



Algoritma Prim & Algoritma Kruskal

MUTIA KARIMAH (140810170002)



Algoritma Prim



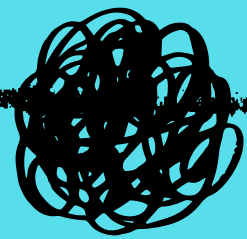
Pengertian Algoritma Prim

Algoritma pohon merentang minimum yang menambahkan edge dari simpul/vertex/node yang sudah dikunjungi.

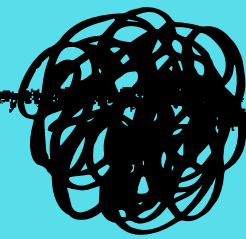
Algoritma



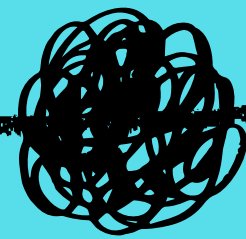
Ambil sisi dari node sembarang graf G yang berbobot minimum, masukkan ke dalam T



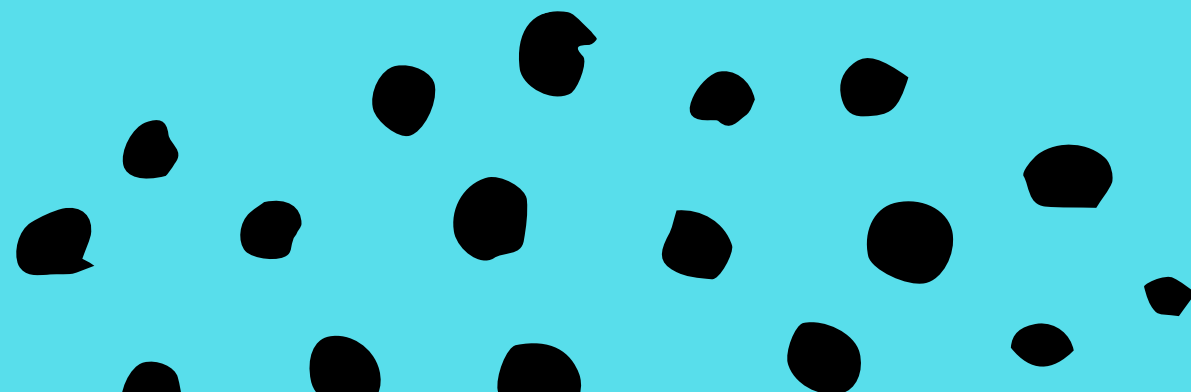
Pilih sisi (u, v) yang mempunyai bobot minimum dan bersisian dengan simpul di T , tetapi (u, v) tidak membentuk sirkuit di T .



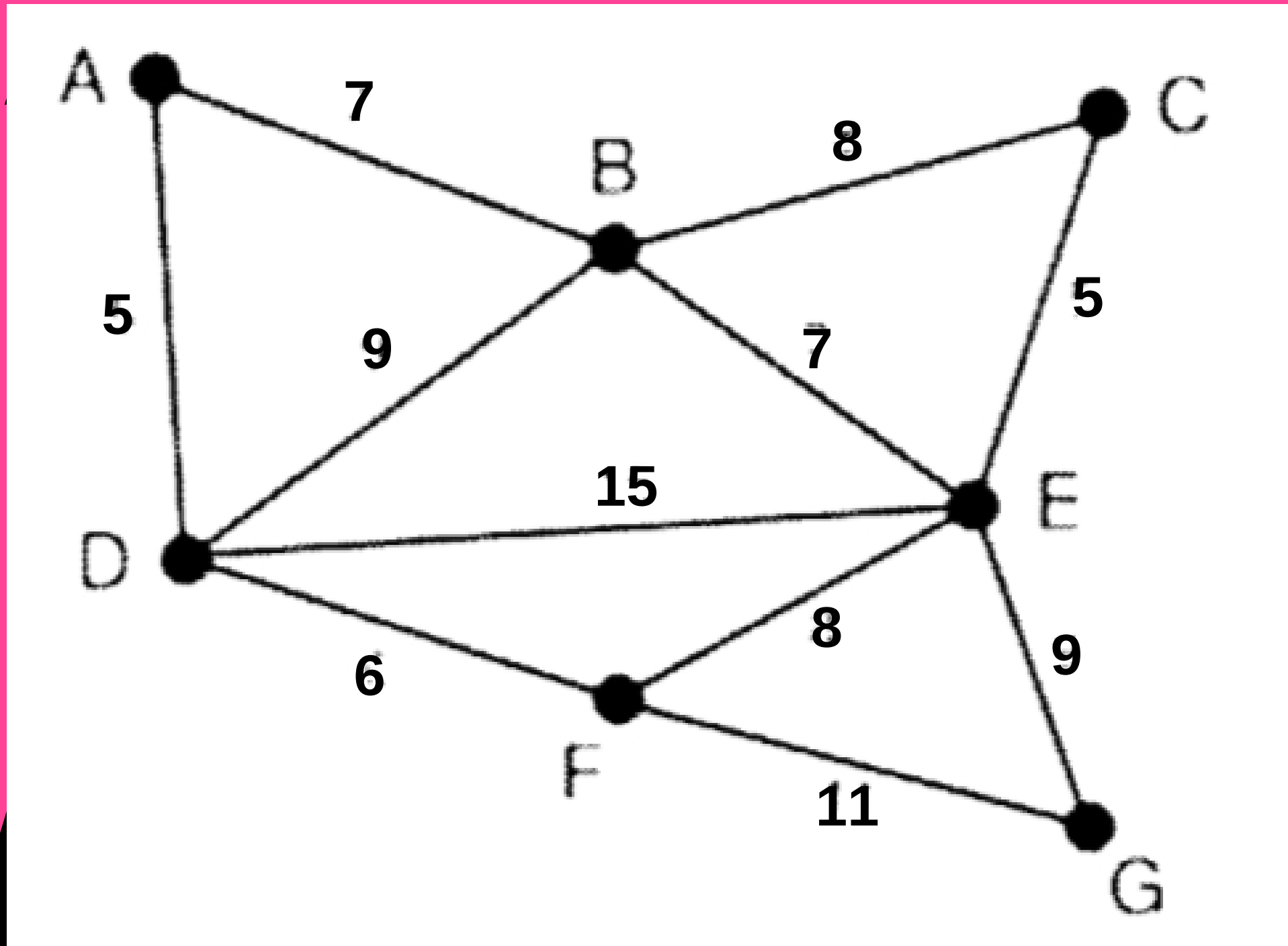
Tambahkan (u, v) ke dalam T



Ulangi langkah-langkah sebelumnya hingga membentuk MST

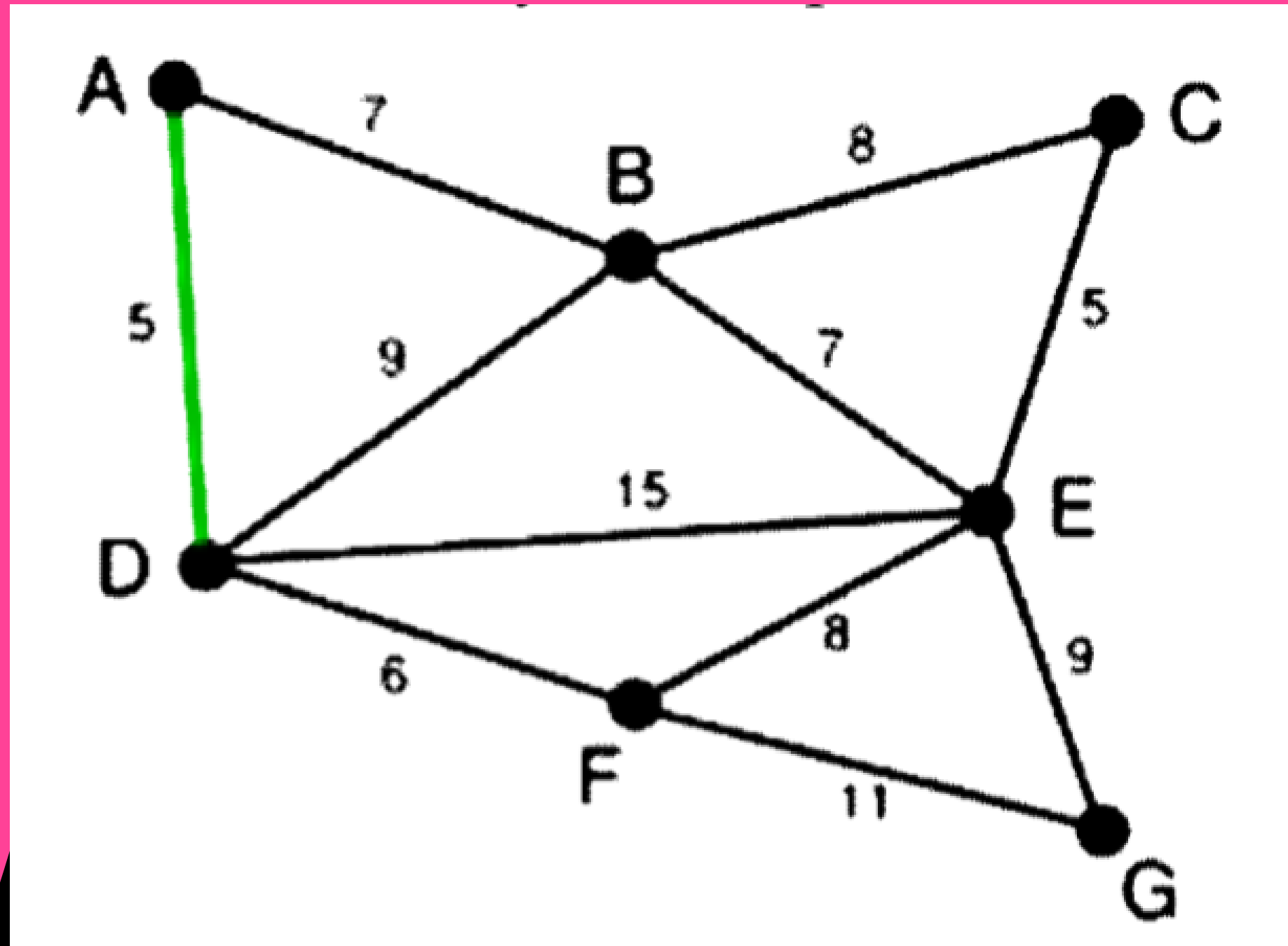


Contoh Soal



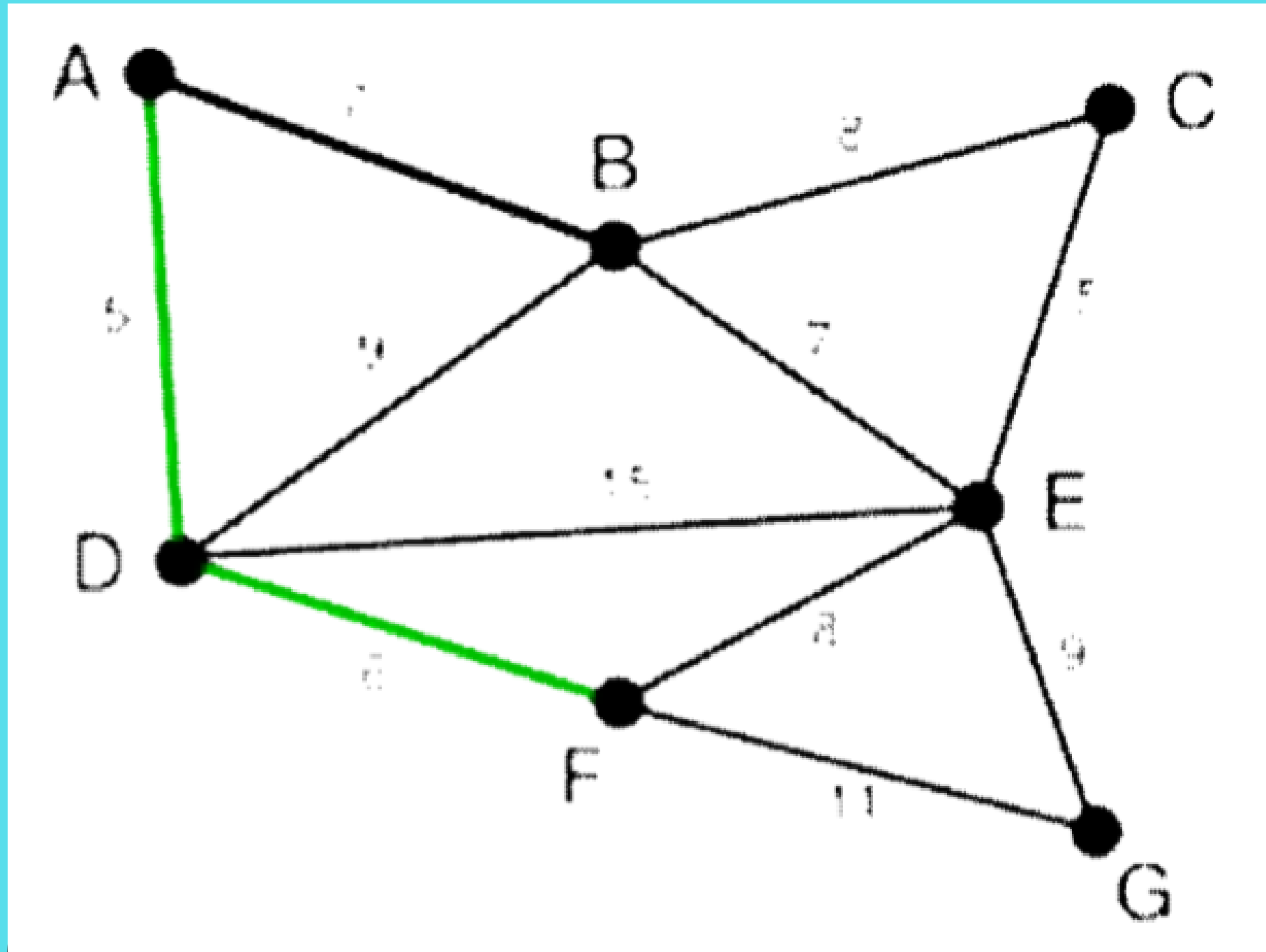
Berikut adalah graf berbobot, bobot sebesar 90, yang akan dicari pohon merentang minimumnya, pilih sembarang simpul, pada contoh ini, simpul D.

Jawaban



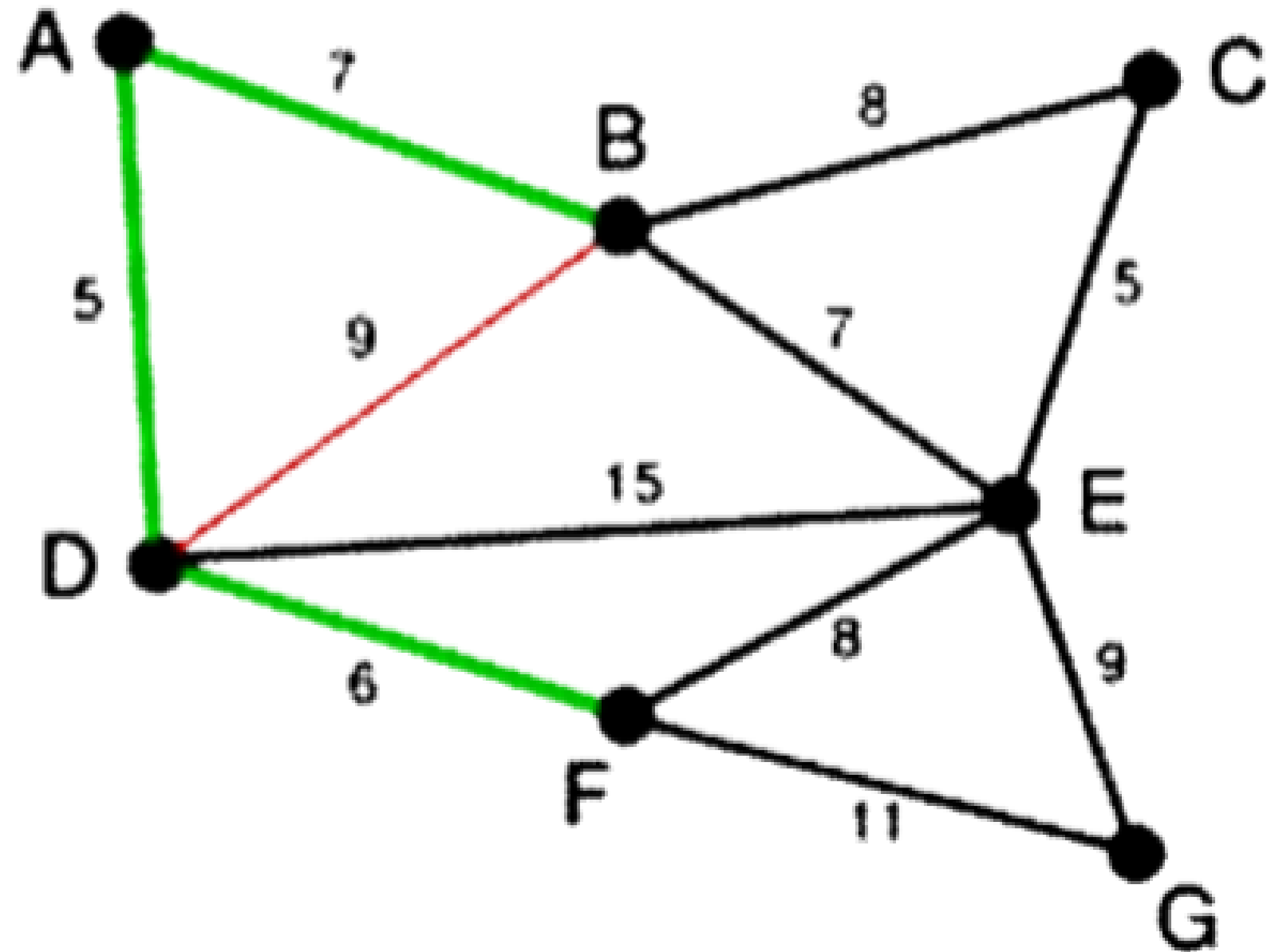
Simpul kedua yang dipilih adalah simpul yang terdekat ke D, yaitu simpul A.

Jawaban



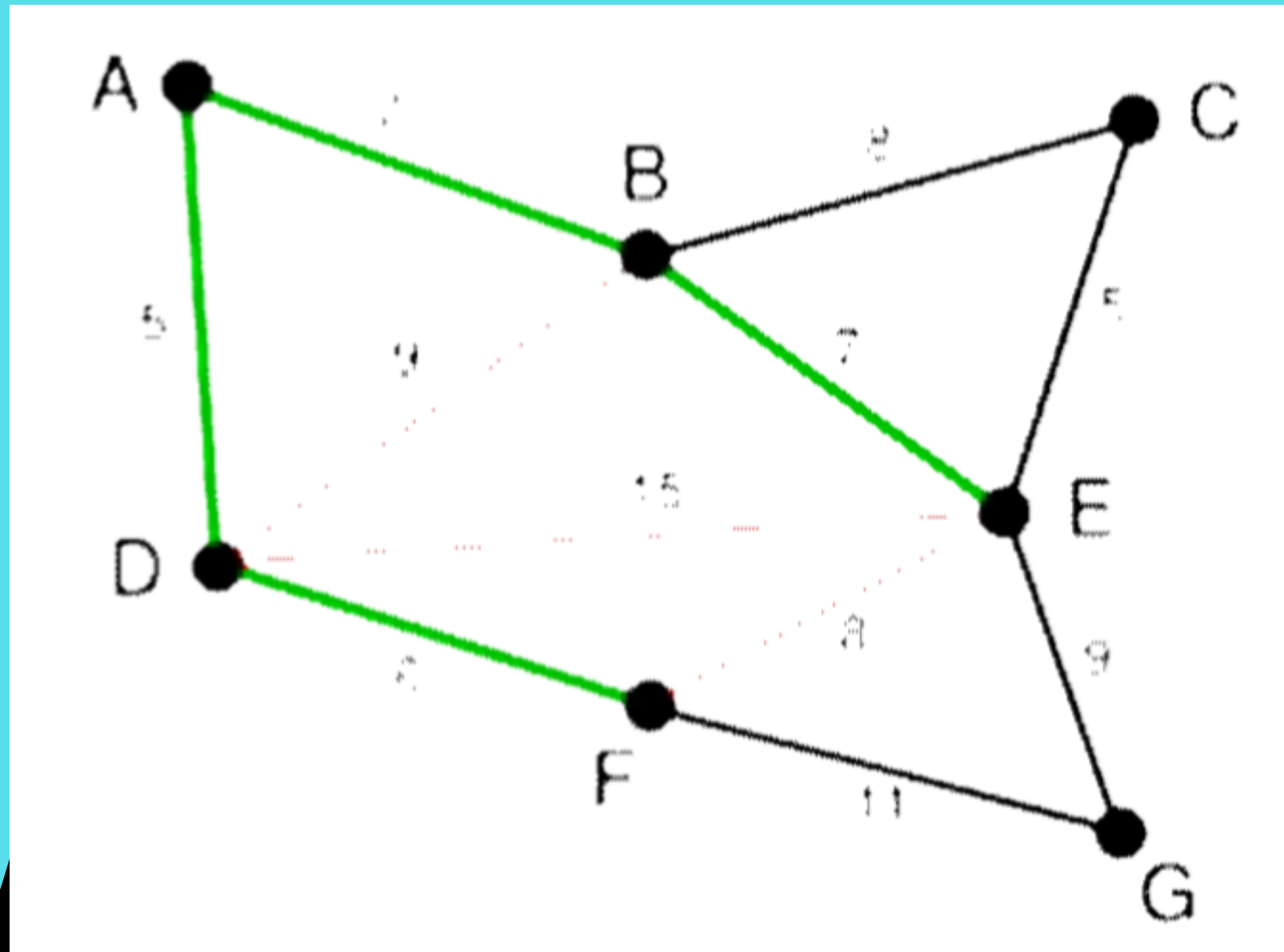
Simpul berikutnya adalah simpul yang terdekat ke simpul D atau A, yaitu simpul F, yang berbobot 6 dari D

Jawaban



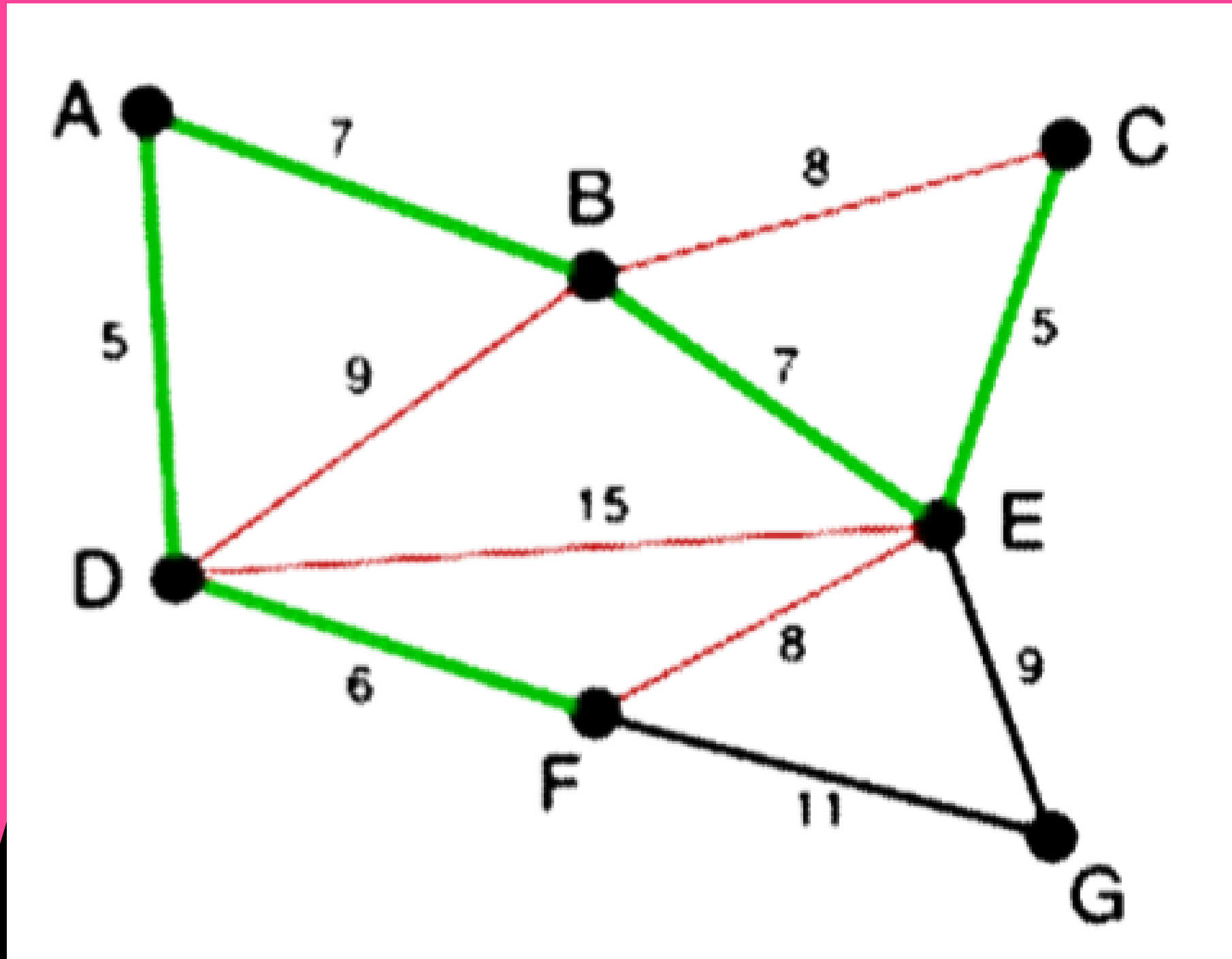
Seperti langkah sebelumnya,
pilih simpul B yang memiliki
bobot 7 dari A

Jawaban



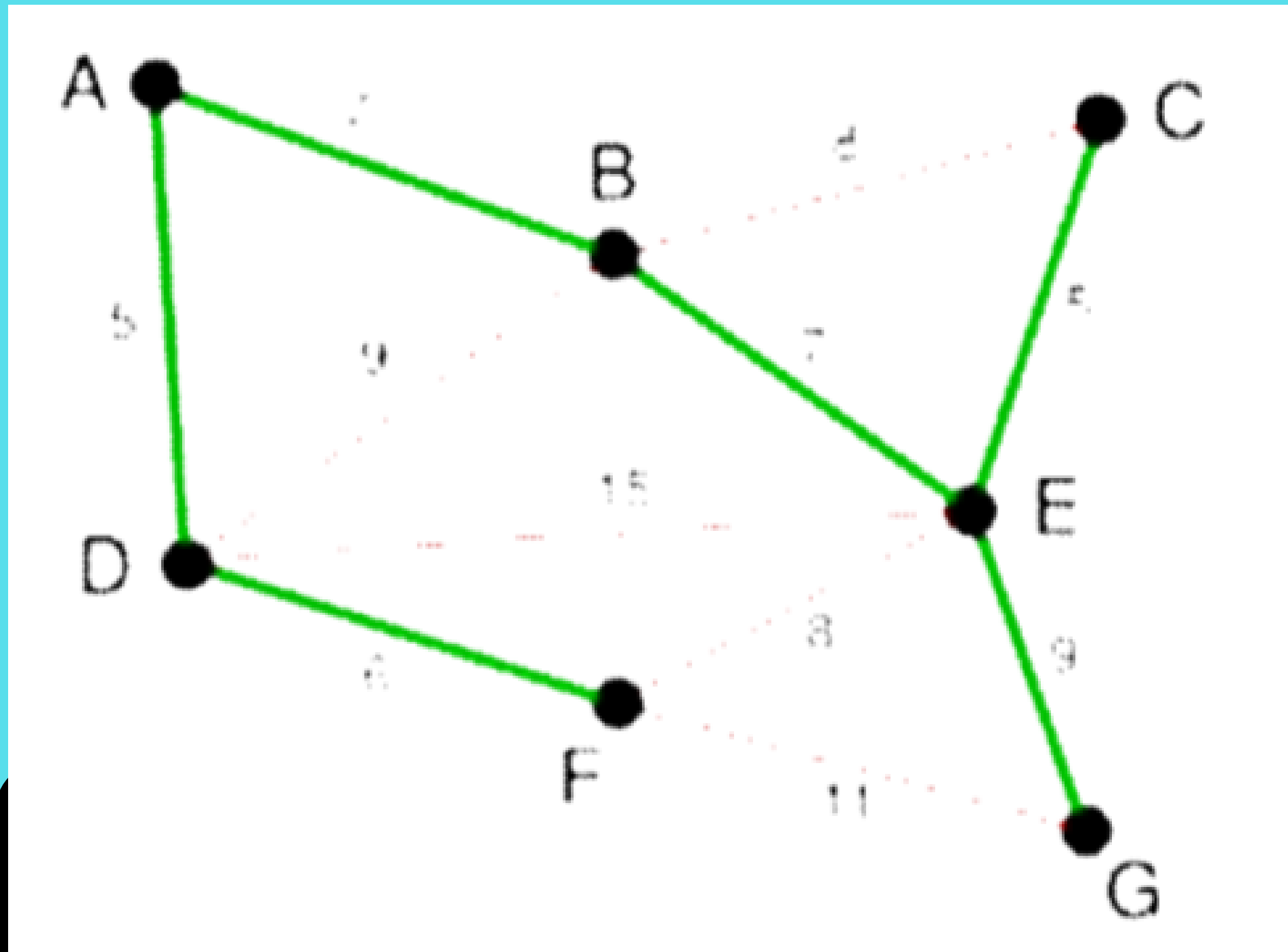
Selanjutnya, pilih simpul E yang terdekat dengan simpul B.

Jawaban



Selanjutnya, pilih simpul C yang terdekat dengan simpul E.

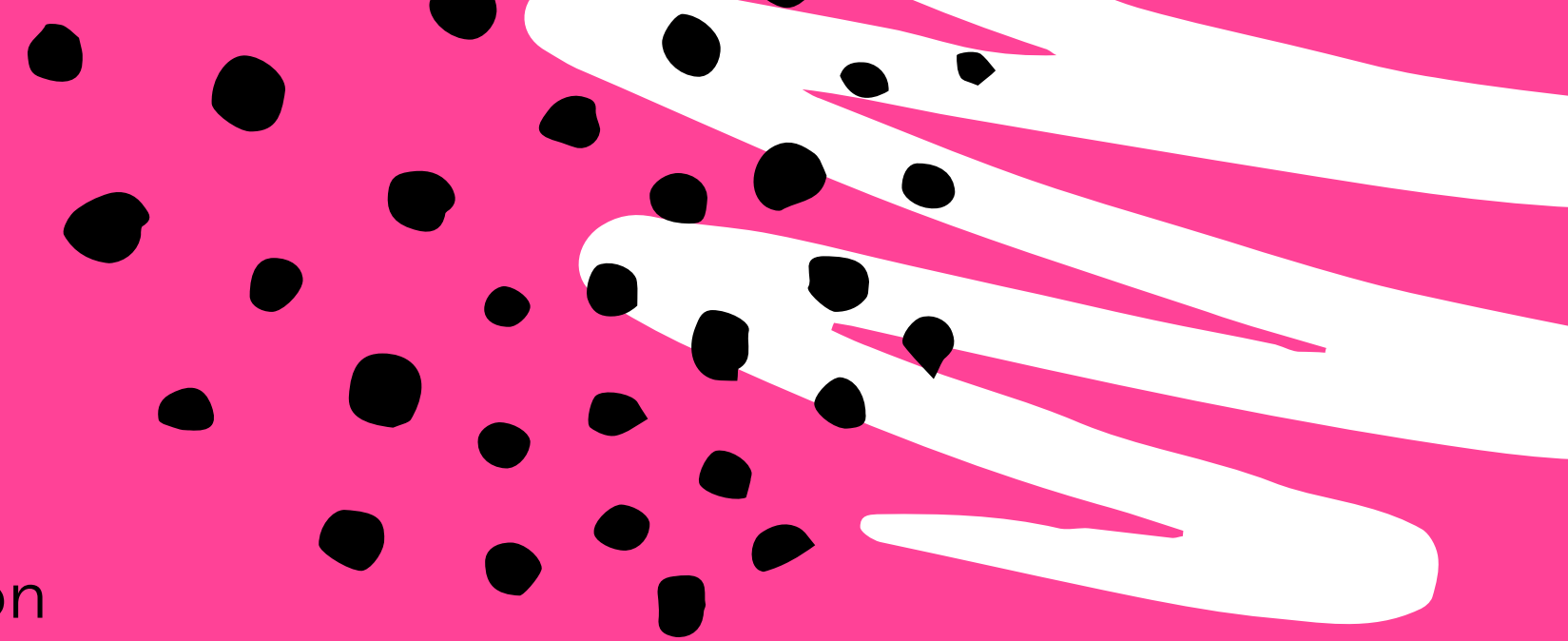
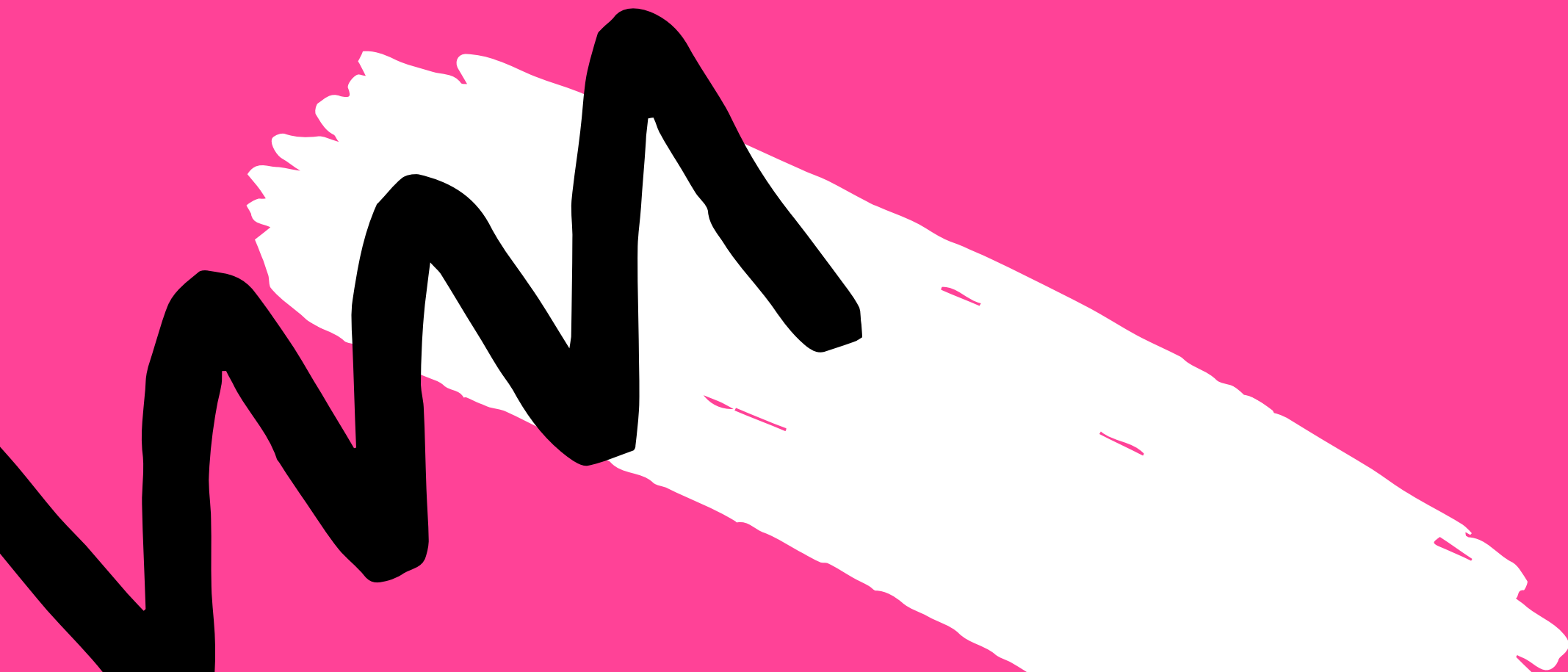
Jawaban



Karena simpul G adalah simpul terakhir yang tersisa, dan memiliki jarak/bobot terdekat dengan E, maka EG dipilih

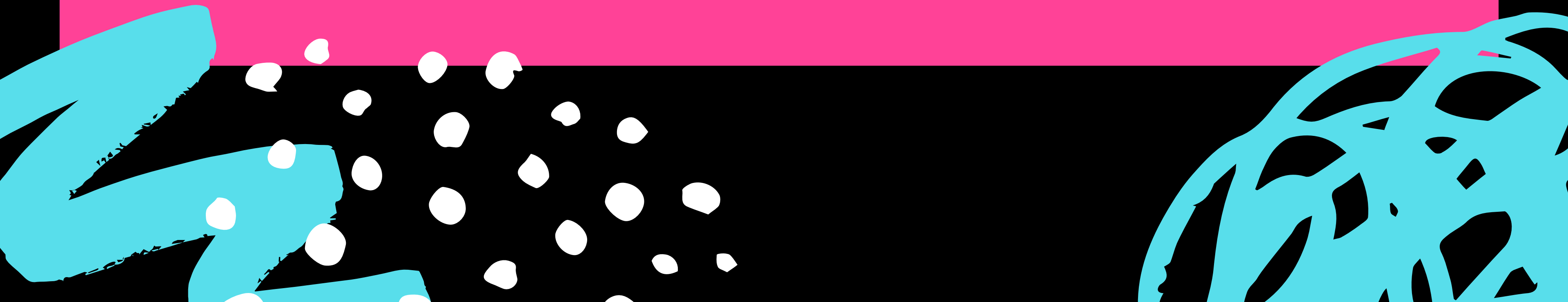
Jawaban

Karena semua simpul telah dipilih, maka pohon merentang minimum ditunjukkan dengan warna hijau. Dengan bobot sebesar 39.



Kompleksitas

Sebuah implementasi sederhana menggunakan graf representasi matrix adjacency (berdekatan) dan mencari tabel bobot untuk menemukan sisi dengan bobot minimum untuk ditambahkan memerlukan kompleksitas waktu $O(V^2)$. Namun, jika menggunakan sebuah struktur data sederhana binary heap dan sebuah representasi adjacency list, maka kompleksitas waktu $O(E \log V)$. Dan jika menggunakan sebuah Fibonacci heap yang rumit, maka kompleksitas waktu $O(E + V \log V)$. Dimana E adalah jumlah sisi dan V jumlah simpul.





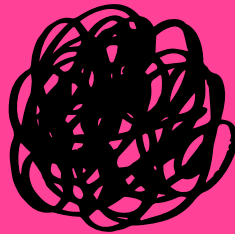
Algoritma Kruskal

Pengertian Algoritma Kruskal

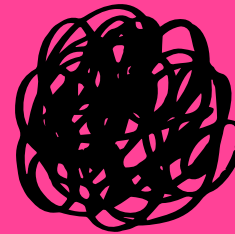
Pada algoritma kruskal, sisi-sisi graf diurut terlebih dahulu berdasarkan bobotnya dari kecil ke besar. Sisi yang dimasukkan ke dalam himpunan T adalah sisi graf G sedemikian sehingga T adalah pohon. Pada keadaan awal, sisi-sisi sudah diurut berdasarkan bobot membentuk hutan, masing-masing pohon di hutan hanya berupa satu buah simpul, hutan tersebut dinamakan hutan merentang (spanning forest). Sisi dari graf G ditambahkan ke T jika ia tidak membentuk siklus di T .

Algoritma

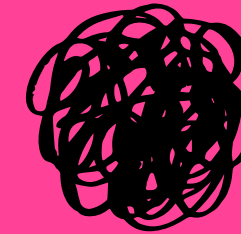
(Asumsi: sisi-sisi dari graf sudah diurut menaik berdasarkan bobotnya dari kecil ke besar)



T masih kosong

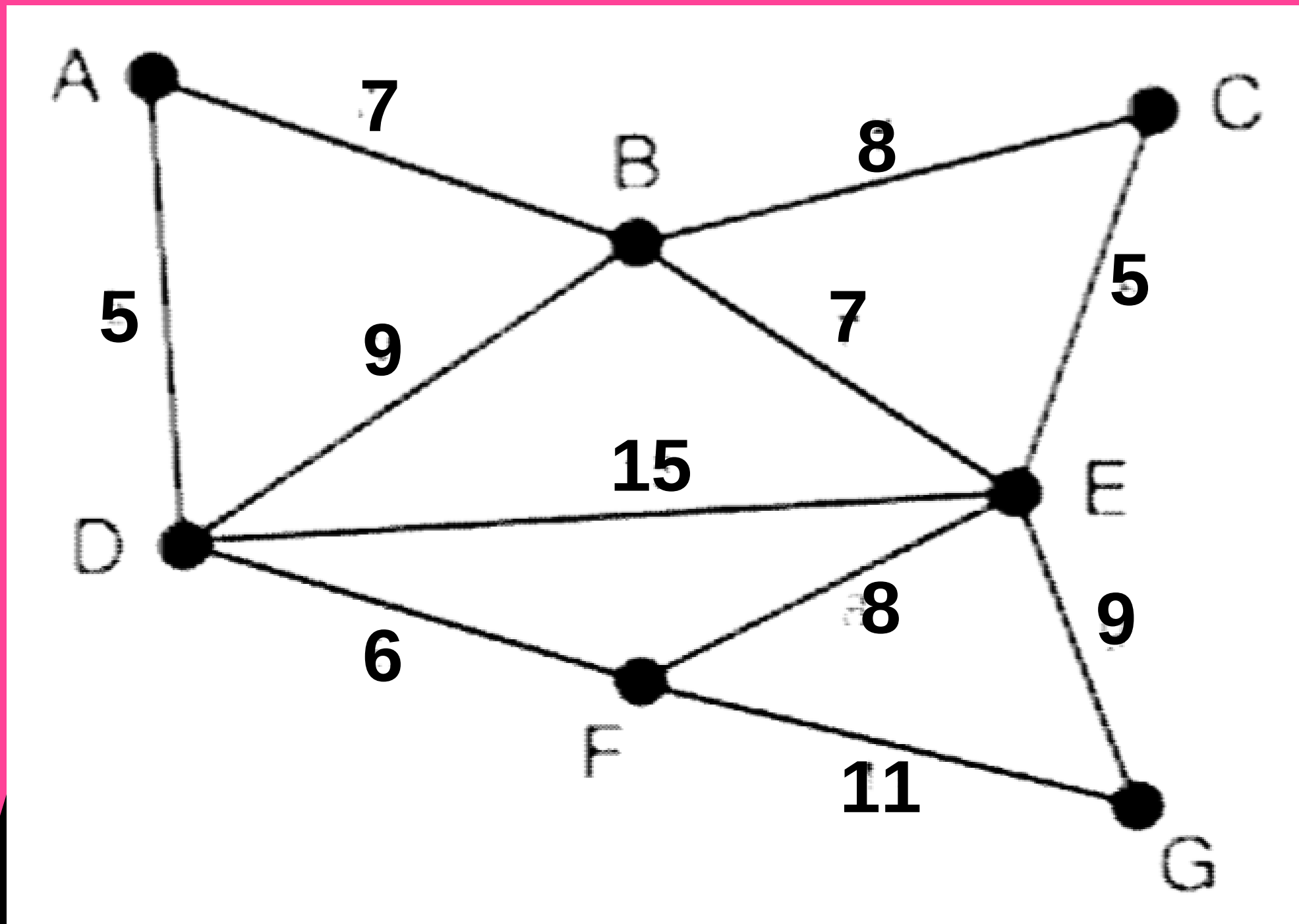


Pilih sisi (u,v)
dengan bobot minimum
yang tidak membentuk
sirkuit di T.
Tambahkan (u,v) ke
dalam T.



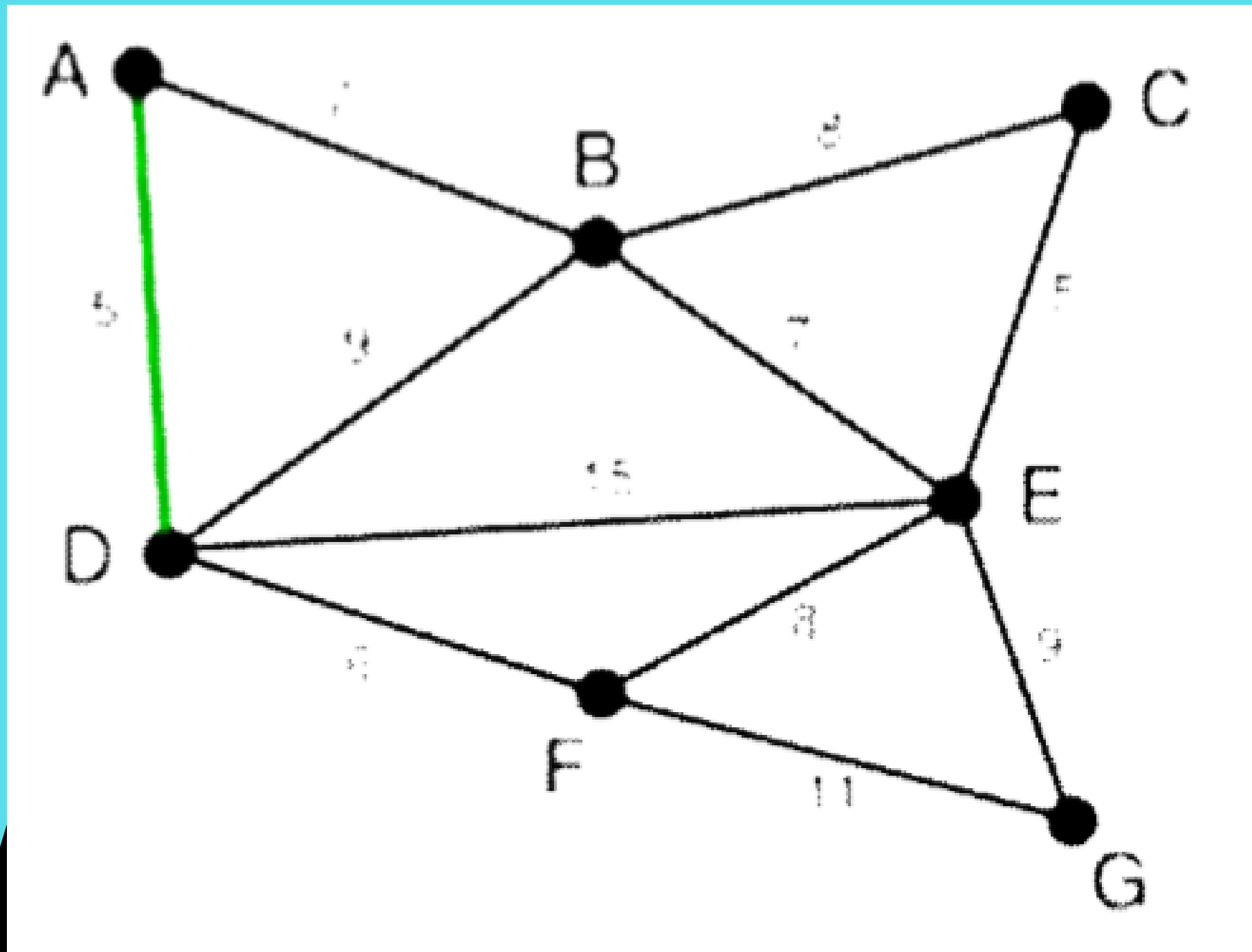
Ulangi langkah-
langkah
sebelumnya

Contoh Soal



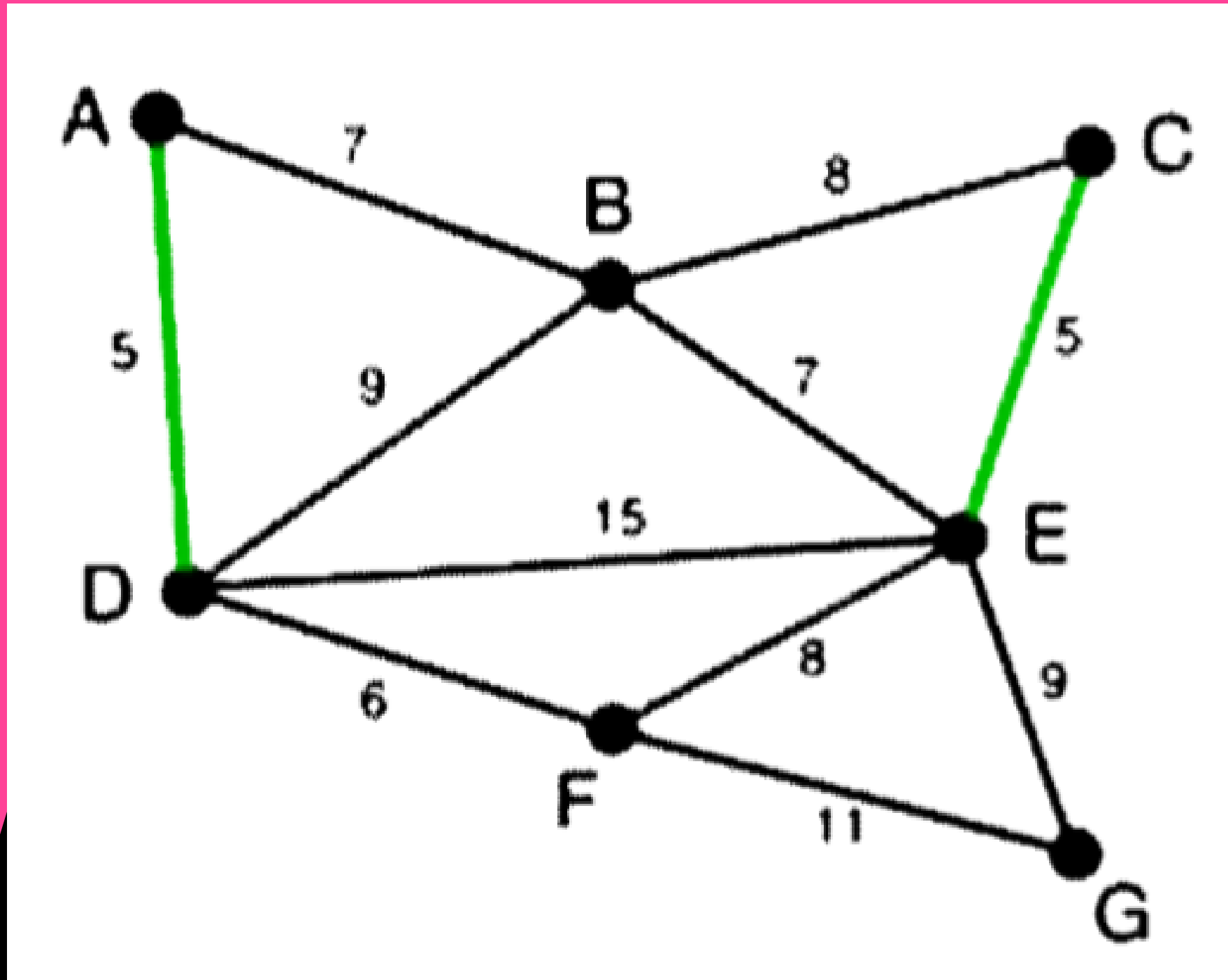
Berikut adalah graf berbobot yang akan dicari pohon merentang minimum nya, dengan bobot sebesar 90.

Jawaban



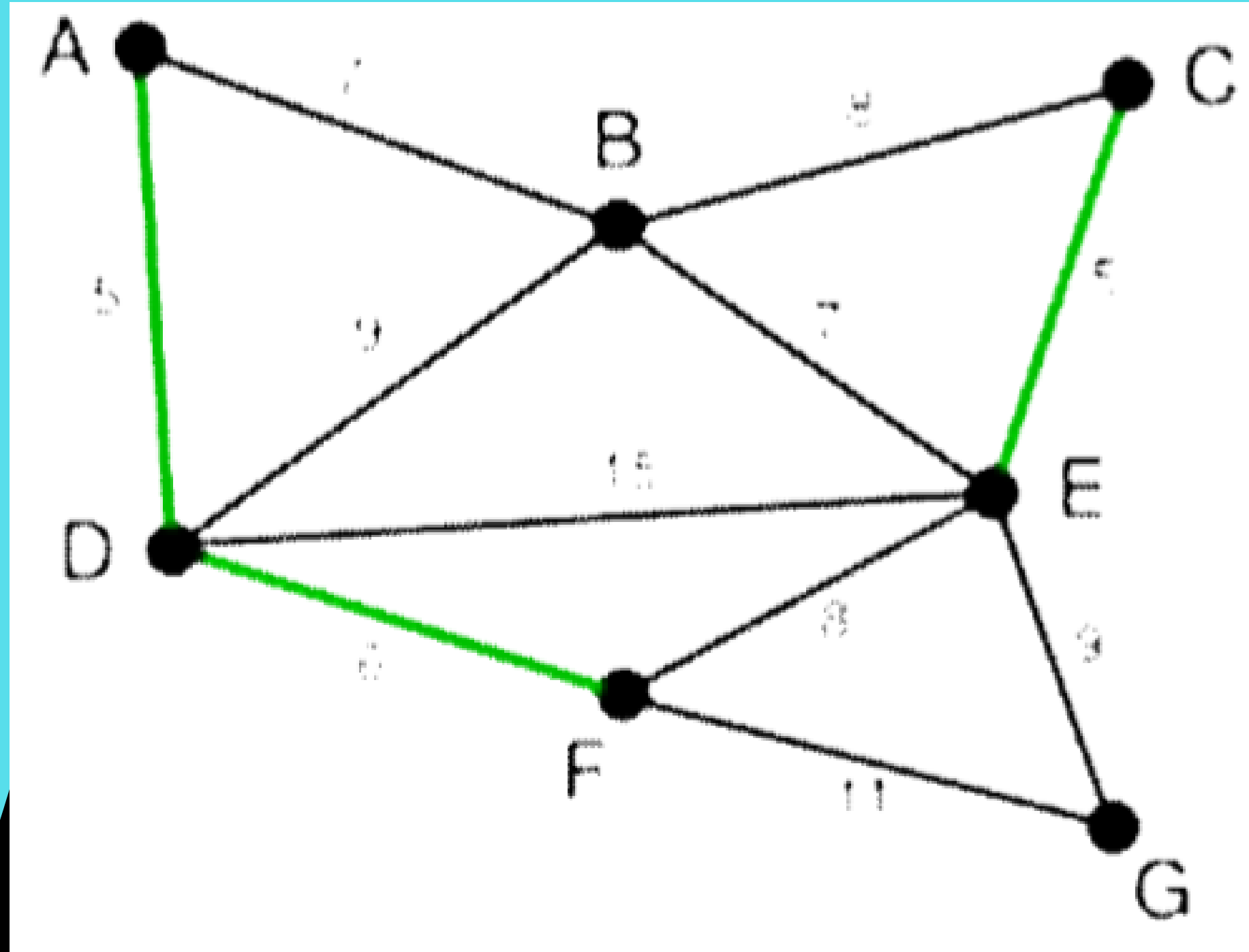
AD dan CE adalah sisi dengan bobot terkecil, pilih sembarang antara AD dan CE, pilih AD.

Jawaban



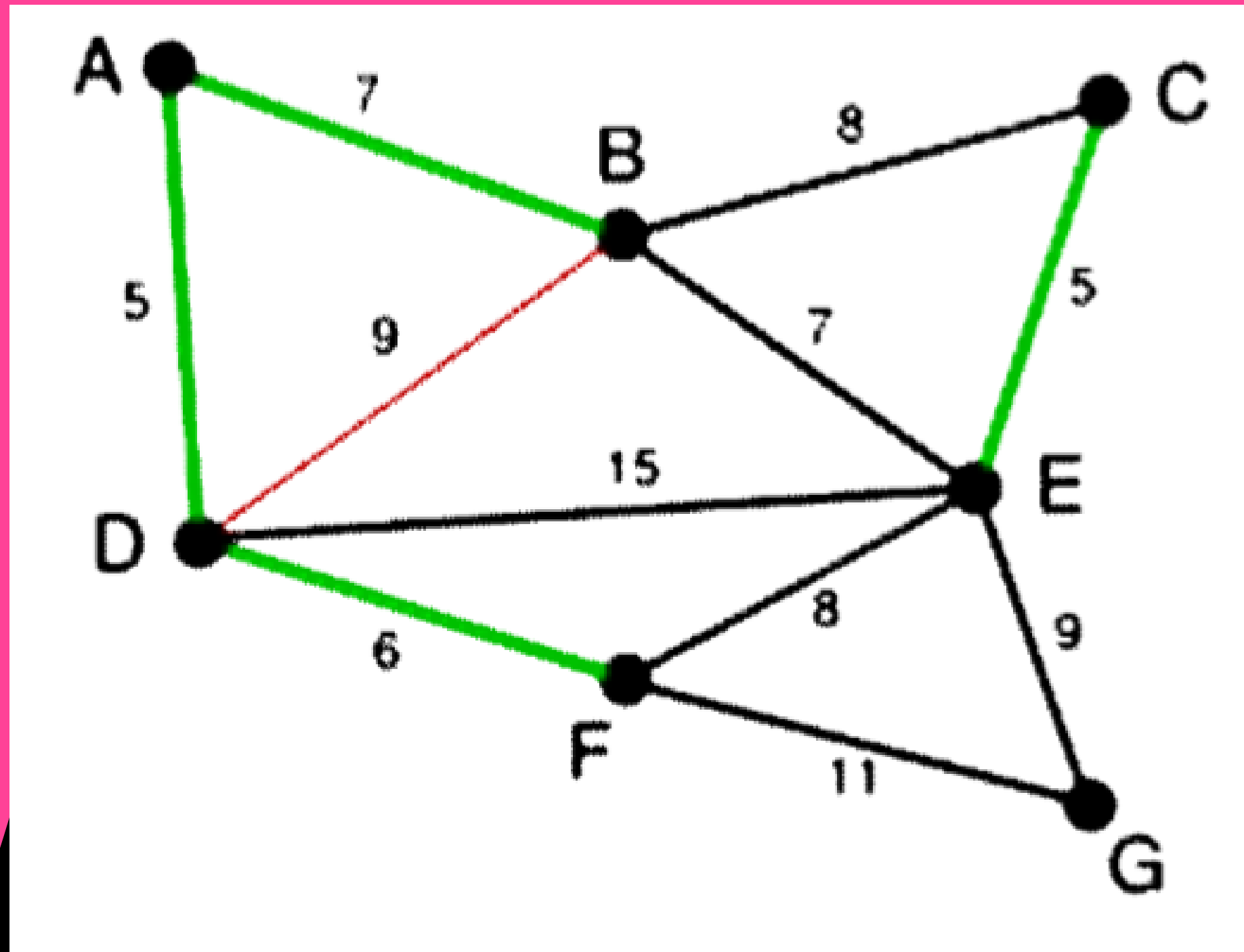
Karena CE sisi terkecil berikutnya, pilih CE

Jawaban



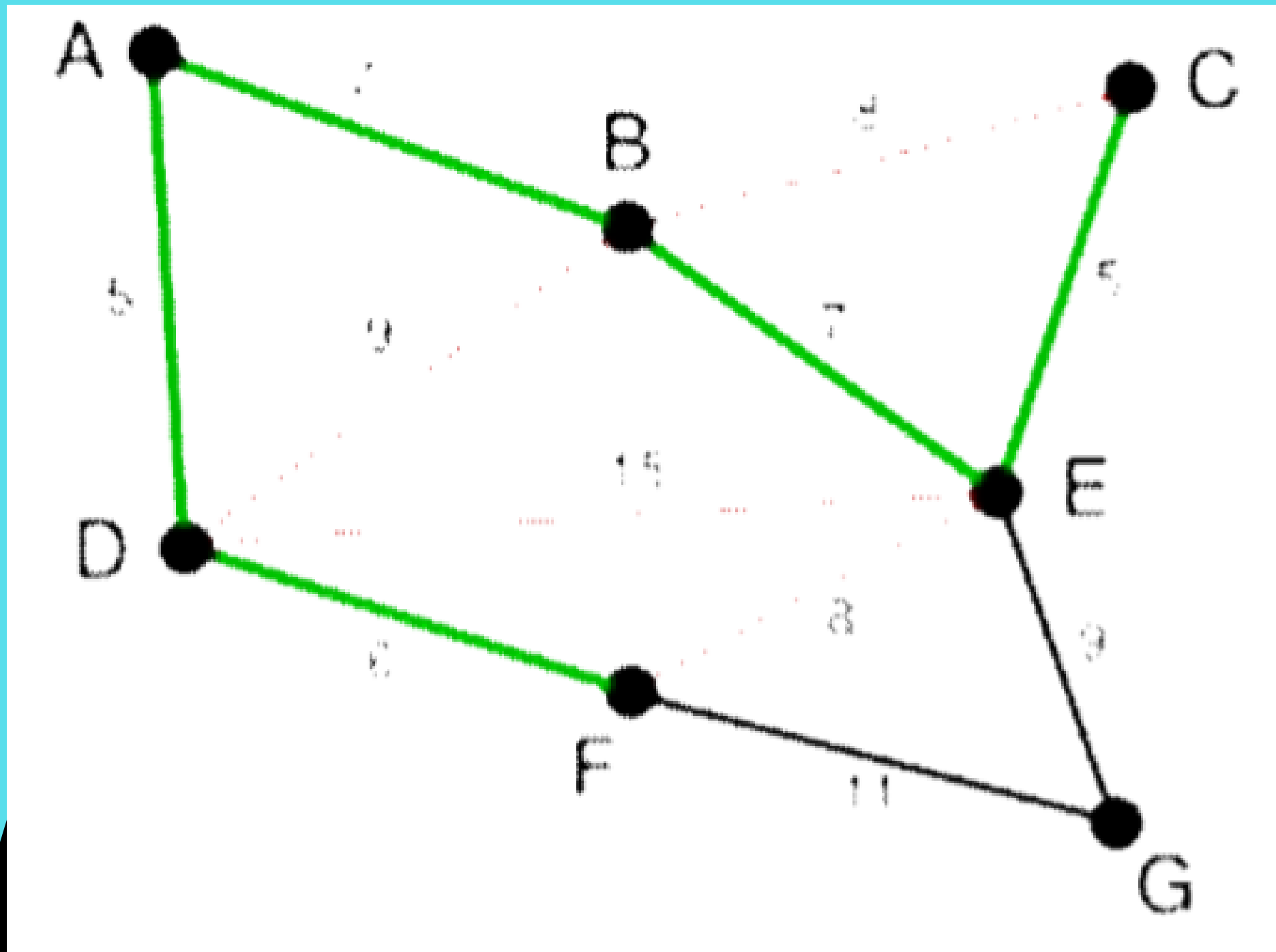
Selanjutnya, pilih sisi DF dengan bobot 6

Jawaban



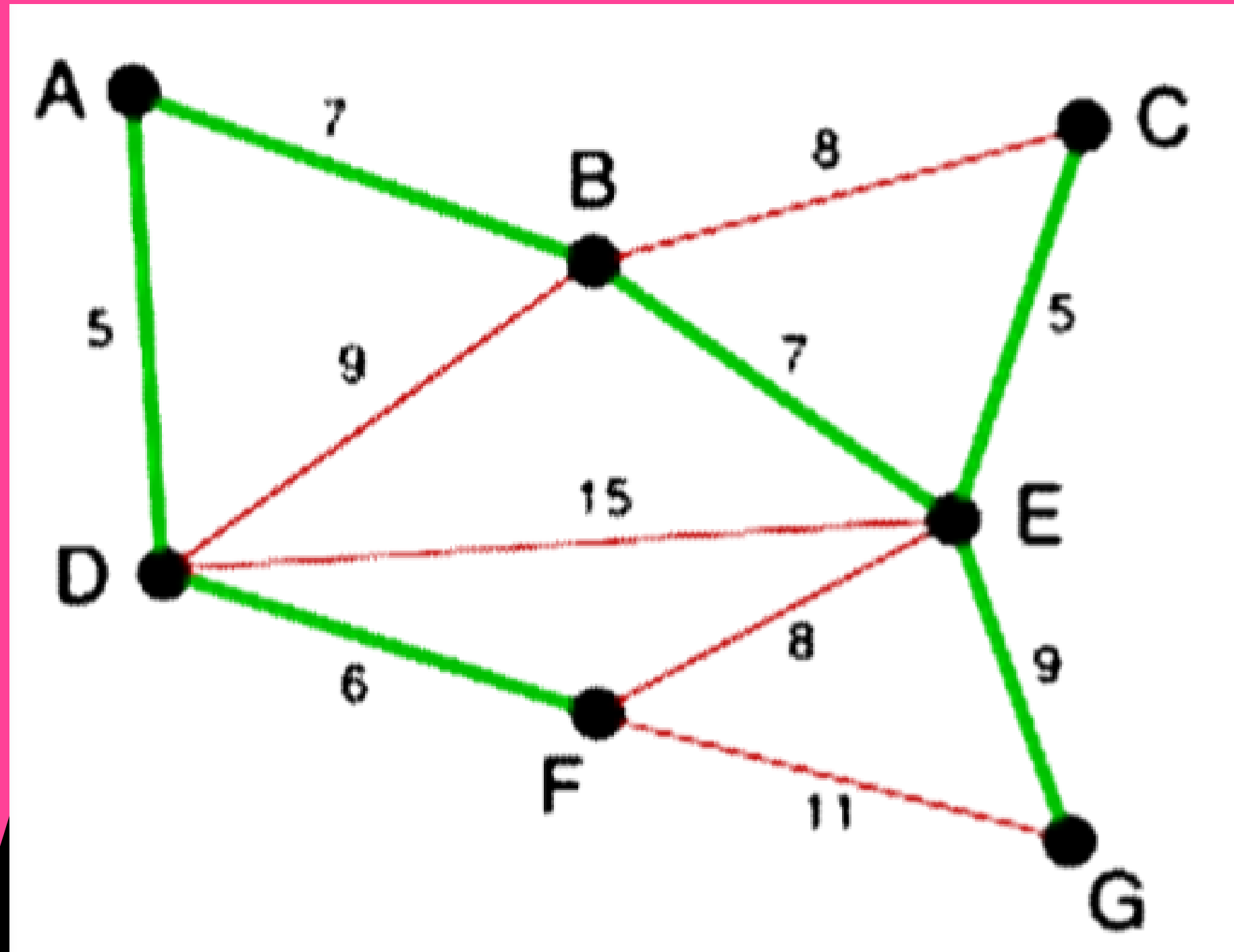
AB dan BE adalah sisi dengan bobot terkecil yang belum dipilih, pilih sembarang antara AB dan BE, pilih AB.

Jawaban



Karena BE sisi terkecil berikutnya, pilih BE

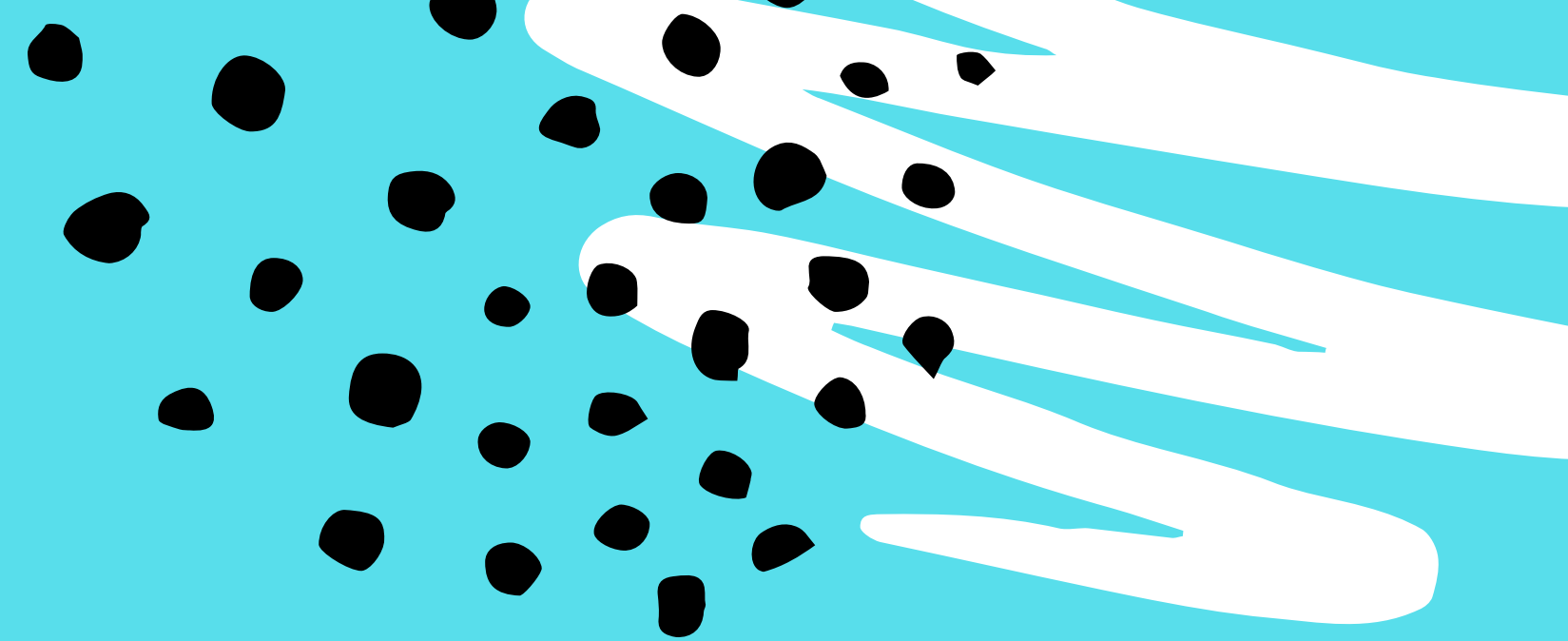
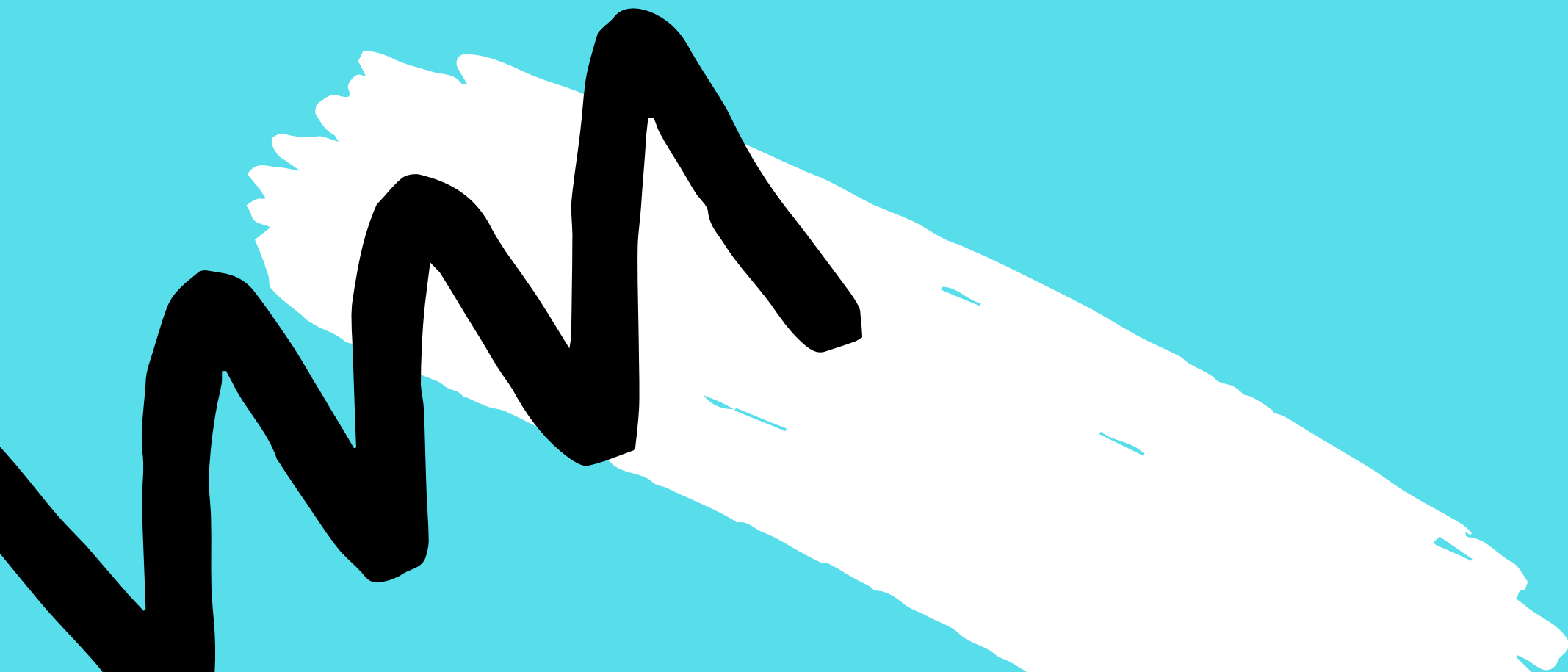
Jawaban



Simpul G adalah simpul terakhir yang belum dipilih, oleh karena itu pilih sisi yang terhubung dengan simpul G, yaitu GE dan GF, karena GE memiliki bobot lebih kecil, pilih GE.

Jawaban

Karena semua simpul telah dipilih, maka pohon merentang minimum ditunjukkan dengan warna hijau. Dengan bobot sebesar 39.



Penjelasan

Dengan menggunakan struktur data sederhana kompleksitas waktu algoritma ini adalah $O(E \log E)$ yang ekuivalen dengan $O(E \log V)$. Keduanya ekuivalen karena :

- E bernilai maksimal V^2 dan $\log V^2 = 2 \times \log V = O(\log V)$
- Jika kita tidak menghiraukan simpul yang terisolasi, $V \leq 2E$, jadi $\log V$ adalah $O(\log E)$


Hasil tsb diperoleh dengan melalui beberapa tahap, pertama, urutkan sisi-sisi dengan menggunakan comparison sort dengan waktu $O(E \log E)$. Kemudian, gunakan himpunan struktur data yang disjoint untuk menjaga tempat simpul yang mana ada di komponen mana. Kemudian, tunjukkan $O(E)$ operasi, untuk menemukan operasi – operasi dan kemungkinan satu union untuk setiap sisi. Sehingga, waktu total adalah $O(E \log E) = O(E \log V)$.

Misalkan sisi-sisi yang ada telah diurutkan atau bisa diurut dalam waktu yang linear (dengan counting sort atau radix sort), algoritma ini dapat menggunakan himpunan struktur data yang disjoint dengan lebih rumit, sehingga memiliki kompleksitas waktu $O(E \alpha(V))$, dimana α adalah inverse dari fungsi Ackermann yang bernilai single, dan tumbuh dengan sangat lambat.

Kompleksitas

```
MST-KRUSKAL( $G, w$ )  
O(1) 1  $A = \emptyset$   
O(V) 2 for each vertex  $v \in G.V$   
      3   MAKE-SET( $v$ )  
O(E log E) 4 sort the edges of  $G.E$  into nondecreasing order by weight  $w$   
           5 for each edge  $(u, v) \in G.E$ , taken in nondecreasing order by weight  
           6   if FIND-SET( $u$ )  $\neq$  FIND-SET( $v$ )  
O(V log V) 7      $A = A \cup \{(u, v)\}$   
           8     UNION( $u, v$ )  
           9 return  $A$ 
```

Kompleksitas waktu untuk algoritma ini adalah $O(E \log E)$ yang ekuivalen dengan $O(E \log V)$, dimana E menyatakan jumlah sisi dan V jumlah simpul dari graph G .



TERIMA KASIH

