1 Minimal Spanning Tree Algorithm

1.1 Introduction

This project mainly contains

- Prim Algorithm
- Kruskal Algorithm

1.2 Prim Algorithm

1.2.1 Pseudo code

Algorithm 1 Prim algorithm

function PRIM(V, E) ▷ V denotes vertices, E denotes edges

Require: A weighted, connected map which vertices set as V and edges set as E.

Ensure: Using sets V_{new} and E_{new} which describe the minimal spanning tree.

$V_{new} \leftarrow \{x\}$ ▷ $x \in V$, x as the start vertex

$E_{new} \leftarrow \{\}$ ▷ set E_{new} as empty set

while $V_{new} \neq V$ **do**

Find the minimal edge $\langle u, v\rangle$ from E , s.t. $u \in V_{new}, v \notin V_{new}, v \in V$ ▷ If there were multi answers, choose one randomly

Push v in V_{new} and push $\langle u, v\rangle$ in E_{new}

end while

end function

1.2.2 Flowchart

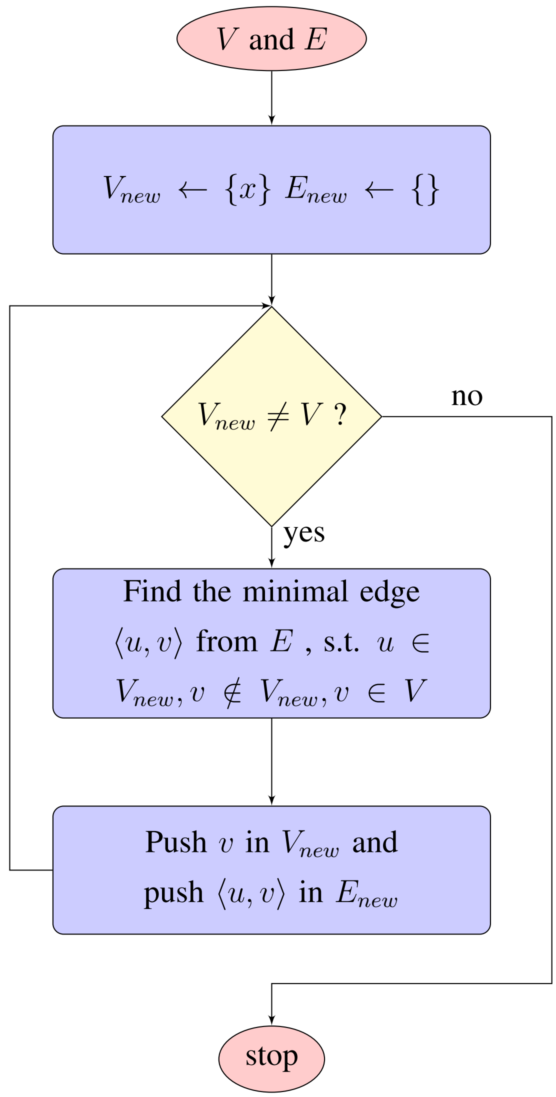

图 1: Prim algorithm flowchart

1.2.3 Analysis

Let v denotes the sum of vertices and e denotes the sum of edges, then, this algorithm's time complexity is:

- Adjacent matrix: $O\left(v^2\right)$
- Adjacent table: $O\left(e \log_2 v\right)$

1.3 Kruskal Algorithm

1.3.1 Pseudo Code

Algorithm 2 Kruskal algorithm

function KRUSKAL(V, E) ▷ V denotes vertices, E denotes edges

Require: A weighted, connected map which vertices set as V and edges set as E.

Ensure: Using map G_{new} which describe the minimal spanning tree.

  $G_{new} \leftarrow \{v_0, e_0 \mid v_0 = V, e_0 \in \emptyset\}$ ▷ v_0 has the same vertices number as V, e_0 denotes empty set

  $E_s \leftarrow sortFromSmallToLarge\,(E)$

  $V_{connected} \leftarrow \{v_0, v_1 \mid \langle v_0, v_1 \rangle \in E_s\,[0]\}$

  for all $e_i \in E_s$ **do** ▷ From small to large

    if $\forall v_t \in V, v_t \in G_{new}$ **then**

      break

    end if

    if $v_0 \in V_{connect} \quad \text{and} \quad v_1 \notin V_{connect} \quad \text{s.t.} \quad \langle v_0, v_1 \rangle \in e_i$ **then**

      add e_i to G_{new}

      add v_1 to $V_{connected}$

    end if

  end for

end function

1.3.2 Flowchart

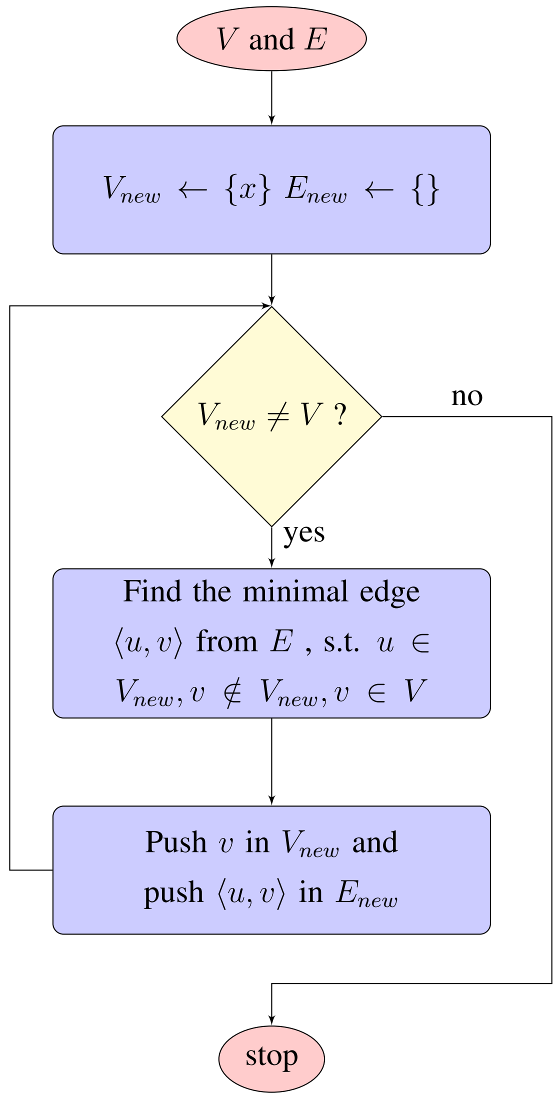

图 2: Kruskal algorithm flowchart

1.3.3 Analysis

Let v denotes the sum of vertices and e denotes the sum of edges, then, this algorithm's time complexity is: $O(e \log_2 e)$