```
dfs(graph, start, goal)
    q = [[start]]
    while q not empty:
        Parent = q.pop(0)
        check if Parent ends in goal -> if so return Parent
    extendedPaths = extensions of parent
    ignore Extended Paths that add end creating a loop
    q = nonLoopPaths + q

bfs(graph, start, goal)
    q = [[start]]
    while q not empty:
        Parent = q.pop(0)
        check if Parent ends in goal -> if so return Parent
    extendedPaths = extensions of parent
    ignore Extended Paths that add end creating a loop
    q = q + nonLoopPaths

hillClimbing(graph, start, goal)
    q = [[start]]
    while q not empty:
        Parent = q.pop(0)
        check if Parent ends in goal -> if so return Parent
    extendedPaths = extensions of parent
    ignore Extended Paths that add end creating a loop
    sort extended Paths by HEURISTIC (of end -> goal)
    q = nonLoopPaths + q

bestFirst(graph, start, goal)
    q = [[start]]
    while q not empty:
        sort ENTIRE q by heuristic
        Parent = q.pop(0)
        check if Parent ends in goal -> if so return Parent
    extendedPaths = extensions of parent
    ignore Extended Paths that add end creating a loop
    q = nonLoopPaths + q
```

```
beamSearch(graph, start, goal)
    q = {0:[[startNode]]}
    level = 0
    while there are still paths in our level:
        parent = level.pop(0)
        if parent ends in goal -> return parent
        find nonLoop extensions of parent
        nextLevel = nextLevel + nonLoopExtensions
        sort next level paths
        remove after first W
        if our current level is out of paths to pop/extend
            -> start at next level

branchAndBound(graph, start, goal)
    q = [[start]]
    while q not empty:
        sort ENTIRE q by LENGTH
        parent = q.pop(0)
        check if parent ends in goal -> if so return parent
        extendedPaths = extensions of parent
        ignore extended paths that add end creating a loop
        q = nonLoopPaths + q

aStar(graph, start, goal)
    q = [[start]]
    while q not empty:
        sort Q by LENGTH + HEURISTIC
        parent = q.pop(0)
        check if parent ends in goal -> if so return parent
        extendedPaths = extensions of parent
        ignore extended paths that add end creating a loop
        q = q + nonLoopPaths
```