J		and the second s			
1	Exponent -	02			
	the transport of the second of the second	Line is the same because the			
	Aim: Implement useedy BFS	and A * seasch technique ternique			
-	in any programming lang	wage			
	West to the other ways and	la tenta la			
	Theory:	the take to control (a)			
	which with the party	11111			
	* Informed spearch tach Ri	ques ?			
		The Mark the Same and the			
	It sefors to search algorith	my which help in po navigating			
	It sefors to search algorithms which help in in navigating large databases with certain available information about the				
	end and in space and r	nost widely used in large			
	end goal in season and redatabases whose unintermed	I seased algorithms can't			
	accusably to wrate preci-	se zent.			
	There one different types	of informed search strategies:			
1	17 Pure Heyristic or Mears	h.			
	I Pure Heuristic & search  I Best first or yreedy  I A* tree search  I A* graph search	Soort.			
	3 A# tree reach				
	4 A # graph search.				
	* Compare Best floot su	earth and A * search algorithm			
	But first spearch	A* seasch			
-	This algorithm visits the -	This algorithm visits the next state			
	next state based on heusilitis	based on heusistics fen ) = htg whome			
	function f (n) = h with longit	based on heusistics fen ) = htg where h' is cost of towel from particular			
		state & g is cost from start to good.			
1	FOR E	DUCATIONAL USE			

(Sundaran)

	16 i country	
_	- It doesn't consider the cost - It doesn't choose the m	act state
contractor	of the path to that portioner only with the lowest hi	eurajstic
	state. All it takes about value but it select	s the one
	is that which next state that gives the lowest vo	lux when
	from the worent state has considering its houristic w	it of
	is that which next state that gives the lowest volumes the woment state has considering its houristic a lower houristic getting to the state.	W
	Joseph Methodic Jeans Jeans	4
	Time maleritais D (1)	1
	Space complexity = 0 (b) Space complexity: 0 (b)	0)
	The state of the s	A pr
	The state of the s	to be
	The second second second	<u>[]</u>
	The distribution of the line	
		n)
	and the second in any three the second	
	Le Lie de la	
	- Land and the second	
	Habry Harry	
	maked dell by the	le .
	The was a series of him terres . That the majories	5
	in the state of th	
e pt	The same of the same that the same of the	al T
	The Comment which is made in many and the	
Contract 1		
relación	the same of the sa	

	Movash Bean DIZC U8
14, 1	. R of support on a try dish dred 20 Bl
	6 1
a	2 (3) (6) 99
J	a called the letter was est
	$(\alpha)$
TITE S	on a relievo las for land of mair o
	3 (e) — (D) white white out the non- 900
	7 6
	Motor of the fact burning
	Find optimal path form A-a using
÷	BLUDONNE BESTON AND LONG TO COLOR
Vijeas	5] H* seach
	CJ A" Stach
	Ans: a) Best first seach
	initially open = [7] closed [7]
	A > open[A] dosed[]
	initially Open = [], closed []  A -> open [A], closed []  A hugs + & B as neighbors so,  open = [B,t) closed = [A]
	open = [B,t] closed =[A]
	node (n) A B C D + G
	h(n) 11 6 99 1 7 0
	node (n) A B C D + G  h(n) 11 6 99 1 7 0  gln) 0 2 3 9 3 11
	hevoistic function for bfs = f(n) = g(n)
	The same was a second of the same of the s
4 - 4	
Sundaram	FOR EDUCATIONAL USE

Since, B is lest cost path so we traverse to B and add it to closed set. i.e. open = [t] closed = [A,B] e since or is goal state and available in open lot in we consider the path  $A \rightarrow B \rightarrow y$ and wat of a path is = 2 + 9 = 11Joseedy Best first Sousch

f(n) = h(n) -> heuristic function for greathgreedy

Best first & Bearch. - Add Pritial state A to open list

open = [A] closed = [7]

Add neighbors of A i.e B & t to open list

open = [B, t] closed = [A]

\$\frac{4(B)}{5(B)} = h(B) = 6

\$\frac{6(t)}{5(B)} < \frac{6}{5(E)} = \frac{6}{ open = [c, G, f] closed = [A, B] f(c) = 99f(G) = 0 i f(G) is minimum so we choose this path

The Ainall optimal path is A-B-G cost=249511

FOR EDUCATIONAL USE Sundaram)

Alexander Stem Mize CX

[] A seasch algorithm. heusistic function f(n) = g(n) + h(n) for open=[A] closed=[] Neighbors of A-B, topen = [B, t], closed = [A] f(B) = g(B) + h (B) = 7+6=15 f(t) > g(t) + h(t) = 7+4=14 : F(B) > F(E) .. We add to closed list & traverse it & add its neighbour to open open=[D, G, B] double [A, 186] f(D) = 1+1=2 f(G)=11+0 f(B)= ; f(D) is minimum we add D to Uosed open = [G, B] closed = [A, E, D] f(G)-11 f(B) = 15 of the have reached goal state. Fio Final Path A - 6 - D - G cot cost = 3+6+1=10// (Sundaram) FOR EDUCATIONAL USE

### **Experiment 05**

## 1. Best First Search

#### Code:

```
from queue import PriorityQueue
graph = [[] for i in range(v)]
def bfs(source, target, n):
    visited = [False] * n
    pq = PriorityQueue()
    pq.put((0, source))
    while pq.empty() == False:
        u = pq.get()[1]
        print(u, end=" ")
        visited[u]=True
        if u == target:
            break
        for v, c in graph[u]:
            if visited[v] == False:
                visited[v] = True
                pq.put((c, v))
    print()
def addedge(x, y, cost):
    graph[x].append((y, cost))
    graph[y].append((x, cost))
addedge(0, 1, 3)
addedge(0, 2, 6)
addedge(0, 3, 5)
addedge(1, 4, 9)
addedge(1, 5, 8)
addedge(2, 6, 12)
addedge(2, 7, 14)
addedge(3, 8, 7)
addedge(8, 9, 5)
addedge(8, 10, 6)
addedge(9, 11, 1)
addedge(9, 12, 10)
addedge(9, 13, 2)
source = 0
```

```
target = 9
print(f"Source: {source} , Destination: {target}")
print("The Path is as follows")
bfs(source, target, v)
```

### **OUTPUT:**

```
Source: 0 , Destination: 9
The Path is as follows
0 1 3 2 8 9
```

### 2. A\* Search

### Code:

```
from collections import deque
class Graph:
    def___init__(self, adjacency_list):
        self.adjacency_list = adjacency_list
    def get_neighbors(self, v):
        return self.adjacency_list[v]
    def h(self, n):
        H = {
            'A': 1,
            'B': 4,
            'C': 1,
            'D': 3,
            'E': 2
        }
        return H[n]
    def a_star_algorithm(self, start_node, stop_node):
        open_list = set([start_node])
        closed_list = set([])
        g = \{\}
        g[start_node] = 0
        parents = {}
        parents[start_node] = start_node
```

```
n = None
            for v in open list:
                if n == None \text{ or } g[v] + self.h(v) < g[n] + self.h(n):
                    n = v;
            if n == None:
                print('Path does not exist!')
                return None
            if n == stop_node:
                reconst_path = []
                while parents[n] != n:
                    reconst_path.append(n)
                    n = parents[n]
                reconst_path.append(start_node)
                reconst_path.reverse()
                print('Path found: {}'.format(reconst_path))
                return reconst_path
            for (m, weight) in self.get_neighbors(n):
                if m not in open_list and m not in closed_list:
                    open_list.add(m)
                    parents[m] = n
                    g[m] = g[n] + weight
                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parents[m] = n
                        if m in closed list:
                             closed_list.remove(m)
                             open_list.add(m)
            open_list.remove(n)
            closed_list.add(n)
        print('Path does not exist!')
        return None
if___name__=="__main__":
        adjacency_list = {
        'A': [('C', 10), ('B', 7)],
        'B': [('D', 5), ('C', 2)],
```

while len(open\_list) > 0:

```
'C': [('E', 12)],
'D': [('E', 13)]
}
graph1 = Graph(adjacency_list)
graph1.a_star_algorithm('A', 'E')
```

### **OUTPUT:**

```
Path found: ['A', 'B', 'C', 'E']
```

# **Conclusion:**

In this experiment we learned about different informed search strategies and have compared them with each other. We also have successfully implemented best first search and A\* search in python getting desired results from the program.