Exp. No 4

Date: A* Search

Aim:

To implement A* search to find shortest path.

Algorithm:

Step 1: Start

Step 2: Add start node to open set.

Step 3: Remove the node with lowest 'f' value.

Step 4: Current node = goal node, reconstruct the path.

Step 5: For each node, calc 'g', 'h', 'f' values.

Step 6: If the node is not in open set or a lower cost,

Step 7: Stop. If goal is reached else repeat.

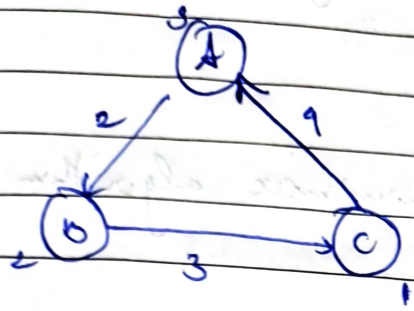
Program:

```
import heapq

def a_star(start, goal, h, nc):
    os = []
```

```
for _ in range(n):
    node = input("enter node:")
    neighbor_input = input(f"enter neighbor
                           cost:").split()

    neighbors = [neighbor_input[i], int(neighbor_
                 input(i+1)) for i in range
                 (0, len(neighbor_input), 2)]

    graph[node] = neighbors

    heur_val = int(input("Enter H(n))
    heur[node] = heur_val.

start = input("enter start node:")
goal = input("enter goal node:")

if start == goal.
    print("start & goal are same.
    return
path = a_star(start, goal, graph, heur)

if path:

    print("path found:", path)

else:

    print("no path found:")

if __name__ == "__main__":
    main()
```

Page No.:
Date:
YOUVA

Output:



Enter no. of nodes : 3
Enter Node: A
Enter neighbor : B 2 C 9
Enter h(n) : 3
Enter node : B
Enter neighbor : C 3
Enter h(n) : 2
Enter node : C
Enter Neighbor:
h(n) : 1
Enter Start : A
Enter goal : C

A B C

Result:

Thus A* algorithm successfully executed & verified.