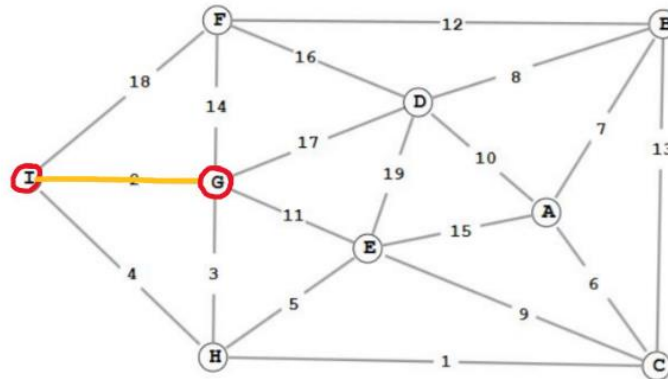


1. Mulai dari titik I dan sort jarak yang dapat ditempuh dan pergi ke node yang memiliki jarak terpendek.



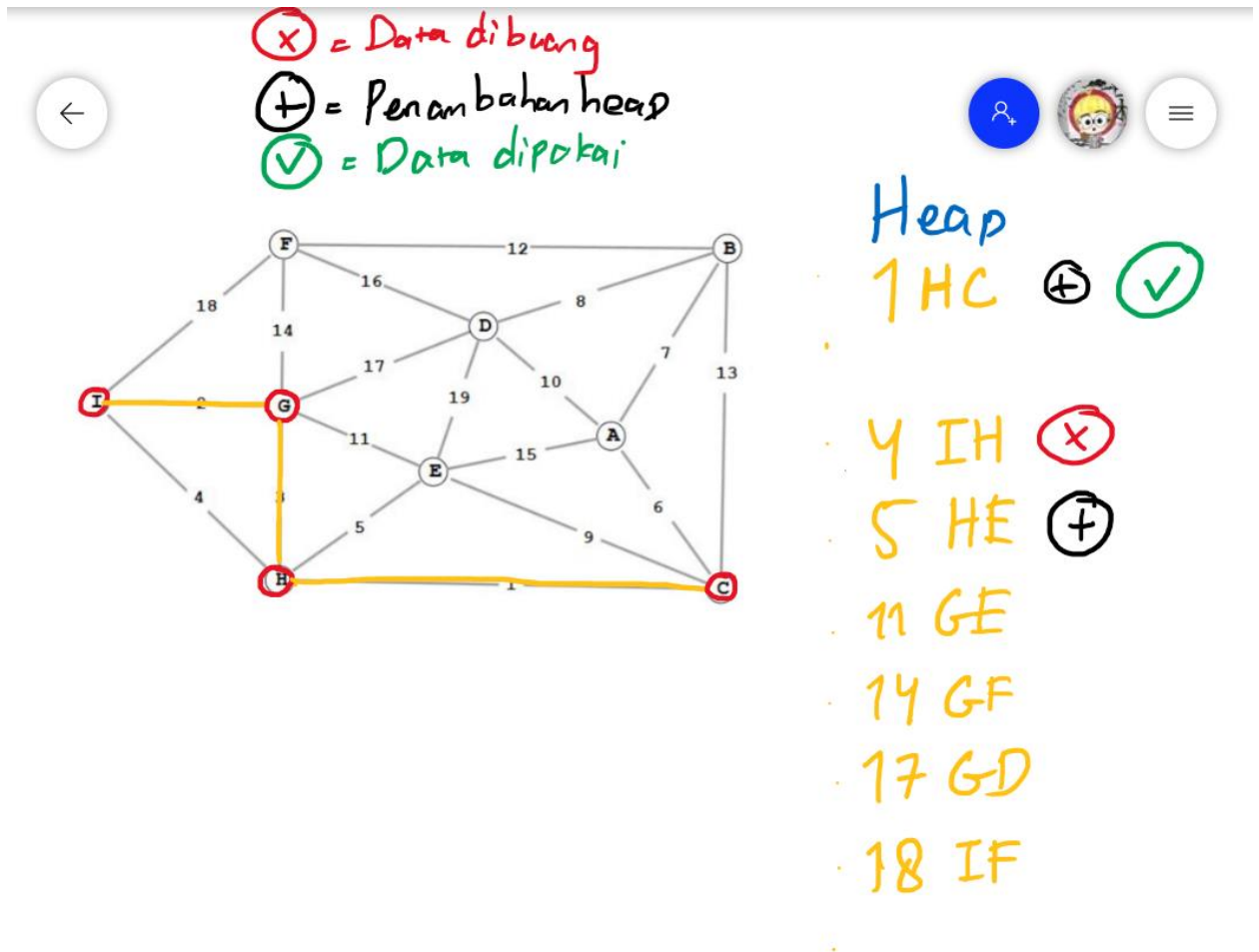
Heap

~~2 IG~~

4 IH

18 IF

2. Hapus yang telah dipilih. Tambahkan lagi node yang bisa dikunjungi dari node yang dituju tadi dan masukkan ke dalam heap. Pilih yang mempunyai nilai terkecil.



3. Lakukan langkah tadi sampai semua node telah terhubung. Jika ada yang membentuk cycle, jalur tersebut harus dibuang.

←

⊗ = Data di buang

⊕ = Penambahan heap

✓ = Data dipakai

🔍

👤

☰

Heap

1 HC ⊕ ✓

4 IH ⊗

5 HE ⊕

11 GE

14 GF

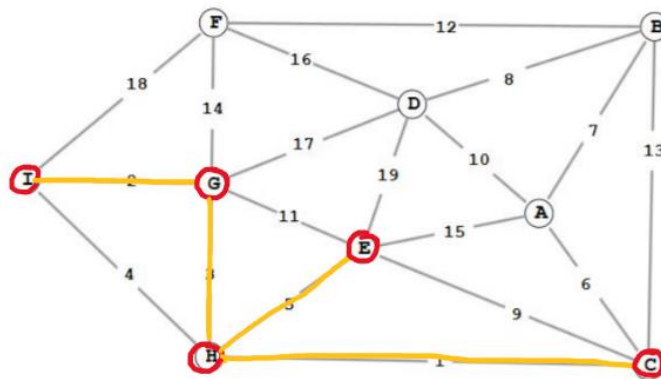
17 GD

18 IF

4. Node C ditambahkan



⊗ = Data dibuang Krm cycle  
⊕ = Penambahan heap  
✓ = Data dipakai



Heap

5 HE ✓  
6 CE ⊕ ⊗  
9 CA ⊕  
11 GE ⊗  
13 CB ⊕  
14 GF  
17 GD  
18 IF

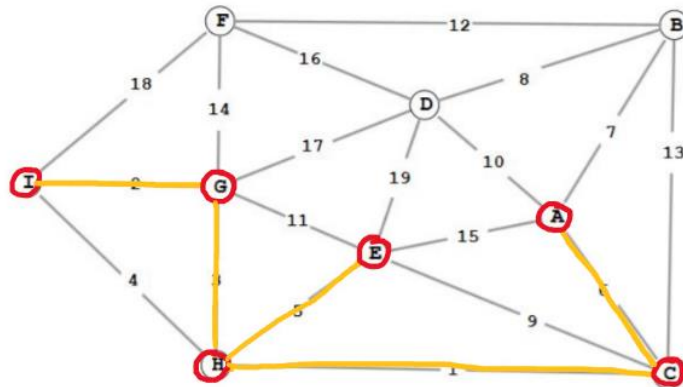
5. Node E ditambahkan



⊗ = Data di buang krm cycle

⊕ = Penambahan heap

✓ = Data dipakai



Heap

9 CA ✓

13 CB

14 GF

15 EA ⊕

17 GD

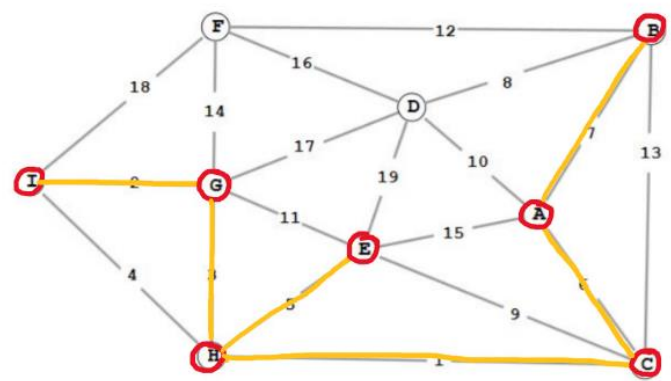
18 IF

19 ED ⊕

6. Node A ditambahkan



⊗ = Data dibuang krm cycle  
⊕ = Penambahan heap  
✓ = Data dipakai



Heap

- 7 AB ⊕ ✓
- 10 AD ⊕
- 13 CB
- 14 GF
- 15 EA ⊗
- 17 GD
- 18 IF
- 19 ED

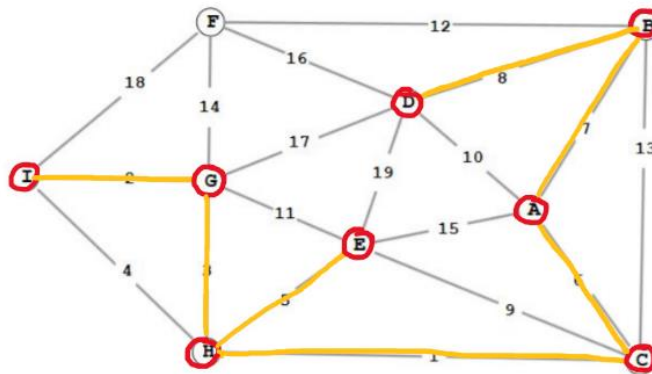
7. Node B ditambahkan



(X) = Data dibuang km cycle

(+) = Penambahan heap

(✓) = Data dipakai



Heap

- 8 BD (+) (✓)
- 10 AD
- 12 BF
- 13 CB (X)
- 14 GF
- 17 GD
- 18 IF
- 19 ED

8. Semua node telah dikunjungi. Algoritma selesai

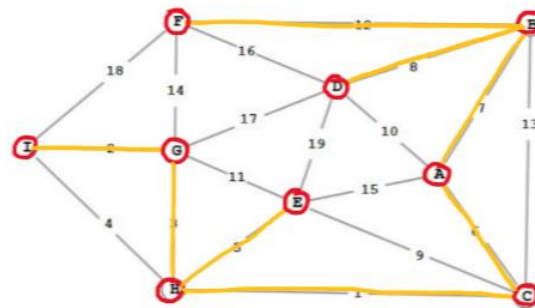


(X) = Data di buang km cycle

(+) = Penambahan heap

(✓) = Data dipakai

Heap



- 10 AD (X)
- 12 BF (✓)
- 14 GF
- 16 DF
- 17 GD (X)
- 18 IF
- 19 ED (X)

Hasil akhir

