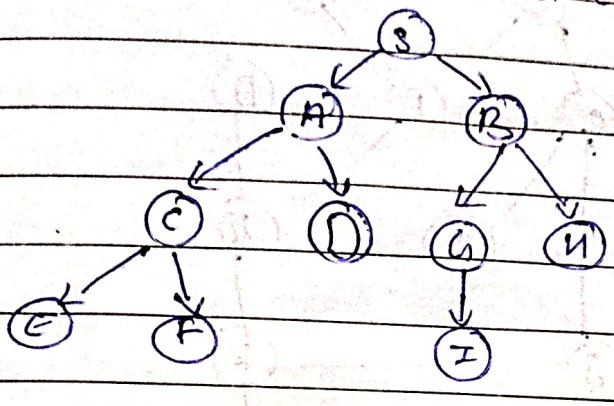
Rufna Gold
Date: Page:

# Laboratory - 2

→ Solutions for given problems and pseudocode

1. perform Breadth first search (BFS)




frontier [S | A | B | C | D | G | H | E | F | I | K]

Explored: [S | A | B | C | D | G | H | E] → Goal reached.

pseudo code:

function BFS (problem) returns a solution node or failure

node ← NODE (problem, INITIAL)

if problem is GOAL (node, STATE) then return node

frontier ← a fifo queue, with node as an element

reached ← {problem, initial}

while not ISEMPTY (frontier) do

node ← pop (frontier)

for each child in Expand (Problem, node) do

s ← child STATE

if problem IS_GOAL (s) then return child

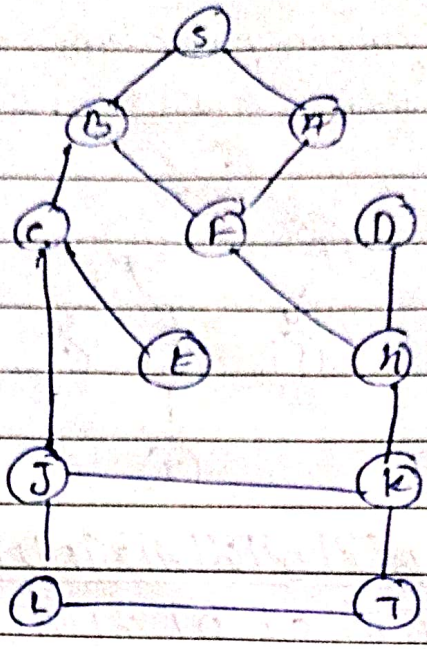
if s is not in reached then

add s to reached

add child to frontier

return failure

2) Performing DFS both tree search & Graph search




pseudocode:

function DFS_tree_search(problem) returns a solution node or failure:
    frontier ← a LIFO queue (stack) with NODE (problem, initial) as an element
    while not ISEMPTY(frontier) do
        node ← pop(frontier)
        if problem.IS_GOAL(node.STATE) then return node
        if not IS_CYCLE(node) then
            for each child in EXPAND(problem, node) add child to frontier
    return failure

Frontier:

S

~~B~~ A

~~C~~ F A

~~J~~ E F A

~~K~~ K E F A

~~T~~ K E F A

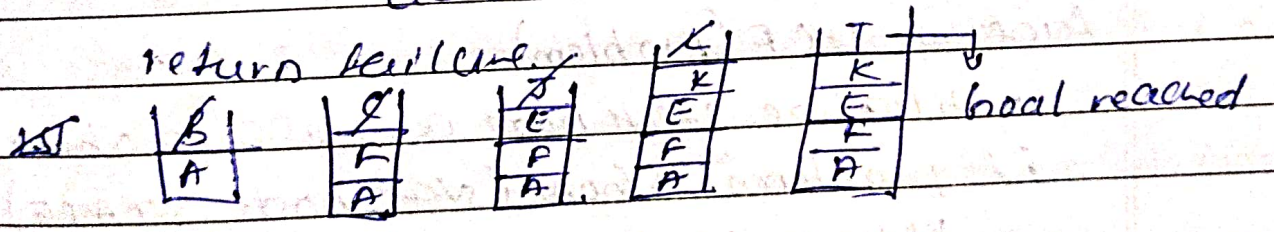
↓

Goal reached

path: S → B → C → J → L

DFS Graph-SEARCH

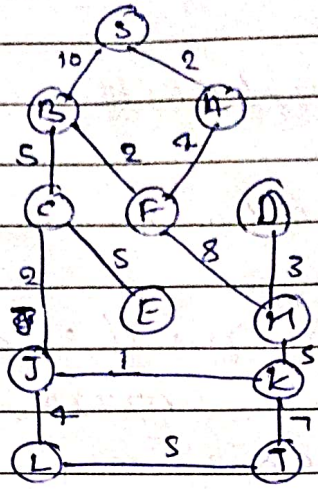
function G-S (problem) returns a solution node as failure.
frontier ← a LIFO Queue (stack) with NODE (problem INITIAL) as an element
explored ← empty set.
while not is Empty (frontier) do
node ← POP (frontier)
if problem IS_Goal (node STATE) then return node
if node STATE not in explored then
for each child in EXPAND (problem node) do
add child to frontier
return failure.




Explored: S B C J L

3) DFS Graph-search

Performing initial Search - cost methods




Frontier : S(0) | A(2) | B(10) | F(6) | H(14) | C(15) | K(19) | D(17) | J(17) | E(20) | L(21) | T(16)

&

Explored: S(0) | A(2) | A(6) | B(10) | H(19) | C(15) | J(17) | K(19) | T(16)

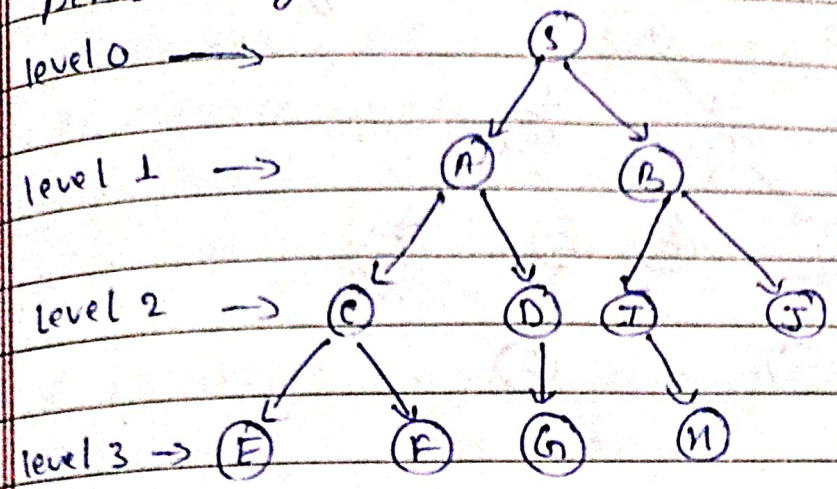
Total cost = 26

path: S → A → F → H → K → T

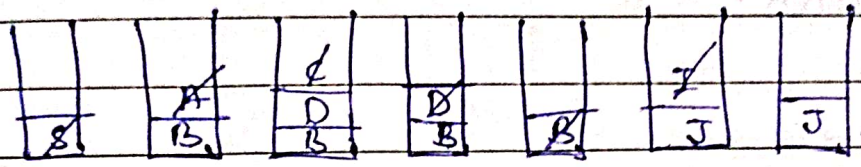
pseudocode:

```
function UCF(problem)
    initilialize start node as NODE(problem, INITIAL)
    if problems IS GOAL(start_node, STATE) then
        return state node solution()
    initialise frontier as empty priority queue
      push start_node into frontier
    initialise reached as empty set add startnode.
      STATE to reached
    while frontier is NOT EMPTY DO
        node ← pop frontier with longest cost
        for each child in problem expand(node) do
            S ← CHILD.STATE
            if problem IS GOAL(s) then
                return child solution
            if S not in reached then
                ADD s to reached
                push child into frontier,
    return failure
```

4) performing DLS




Frontier:




Explored: | S | A | C | D | B | I | J |

path: S → B → J

pseudocode:

function DLS (problem, l) returns a node or failure or cutoff

frontier ← a LIFO queue (stack) with node(problem, initial) as an element.

result ← failure

while not is EMPTY (frontier) do

node ← pop (frontier)

if problem.IS_GOAL(node, STATE) the return node

if DEPTH (node) > l then

result ← cutoff

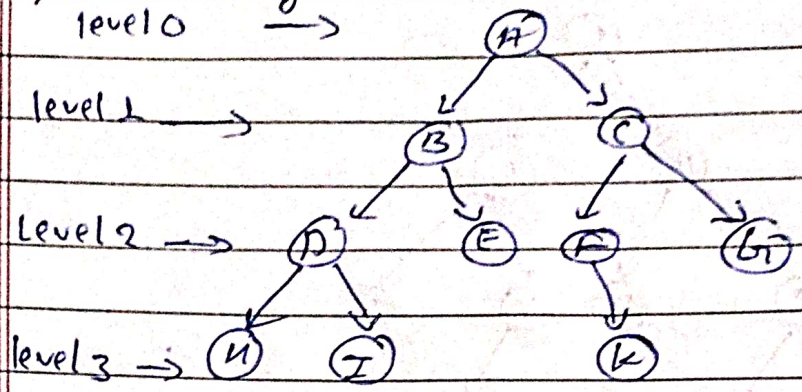
else if not IS_CYCLE (node) do

for each child in EXPAND (problem, node) do

add child to frontier.

return result.

(5) performing iterative deepening search.




Process:

A

A → B → C

A → B → D → E → C

A → B → D → E → C → F → G

path: A → C → G

pseudo code:

function IDS (Problem) returns a solution node or failure

for depth = 0 to ∞ do

result ← DLS (problem, depth)

if result ≠ cutoff then return result

18/10/24