Construction of Dual Graph of a Planar Graph

IHM2014003

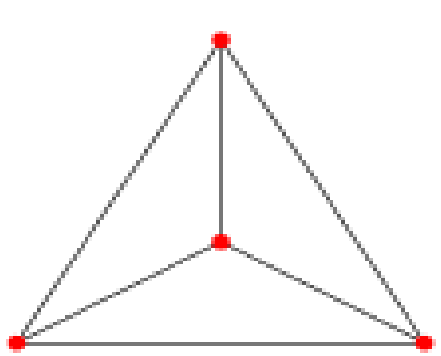
ISM2014001

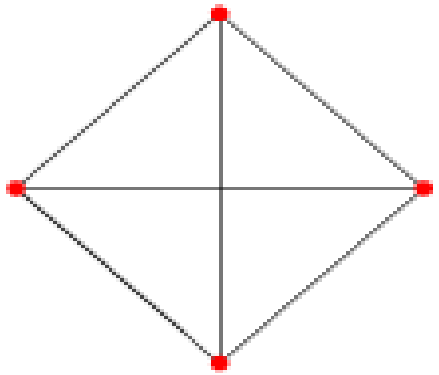
Introduction

1. **Planar Graph:** A planar graph is a graph that can be embedded in the plane i.e., it can be drawn on the plane in such a way that its edges intersect only at their endpoints. In other words, it can be drawn in such a way that no edges cross each other.
2. **Plane Graph:** A planar embedding of a graph.
3. The **dual graph** of a plane graph *G* is a graph that has a vertex for each face of *G*. The dual graph has an edge whenever two faces of *G* are separated from each other by an edge.

Assumptions

1. Input: A plane graph, i.e., a planar embedding of a planar graph.
2. Graph is bi-connected.
3. The edges should be straight lines.
4. Graph should be undirected.





Algorithm to obtain faces

1. Let S = set of edges having 0 face count, initialize all edges
2. Let F = set of all faces
3. while(S is not empty)
    1. Remove edge from S. let it be e(u, v)
    2. Let f = set of edges of the current face
    3. For all edges emerging from v except u
        1. *While vertex != v*
            1. Select the edge having maximum angle and is CCW with e.
            2. If there are no CCW edges, take the maximum CW edge.
            3. Let the edge selected be {x, y}.
            4. Vertex = y
        2. *Add f to F*

Construction of Dual Graph

Algorithm 2: Construction of the dual graph of G

Input: Set of faces F, where each face is represented as a set of edges

Output: The dual graph G'

```
Let n = number of faces in F
for i = 1 to n
    for j = i + 1 to n
        if set_intersection(F[i], F[j]) > 0
            add edge(i, j) to G'
return G'
```

Construction of Dual Graph

Algorithm 3: Construction of the dual graph of G

Input: Set of faces F, where each face is represented as a set of edges

Output: The dual graph G'

Let n = number of faces in F

Let M = map of faces involved with each edge.

for i = 1 to n

add each i to each edge of F[i] in M

For each edge
Let f1, f2 be the two faces.
Add and edge from f1 to f2 in G'.

return G'