LAPORAN PRAKTIKUM
STRUKTUR DATA DAN ALGORITMA

ANALISA GRAF MENGGUNAKAN ALGORITMA PRIM DAN ALGORITMA KRUSKAL

DISUSUN OLEH

Muhammad Alwiza Ansyar M0520051

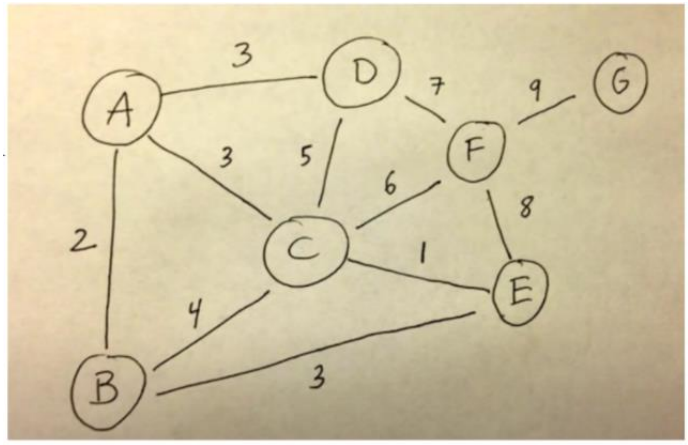
PROGRAM STUDI INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SEBELAS MARET

2021

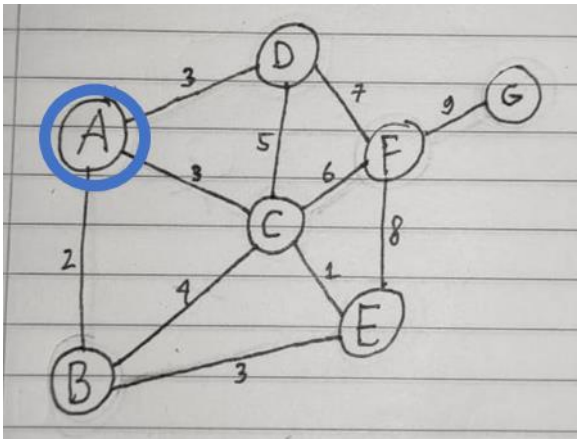
Diberikan sebuah graf sebagai berikut



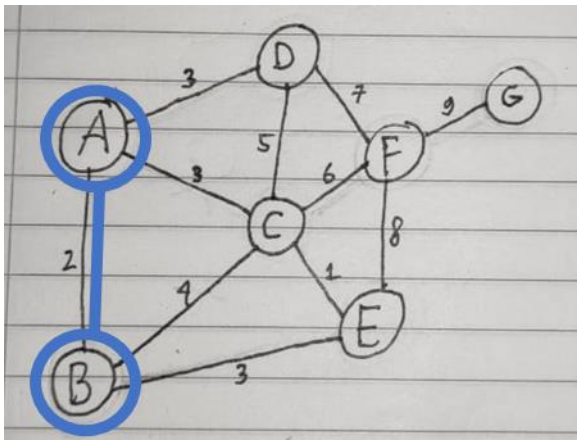
I. ANALISA

A. Algoritma Prim

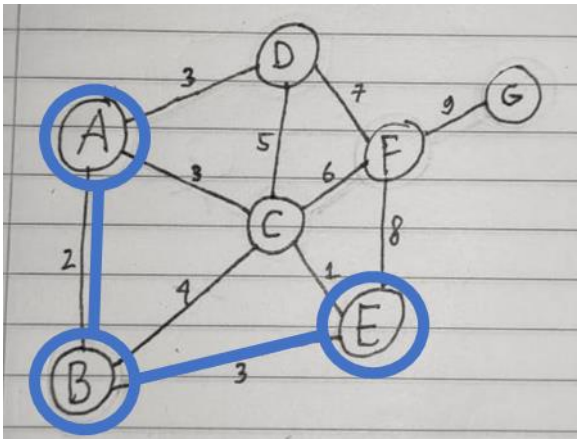
1. Initial vertex: A



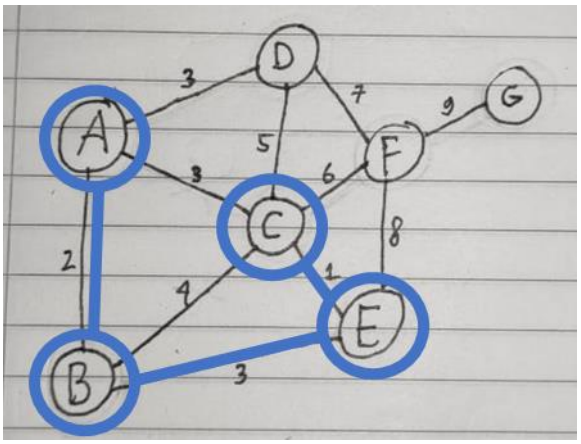
2. Visited vertex: A
   Visited edge: -
   Available adjacent edges: AB(2), AC(3), AD(3)
   Lightest edge: AB → AB is selected
   B added to visited vertex and AB added to visited edge



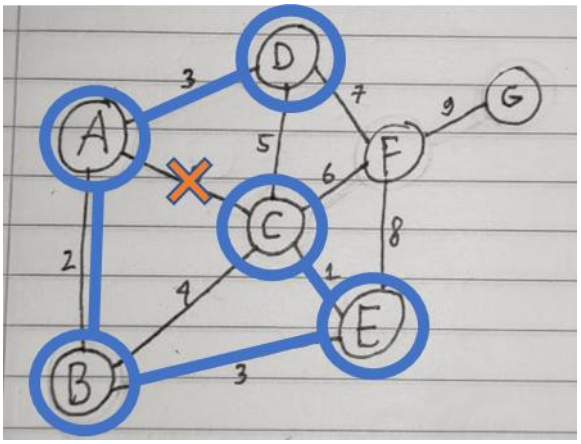
3. Visited vertex: A, B
   Visited edge: AB
   Available adjacent edges: AC(3), AD(3), BC(4), BE(3)
   Lightest edge: AC, AD, and BE → BE is selected (randomly)
   E added to visited vertex and BE added to visited edge



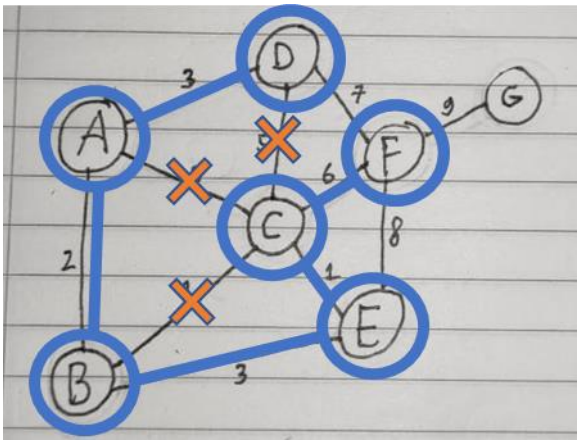
4. Visited vertex: A, B, E
   Visited edge: AB, BE
   Available adjacent edges: AC(3), AD(3), BC(4), EC(1), EF(8)
   Lightest edge: EC → EC is selected
   C added to visited vertex and EC added to visited edge



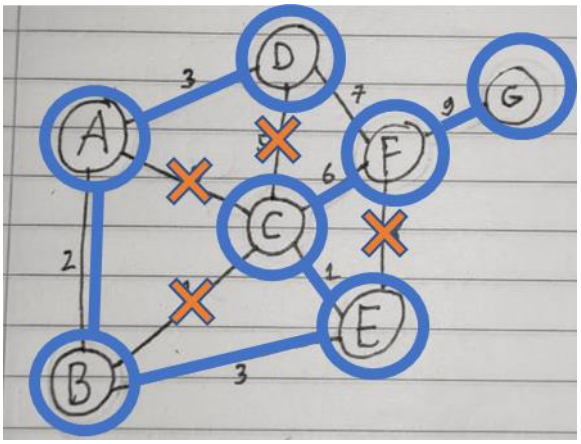
5. Visited vertex: A, B, E, C
   Visited edge: AB, BE, EC
   Available adjacent edges: AC(3), AD(3), BC(4), EF(8), CD(5), CF(6)
   Lightest edge: AC, AD → AC make a cycle → AC is skipped
   Lightest edge: AD → AD is selected
   D added to visited vertex and AD added to visited edge



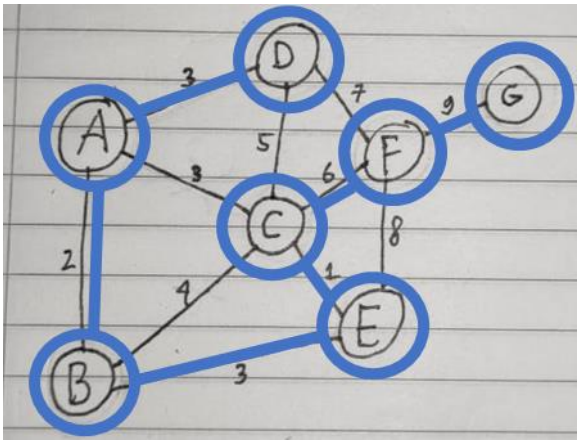
6. Visited vertex: A, B, E, C, D
   Visited edge: AB, BE, EC, AD
   Available adjacent edges: BC(4), EF(8), CD(5), CF(6), DF(7)
   Lightest edge: BC → BC make a cycle → BC is skipped
   Lightest edge: CD → CD make a cycle → CD is skipped
   Lightest edge: CF → CF is selected
   F added to visited vertex and CF added to visited edge



7. Visited vertex: A, B, E, C, D, F
   Visited edge: AB, BE, EC, AD, CF
   Available adjacent edges: EF(8), DF(7), FG(9)
   Lightest edge: DF → DF make a cycle → DF is skipped
   Lightest edge: EF → EF make a cycle → EF is skipped
   Lightest edge: FG → FG is selected
   G added to visited vertex and FG is added to visited edge

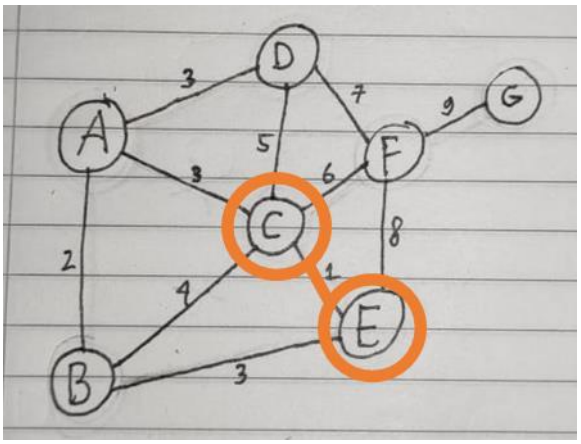


8. Visited vertex: A, B, E, C, D, F, G → all vertex has been visited
   Minimum Spanning Tree: AB, BE, EC, AD, CF, FG
   Total weight: 2 + 3 + 1 + 3 + 6 + 9 = 24

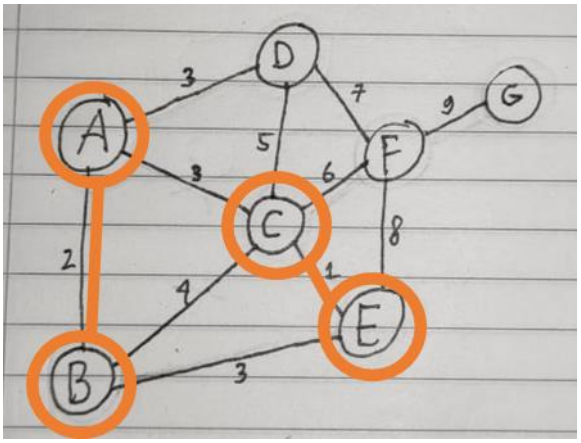


B. Algoritma Kruskal

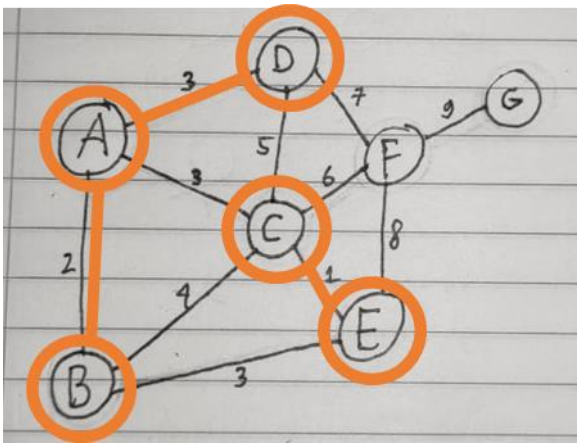
1. Visited vertex: -
   Visited edge: -
   Available edges sorted: CE(1), AB(2), AC(3), AD(3), BE(3), BC(4), CD(5), CF(6), DF(7), EF(8), FG(9)
   Lightest edge: CE → CE is selected
   C and E added to visited vertex and CE added to visited edge



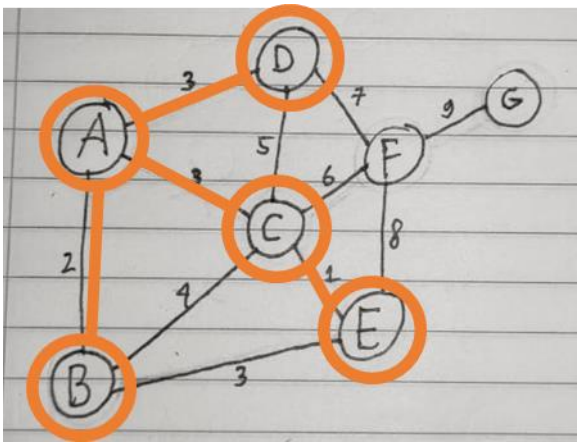
2. Visited vertex: C, E
   Visited edge: CE
   Available edges sorted: AB(2), AC(3), AD(3), BE(3), BC(4), CD(5), CF(6), DF(7), EF(8), FG(9)
   Lightest edge: AB → AB is selected
   A and B added to visited vertex and AB added to visited edge



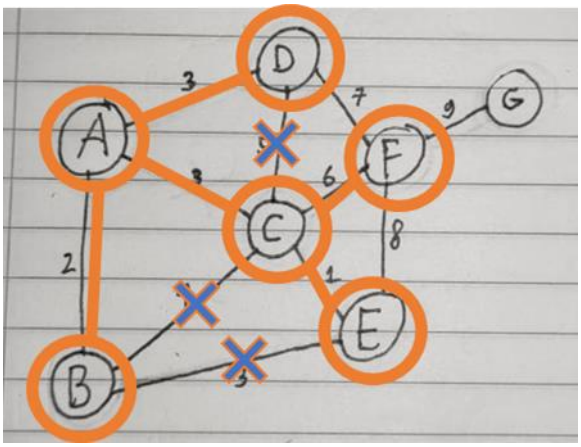
3. Visited vertex: C, E, A, B
   Visited edge: CE, AB
   Available edges sorted: AC(3), AD(3), BE(3), BC(4), CD(5), CF(6), DF(7), EF(8), FG(9)
   Lightest edge: AC, AD, and BE → prioritize edge that connect least visited vertex → AD is selected
   D added to visited vertex and AD added to visited edge

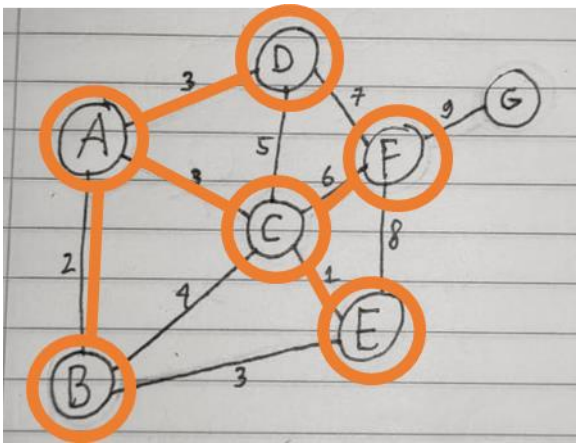


4. Visited vertex: C, E, A, B, D
   Visited edge: CE, AB, AD
   Available edges sorted: AC(3), BE(3), BC(4), CD(5), CF(6), DF(7), EF(8), FG(9)
   Lightest edge: AC and BE → AC is selected (randomly)
   AC added to visited edge

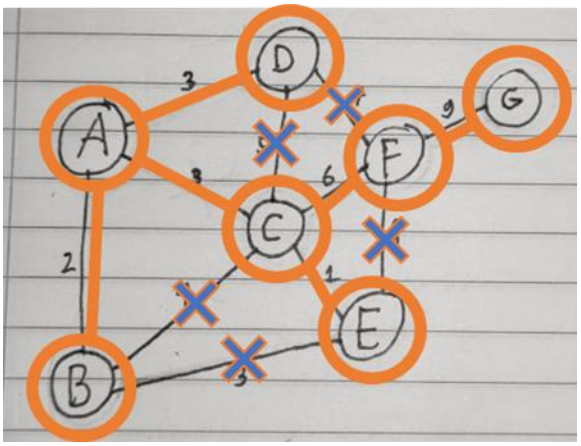


5. Visited vertex: C, E, A, B, D
   Visited edge: CE, AB, AD, AC
   Available edges sorted: BE(3), BC(4), CD(5), CF(6), DF(7), EF(8), FG(9)
   Lightest edge: BE → BE make a cycle → BE is skipped
   Lightest edge: BC → BC make a cycle → BC is skipped
   Lightest edge: CD → CD make a cycle → CD is skipped
   Lightest edge: CF → CF is selected
   F added to visited vertex and CF added to visited edge

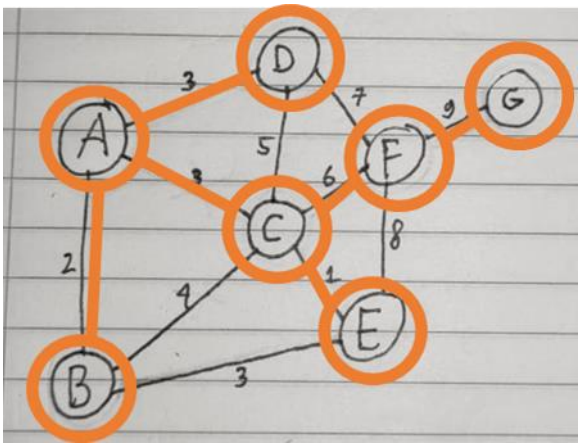




6. Visited vertex: C, E, A, B, D. F
   Visited edge: CE, AB, AD, AC, CF
   Available edges sorted: DF(7), EF(8), FG(9)
   Lightest edge: DF → DF make a cycle → DF is skipped
   Lightest edge: EF → EF make a cycle → EF is skipped
   Lightest edge: FG → FG is selected
   G added to visited vertex and FG added to visited edge

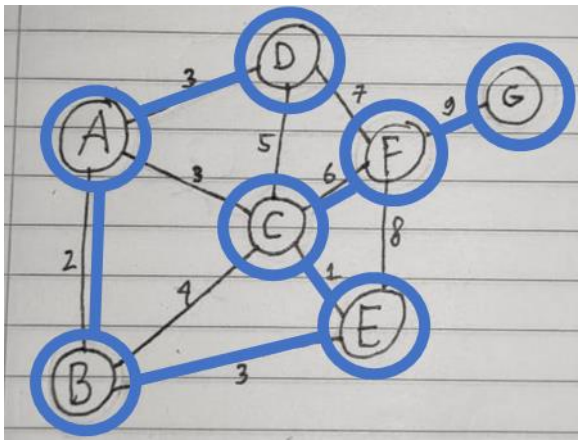


9. Visited vertex: C, E, A, B, D. F, G → all vertex has been visited
   Minimum Spanning Tree: CE, AB, AD, AC, CF, FG
   Total weight: 1 + 2 + 3 + 3 + 6 + 9 = 24

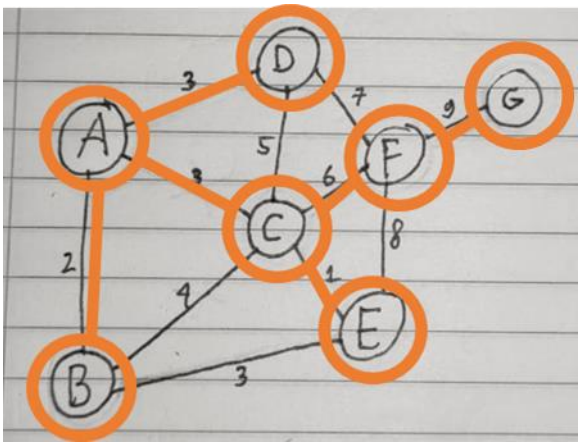


C. Kesimpulan

Hasil dari Algoritma Prim



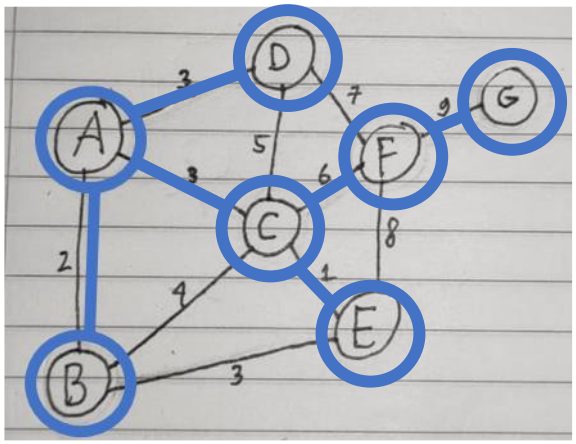
Hasil dari Algoritma Kruskal



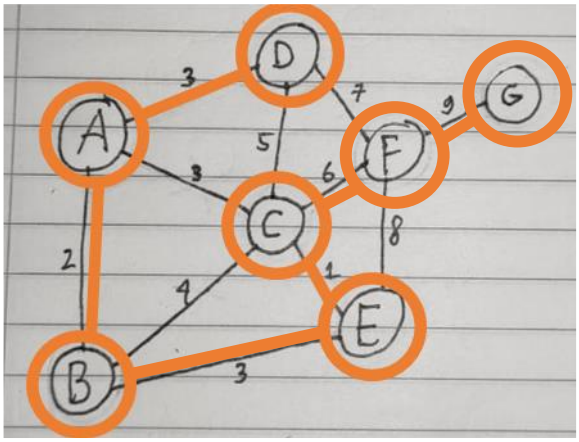
Graf yang telah disediakan memiliki dua MST dengan total weight ialah 24. Hasil dari Prim dan Kruskal berbeda dikarenakan tahap yang memiliki pemilihan acak menghasilkan hasil yang membuat kedua algoritma menghasilkan hasil yang berbeda. Tahap tersebut ialah:

- [Prim tahap 3] Lightest edge: AC, AD, and BE → BE is selected (randomly)
- [Kruskal tahap 4] Lightest edge: AC and BE → AC is selected (randomly)

Apabila pada Prim tahap 3, edge yang dipilih adalah AC atau AD, maka hasil akhir akan menjadi



Apabila pada Kruskal tahap 4, edge yang dipilih adalah BE, maka hasil akhir akan menjadi



2. PERBEDAAN

- Algoritma Prim memerlukan sebuah vertex awal, algoritmanya ialah dengan mendata semua edge yang adjacent dengan visited vertex lalu memilih edge dari pendataaan tersebut dengan weight terendah serta tidak menimbulkan sebuah cycle
- Algoritma Kruskal tidak memerlukan vertex awal, algoritmanya ialah dengan mendata semua edge yang ada pada graf lalu mengurutkannya berdasarkan weight. Kemudian, edge dipilih dari yang terendah dan tidak menimbulkan sebuah cyle