**LAPORAN PRAKTIKUM**

**ANALISIS ALGORITMA**



**Disusun Oleh:**

Nama : Kevin Akbar Adhiguna

NPM : 140810170055

Kelas : A

**FAKULTAS MATEMATIKA ILMU PENGETAHUAN ALAM**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**UNIVERSITAS PADJADJARAN**

**2019**

**MINIMUM SPANNING TREE**

**DASAR TEORI**

Minimum spanning tree merujuk kepada pohon rentangan yang memiliki kompleksitas terendah di antara semua pohon rentangan. Berikut adalah beberapa kegunaan Minimum spanning tree :

* rancang jaringan
* merumuskan solusi pada persoalan travelling salesman
* multi-terminal minimum cut
* minimum-cost weighted perfect matching
* Cluster Analysis (Analisis Kluster)
* Handwriting recognition (Pengenalan tulisan tangan)
* Image segmentation (Segmentasi gambar)

Dalam Minimum Spanning Tree, terdapat dua algoritma, yaitu : Algoritma Kruskal & Algoritma Prim.

1. **Algoritma Kruskal**

Algoritma Kruskal membentuk spanning tree dengan menambahkan satu persatu edge menjadi spanning tree yang terus tumbuh. Langkah-langkahnya adalah sebagai berikut :

* Sorting edge graf berdasarkan weight
* Mulai tambahkan edge ke MST dari edge dengan bobot terkecil hingga edge dengan bobot terbesar
* Hanya tambahkan edge yang tidak membentuk siklus, yaitu edge yang hanya terhubung ke komponen yang terputus

Kompleksitas Waktu :

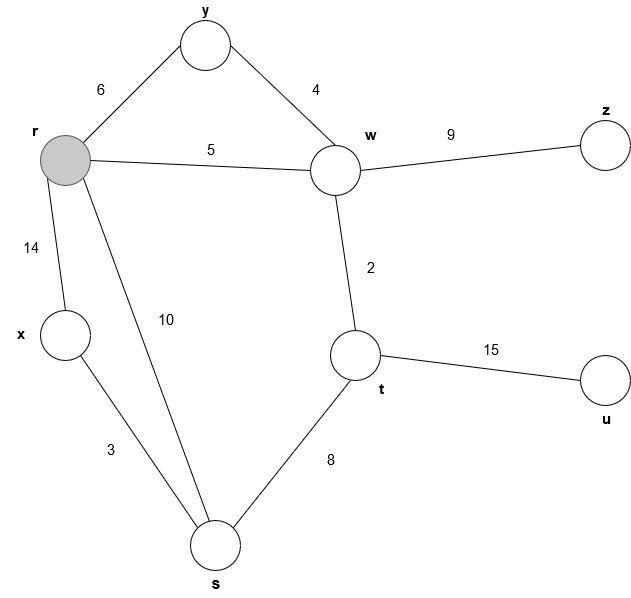
1. **Algoritma Prim**

Algoritma Prim mengembangkan spanning tree dari posisi asal. Bagian yang berbeda dari algoritma Kruskal adalah ditambahkannya **edge** ke growing spanning tree, sedangkan pada algoritme Prim, **vertex** yang ditambahkan ke growing spanning tree. Langkah-langkahnya adalah sebagai berikut :

* Terdapat dua disjoint set vertex. Salah satu disjoint set berisi vertex dalam growing spanning tree, dan yang lainnya berisi vertex yang tidak dalam growing spanning tree.
* Pilih vertex dengan cost teringan yang terhubung dengan growing spanning tree dan yang tidak dalam growing spanning tree lalu tambahkan ke growing spanning tree.
* Pastikan tidak ada siklus yang terjadi.

Kompleksitas Waktu :

**ALGORITMA KRUSKAL**



|  |  |  |
| --- | --- | --- |
| Edge | Weight | Rank |
| r,x | 14 | 9 |
| r,y | 6 | 5 |
| r,w | 5 | 4 |
| r,s | 10 | 8 |
| y,w | 4 | 3 |
| w,z | 9 | 7 |
| w,t | 2 | 1 |
| x,s | 3 | 2 |
| s,t | 8 | 6 |
| t,u | 15 | 10 |

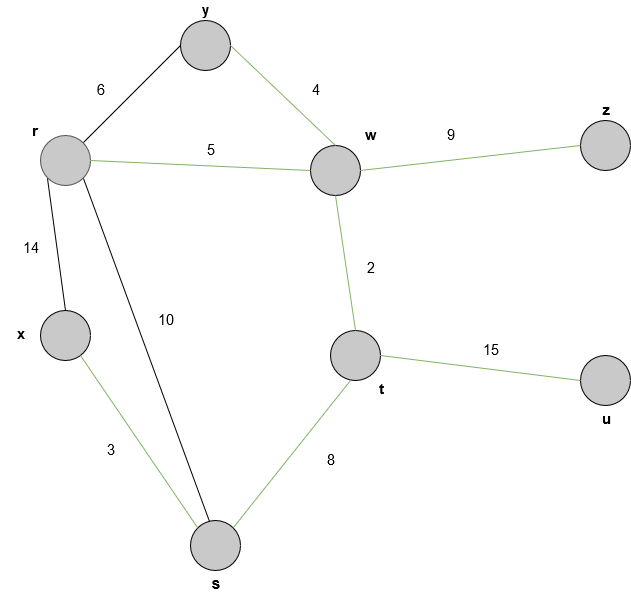
Setelah sorting :

|  |  |
| --- | --- |
| Edge | Weight |
| w,t | 2 |
| x,s | 3 |
| y,w | 4 |
| r,w | 5 |
| r,y | 6 |
| s,t | 8 |
| w,z | 9 |
| r,s | 10 |
| r,x | 14 |
| t,u | 15 |

Lalu menjumlahkan cost sesuai dengan hasil sorting, dengan syarat **tidak membentuk siklus**

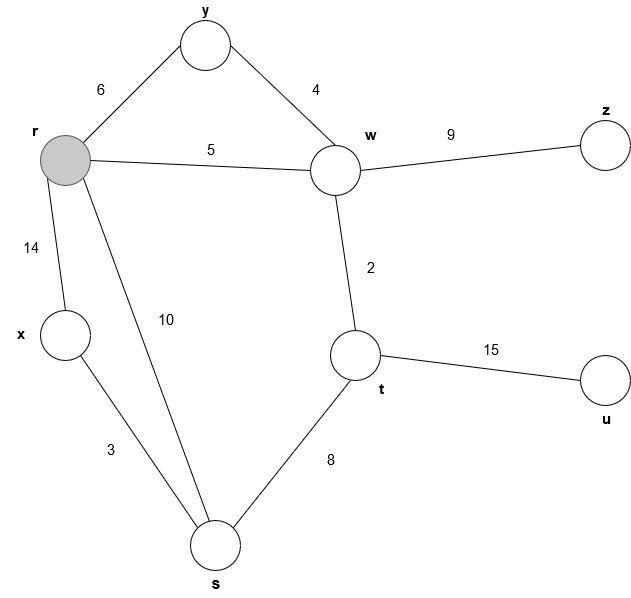
Cost = 2 + 3 + 4 + 5 + 8 + 9 + 15 = 46

Visualisasi Hasil :

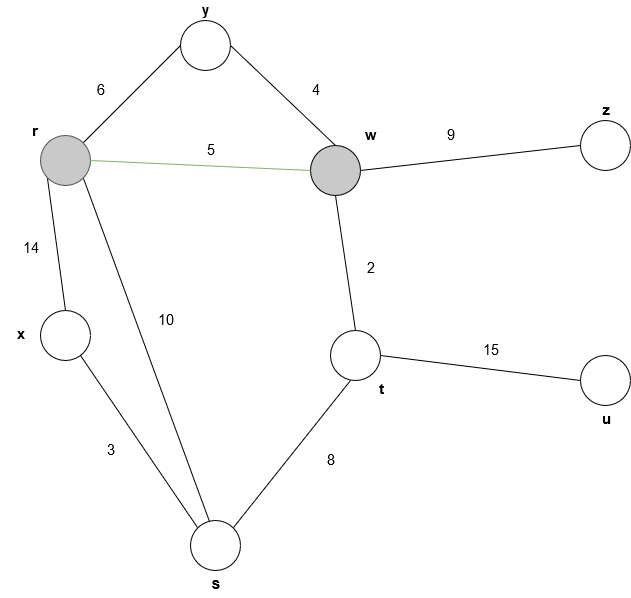


**ALGORITMA PRIM**

Sesuai dengan soal, vertex awal adalah **r** .

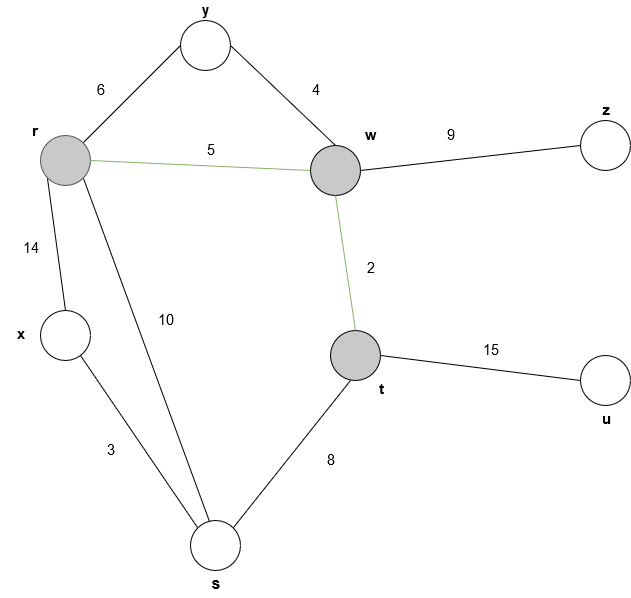


|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |



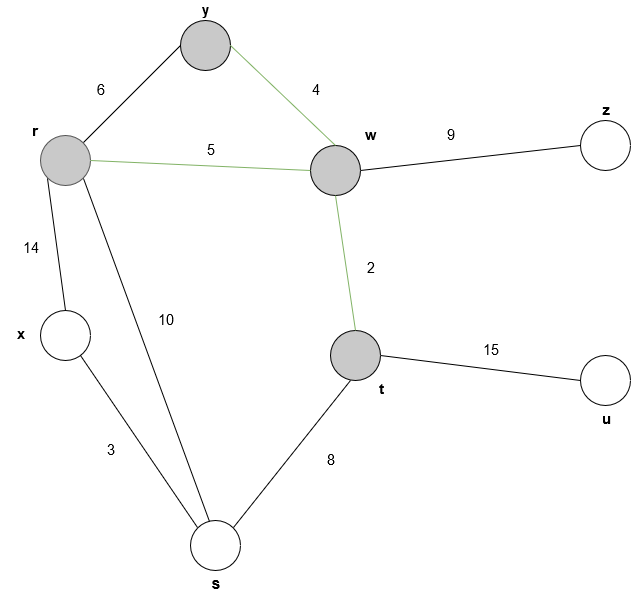
|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |

Karena diantara vertex yang terjangkau, (edge : 14, 10, 6, 5 ) , edge terkecil adalah 5, maka memilih vertex w.



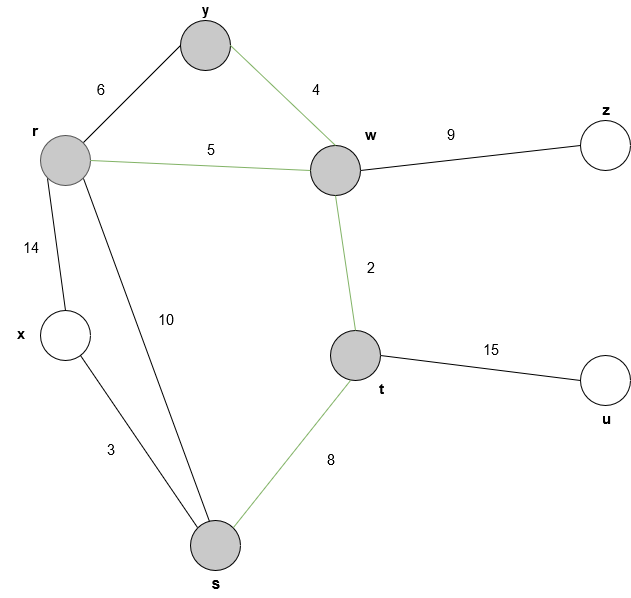
|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |

Karena diantara vertex yang terjangkau, (edge : 14, 10, 6, 4, 9, 2 ), edge terkecil adalah 2, maka memilih vertex t.



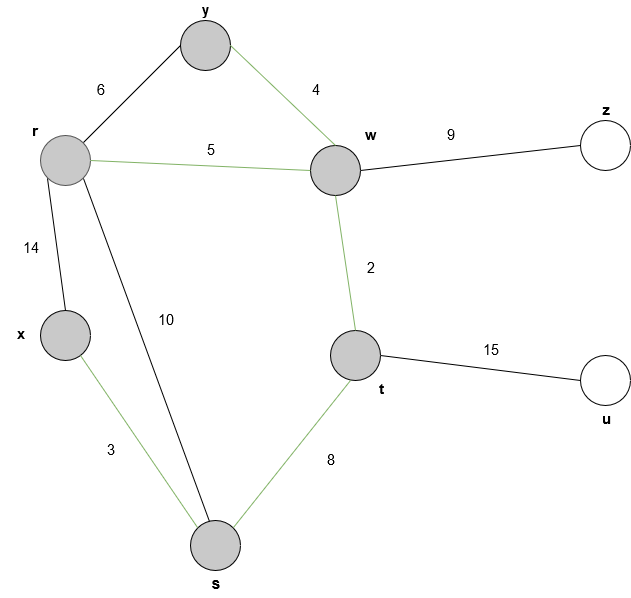
|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |
| **y** | 4 | w |

Karena diantara vertex yang terjangkau, (edge : 14, 10, 6, 4, 9, 8, 15 ), edge terkecil adalah 4, maka memilih vertex y.



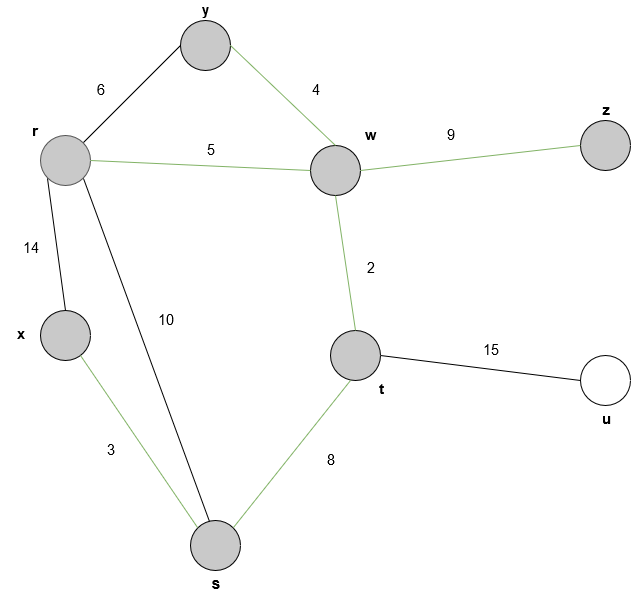
|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |
| **y** | 4 | w |
| **s** | 8 | t |

Karena diantara vertex yang terjangkau, (edge : 14, 8, 9, 15 ), edge terkecil adalah 8, maka memilih vertex s.



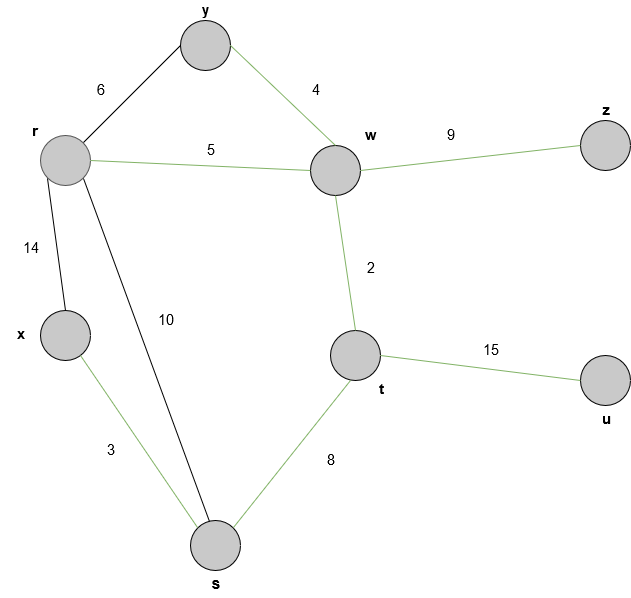
|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |
| **y** | 4 | w |
| **s** | 8 | t |
| **x** | 3 | s |

Karena diantara vertex yang terjangkau, (edge : 9, 3, 15 ), edge terkecil adalah 3, maka memilih vertex x.



|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |
| **y** | 4 | w |
| **s** | 8 | t |
| **x** | 3 | s |
| **z** | 9 | w |

Karena diantara vertex yang terjangkau, (edge : 9 , 15 ), edge terkecil adalah 9, maka memilih vertex z.



|  |  |  |
| --- | --- | --- |
| Vertex | Key | Parent |
| **r** | 0 | - |
| **w** | 5 | r |
| **t** | 2 | w |
| **y** | 4 | w |
| **s** | 8 | t |
| **x** | 3 | s |
| **z** | 9 | w |
| **u** | 15 | t |

Karena hanya ada vertex u yang terjangkau, maka memilih vertex u .

**REFERENSI**

<https://indonesia.hackerearth.com/minimum-spanning-tree-pohon-rentangan-minimum/>