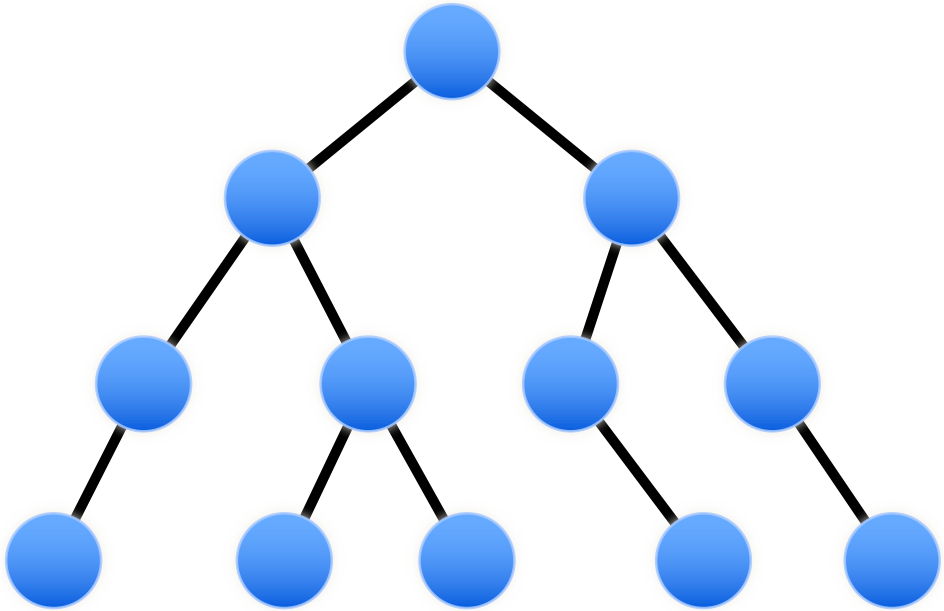
- Every node has at most two child nodes



```
public class BinaryTreeNode {

  public Object element;

  public BinaryTreeNode left;

  public BinaryTreeNode right;
}

public class BinaryTreeCollection implements ICollection {
  private BinaryTreeNode root = null;
  ...
}
```

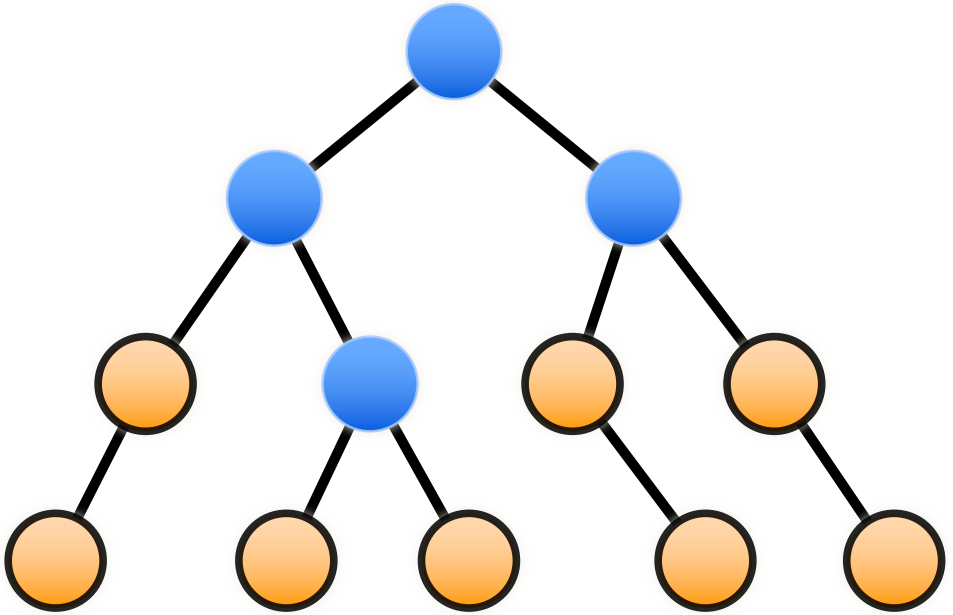
add

- Where can a new element be added?

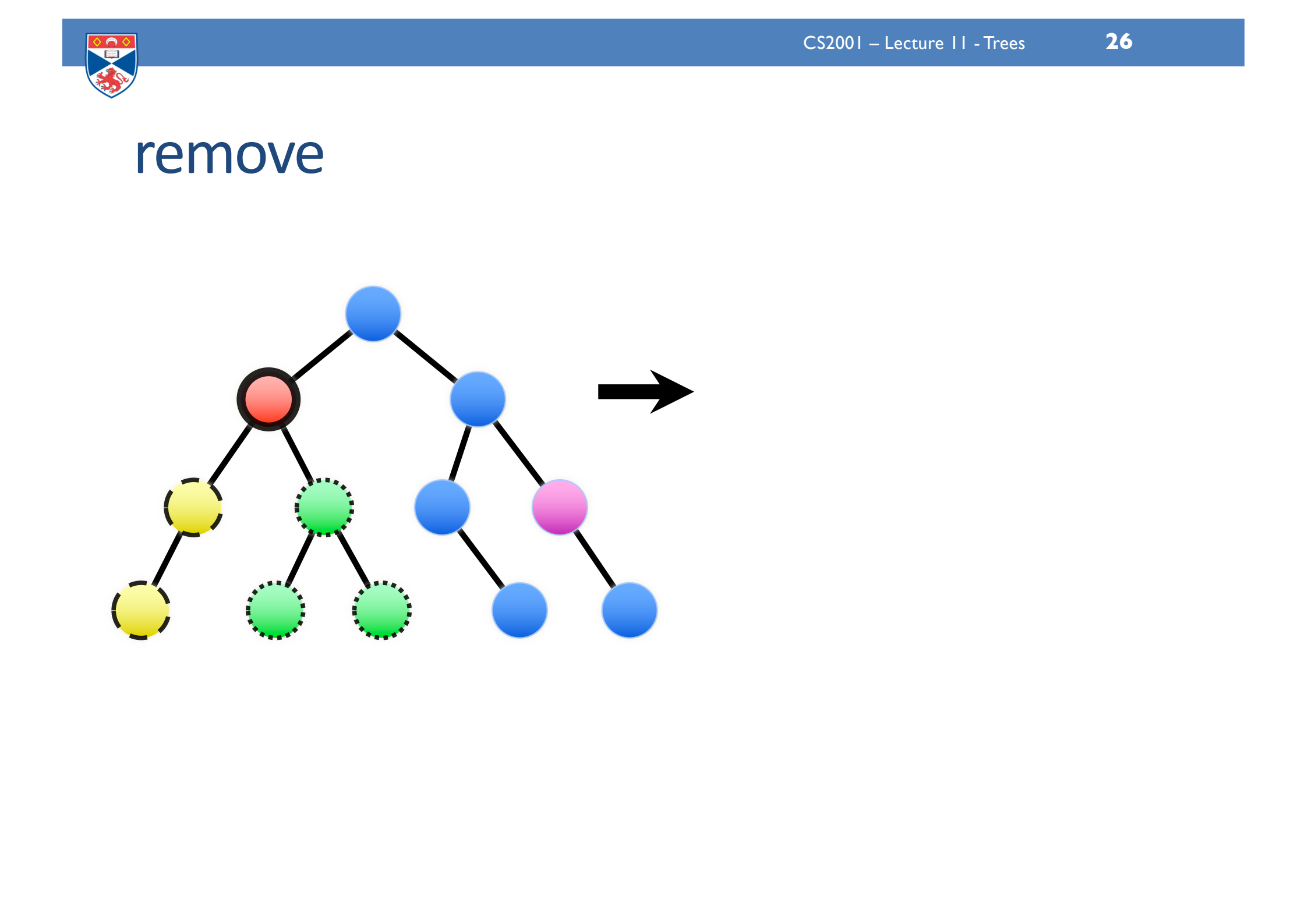


add

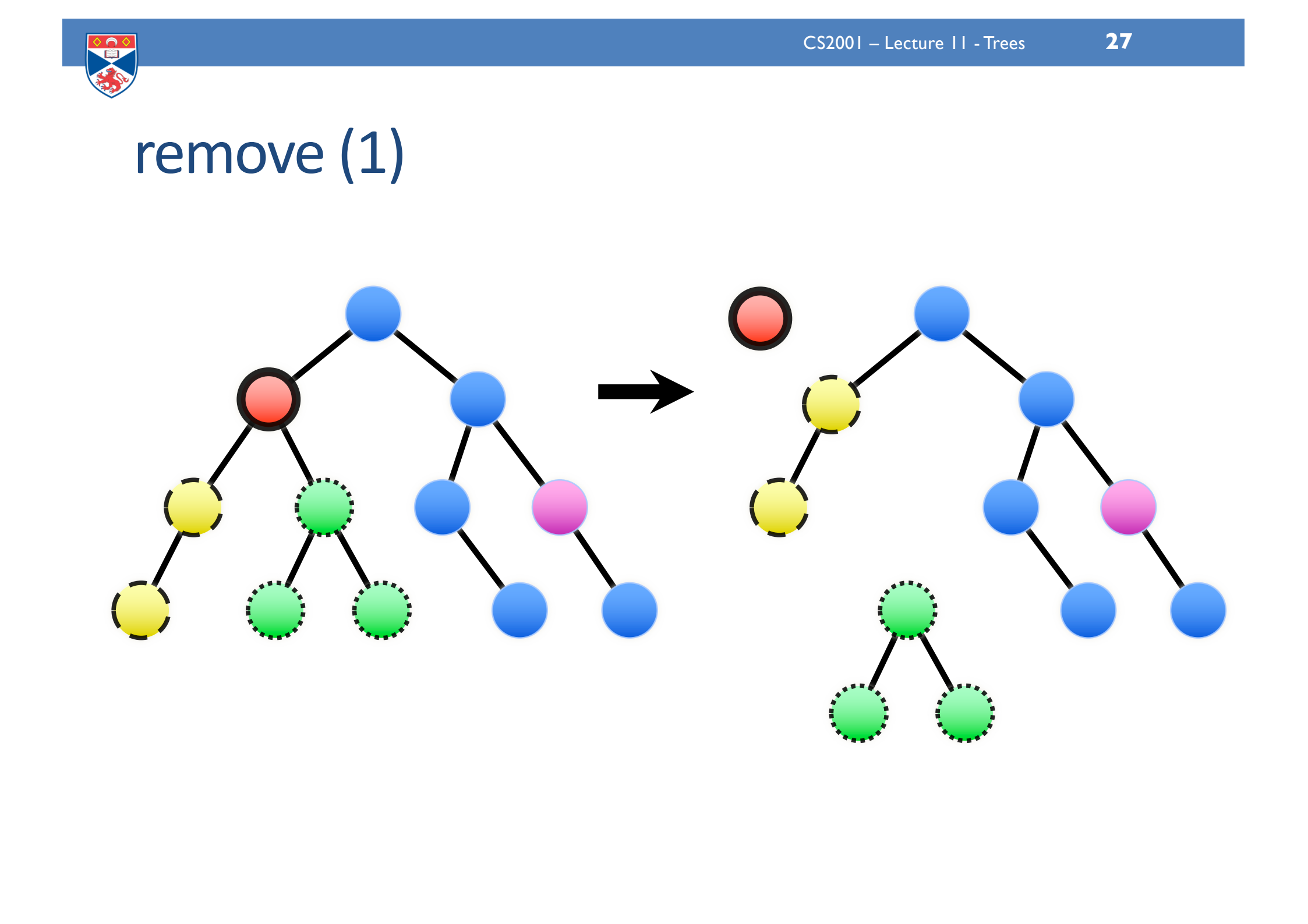
- Where can a new element be added?



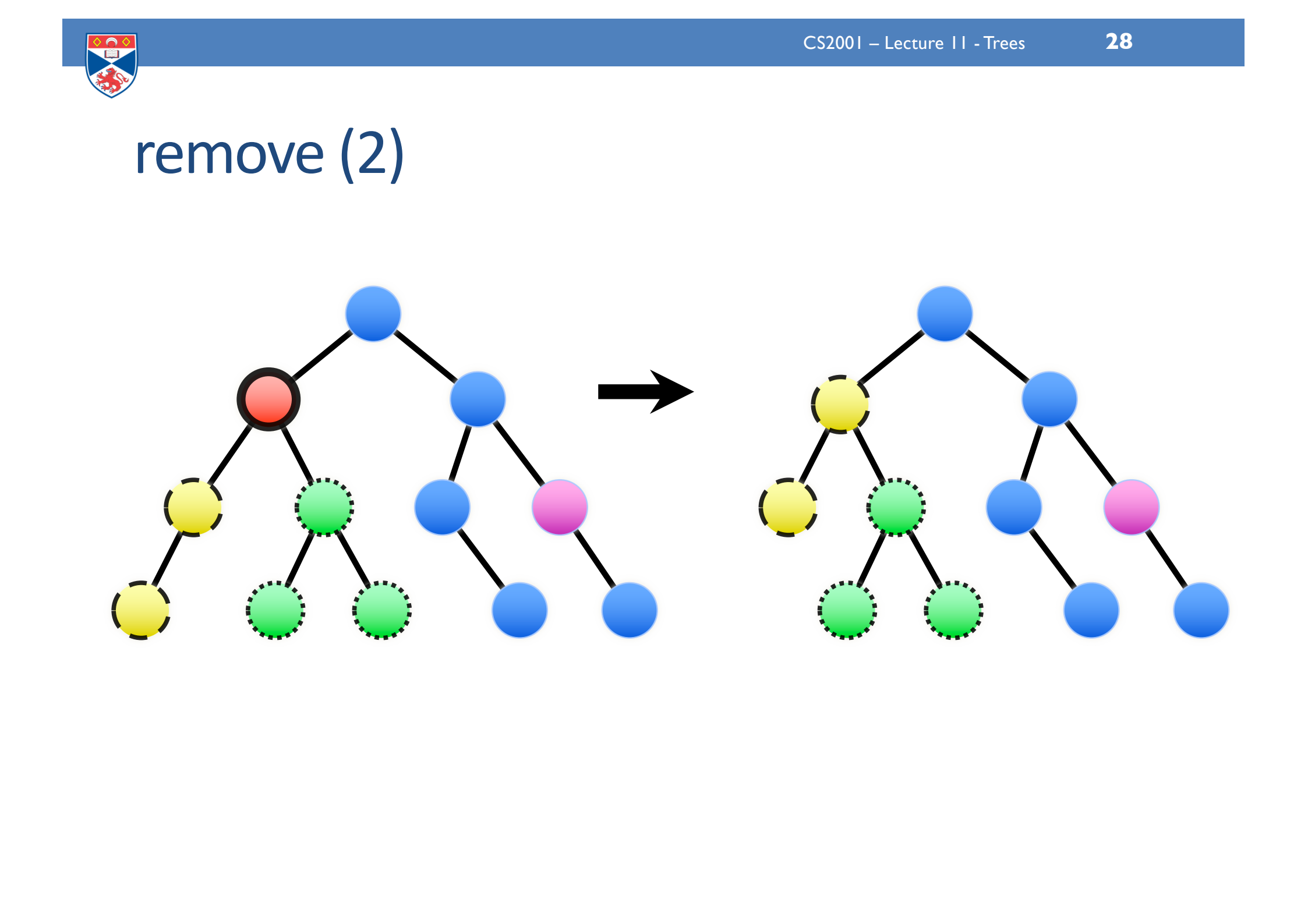
```
public void add(Object element) {
  BinaryTreeNode node = new BinaryTreeNode(element);
  if (root == null) root = node;
  else add(node, root);
}

private void add(BinaryTreeNode subtree, BinaryTreeNode parent) {
  if (subtree != null)
    if (parent.left == null) parent.left = subtree;
    else if (parent.right == null) parent.right = subtree;
         else if (size(parent.left) < size(parent.right))
                   add(subtree, parent.left);
              else add(subtree, parent.right);
}
```



remove



remove (1)



remove (2)

```
public void remove(Object element) {
  BinaryTreeNode parent = findParent(element);
  BinaryTreeNode node = findNode(element);

  if (node != null) {
    if (parent == null) {
      root = node.left;
      add(node.right, root);
    }
    else {
      if (parent.left == node) parent.left = node.left;
      else parent.right = node.left;
      add(node.right, parent);
    }
  }
}
```

```
public int size() {
  return size(root);
}

private int size(BinaryTreeNode subtree) {
  if (subtree == null) return 0;
  else return 1 + size(subtree.left) + size(subtree.right);
}
```

Trees: Complexity

- insert linear
- contains linear
- remove linear
- size linear

www.flickr.com/photos/ericsbinaryworld/

Key Points

- Trees suitable for storing hierarchical data
- Many applications for in-memory and on-disk data structures
- Binary trees restrict nodes to two children
- Many applications use sorted trees: still to come

Read More

- Sedgewick sections 5.4-5.7
- Goodrich & Tamassia section 7.1-7.3