Tugas Individu 1

The A* Informed Search Algorithm for Finding the Shortest Distance

Salsabila Fatma Aripa - 5025211057

Algoritma A*

Pencarian **A* (A-star)** adalah algoritma pencarian yang digunakan untuk mencari jalur terpendek atau solusi terbaik dari suatu masalah, dengan mempertimbangkan biaya atau jarak yang ditempuh serta estimasi biaya atau jarak yang tersisa. Rumus umum untuk algoritma pencarian A* adalah sebagai berikut:

f(n) = g(n) + h(n)

dimana:

f(n) : adalah nilai total biaya atau jarak dari simpul awal ke simpul tujuan.

g(n) : adalah biaya aktual yang telah ditempuh dari simpul awal ke simpul n.

h(n) : adalah heuristik yang memberikan perkiraan biaya atau jarak tersisa dari simpul n ke simpul tujuan.

Penyelesaian Masalah

Tugas : Generate graph node/kota dan edge beserta cost, Tentukan start dan goal dan generate SLD setiap node ke goal, dan selesaikan dengan A*

Lalu didapatkan penyelesaian sebagai berikut :

1. Membuat graph node/kota dan edge beserta cost
2. Menentukan Start dan Goal
3. Generate SLD setiap node ke goal
4. Mengimplementasikan algoritma A* untuk mencari jalur terpendek
5. Menerapkan algoritma A* pada graf

IMPLEMENTASI

```
import heapq

# Generate graph node/kota dan edge beserta cost
graph = {
    'A': {'B': 5, 'D': 9, 'E': 3},
    'B': {'A': 5, 'C': 2},
    'C': {'B': 2, 'G': 7},
    'D': {'A': 9, 'F': 4},
    'E': {'A': 3, 'F': 2},
    'F': {'D': 4, 'E': 2, 'G': 6},
    'G': {'C': 7, 'F': 6}
}

# Tentukan start dan goal
start = 'A'
goal = 'F'
```

IMPLEMENTASI

```
# Generate SLD setiap node ke goal
sld = {
    'A': 11,
    'B': 8,
    'C': 7,
    'D': 6,
    'E': 5,
    'F': 3,
    'G': 0
}
```

IMPLEMENTASI

```python
def astar(graph, start, goal, sld):
    # Inisialisasi nilai heuristik awal dan jarak sejauh ini dari start ke
setiap node
    heuristics = {node: sld[node] for node in graph}
    distances = {node: float('inf') for node in graph}
    distances[start] = 0

    # Inisialisasi priority queue dan masukkan node start
    queue = [(heuristics[start], start)]
    heapq.heapify(queue)

    while queue:
        # Pop node dengan heuristik terkecil dari priority queue
        current_heuristic, current_node = heapq.heappop(queue)

        # Jika node yang di-pop adalah goal, kembalikan jarak sejauh ini
        if current_node == goal:
            return distances[current_node]
```

IMPLEMENTASI

```
        # Loop melalui tetangga dari node saat ini
        for neighbor, weight in graph[current_node].items():
            # Hitung jarak baru dari start ke tetangga melalui node saat ini
            distance = distances[current_node] + weight

            # Jika jarak baru lebih kecil dari jarak sejauh ini dari start ke
tetangga, update nilai
            if distance < distances[neighbor]:
                distances[neighbor] = distance

                # Hitung nilai heuristik baru dari tetangga ke goal dan
tambahkan dengan jarak sejauh ini
                heuristic = distance + sld[neighbor]
                heuristics[neighbor] = heuristic

                # Masukkan tetangga ke priority queue dengan nilai heuristik
baru
                heapq.heappush(queue, (heuristic, neighbor))

    # Jika goal tidak ditemukan, kembalikan None
    return None
```

IMPLEMENTASI

```
# Jalankan algoritma A* dan tampilkan jarak terpendek
distance = astar(graph, start, goal, sld)
if distance:
    print(f"Jarak terpendek dari {start} ke {goal} adalah {distance}")
else:
    print(f"Tidak ada jalur yang menghubungkan {start} dan {goal}")
```

Thank You !

Mengeluh hanya akan membuat kita
semakin tertekan