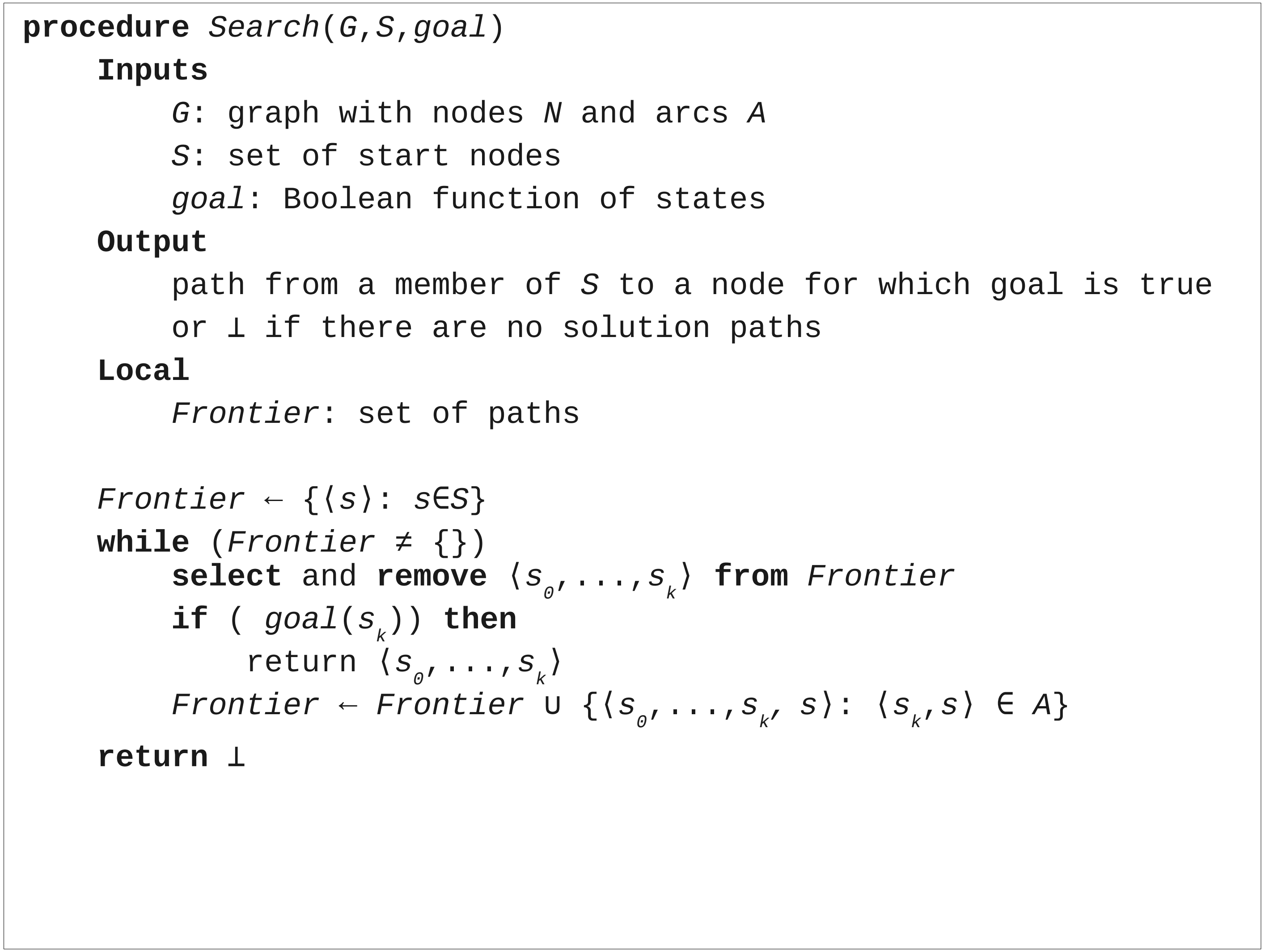
Undirected Search

Basic Search Algorithm



```
procedure Search(G,S,goal)
    Inputs
        G: graph with nodes N and arcs A
        S: set of start nodes
        goal: Boolean function of states
    Output
        path from a member of S to a node for which goal is true
        or ⊥ if there are no solution paths
    Local
        Frontier: set of paths

    Frontier ← $\{\langle s \rangle : s \in S\}$
    while (Frontier ≠ {})
        select and remove $\langle s_0, ..., s_k \rangle$ from Frontier
        if ( goal($s_k$)) then
            return $\langle s_0, ..., s_k \rangle$
        Frontier ← Frontier ∪ $\{\langle s_0, ..., s_k, s \rangle : \langle s_k, s \rangle \in A\}$
    return ⊥
```

(From Artificial Intelligence: Foundations of Computational Agents, Poole and Mackworth, 2010.)

Multiple Path Pruning

```
procedure Search($G$,$S$,$goal$)
    Inputs
        $G$: graph with nodes $N$ and arcs $A$
        $S$: set of start nodes
        $goal$: Boolean function of states
    Output
        path from a member of $S$ to a node for which goal is true
        or ⊥ if there are no solution paths
    Local
        $Frontier$: set of paths
        $Explored$: set of nodes that have been expanded

    $Frontier \leftarrow \{\langle s \rangle : s \in S\}$
    $Explored \leftarrow \{\}$
    while ($Frontier \neq \{\}$)
        select and remove $\langle s_0, \ldots, s_k \rangle$ from $Frontier$
        $Explored \leftarrow Explored \cup \{s_k\}$
        if ( $goal(s_k)$) then
            return $\langle s_0, \ldots, s_k \rangle$
        $Frontier \leftarrow Frontier \cup \{\langle s_0, \ldots, s_k, s \rangle : \langle s_k, s \rangle \in A \wedge$
                                                  $s \notin Frontier \wedge$
                                                  $s \notin Explored\}$

    return ⊥
```

Abuse of notation

Multiple Path Pruning

Depth First Search

```
procedure DepthFirstSearch(G,S,goal)
    Inputs
        G: graph with nodes N and arcs A
        S: set of start nodes
        goal: Boolean function of states
    Output
        path from a member of S to a node for which goal is true
        or ⊥ if there are no solution paths
    Local
        Frontier: a stack of paths
        Explored: set of nodes that have been expanded

    Frontier ← Empty Stack
    Frontier.push(⟨s⟩) for s ∈ S
    Explored ← {}
    while (Frontier is not empty)
        Pop $\langle s_0,...,s_k \rangle$ from Frontier
        Explored ← Explored ∪ $\{s_k\}$
        if ( goal($s_k$)) then
            return $\langle s_0,...,s_k \rangle$
        For all $\{\langle s_k,s \rangle : \langle s_k,s \rangle \in A \wedge s \notin Frontier \wedge s \notin Explored\}$
             Frontier.push($\langle s_0,...,s_k, s \rangle$)
    return ⊥
```

For Breadth-First Search, Just replace the Stack with a Queue.