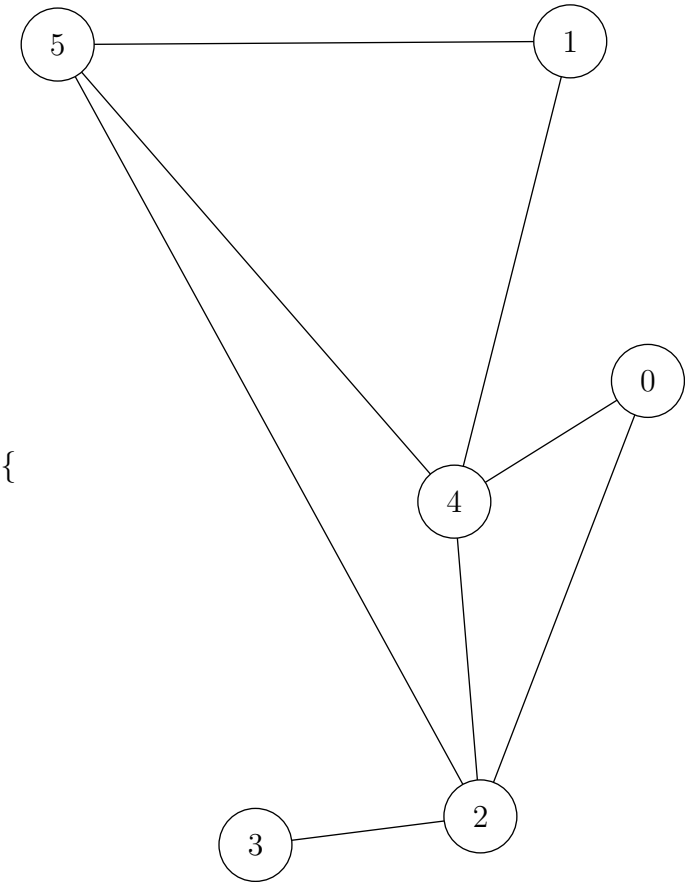
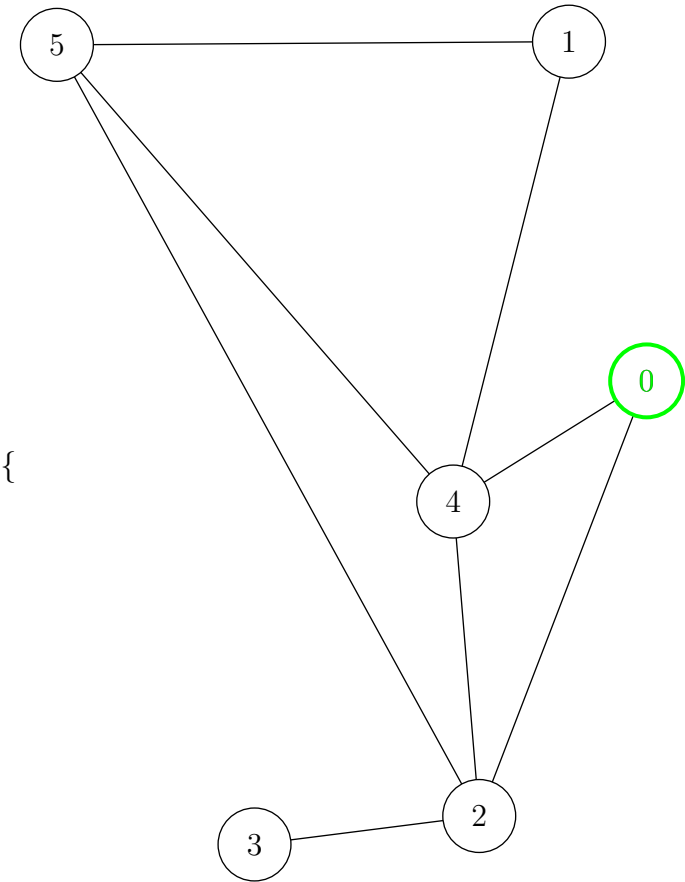
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=nil time=0



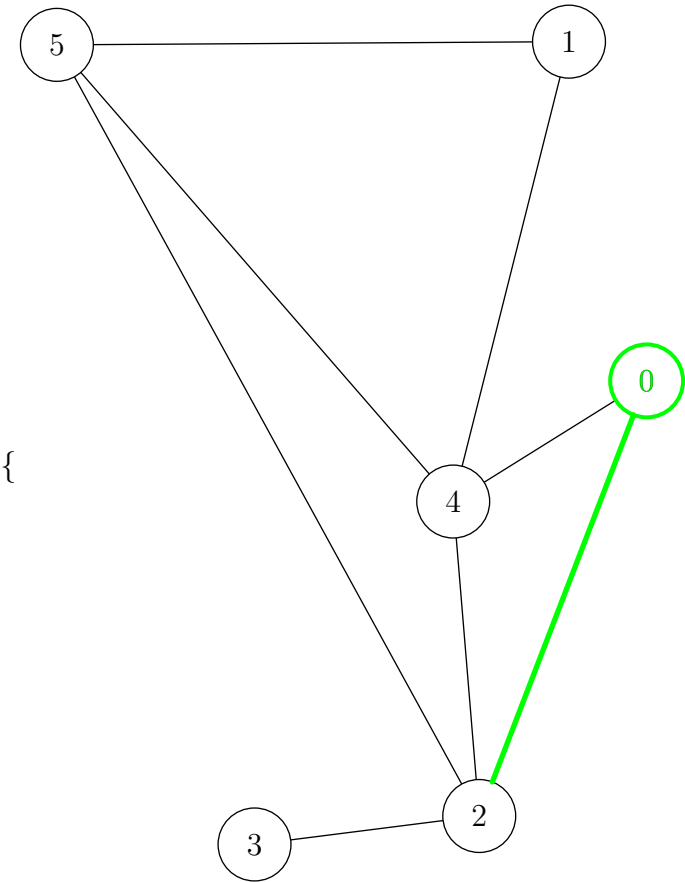
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=nil time=1



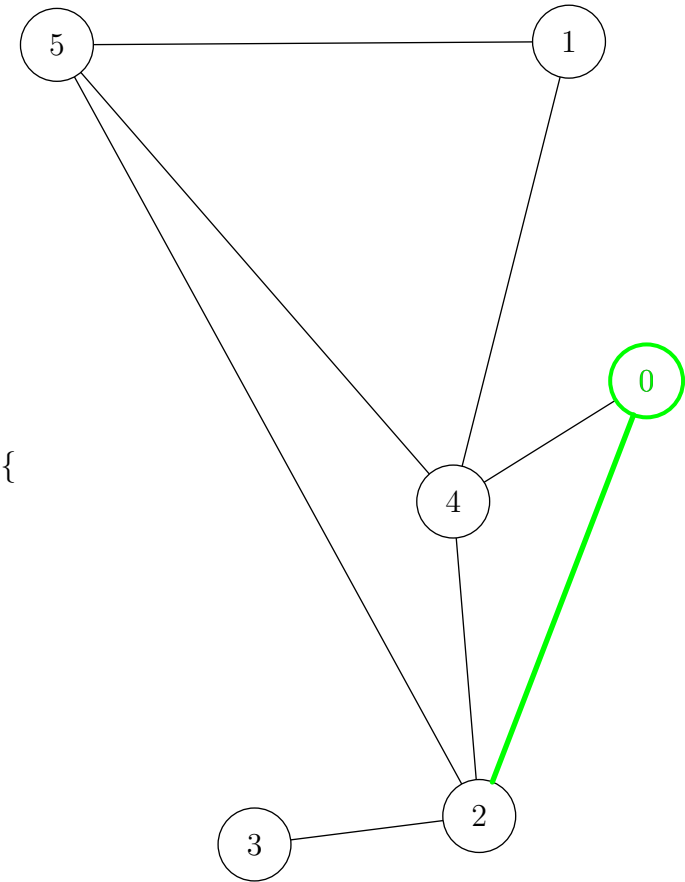
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=2 time=1



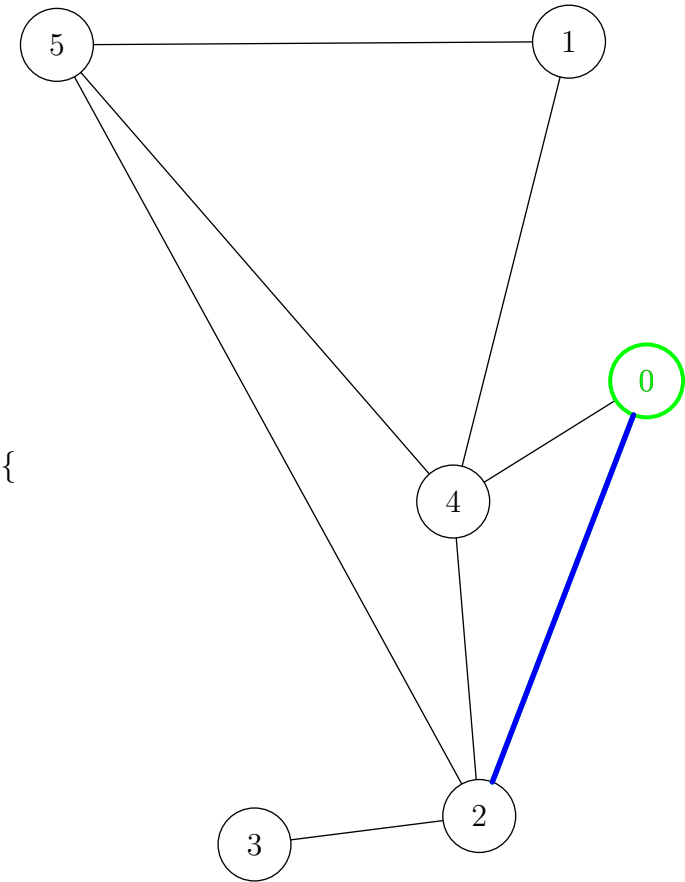
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=2 time=1



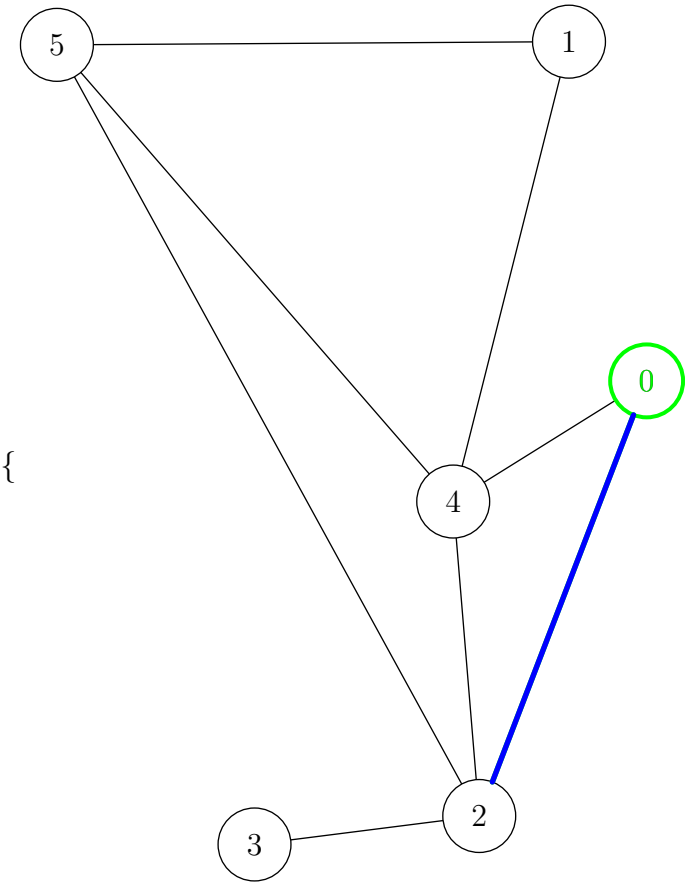
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=2 time=1



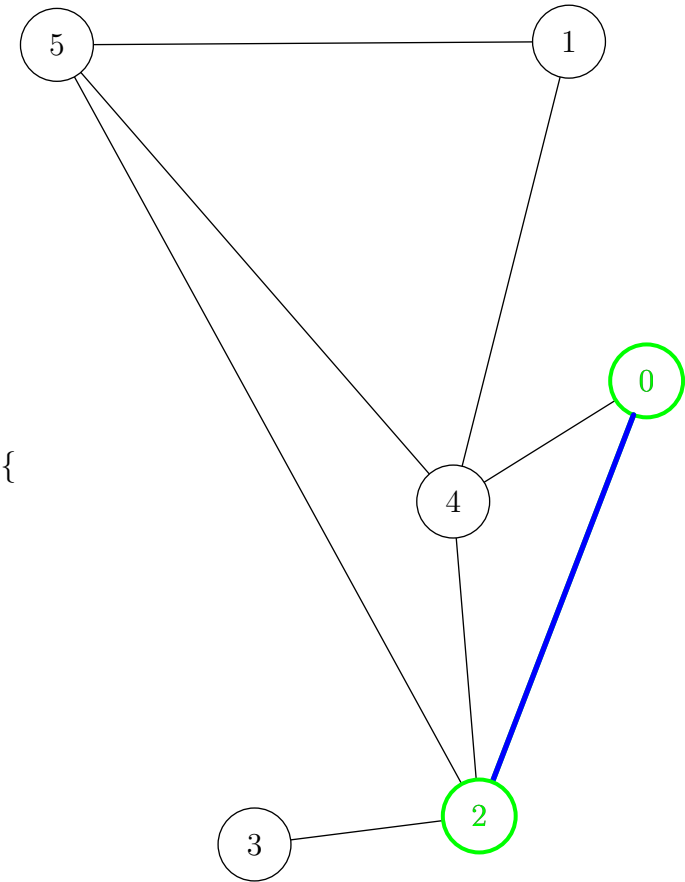
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=2 time=1



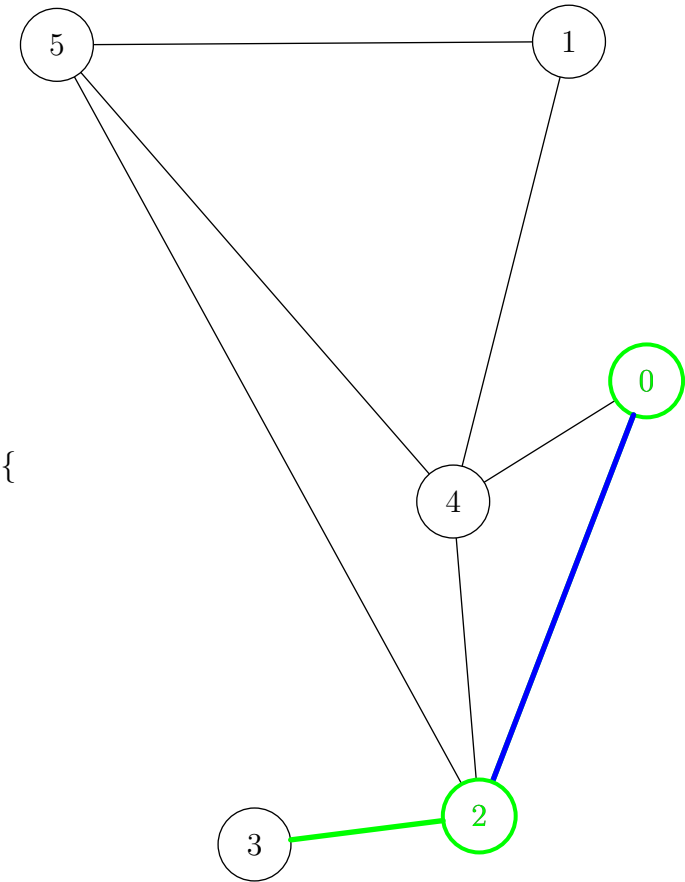
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=nil time=2



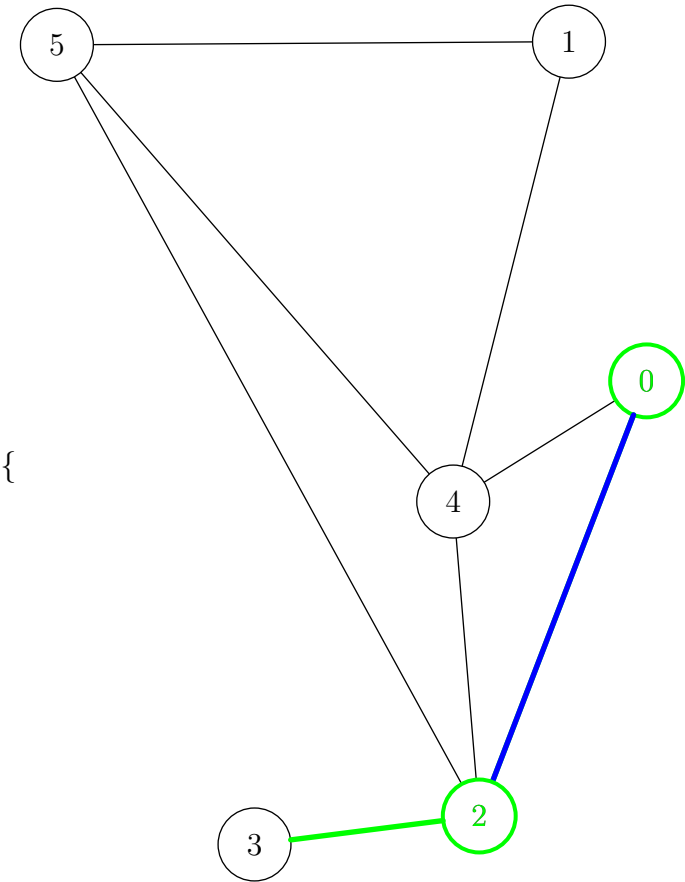
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=3 time=2

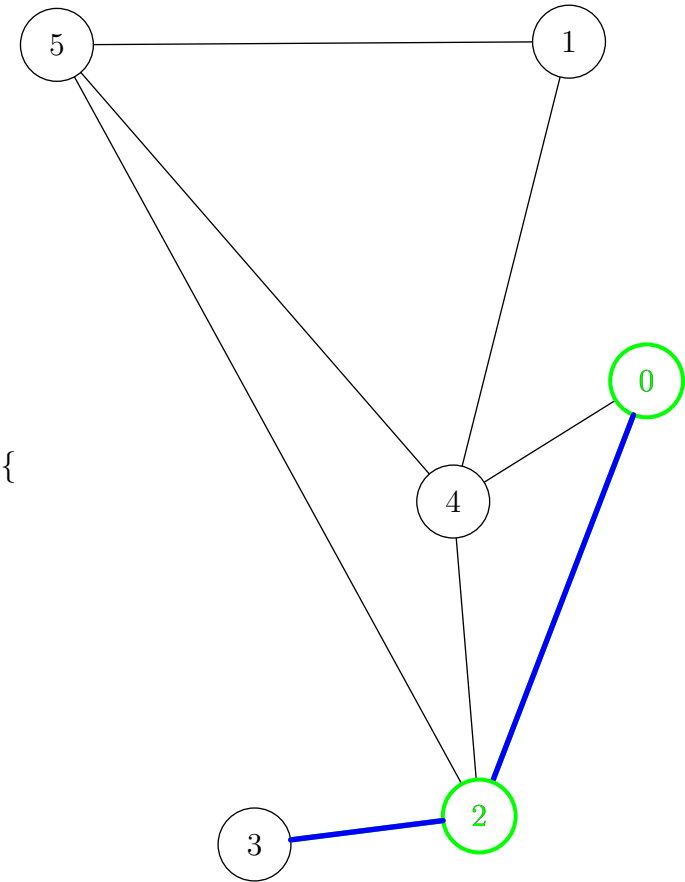


```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=3 time=2



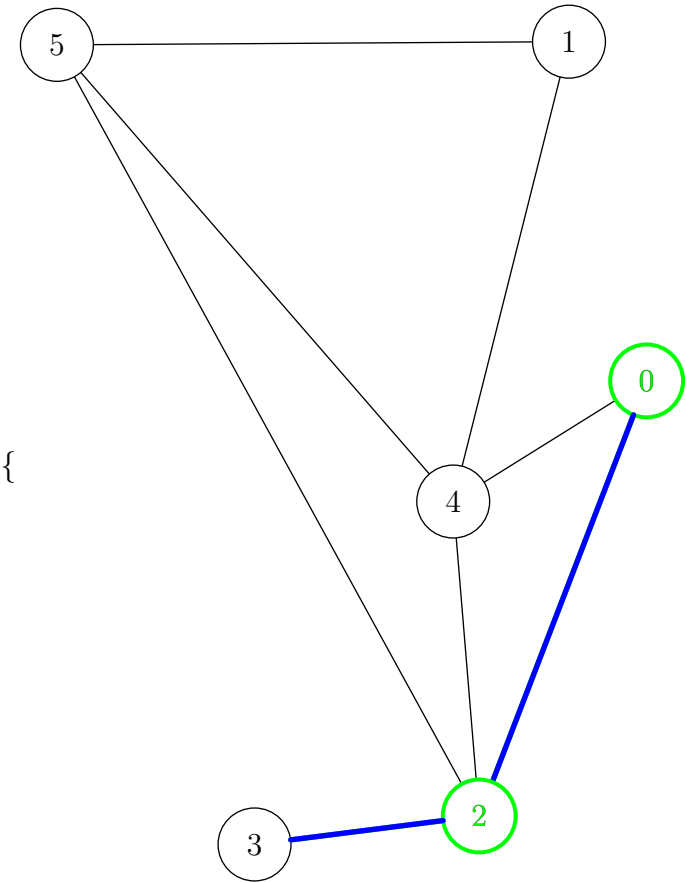
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```



node=2 neighbour=3 time=2

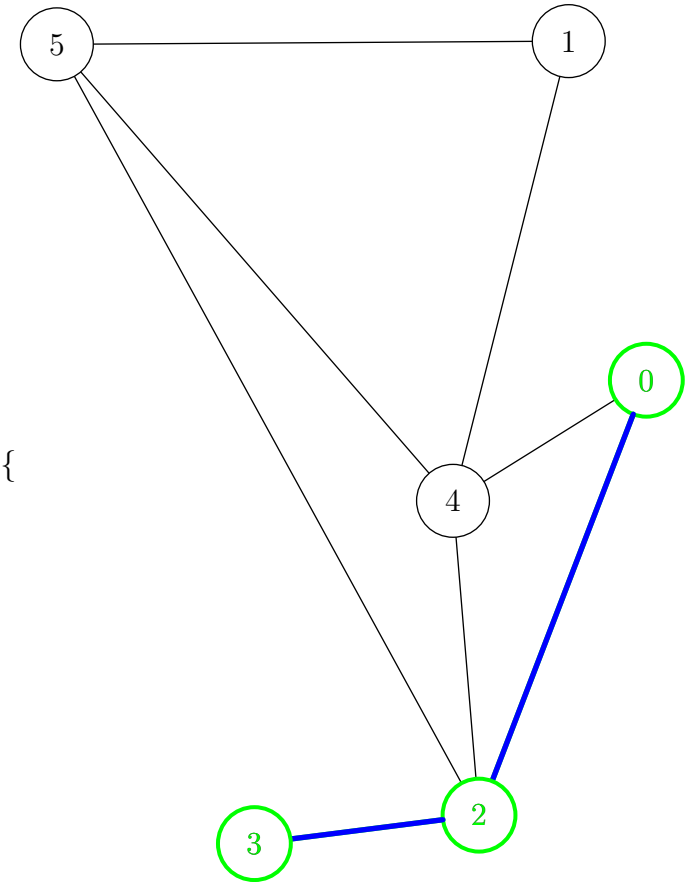
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=3 time=2



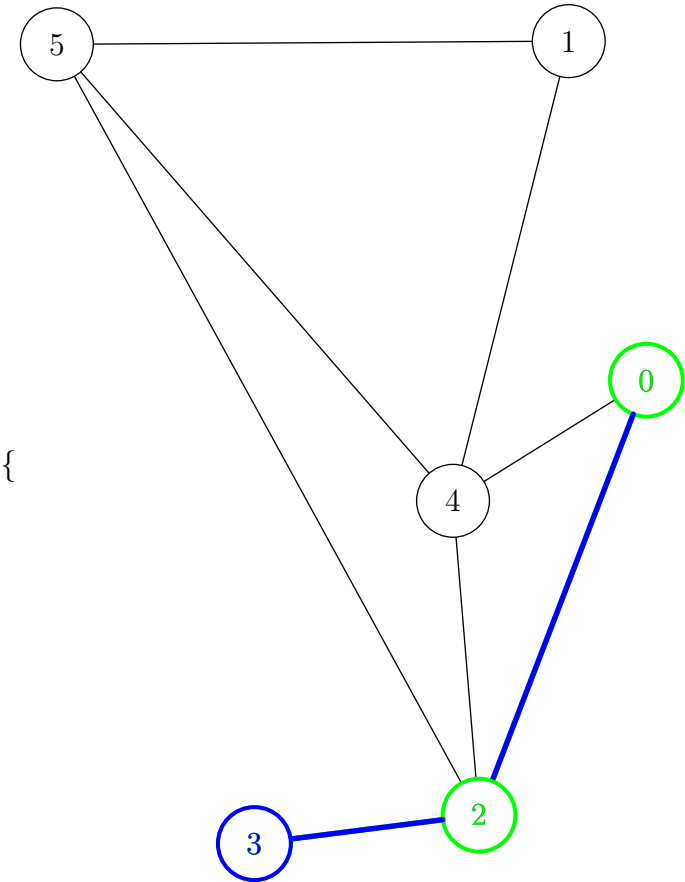
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=3 neighbour=nil time=3



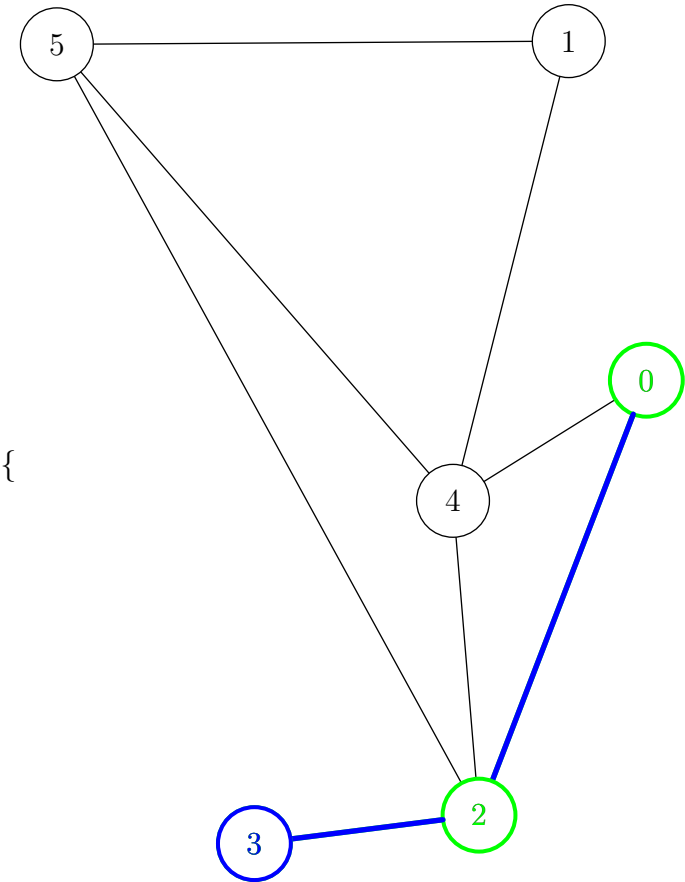
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=3 neighbour=nil time=4



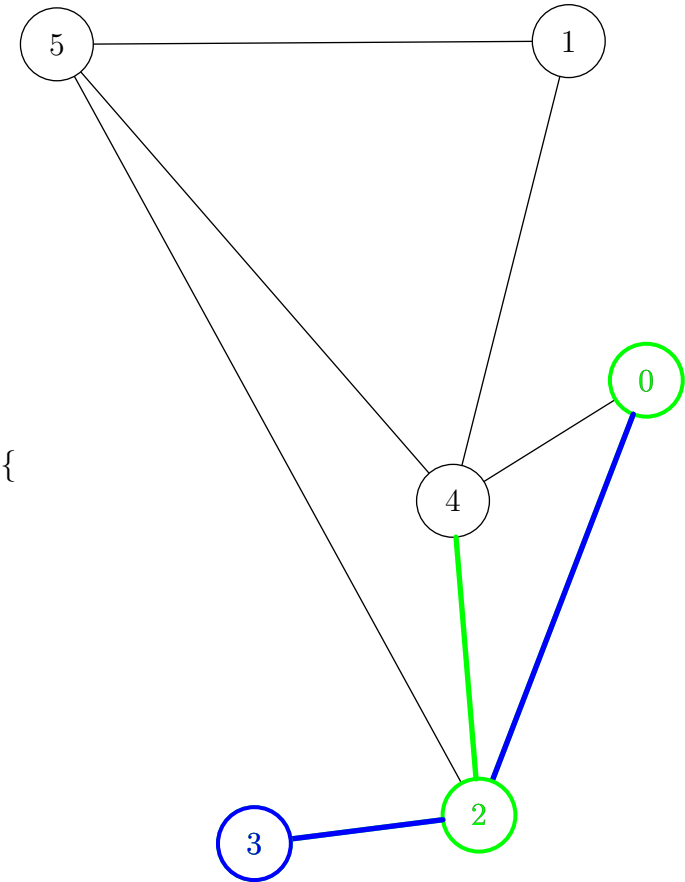
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=3 time=4

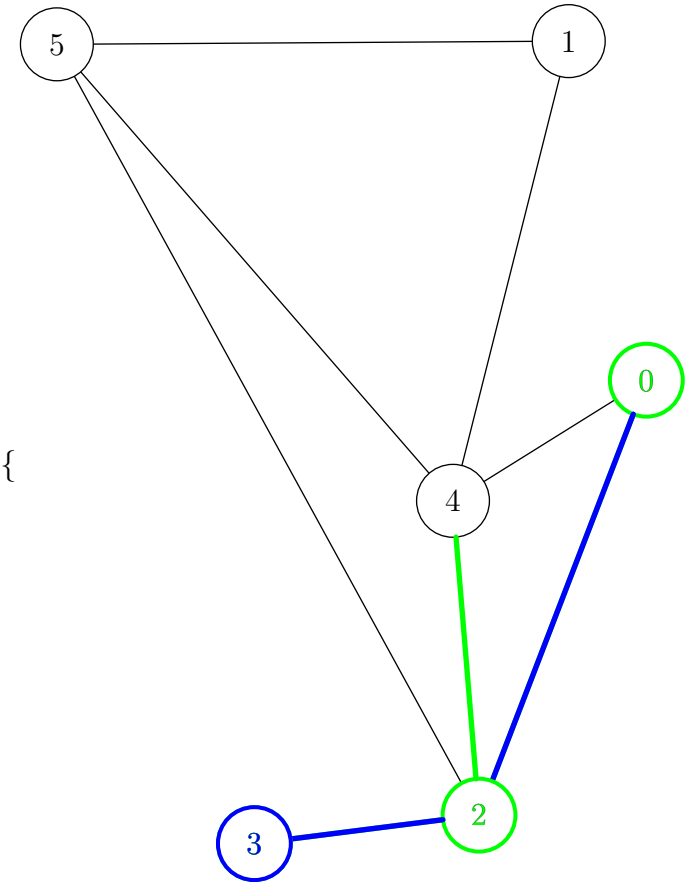


```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=4 time=4



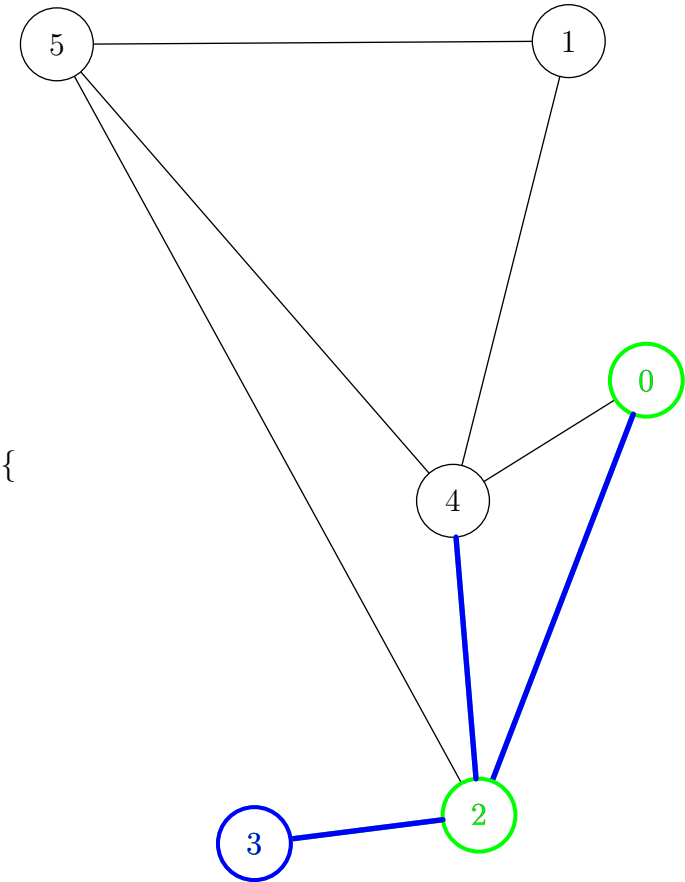
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```



node=2 neighbour=4 time=4

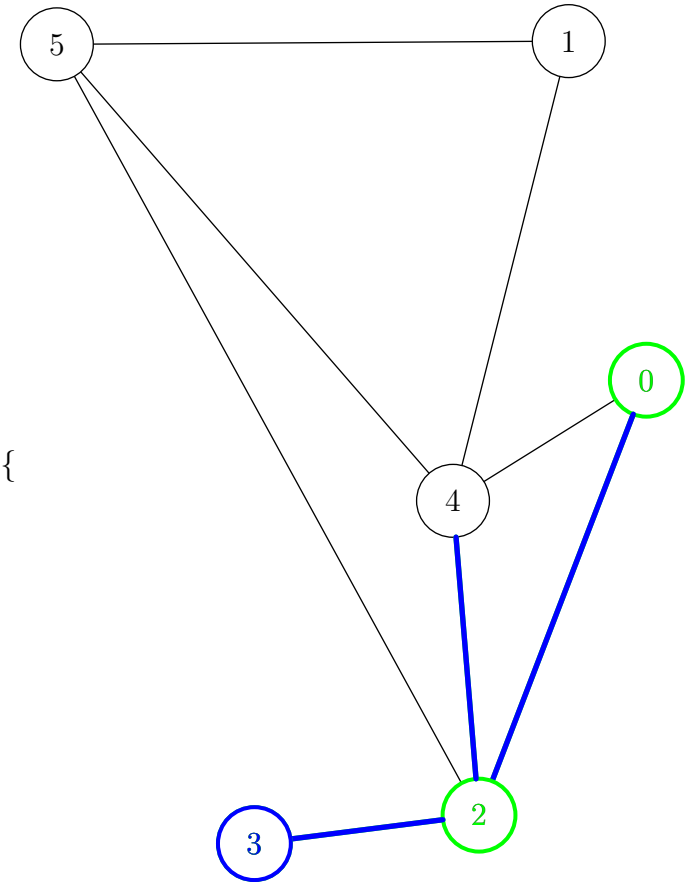
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=4 time=4



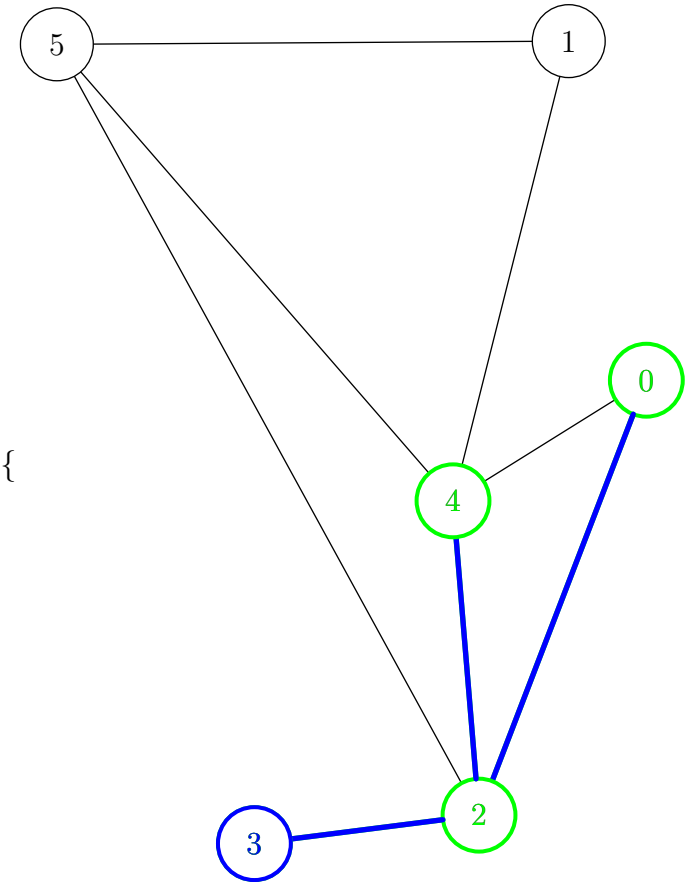
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=4 time=4



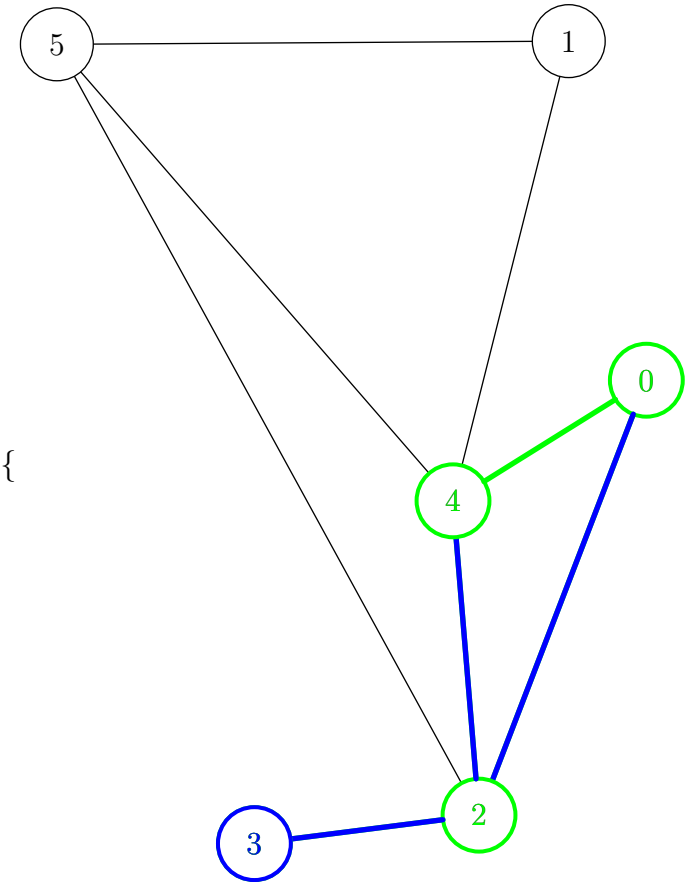
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=5



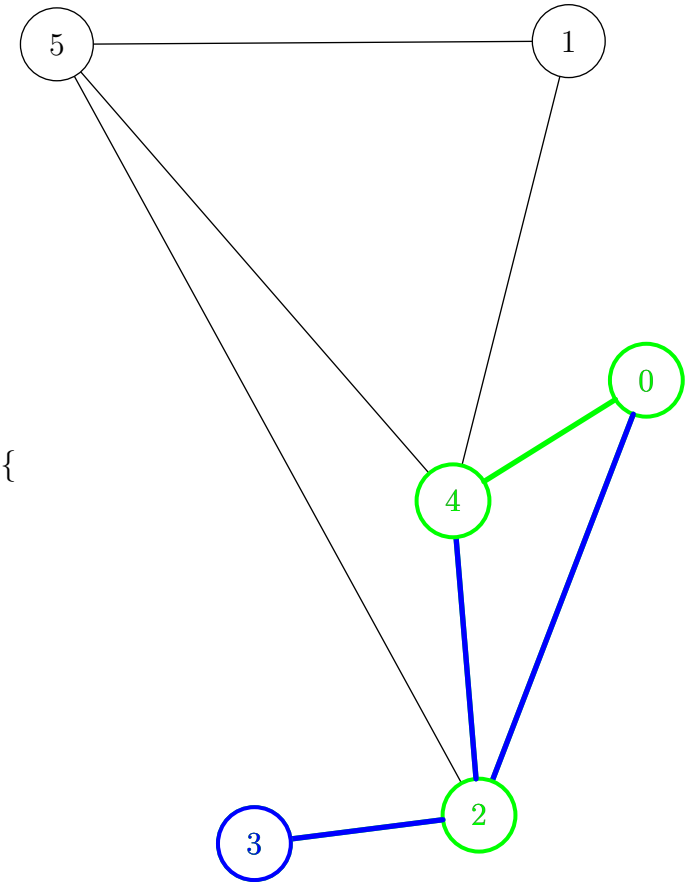
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=5

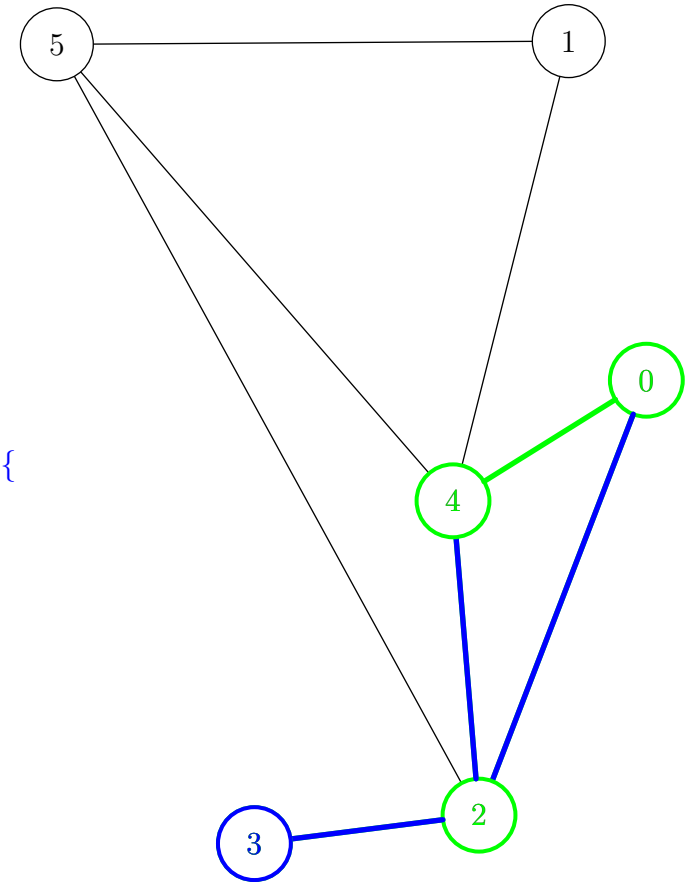


```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=5



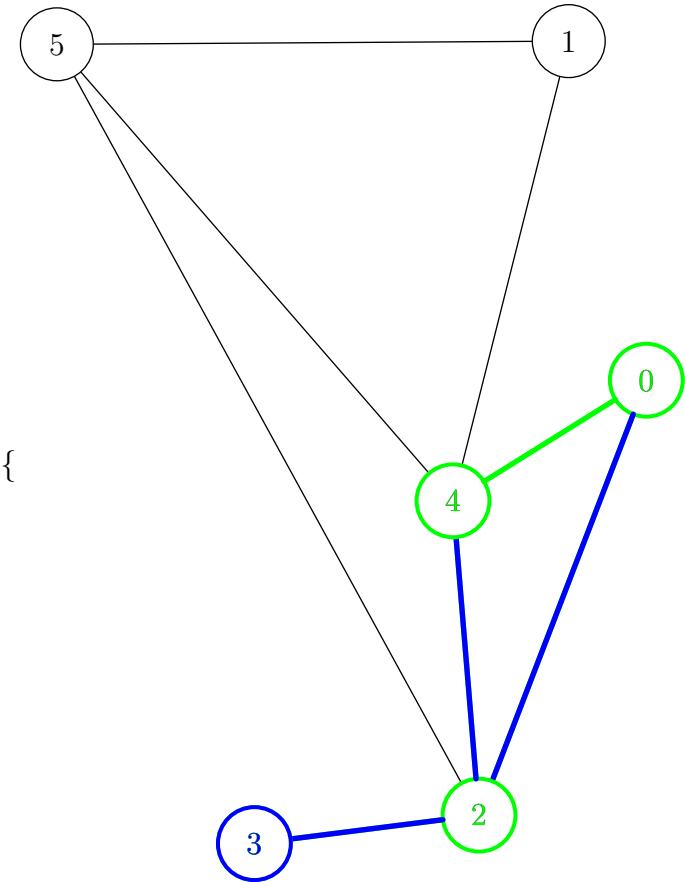
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```



node=4 neighbour=nil time=5

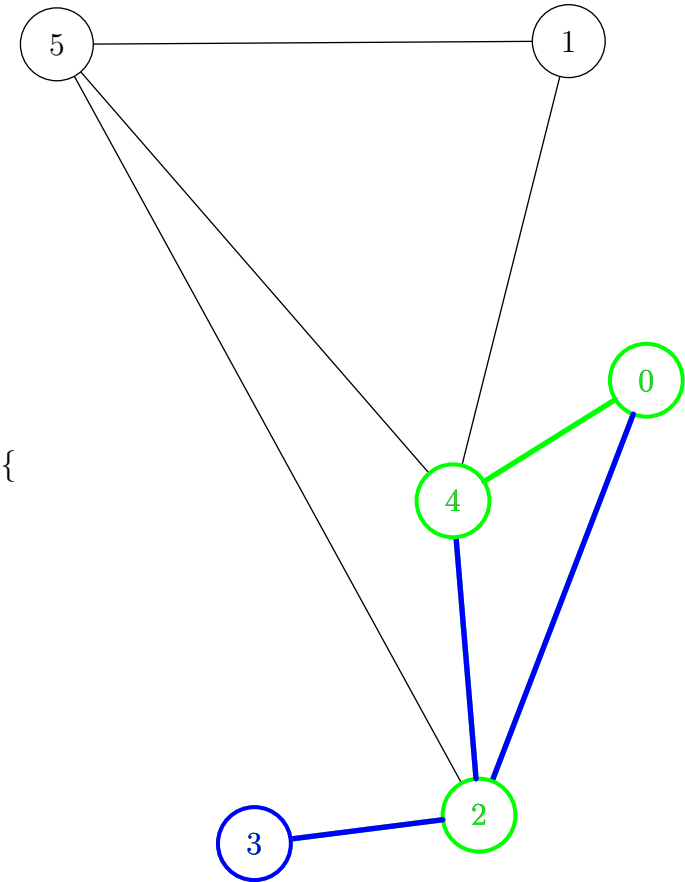
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=5



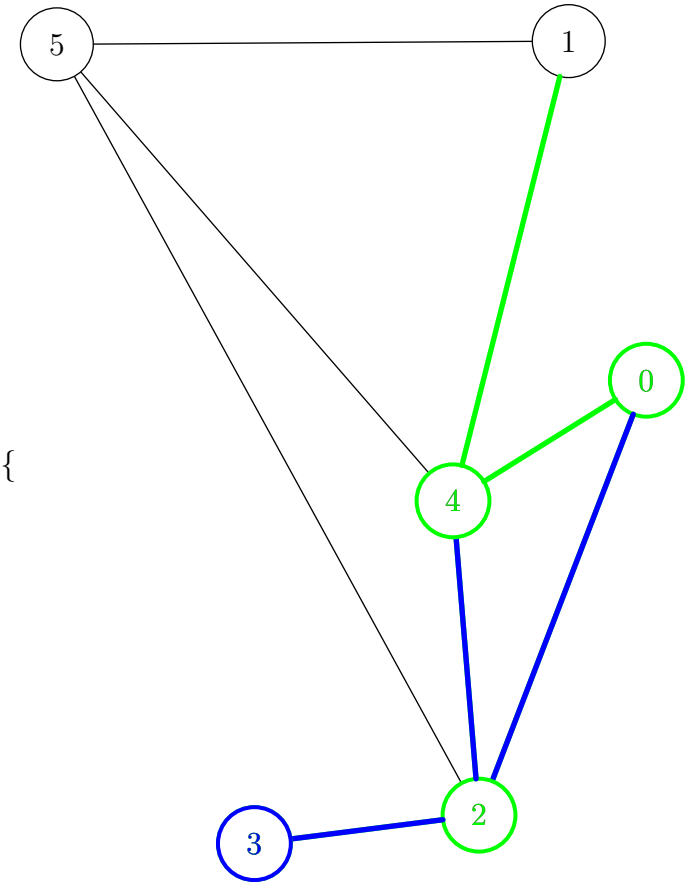
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=5



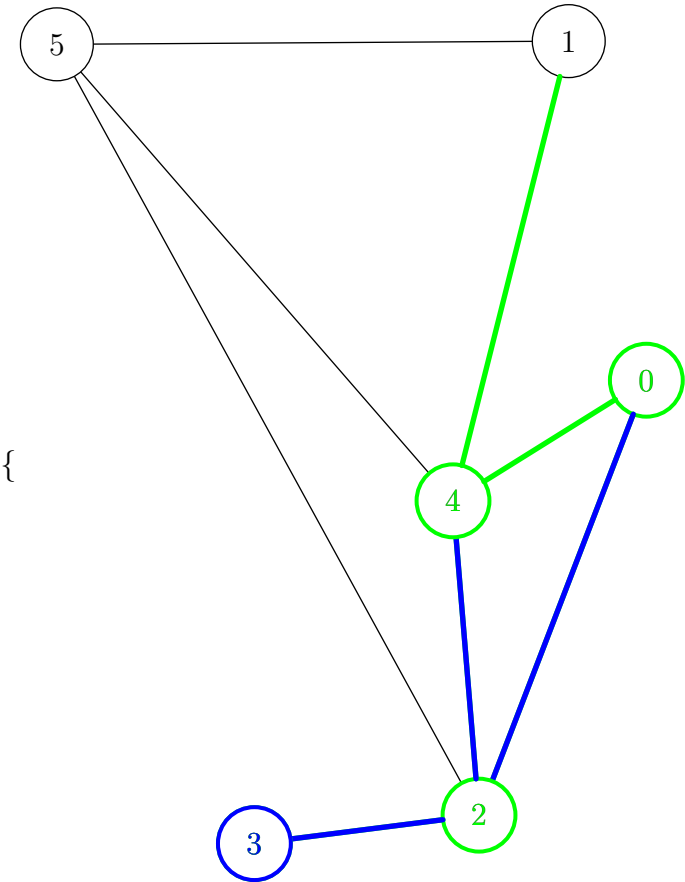
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=1 time=5



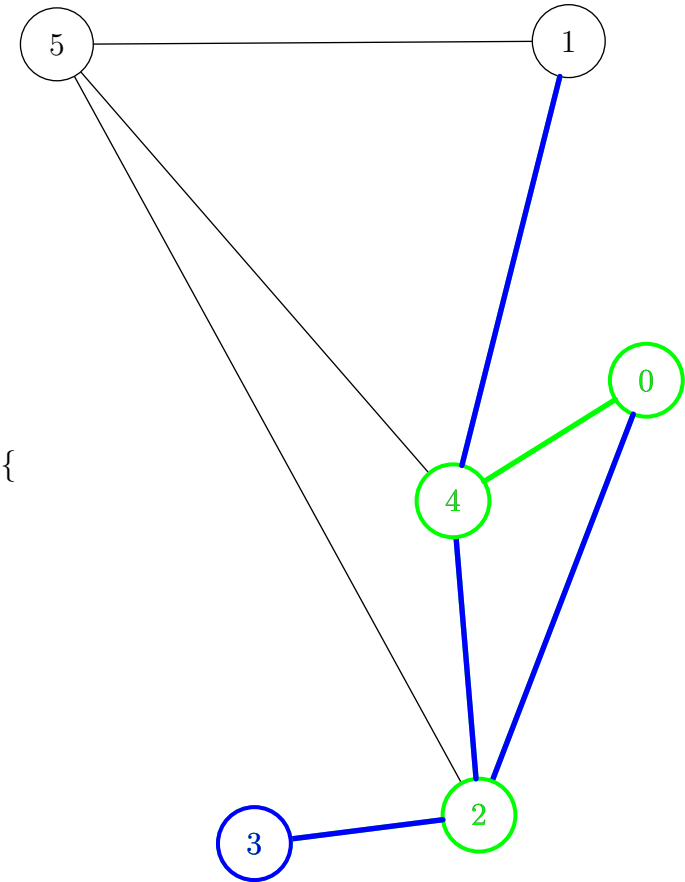
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=1 time=5



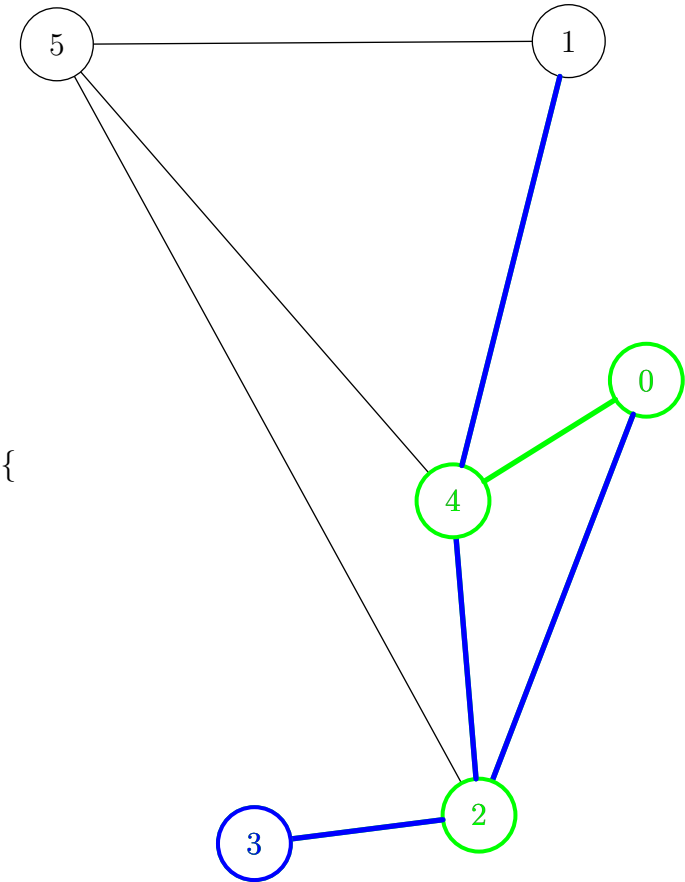
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=1 time=5



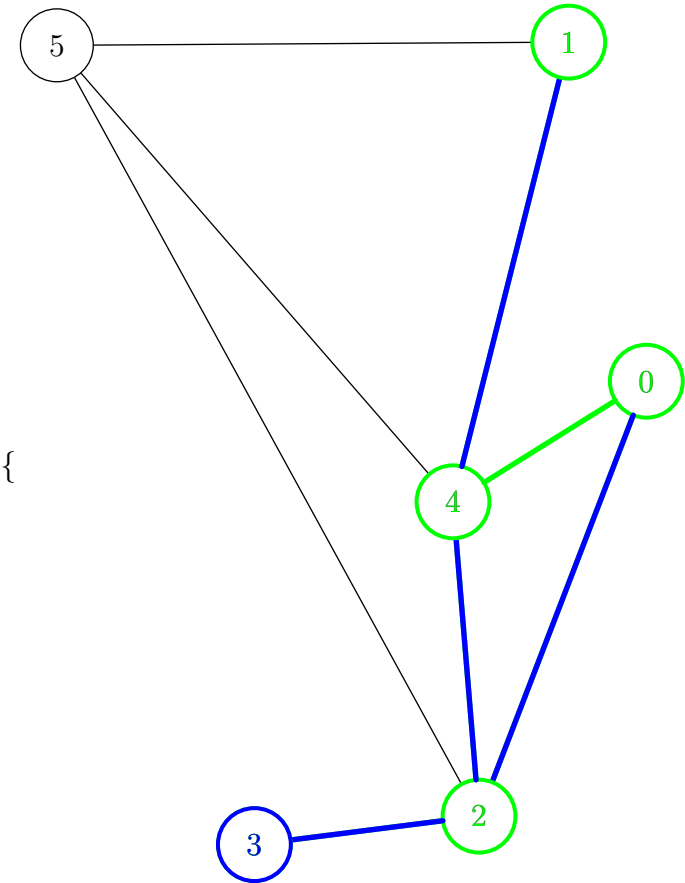
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=1 time=5



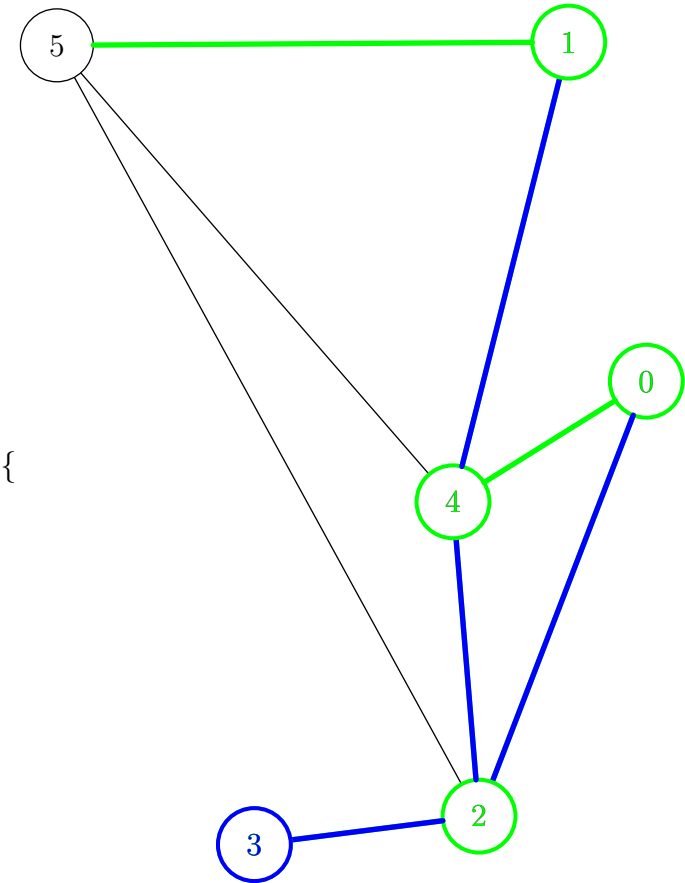
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=nil time=6

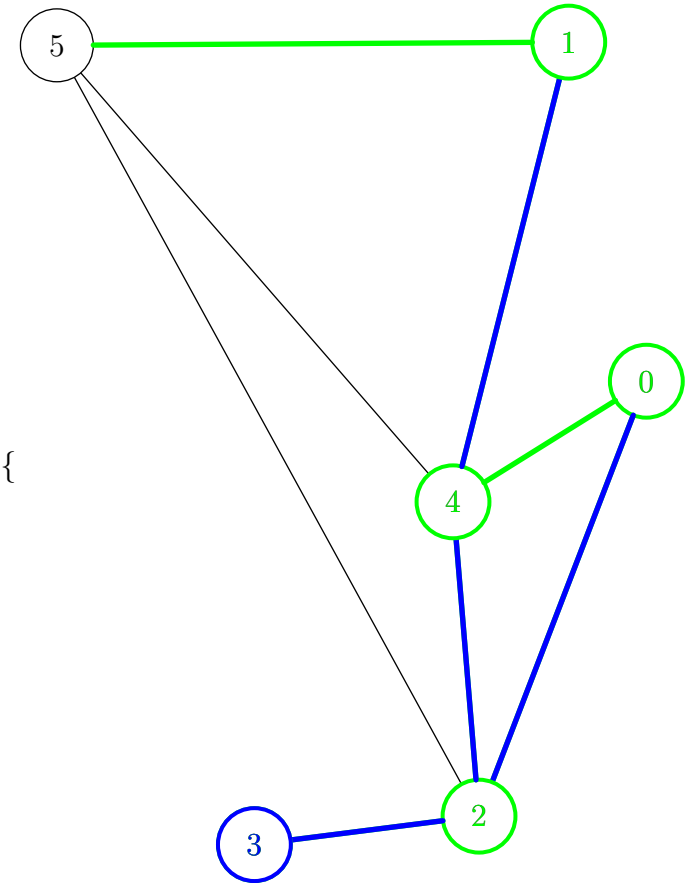


```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=5 time=6



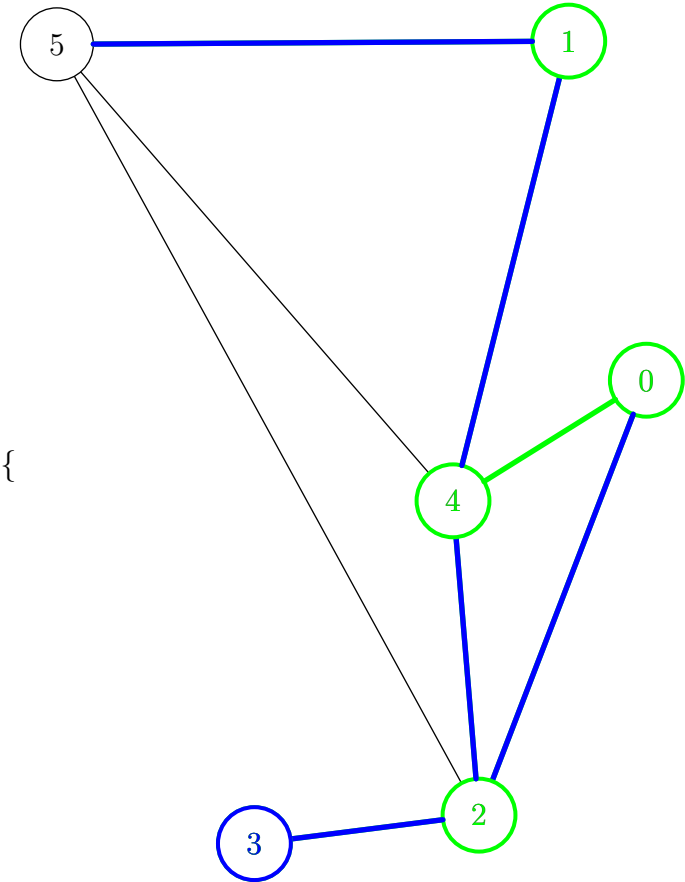
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```



node=1 neighbour=5 time=6

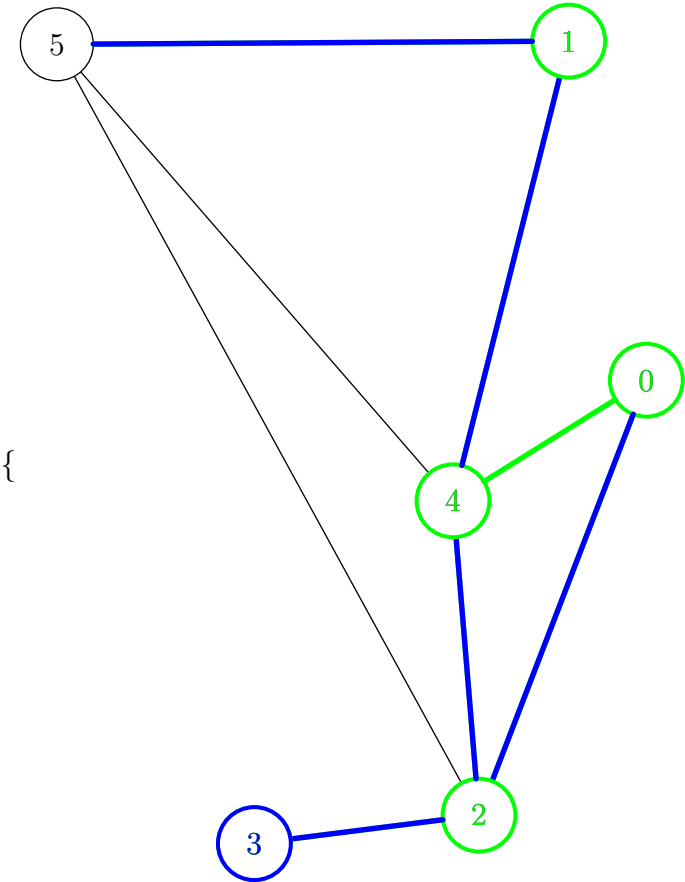
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=5 time=6



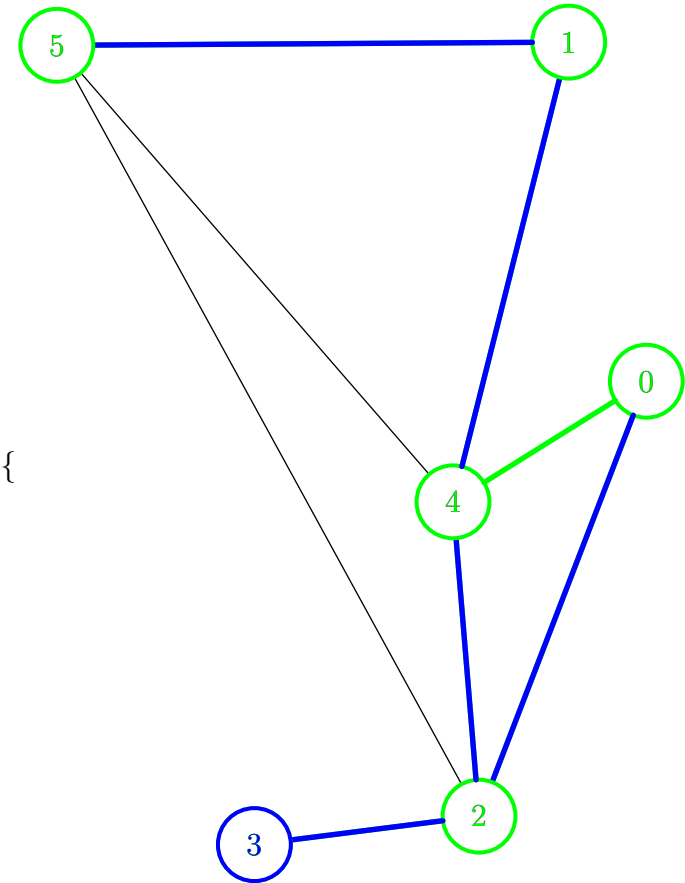
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=5 time=6



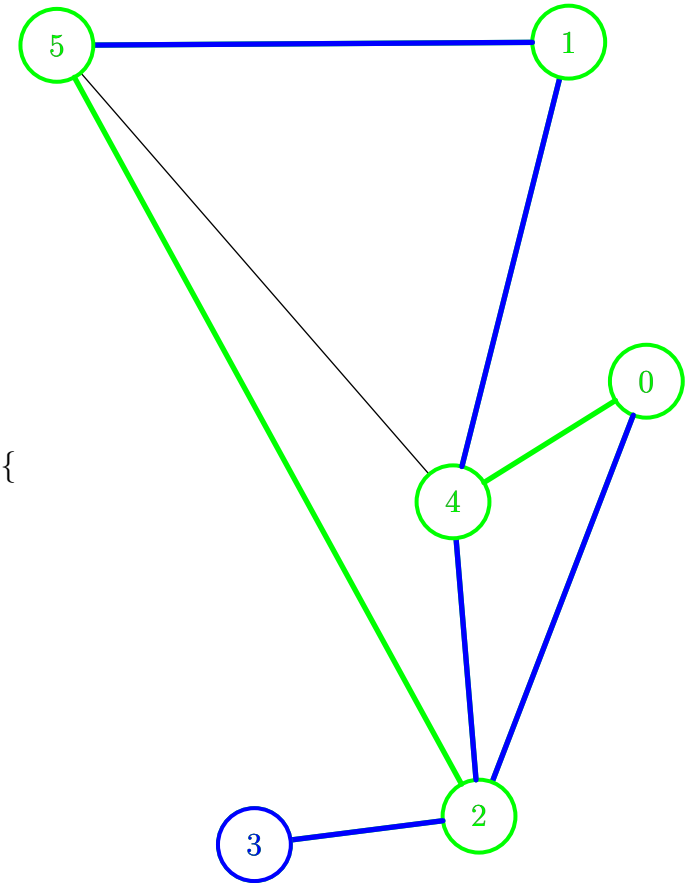
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=nil time=7



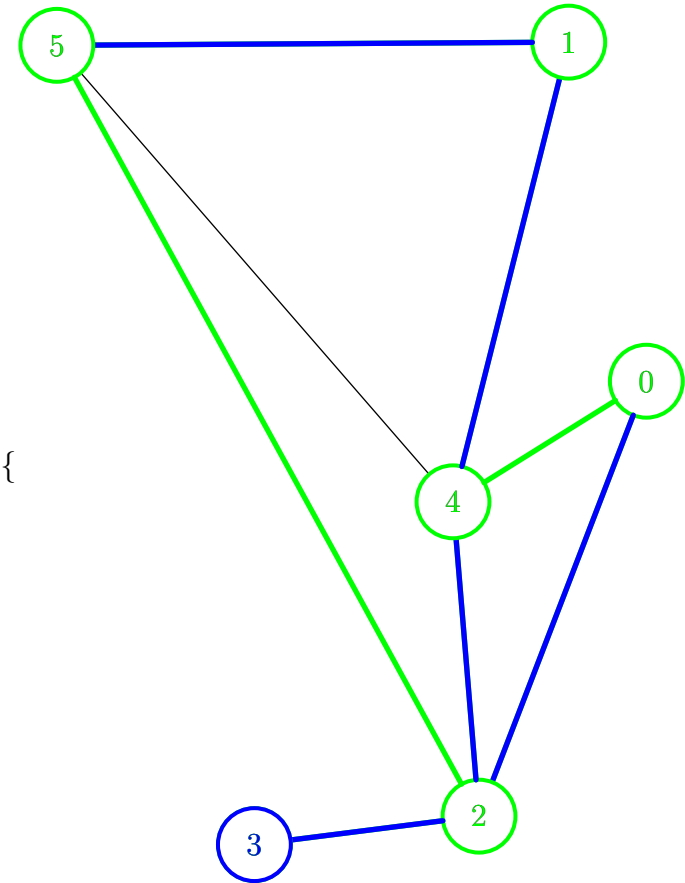
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=2 time=7



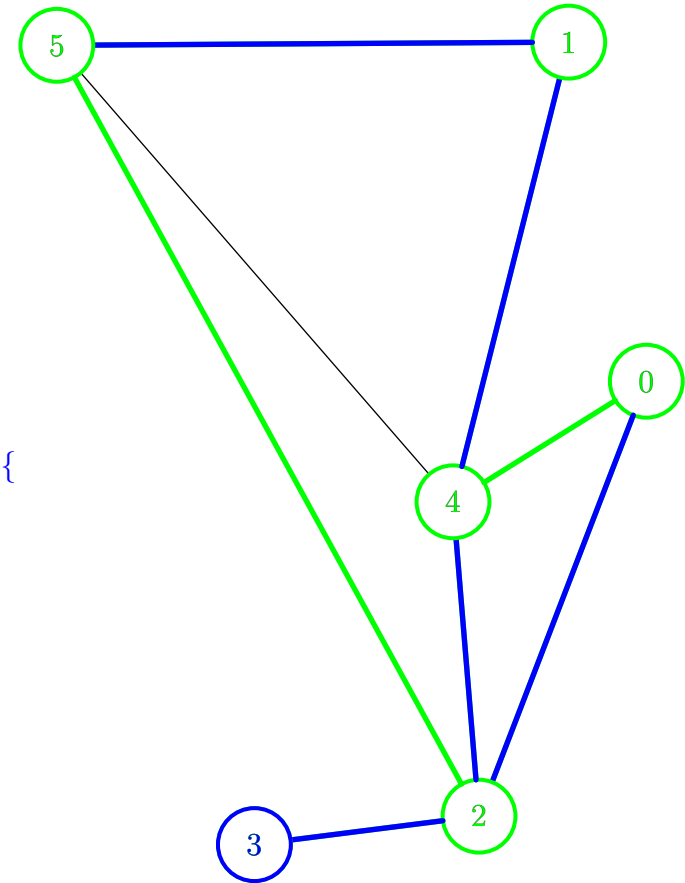
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=2 time=7



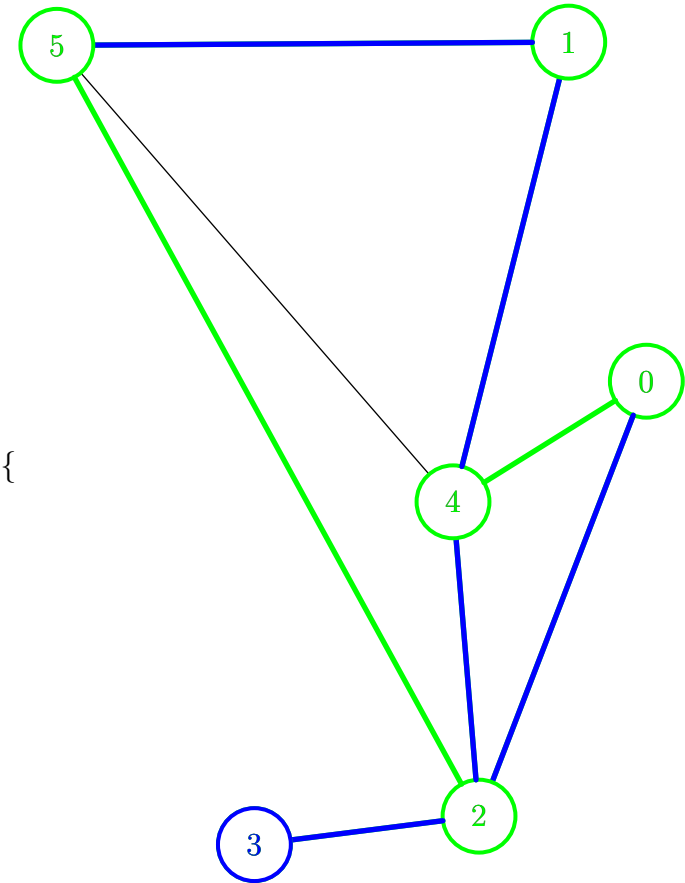
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=2 time=7



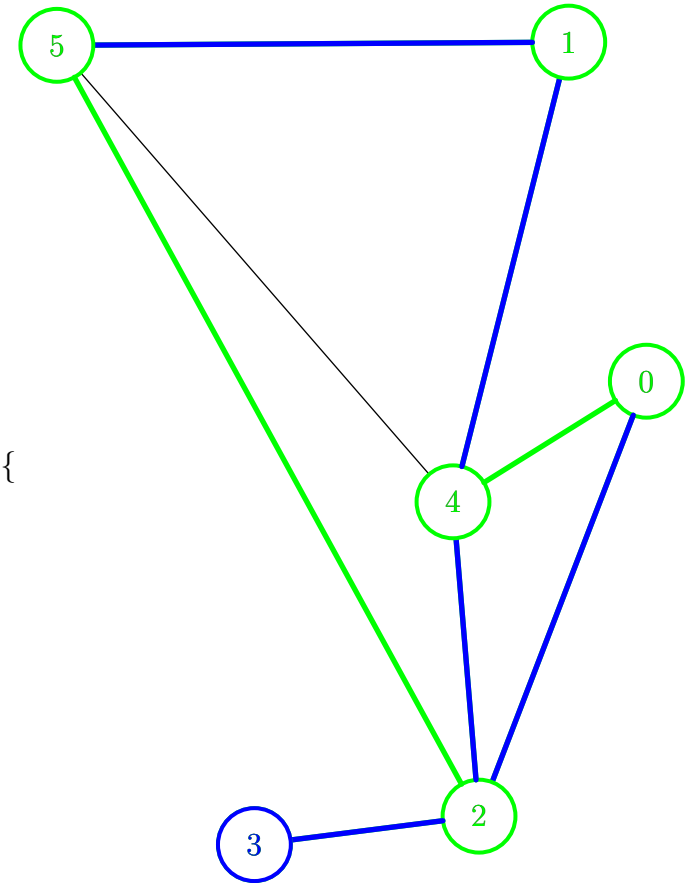
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=2 time=7



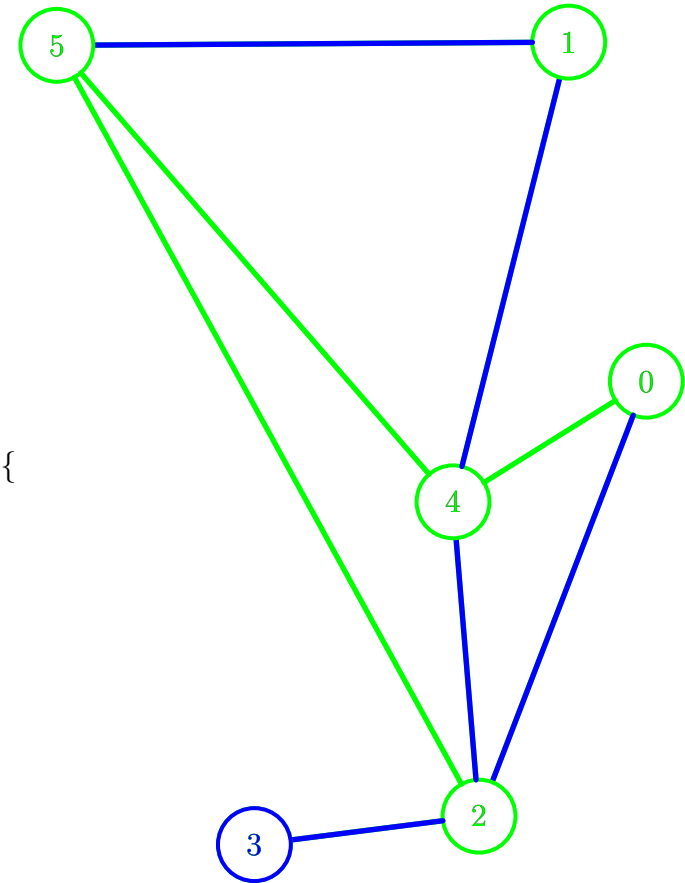
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=2 time=7



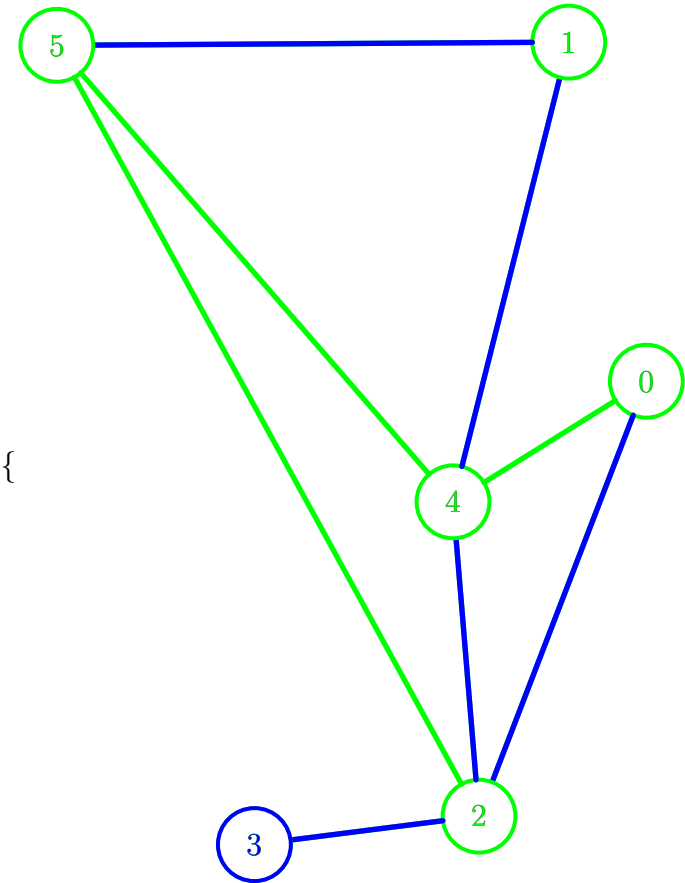
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=4 time=7



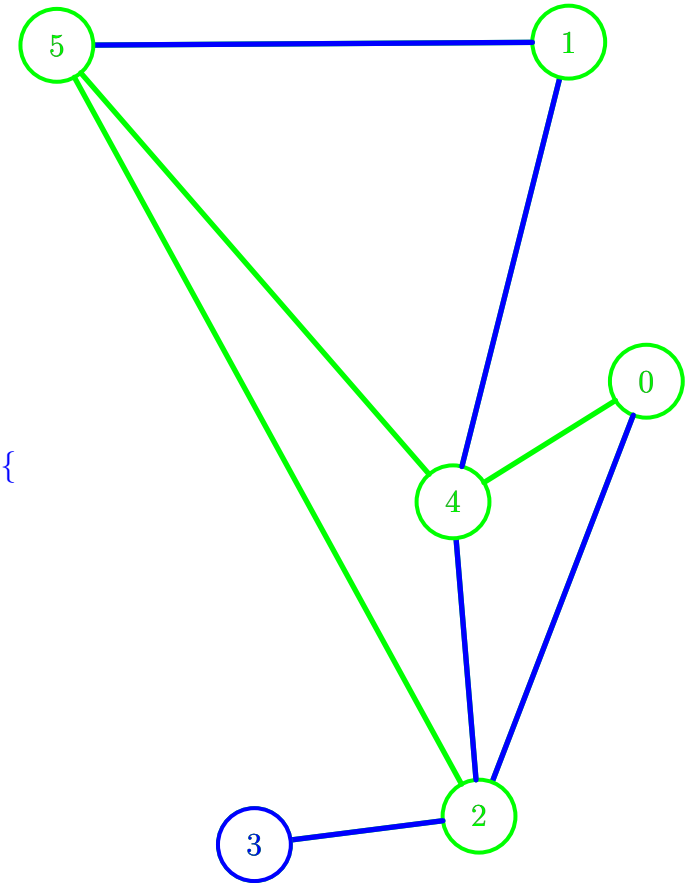
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=4 time=7



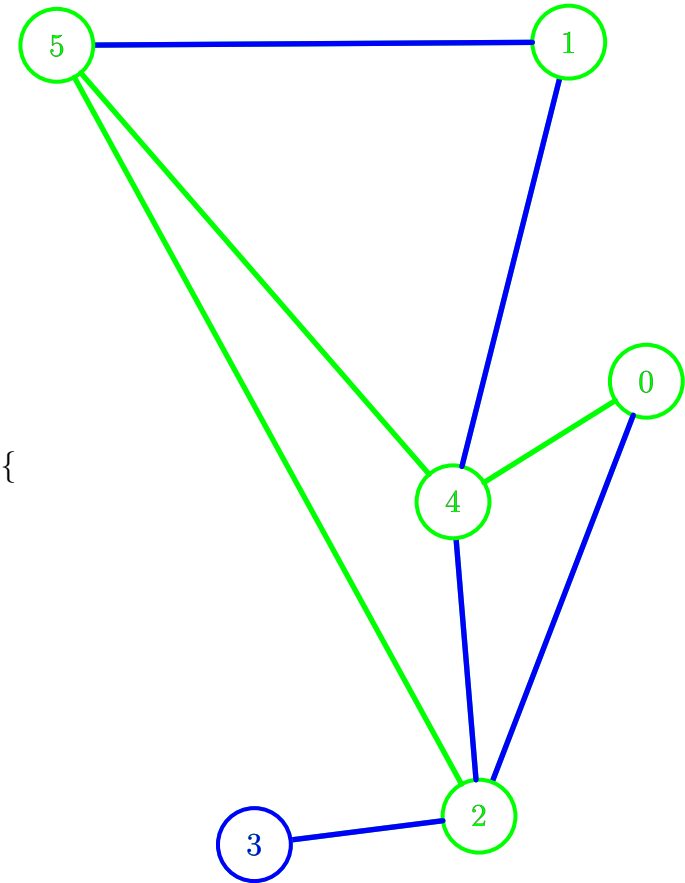
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=4 time=7



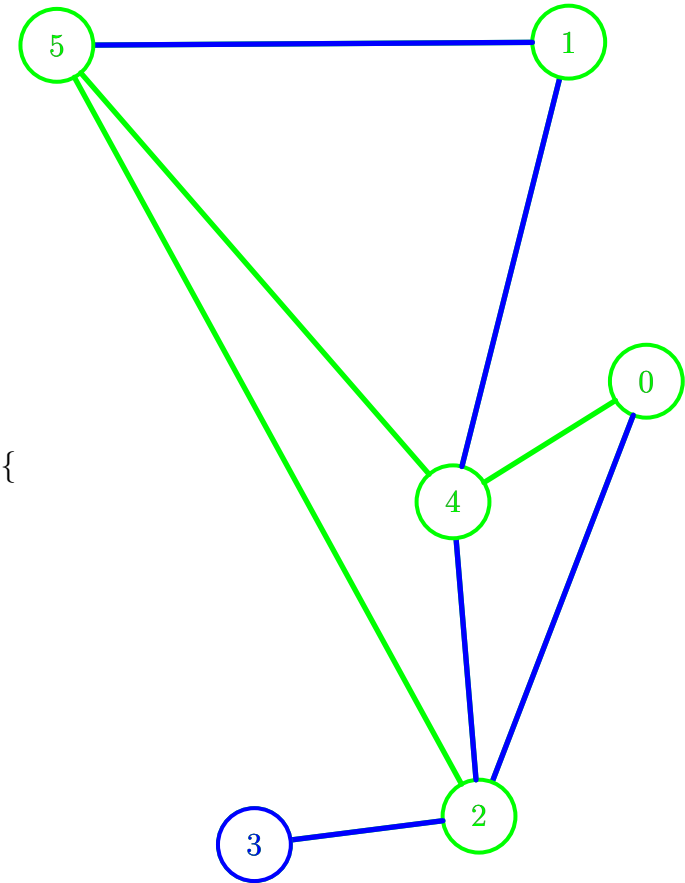
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=4 time=7



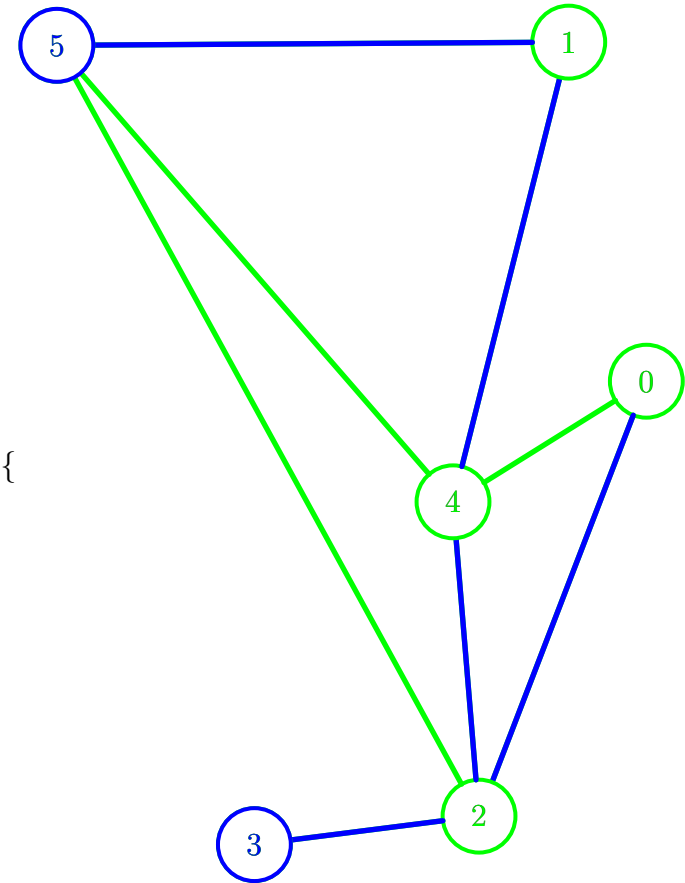
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=4 time=7



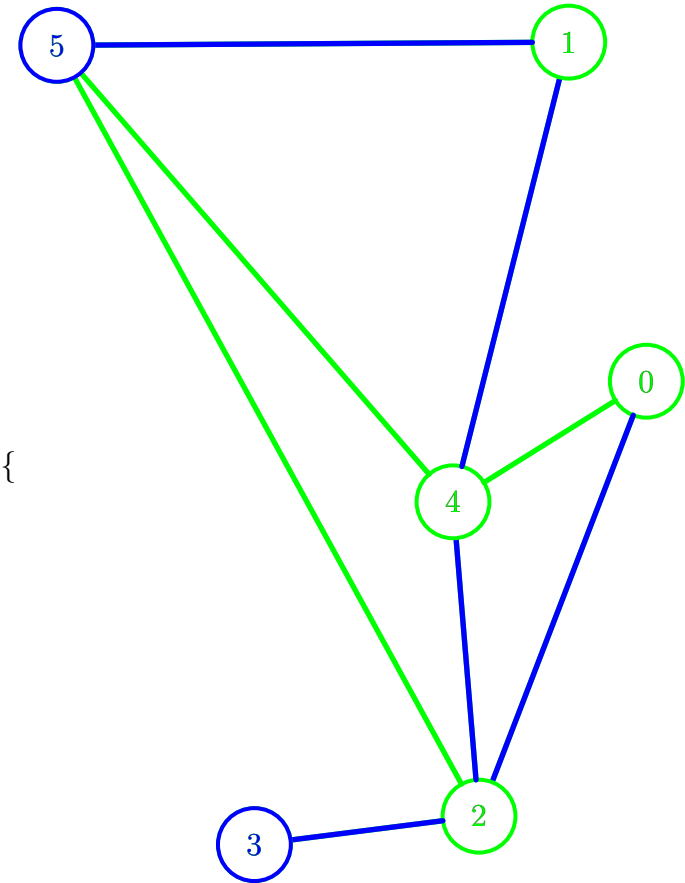
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=5 neighbour=nil time=8



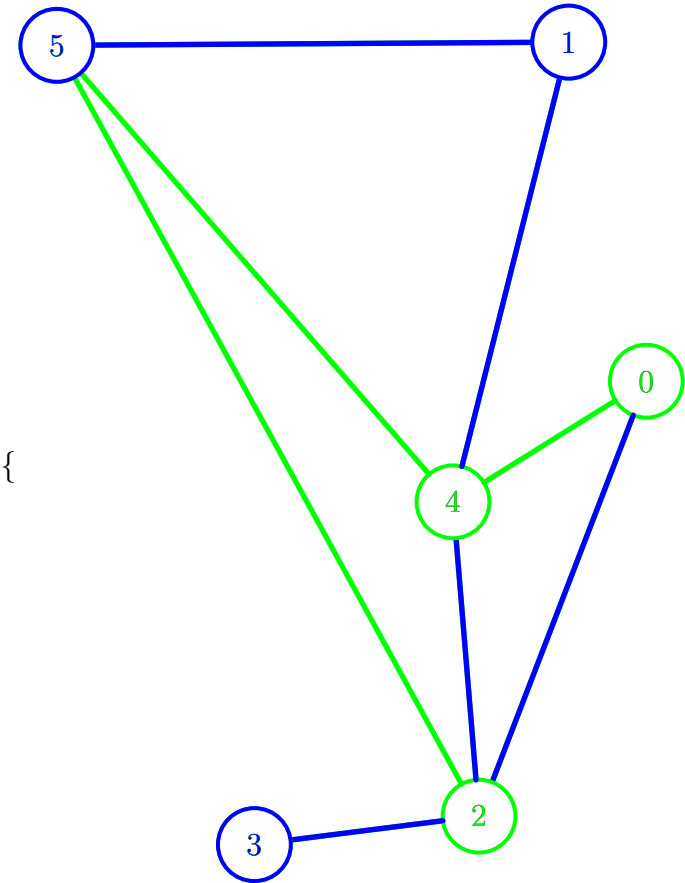
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=5 time=8



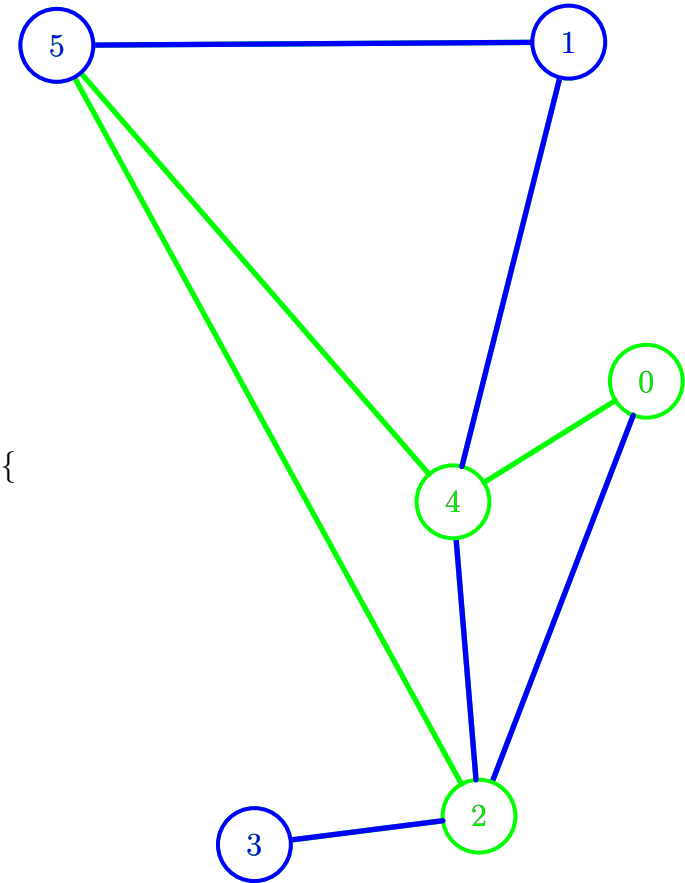
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=1 neighbour=nil time=9



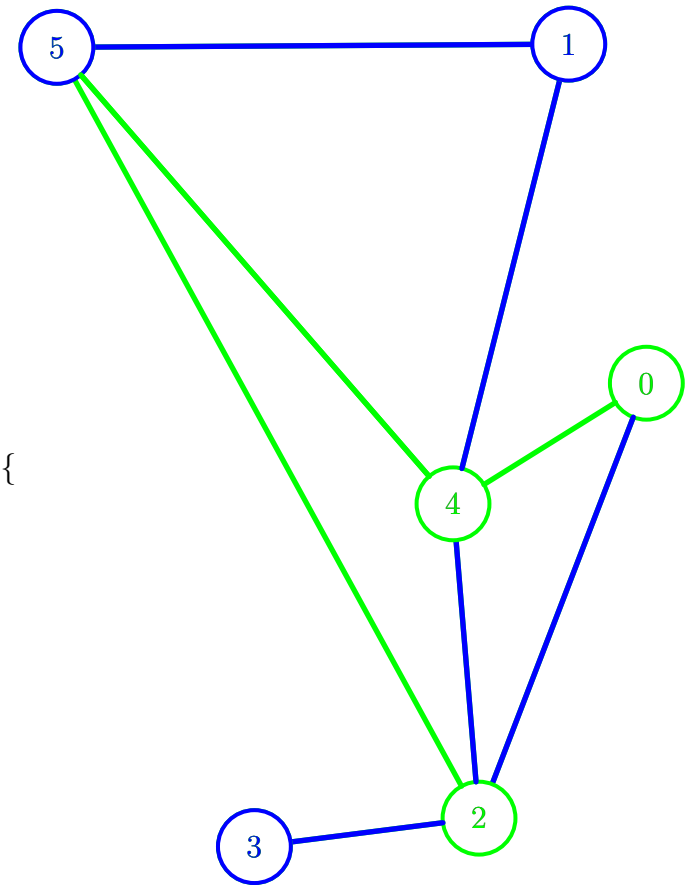
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=1 time=9



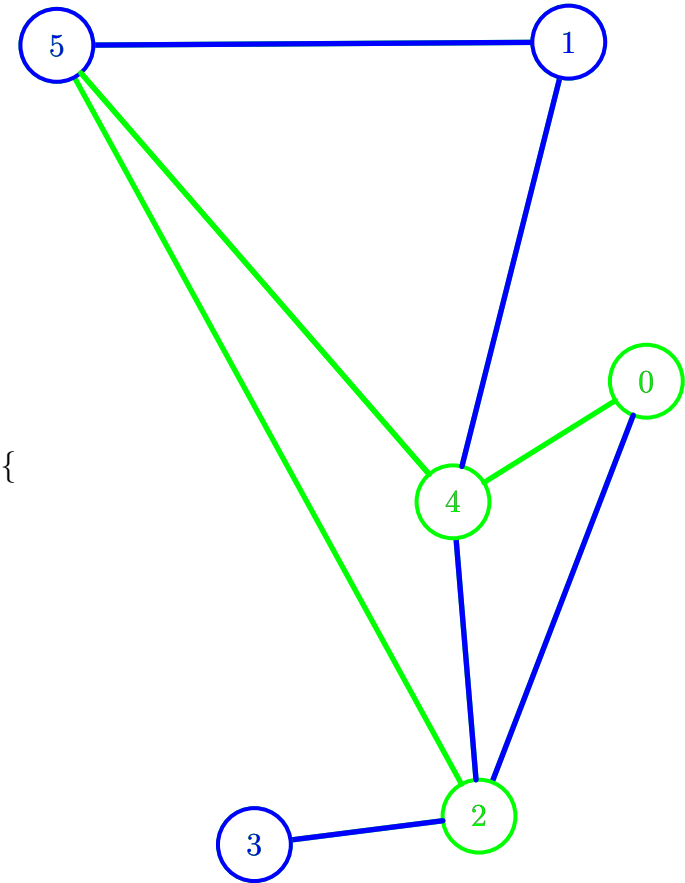
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=5 time=9



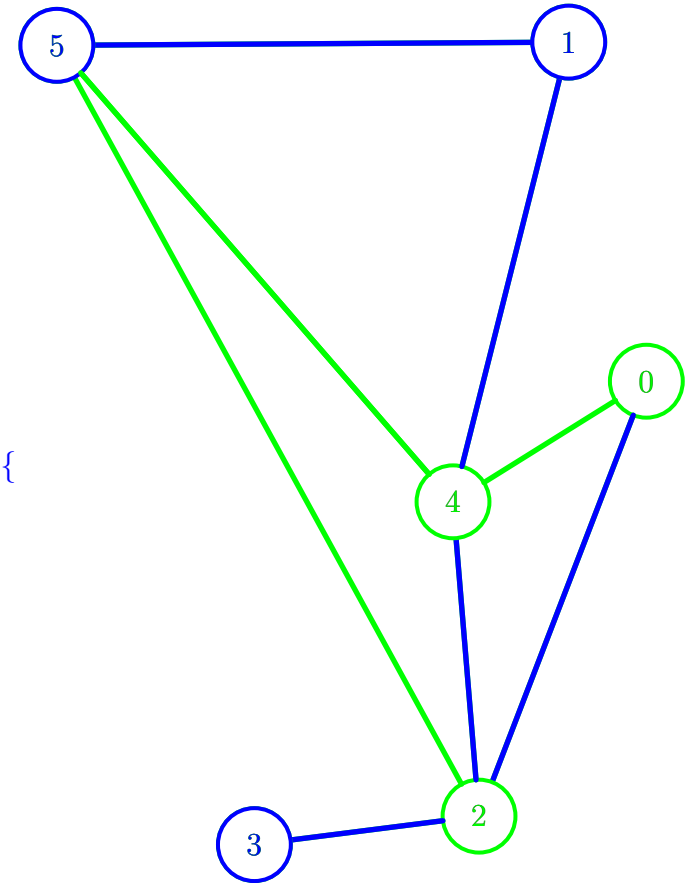
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=5 time=9



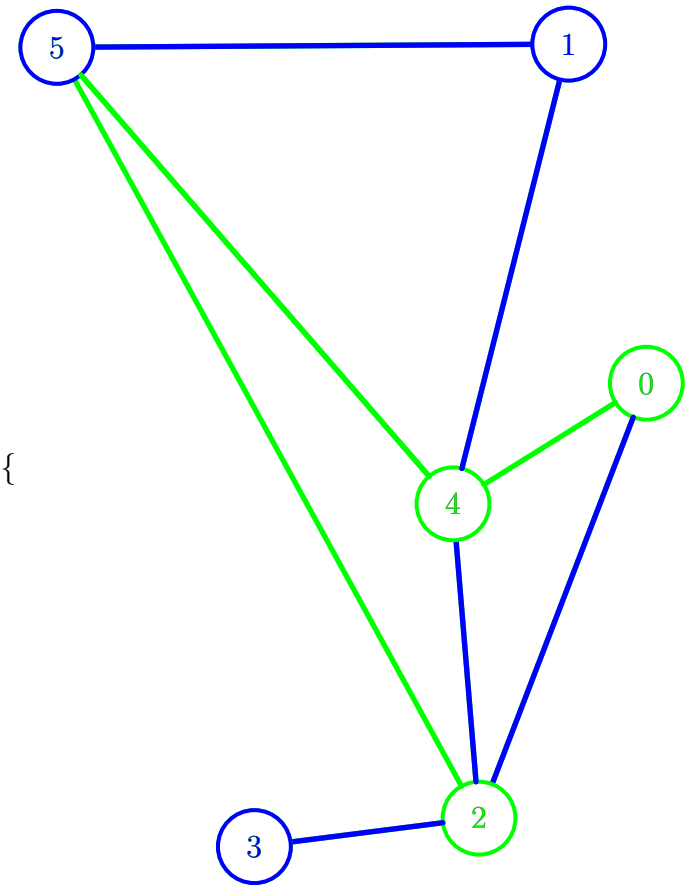
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=5 time=9



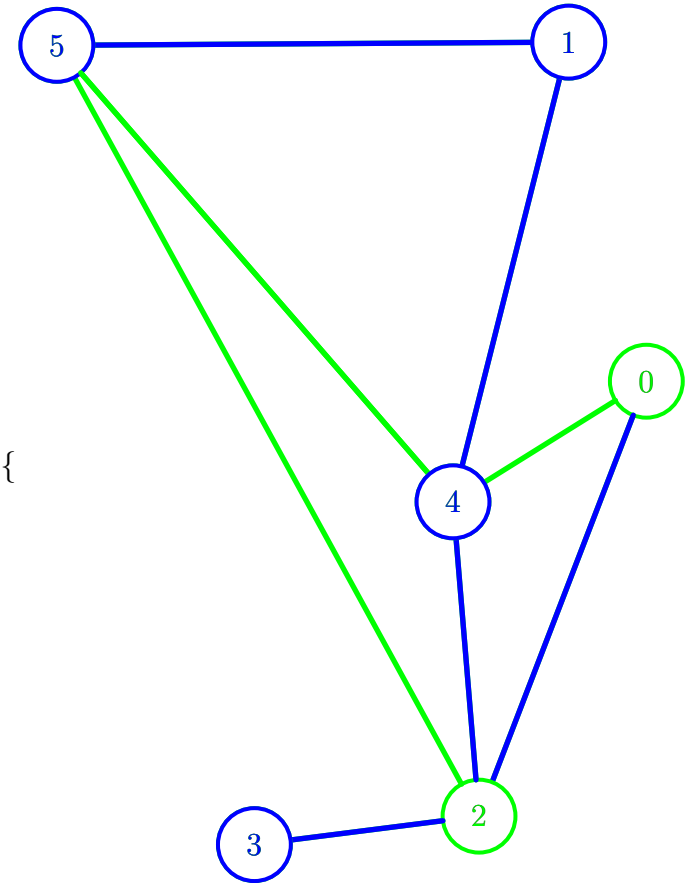
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=5 time=9



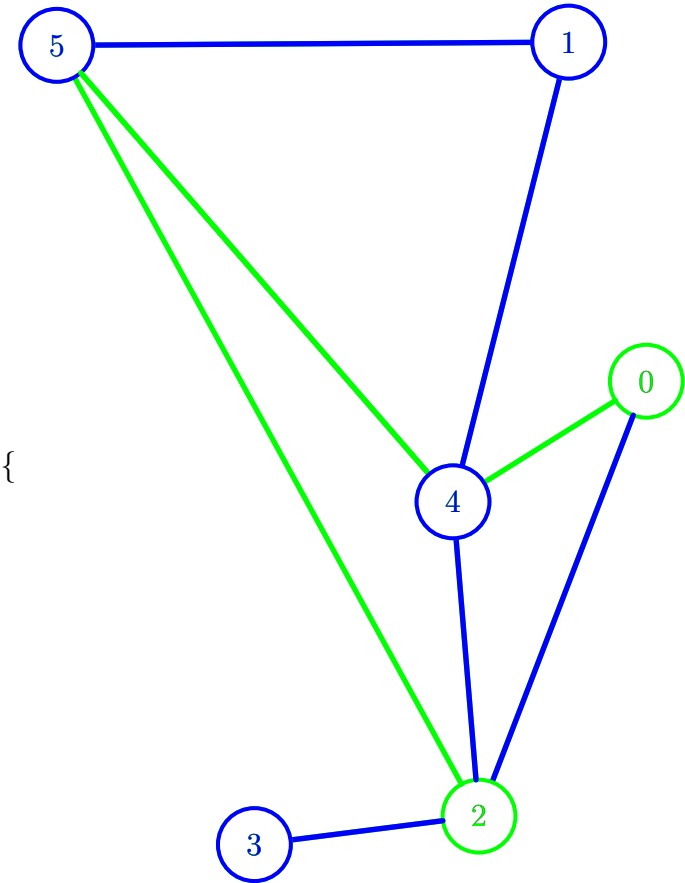
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLlate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=4 neighbour=nil time=10



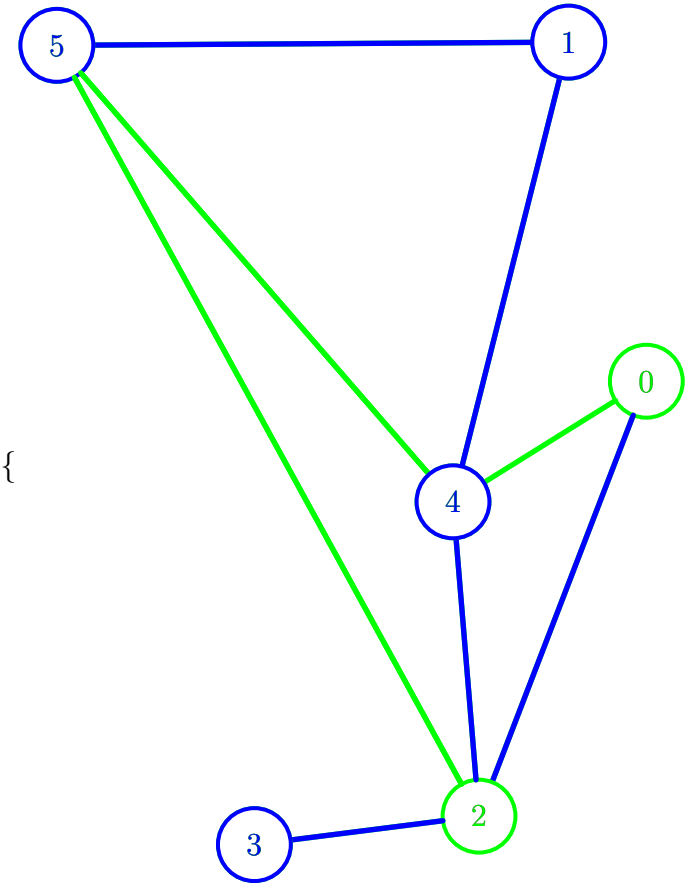
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=4 time=10



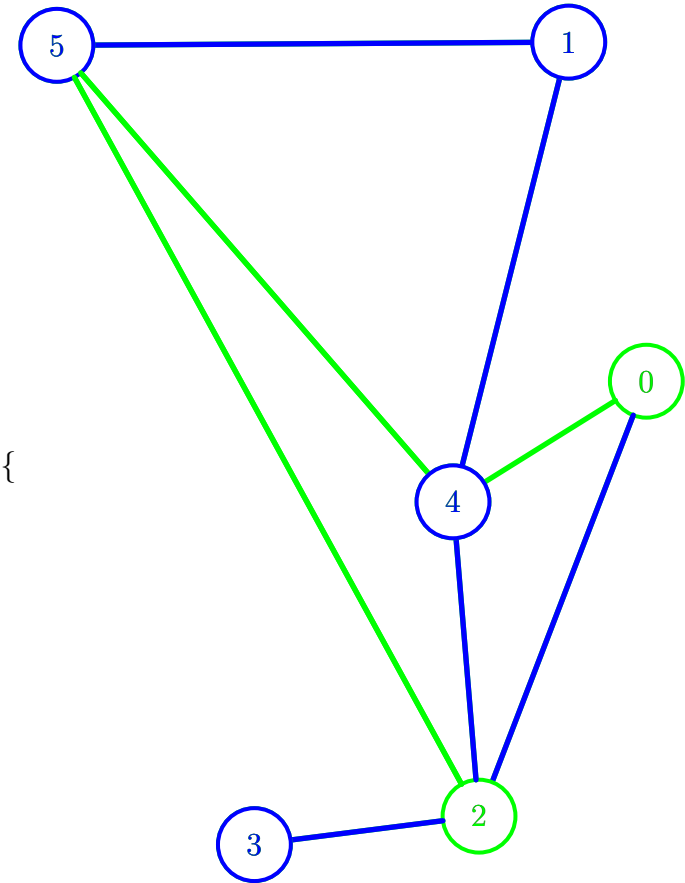
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=5 time=10



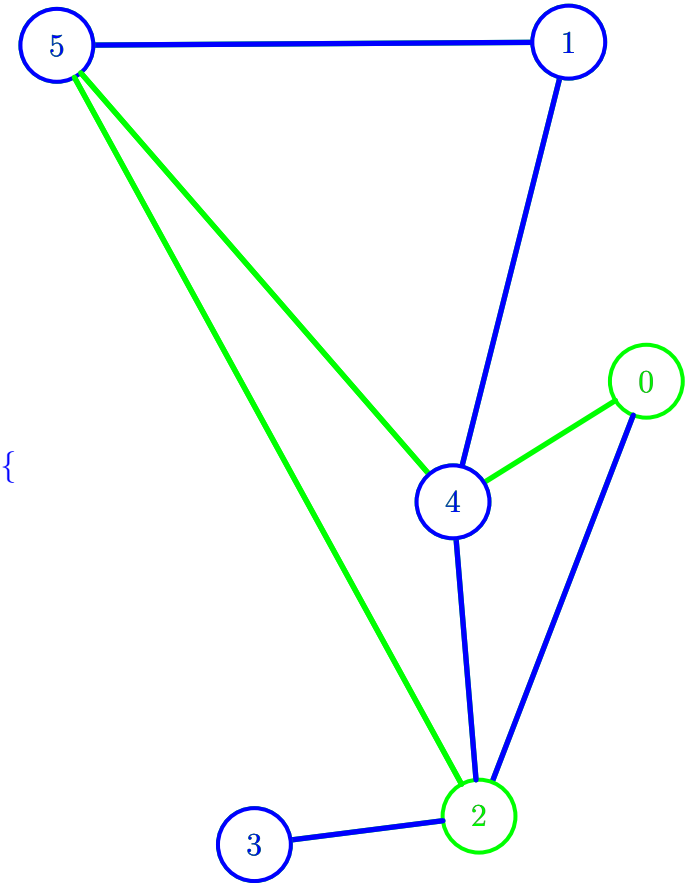
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=5 time=10



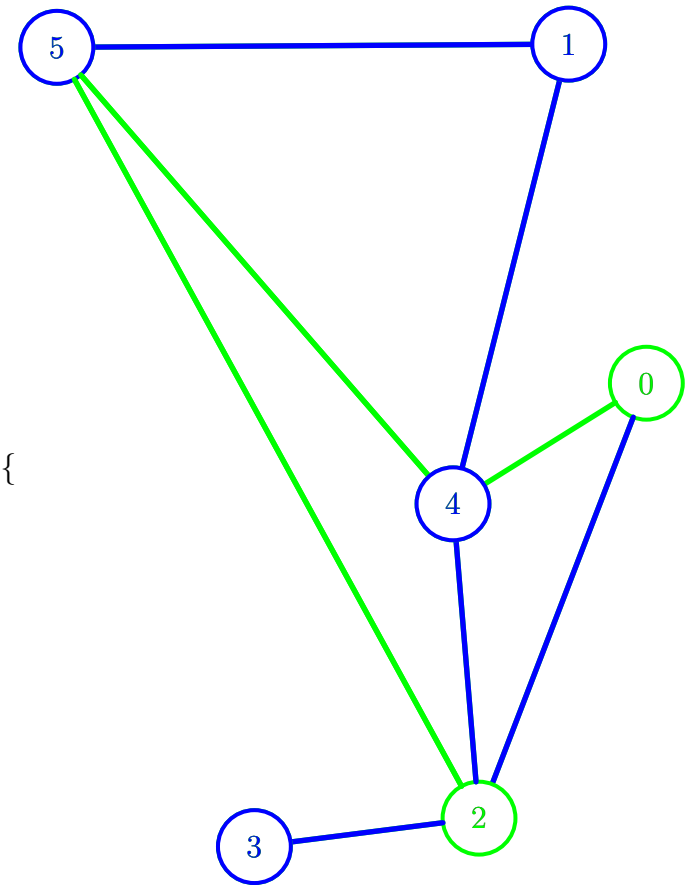
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=5 time=10



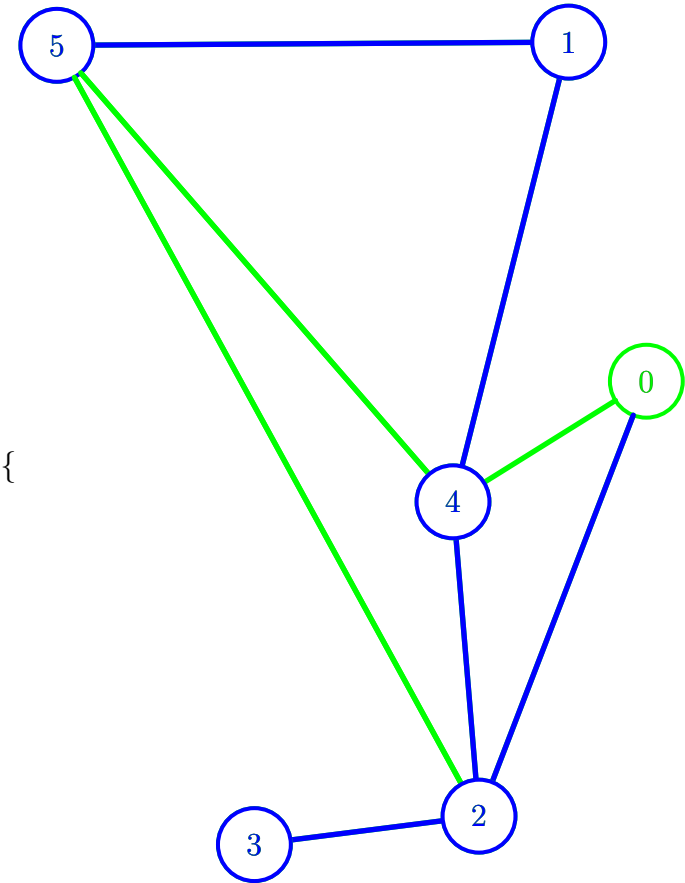
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=5 time=10



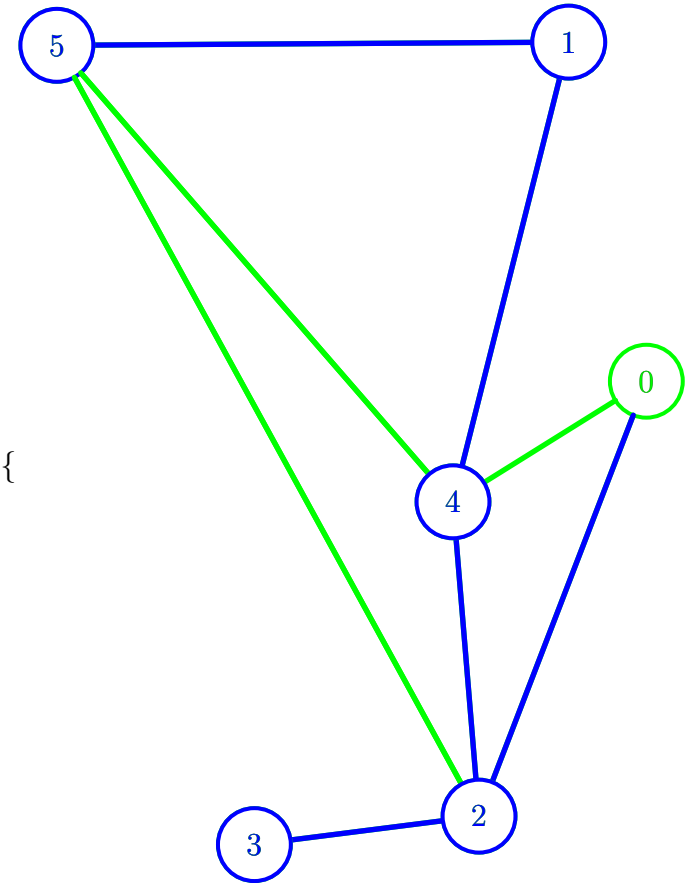
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=2 neighbour=nil time=11



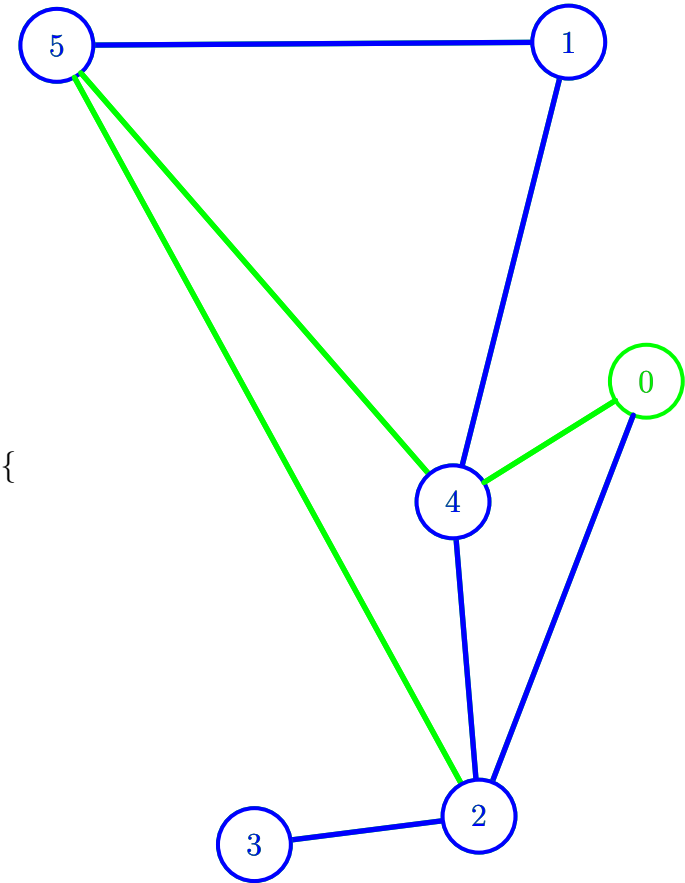
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=2 time=11



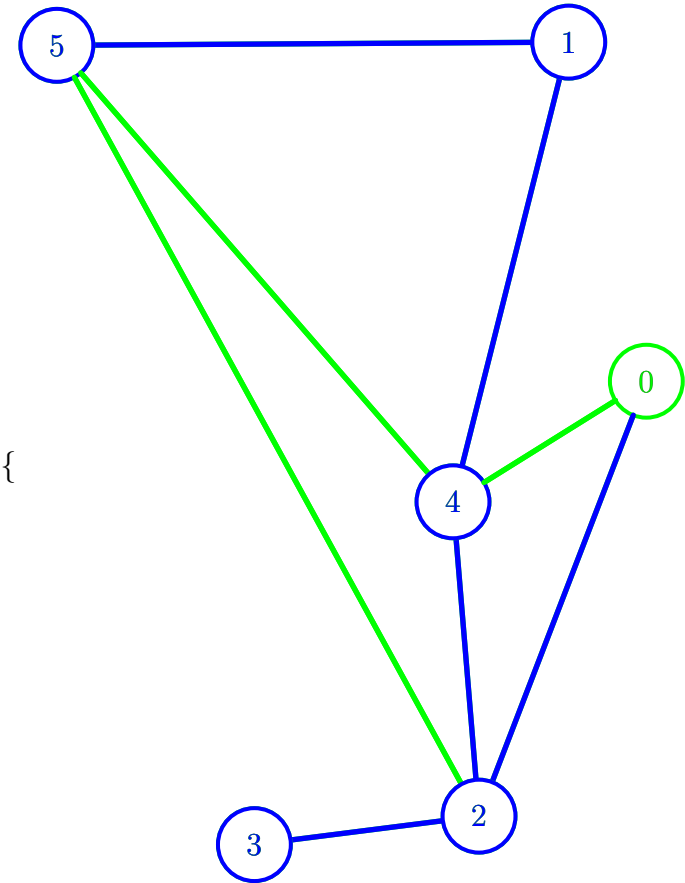
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=4 time=11



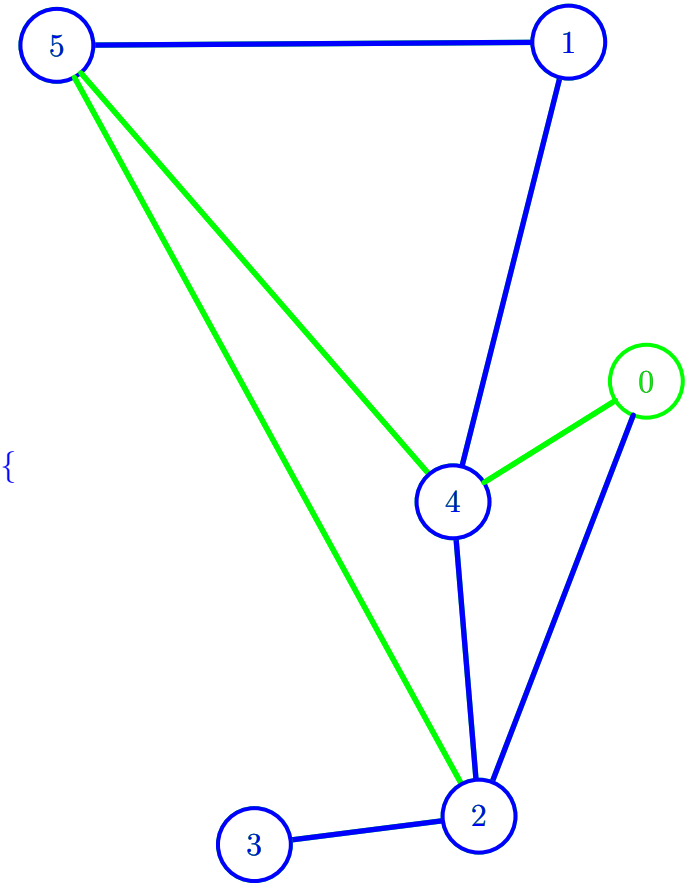
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=4 time=11



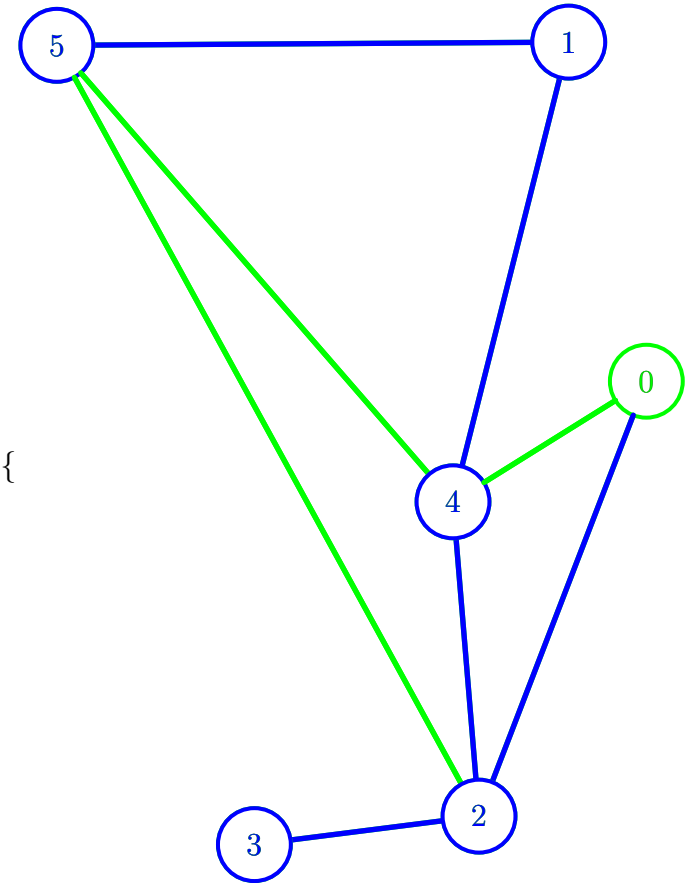
```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=4 time=11



```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=4 time=11



```
dfs(graph, node) {
  state ← Array[n, "undiscovered"]
  finished ← false
  dfs_recur(graph, node)
}

dfs_recur(graph, node) {
  if (finished) return
  state[node] ← "discovered"
  time ← time + 1
  processVertexEarly(node)
  foreach neighbour ∈ Neighbourhood(node) {
    if (state[neighbour] = "undiscovered") {
      parent[neighbour] ← node
      processEdge(node, neighbour)
      dfs_recur(graph, neighbour)
    } else if (state[neighbour] ≠ "processed") {
      processEdge(node, neighbour)
    }
    if (finished) return
  }
  processVertexLate(currentNode)
  state[currentNode] ← "processed"
  time ← time + 1
}
```

node=0 neighbour=nil time=12