Bùi Hà Bảo Khanh

23110168

Bài tập lí thuyết tuần 1

Câu 1: Chạy tay thuật toán BFS, DFS và UCS.

Thuật toán BFS

Node =	L = []	father[] = current node	
v1	v2, v3, v4	v2, v3, v4	v1
v2	v3, v4, v7	v7	v2
v3	v4, v7, v6, v8	v6, v8	v3
v4	v7, v6, v8, v5	v5	v4
v7	v6, v8	v7	v3
v8	v12, v9, v10	v9, v10	v8
v12	v9, v10, v13	v13	v12
v9	v10, v13, v11	v11	v9
v10	v13, v11	v11	v9
v13	v11, v14	v14	v13
v11	v14, v15	v15	v11
v14	v15	v11	v14
v15	v16	v16	v15
v16	v17	v17	v15
v17	v18	v18	v17

Đường đi ngắn nhất: v1 → v3 → v8 → v9 → v11 → v15 → v16 → v17 → v18

Thuật toán DFS

Node	L = []	father[] = current node	
v1	v4, v3, v2	v4, v3, v2	v1
v4	v5, v3, v2	v5	v4
v5	v14, v11, v3, v2	v14	v5
v14	v11, v3, v2	v11	v14
v11	v15, v10, v3, v2	v10, v19	v11
v15	v16, v10, v3, v2	v16	v15
v16	v17, v10, v3, v2	v17	v16
v17	v18, v10, v3, v2	v18	v17

Đường đi ngắn nhất: v1 → v4 → v5 → v14 → v11 → v15 → v16 → v17 → v18

Thuật toán UCS

PQ = {(start, 0)}

PQ = {(v2, 50), (v4,300), (v3, 350)}

PQ = {(v4, 300), (v3, 350), (v7, 650)}

PQ = {(v3, 350), (v7, 650), (v5, 1600)}

PQ = {(v6, 450), (v7, 650), (v8, 1250), (v5, 1600)}

PQ = {(v7, 650), (v12, 1150), (v8, 1250), (v5, 1600)}

PQ = {(v12, 1150), (v8, 1250), (v3, 1450), (v5, 1600)}

PQ = {(v8, 1250), (v3, 1450), (v5, 1600), (v13, 2100)}

PQ = {(v3, 1450), (v10, 1550), (v5, 1600), (v9, 2040), (v13, 2100)}

PQ = {(v10, 1550), (v9, 2040), (v13, 2100), (v14, 300)}

PQ = {(10, 1550), (v13, 2100), (v14, 3000), (v11, 3240)}

PQ = {(v14, 2700), (v11, 3240)}

PQ = {(v11, 3240)}

PQ = {(v10, 4040), (v15, 3640)}

PQ = {(v16, 4760)}

PQ = {(v17, 5530)}

PQ = {(v18, 6720)}

Đường đi ngắn nhất : v1 → v3 → v6 → v12 → v13 → v14 → v11 → v15 → v16 → v17 → v18

Chi phí : 6720

Câu 2: Kiểm tra tính đúng đắn của các thuật toán đã cho sẵn code như trên. Nếu chưa đúng thì em sửa lại như thế nào cho phù hợp?

Thuật toán BFS

Lỗi: một node bị visited 2 lần, điều này không gây lỗi khi chạy chương trình nhưng sai logic BFS

Sửa lại: node sẽ bị visited ngay sau khi được put vào frontier

```
# Cài đặt thuật toán BFS
def BFS(graph, start, end):
    visited = []
    frontier = Queue()

    # Thêm node start vào frontier và visited
    frontier.put(start)
    visited.append(start)

    # start không có node cha
    parent = dict()
```

```
parent[start] = None


path_found = False


while True:
    if frontier.empty():
        raise Exception("No way Exception")
    current_node = frontier.get()


    # Kiểm tra current_node có là end hay không
    if current_node == end:
        path_found = True
        break


    for node in graph[current_node]:
        if node not in visited:
            frontier.put(node)
            visited.append(node)
            parent[node] = current_node


# Xây dựng đường đi
path = []
if path_found:
    path.append(end)
    while parent[end] is not None:
        path.append(parent[end])
```

```
            end = parent[end]
        path.reverse()

    return path
```

Thuật toán DFS

Lỗi: một node bị visited 2 lần, điều này không gây lỗi khi chạy chương trình nhưng sai logic DFS

Sửa lại: node sẽ bị visited ngay sau khi được append vào frontier

```
# Cài đặt thuật toán DFS
def DFS(graph, start, end):
    visited = []
    frontier = []

    #thêm node start vào frontier và visited
    frontier.append(start)
    visited.append(start)

    #start không có node cha
    parent = dict()
    parent[start] = None

    path_found = False

    while True:
        if frontier == []:
```

```
            raise Exception("No way Exception")
        current_node = frontier.pop()

        # Kiểm tra current_node có là end hay không
        if current_node == end:
            path_found = True
            break

        for node in graph[current_node]:
            if node not in visited:
                frontier.append(node)
                parent[node] = current_node
                visited.append(node)

    # Xây dựng đường đi
    path = []
    if path_found:
        path.append(end)
        while parent[end] is not None:
            path.append(parent[end])
            end = parent[end]
        path.reverse()
    return path
```

Thuật toán UCS

Lỗi: một node chỉ nên bị visited khi đã lấy nó ra khỏi hàng đợi, việc đánh dấu node ngay khi được thêm vào frontier sẽ khiến nó không được xét lại khi có đường ngắn hơn

Sửa lại:

```
# Cài đặt thuật toán UCS
def UCS(graph, start, end):
    visited = []
    frontier = PriorityQueue()
    #thêm node start vào frontier và visited
    frontier.put((0, start))
    visited.append(start)
    #start không có node cha
    parent = dict()
    parent[start] = None
    path_found = False

    while True:
        if frontier.empty():
            raise Exception("No way Exception")
        current_w, current_node = frontier.get()
        visited.append(current_node)

        # Kiểm tra current_node có là end hay không
        if current_node == end:
            path_found = True
```

```
            break

        for nodei in graph[current_node]:
            node, weight = nodei
            if node not in visited:
                frontier.put((current_w + weight, node))
                parent[node] = current_node

    # Xây dựng đường đi
    path = []
    if path_found:
        path.append(end)
        while parent[end] is not None:
            path.append(parent[end])
            end = parent[end]
        path.reverse()
    return current_w, path
```

Câu 3: Từ đó, em có nhận xét gì về kết quả chạy tay với kết quả chạy trên máy tính

Kết quả chạy trên máy tính trùng khớp với kết quả chạy tay