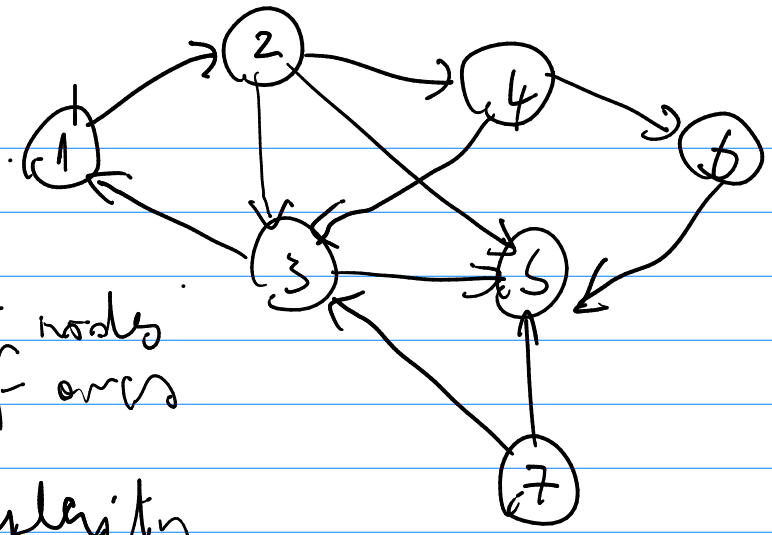
For any graph

we have

n number of nodes

m number of arcs



$n = 7$

$m = 12$

Computational Complexity

$O = n$

```
algorithm search;
begin
    unmark all nodes in N;
    mark node s;
    pred(s) : = 0;
    next : = 1;
    order(s): = s;
    LIST : = {s}
    while LIST ≠ ∅ do
    begin
        select a node i in LIST;
        If node i is incident to an admissible arc (i, j) then
        begin
            mark node j;
            pred( j) : = i;
            next : = next + 1;
            order( j) : = next;
            add node j to LIST;
        end
        else delete node i from LIST;
    end;
end;
```

Figure 3.4 Search algorithm.

```
function graph_search(s,aL,isDepth::Bool=true)
    # Initialization
    n=length(aL) #number of nodes
    pred=zeros(Int,n)
    marked=Int[s]
    List=Int[s]
    # Now I can stard doing the actual work
    while !isempty(List)
        i=isDepth? List[end]:List[1] #This is the current node
        j=find_addmissable(aL,marked,i)
        if j>0 # We found an addmissable arc
            push!(marked,j)
            pred[j]=i
            push!(List,j)
        else
            isDepth? pop!(List):shift!(List)
        end
    end
    pred
end
```

```
function find_addmissable(L,marked_list,tail)
    for head in L[tail]
        if !(head in marked_list)
            return head
        end
    end
    return -1 # Special value to indicate no addmissable arc
end
```