Çizge Kuramı Projesi - Kaynak Kodları

Emre Yıldız

25 Ocak 2025

1 graph.py - Temel Veri Yapıları

```python
from dataclasses import dataclass
from typing import List, Dict, Set, Tuple, Optional
from enum import Enum

class GraphInputType(Enum):
    MATRIX = "Matrix"
    LIST = "List"

@dataclass
class Vertex:
    """Represents a vertex in the graph"""
    label: str
    neighbors: Set[str] = None

    def __post_init__(self):
        if self.neighbors is None:
            self.neighbors = set()

@dataclass
class Graph:
    """Main graph class that handles all operations"""
    vertices: Dict[str, Vertex]
    input_type: GraphInputType

    def __init__(self):
        self.vertices = {}
```

```
        self.input_type = None
```

2 graph_io.py - Dosya İşlemleri

```
def read_graph_from_file(filename: str) -> Graph:
    """Read graph from a file in either Matrix or List
       format"""
    graph = Graph()

    with open(filename, 'r') as f:
        # Read input type
        input_type = f.readline().strip()
        graph.input_type = GraphInputType(input_type)

        if graph.input_type == GraphInputType.MATRIX:
            # Read vertex labels
            labels = f.readline().strip().split()
            if labels[0] == 'M':
                labels = labels[1:]

            # Create vertices
            for label in labels:
                graph.vertices[label] = Vertex(label)

            # Read adjacency matrix
            for i, line in enumerate(f):
                row = line.strip().split()
                current_vertex = labels[i]
                for j, value in enumerate(row[1:]):
                    if value == '1':
                        graph.vertices[current_vertex].neighbors.add(label
```

3 graph_generators.py - Özel Çizge Oluşturucular

```python
def create_complete_graph(n: int) -> Graph:
    """Create a complete graph Kn"""
    graph = Graph()

    # Create vertices
    vertices = [chr(ord('a') + i) for i in range(n)]
    for v in vertices:
        graph.vertices[v] = Vertex(v)

    # Add edges between all pairs of vertices
    for v1, v2 in combinations(vertices, 2):
        graph.vertices[v1].neighbors.add(v2)
        graph.vertices[v2].neighbors.add(v1)

    return graph

def create_cycle_graph(n: int) -> Graph:
    """Create a cycle graph Cn"""
    graph = Graph()

    # Create vertices
    vertices = [chr(ord('a') + i) for i in range(n)]
    for v in vertices:
        graph.vertices[v] = Vertex(v)

    # Add edges to form a cycle
    for i in range(n):
        v1 = vertices[i]
        v2 = vertices[(i + 1) % n]
        graph.vertices[v1].neighbors.add(v2)
        graph.vertices[v2].neighbors.add(v1)

    return graph
```

4 graph_analysis.py - Çizge Analiz Fonksiyonları

```python
def find_connected_components(graph: Graph) -> List[Set[str]]:
    """Find all connected components in the graph using BFS"""
    components = []
    unvisited = set(graph.vertices.keys())

    while unvisited:
        # Start a new component
        start = next(iter(unvisited))
        component = set()
        queue = deque([start])

        # BFS
        while queue:
            vertex = queue.popleft()
            if vertex in unvisited:
                component.add(vertex)
                unvisited.remove(vertex)
                queue.extend(n for n in graph.vertices[vertex].neighbors
                             if n in unvisited)

        components.append(component)

    return components

def is_bipartite(graph: Graph) -> Tuple[bool, Optional[Tuple[Set[str], Set[str]]]]:
    """Check if graph is bipartite and return the two vertex sets if true"""
    if not graph.vertices:
        return True, (set(), set())

    colors = {}  # vertex -> color (0 or 1)

    def try_color_component(start: str) -> bool:
        queue = deque([(start, 0)])

        while queue:
```

```
            vertex, color = queue.popleft()

            if vertex in colors:
                if colors[vertex] != color:
                    return False
                continue

            colors[vertex] = color
            next_color = 1 - color

            for neighbor in
               graph.vertices[vertex].neighbors:
                queue.append((neighbor, next_color))

        return True

    # Try to color each component
    for vertex in graph.vertices:
        if vertex not in colors:
            if not try_color_component(vertex):
                return False, None

    # If we get here, the graph is bipartite
    set0 = {v for v, c in colors.items() if c == 0}
    set1 = {v for v, c in colors.items() if c == 1}
    return True, (set0, set1)
```