Intro to Contemporary Math

Hamiltonian Circuits and Nearest Neighbor Algorithm

Nicholas Nguyen
nicholas.nguyen@uky.edu

Department of Mathematics
UK

Agenda

- Hamiltonian Circuits and the Traveling Salesman Problem
- Nearest Neighbor Algorithm

Announcements

- Homework EC is due on Saturday

Hamiltonian Circuit

A circuit on a graph is a Hamiltonian circuit if it:

- starts and ends at the same vertex
- visits every other vertex on the graph
- does not repeat vertices

Traveling Salesman Problem

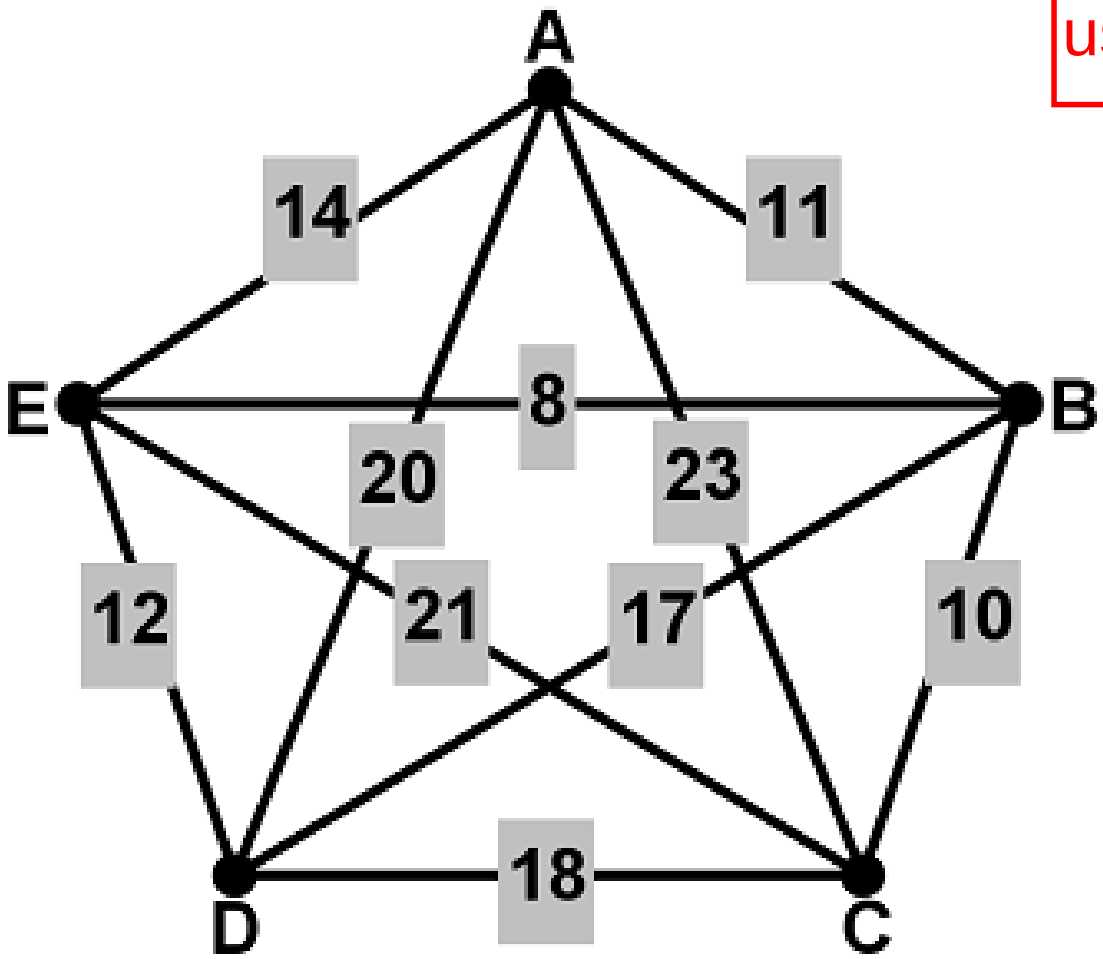
- Given a connected graph with weights, how do we find the Hamiltonian circuit with the lowest total weight?
- Application: A salesman must fly out on a trip to some cities, and then return home. In what order should he visit the cities to minimize the cost of travel?

Hamiltonian Circuit Example

Start at A, visit other vertices, and end at A.

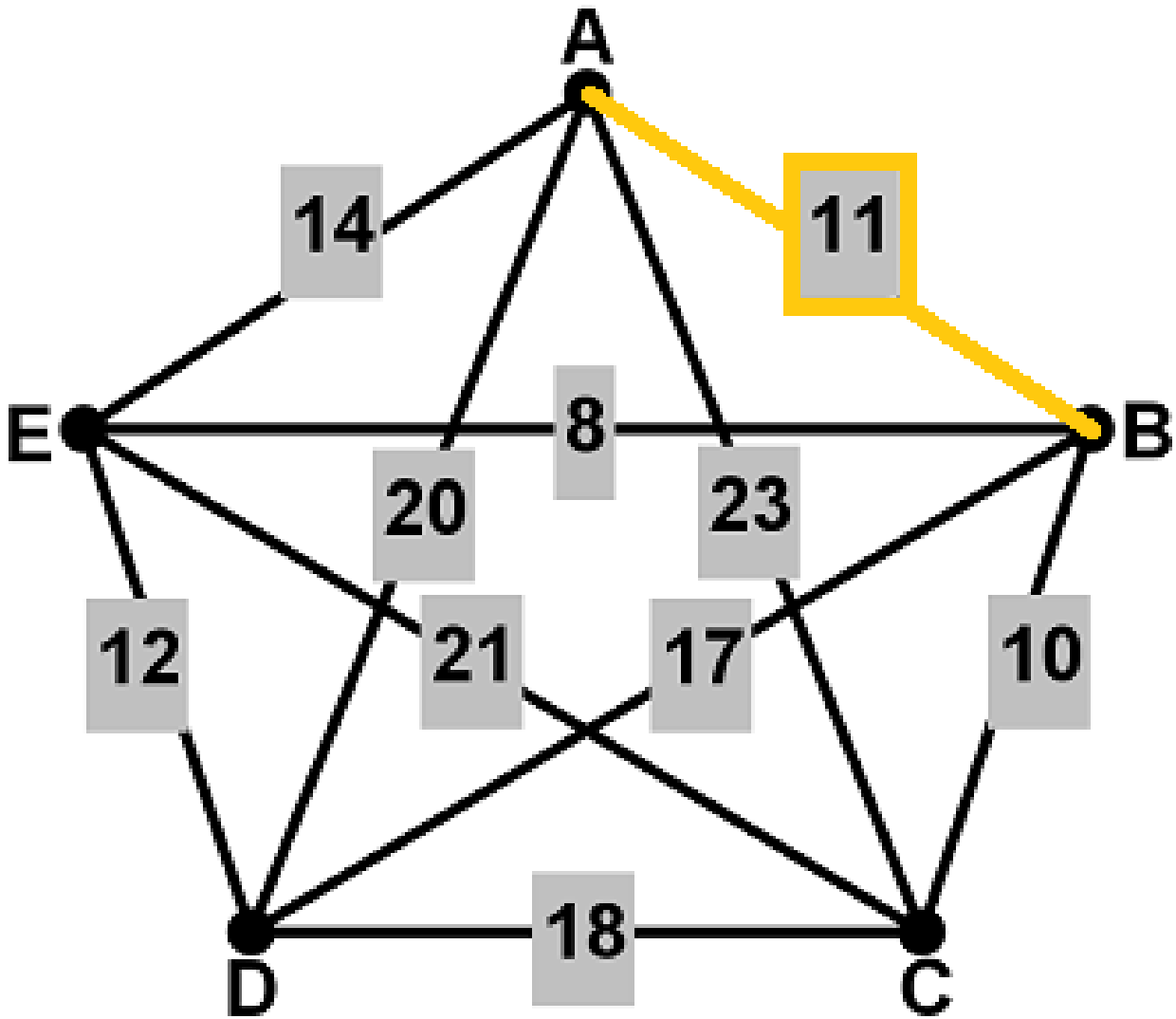
Circuit: A, , , , , A

No special algorithm is used yet.



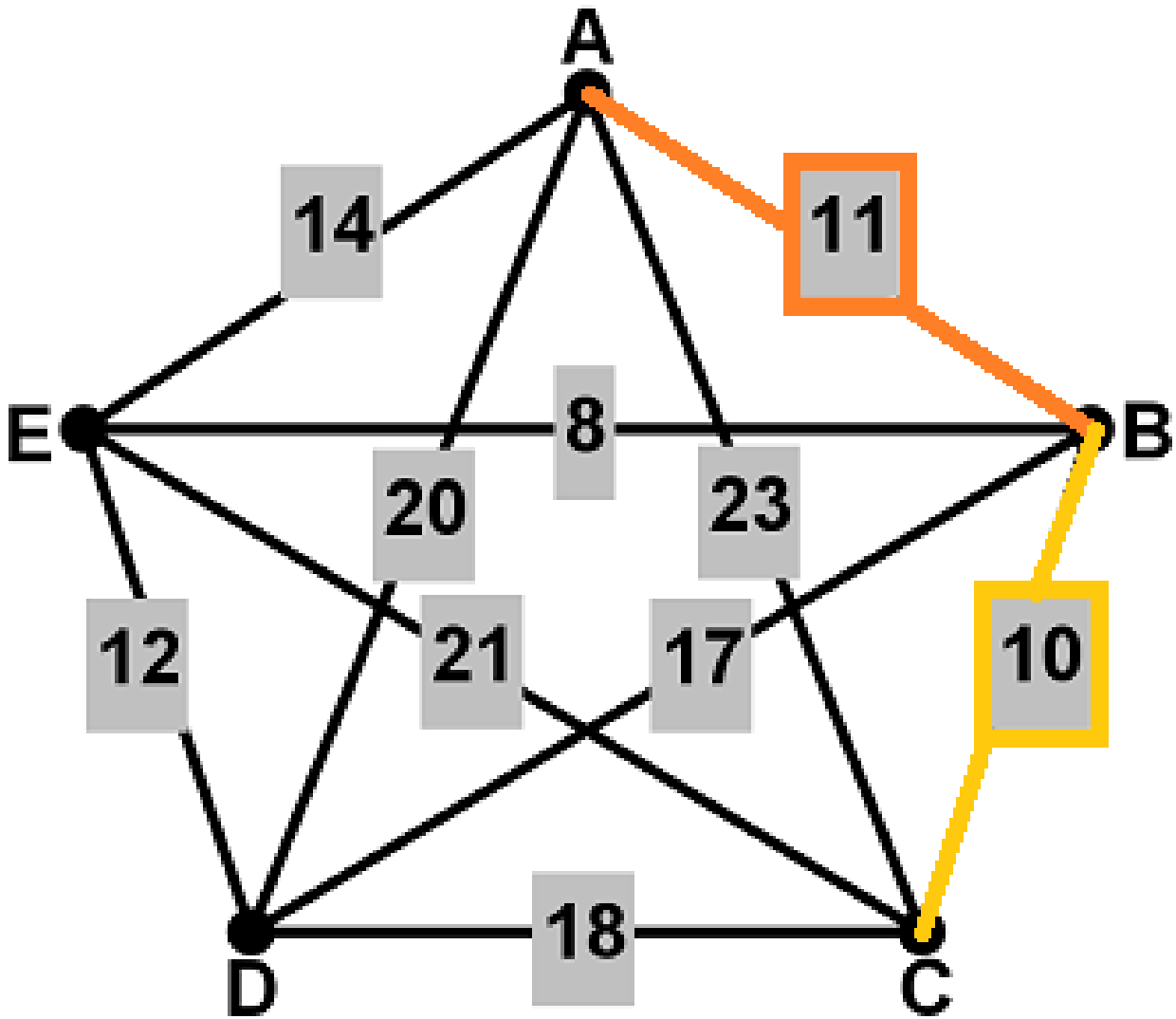
Hamiltonian Circuit Example

Circuit: A, B, , , , A



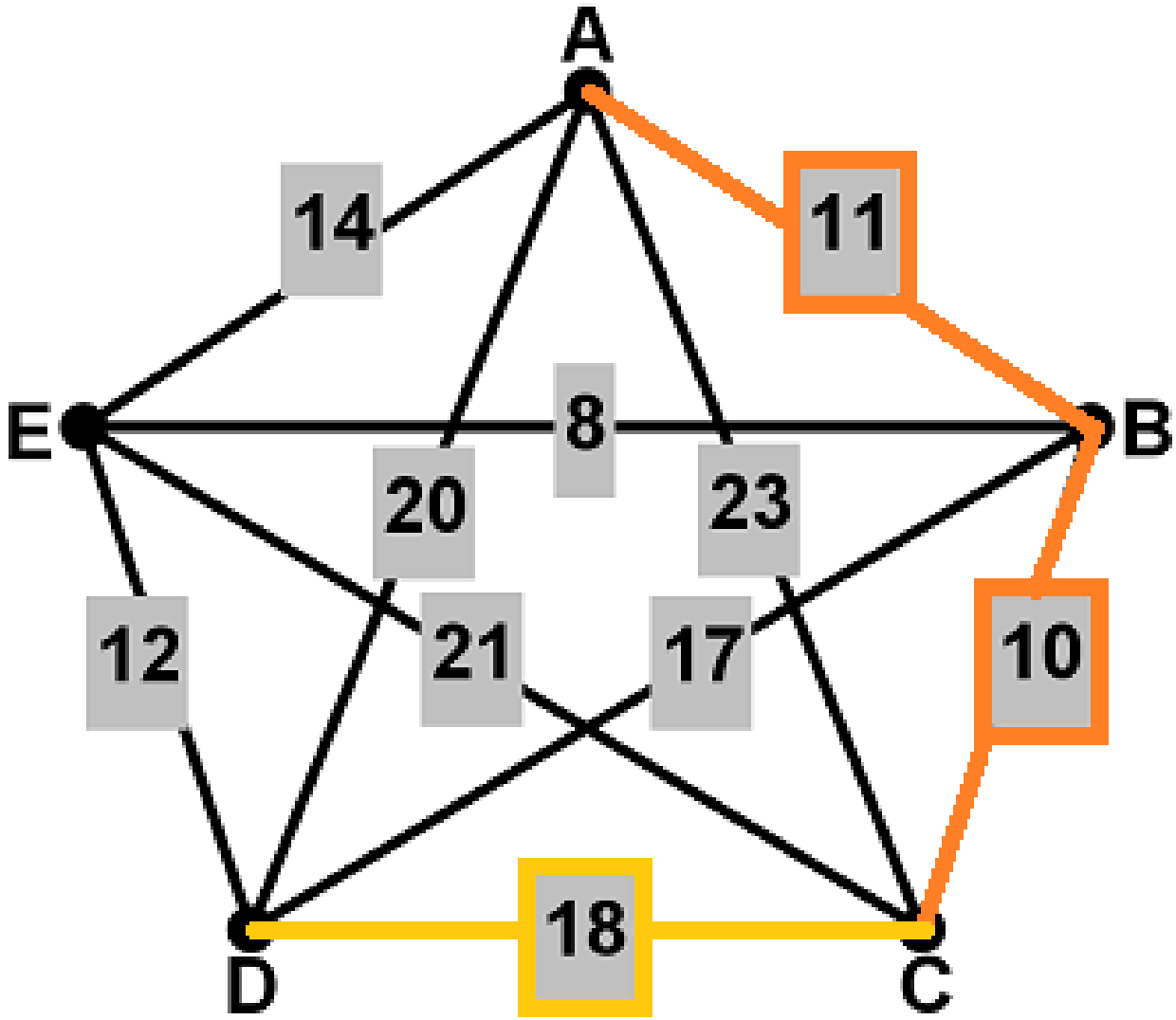
Hamiltonian Circuit Example

Circuit: A, B, C, , , A



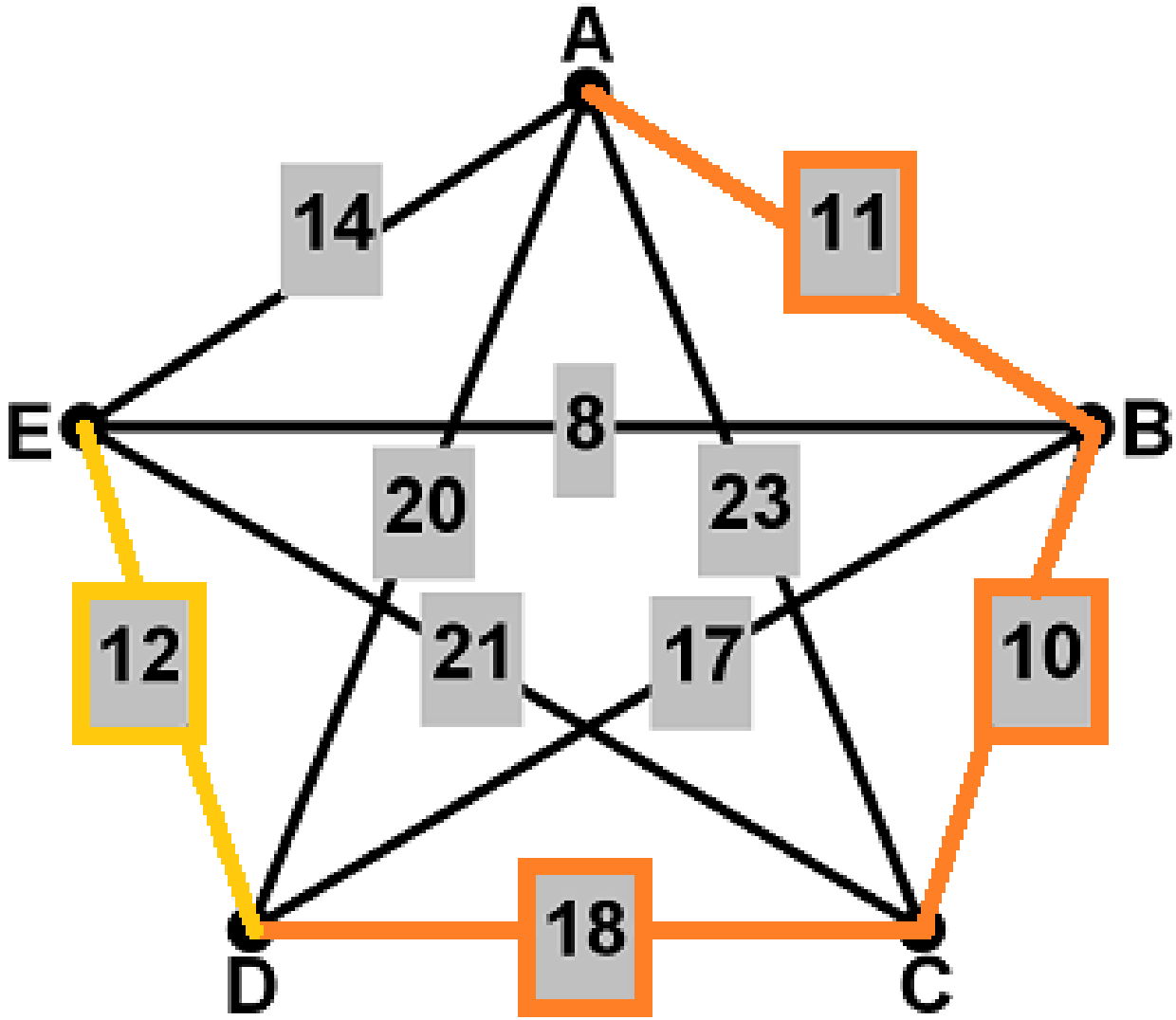
Hamiltonian Circuit Example

Circuit: A, B, C, D, , A



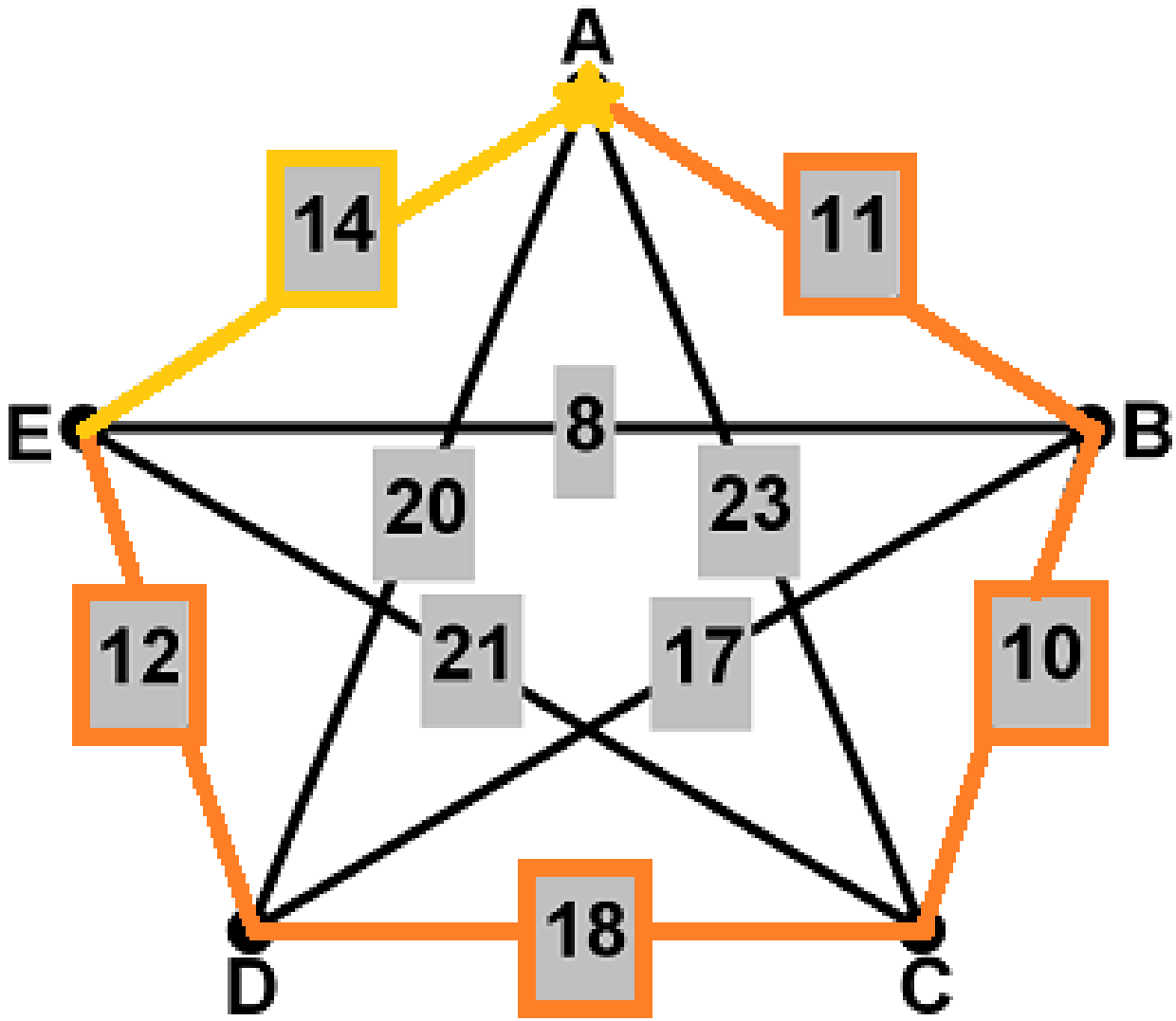
Hamiltonian Circuit Example

Circuit: A, B, C, D, E, A

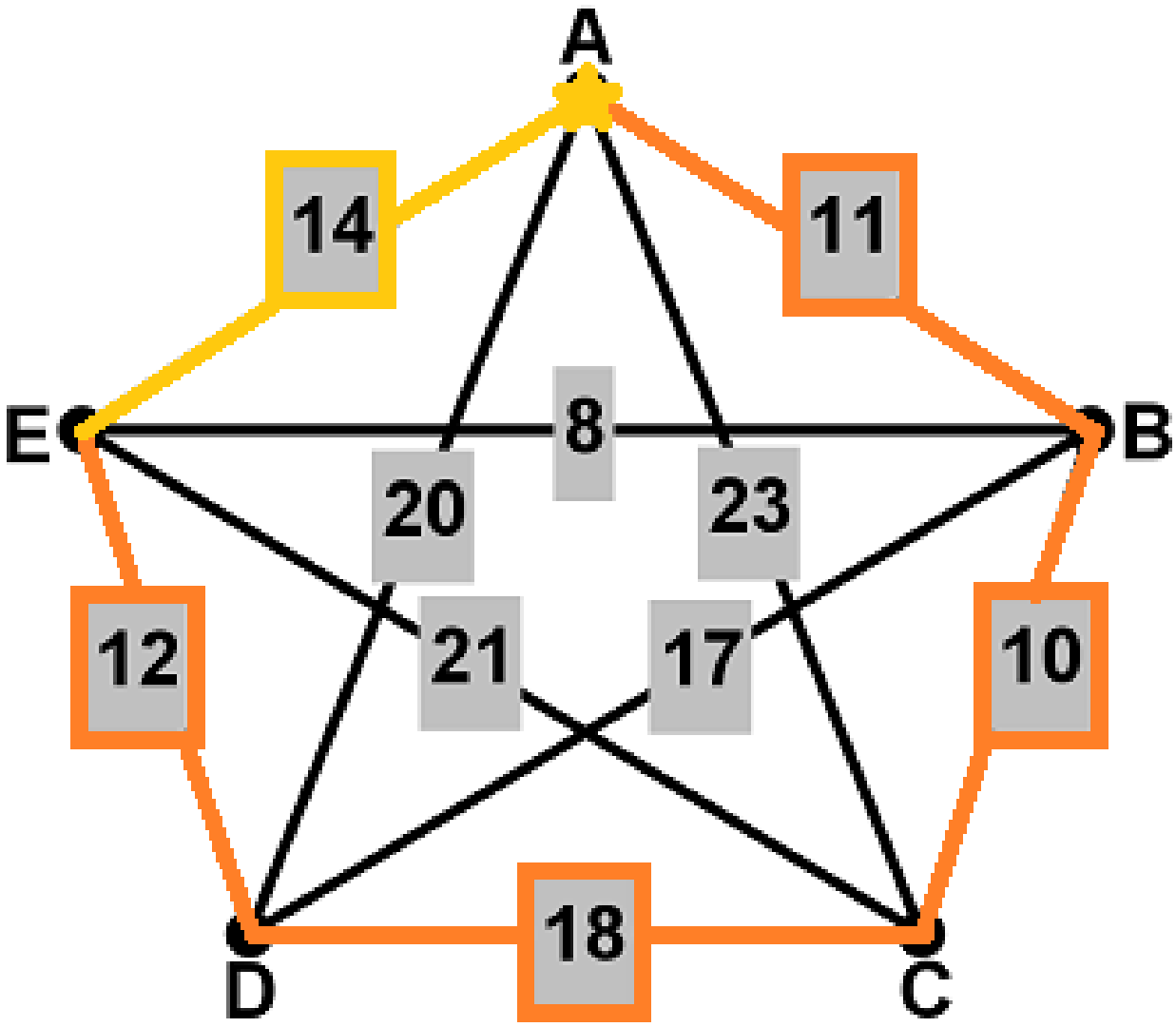


Hamiltonian Circuit Example

Circuit: A, B, C, D, E, A (return)



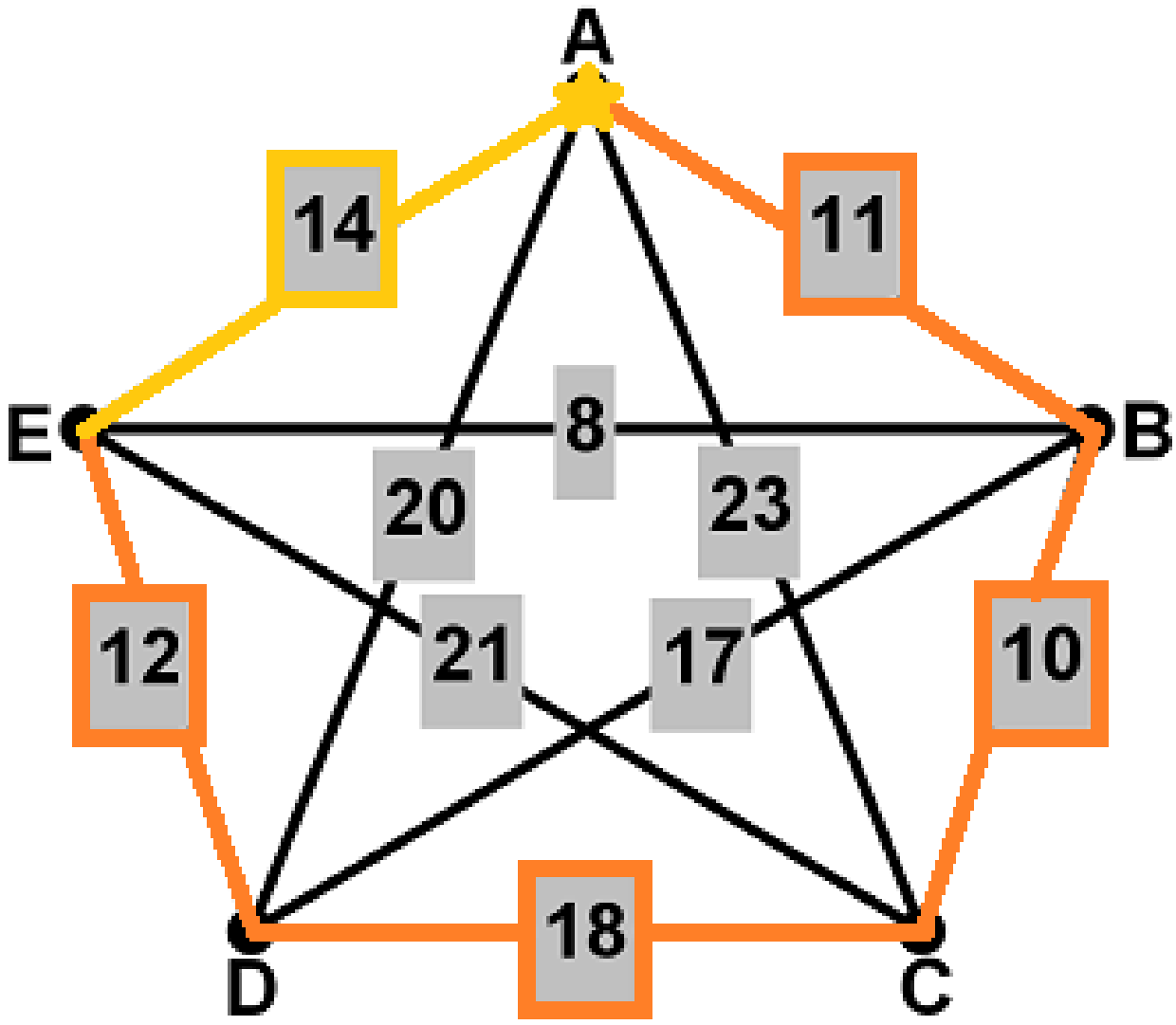
?(7.1) Total Weight



What is the total weight? Type and send a **number**.

Total Weight

$$11 + 10 + 18 + 12 + 14 = \boxed{65}$$



Nearest Neighbor Algorithm

Goal: Find the shortest Hamiltonian circuit
Step 1: Pick a starting vertex.
Step 2: Move to the **nearest unvisited vertex** (along the edge with the lowest weight).
Step 3: Repeat Step 2 until the circuit is complete: once you have visited all other vertices, **go back to the starting vertex.**

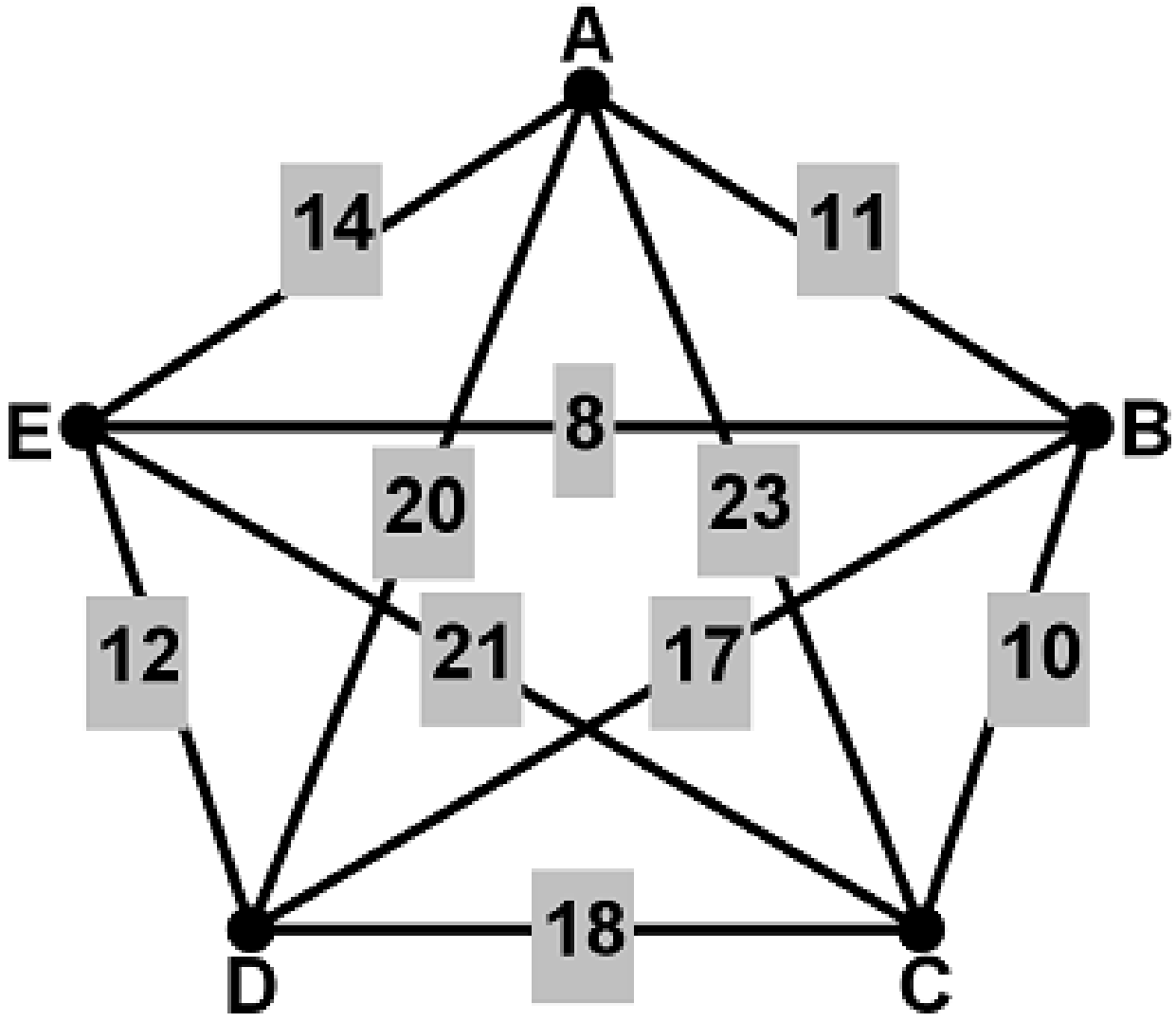
Nearest Neighbor Demo

Use the Nearest Neighbor Algorithm starting at E. Record the vertices in the order you follow them and find the total weight of the circuit.

Circuit: E, _____ , _____ , _____ , _____ , E

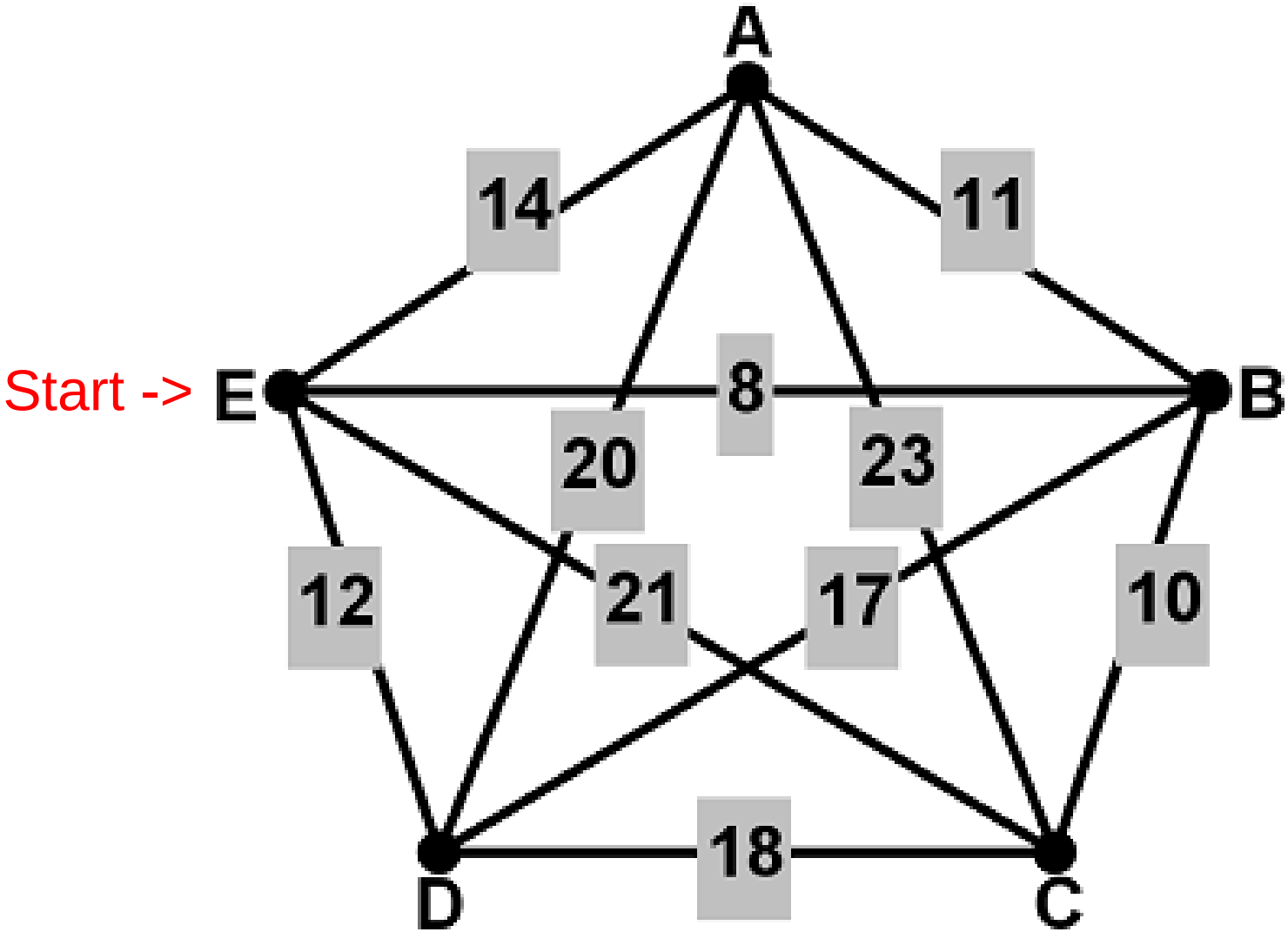
In class, when you see a "?(7.#)" in the upper left corner, press the label of the vertex (A-E) we should go to next according to the algorithm.

Start ->

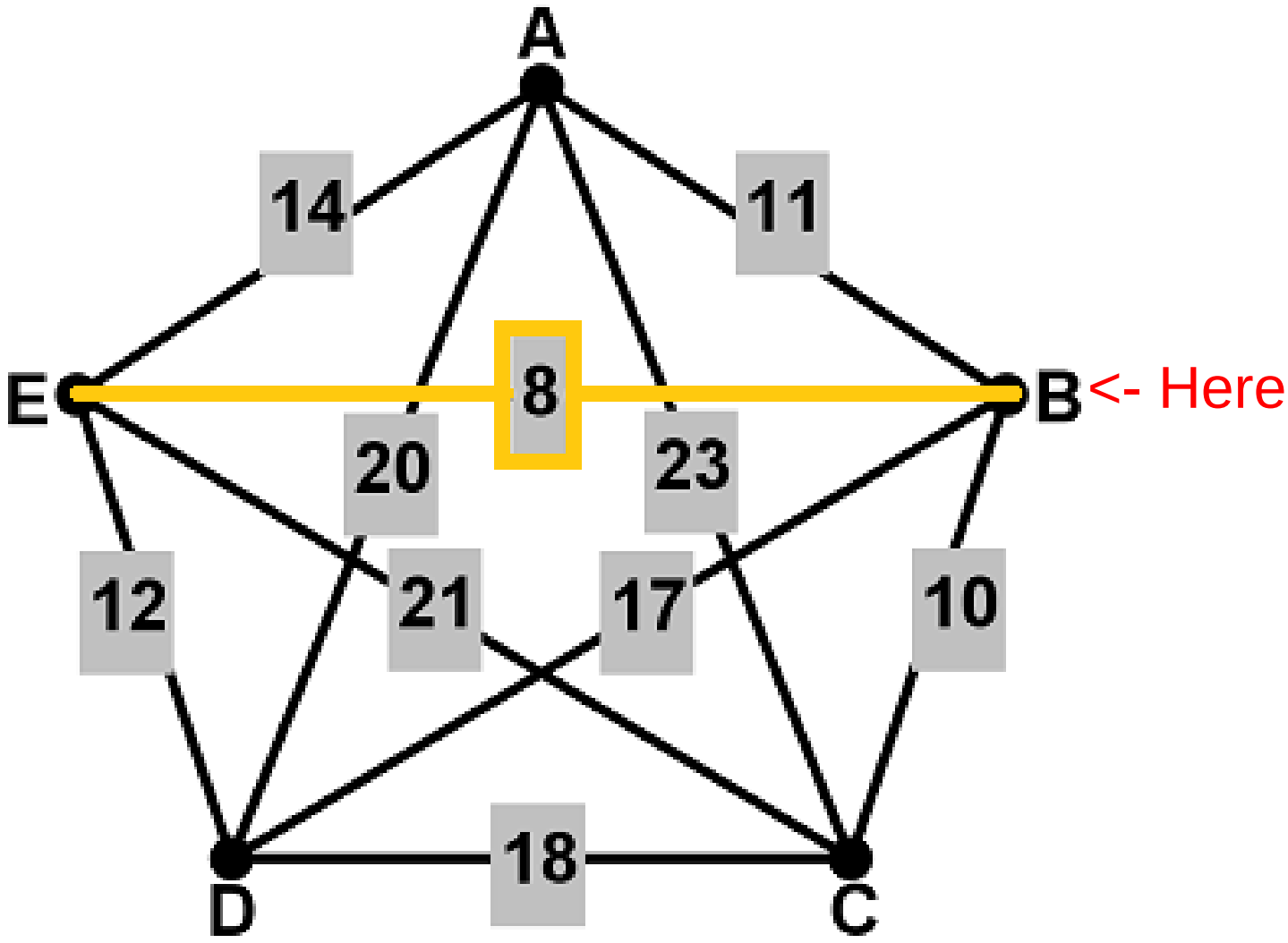


?(7.2) Next?

Press the letter of the next vertex in the circuit.



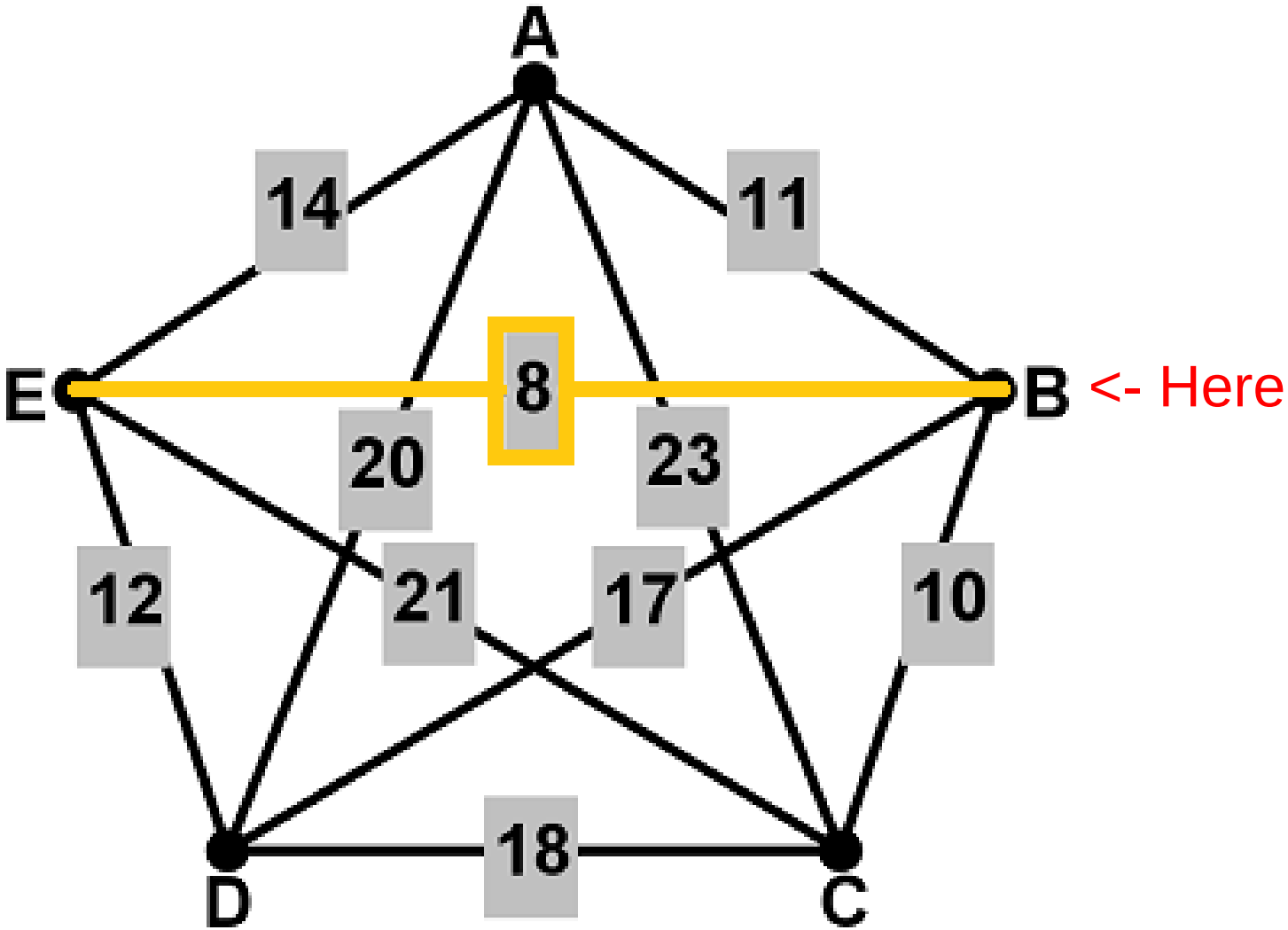
Cheapest edge at E is 8 (goes to B)



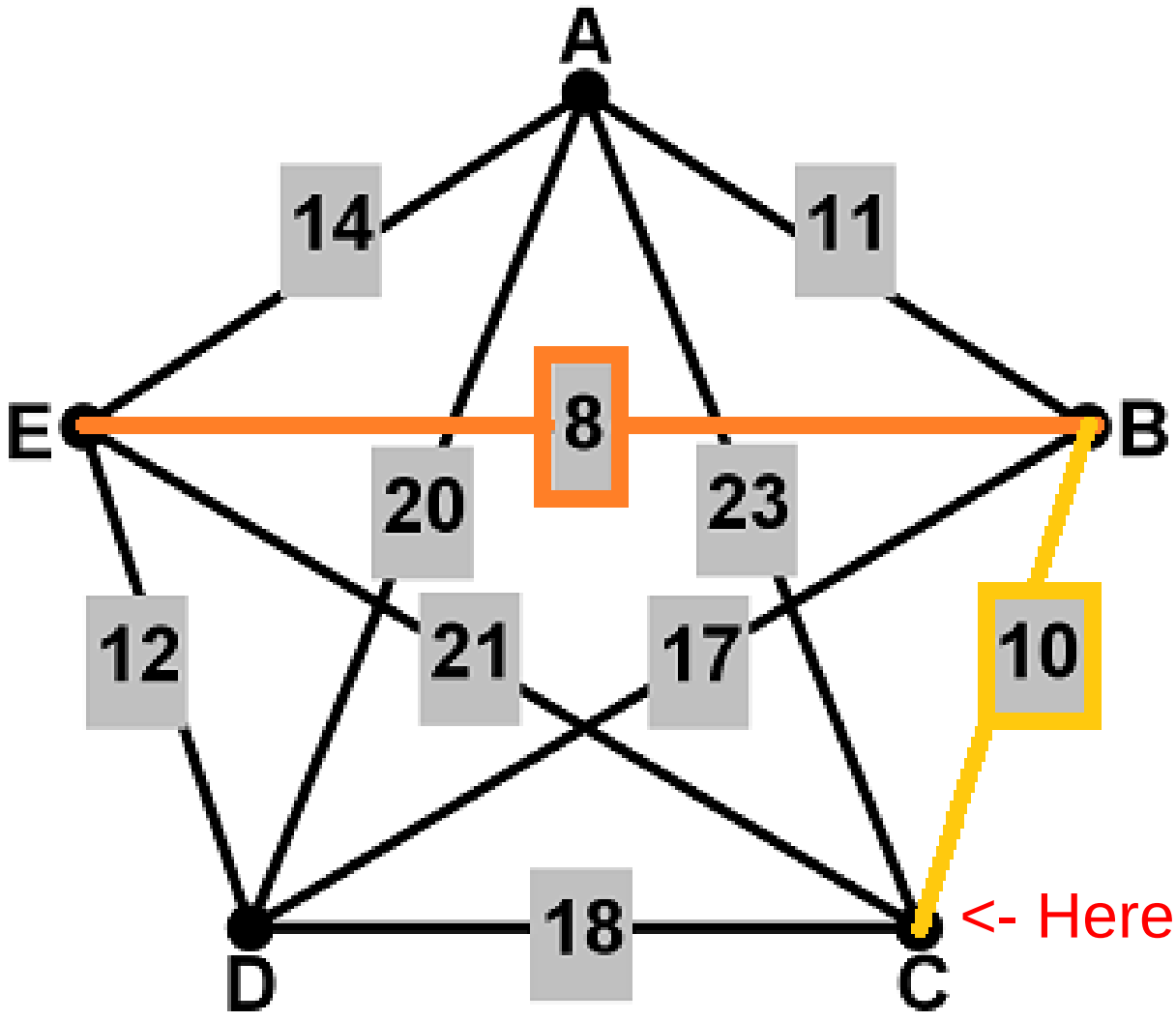
?(7.3) Next?

Cheapest edge at E is 8 (goes to B)

Type the next vertex.



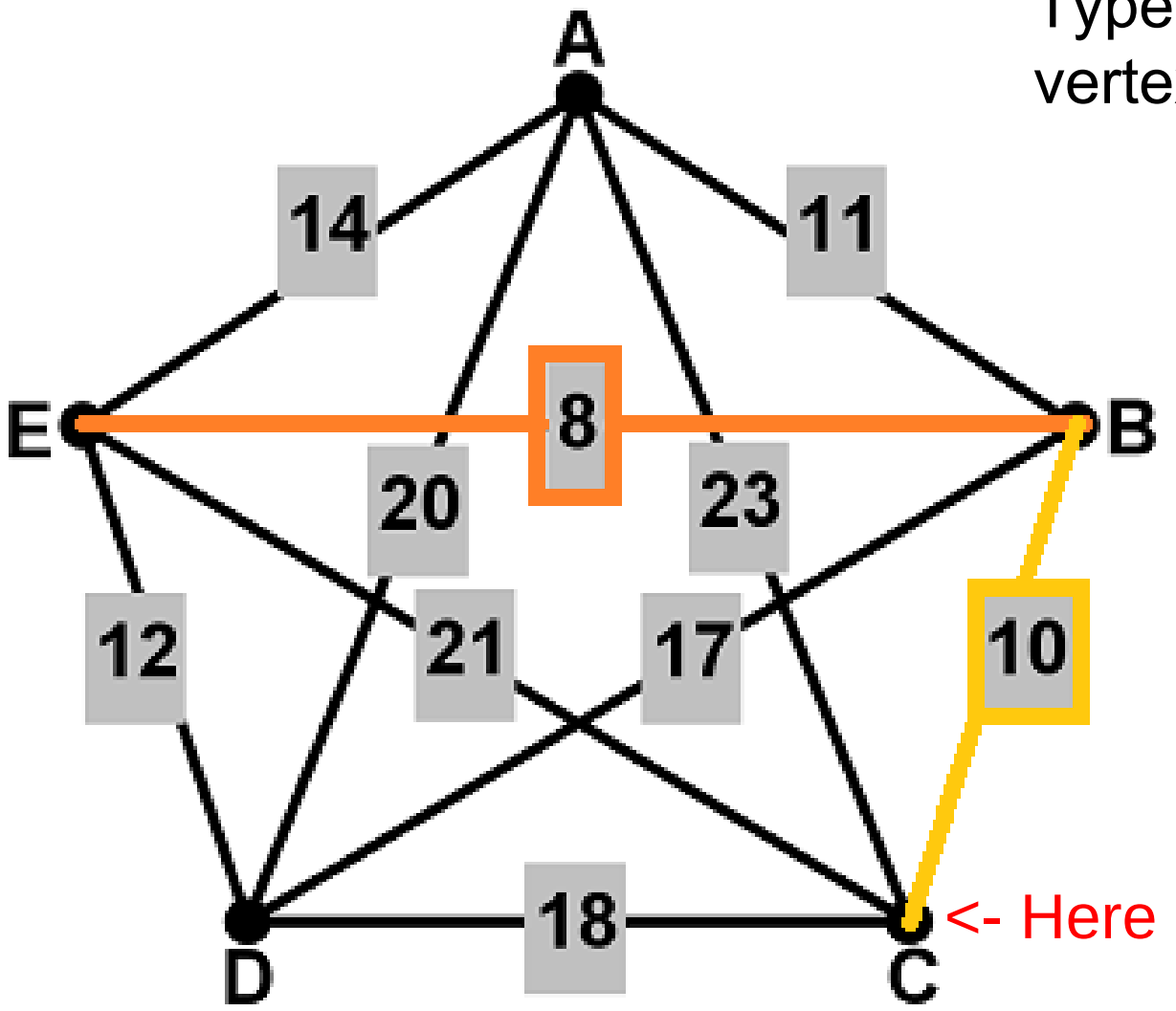
Cheapest edge at B that leads to a new vertex is 10 (to C).



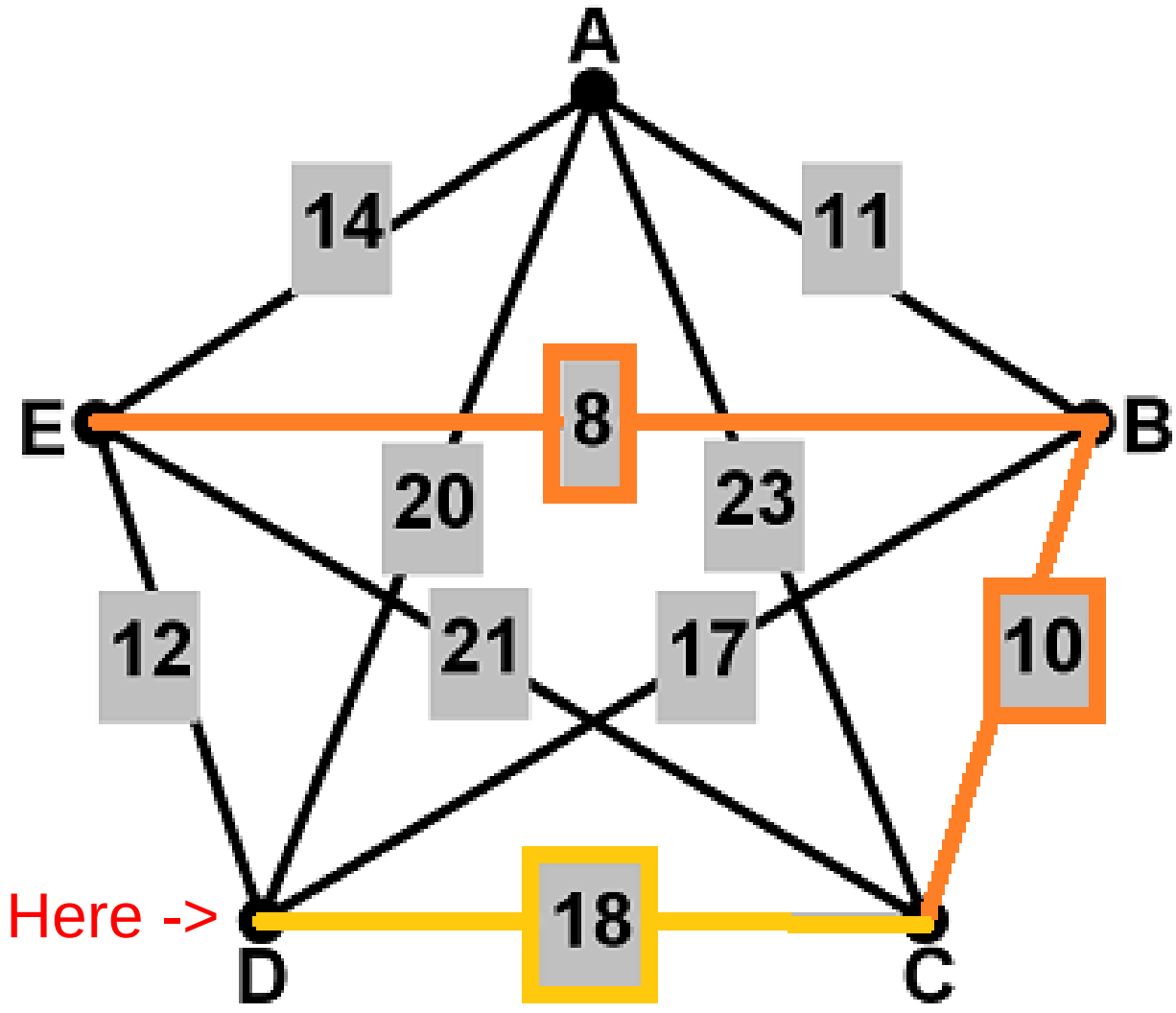
?(7.4) Next?

Cheapest edge at B that leads to a new vertex is 10 (to C).

Type the next vertex.



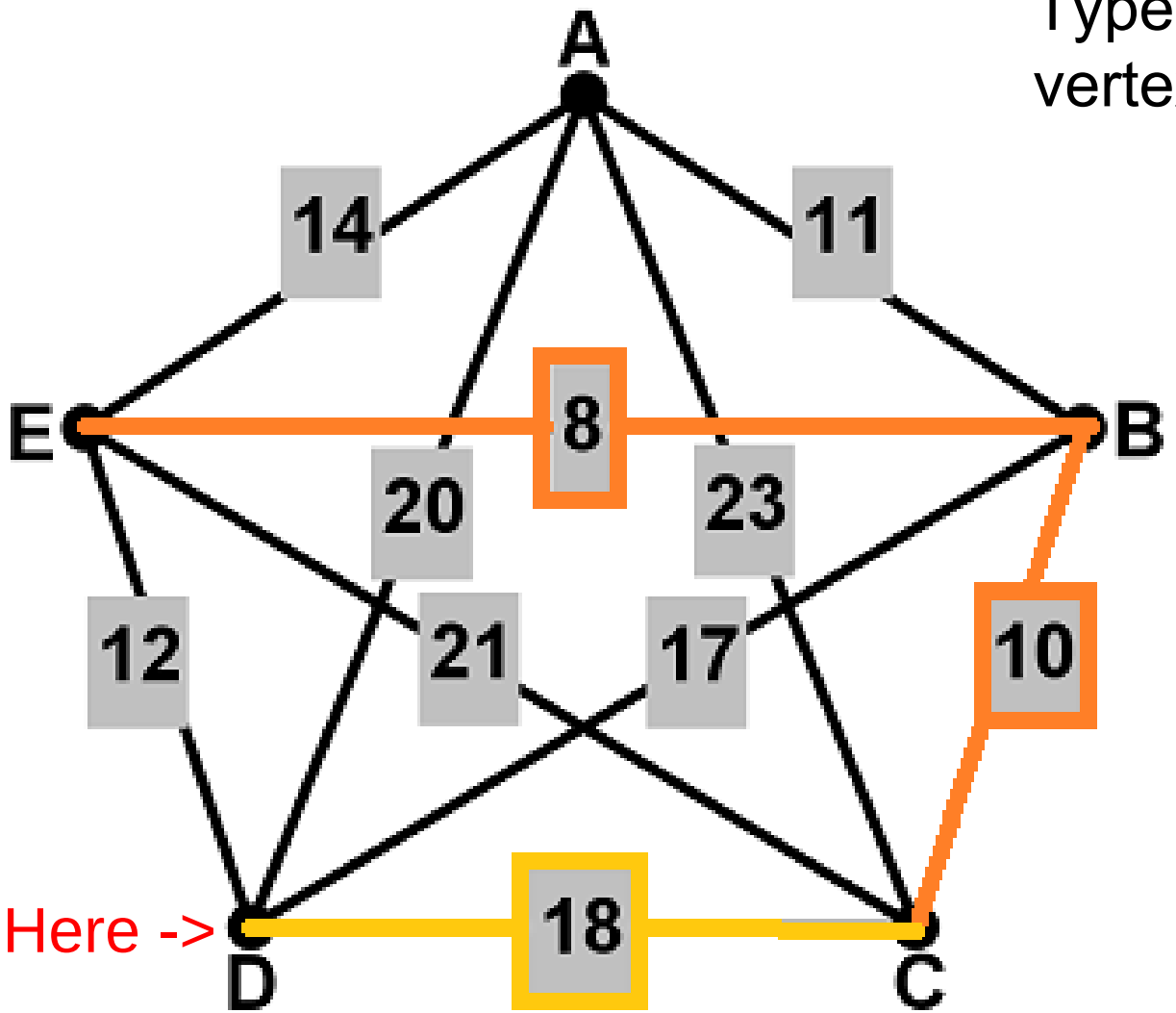
Cheapest edge at C that leads to a new vertex is 18 (to D).



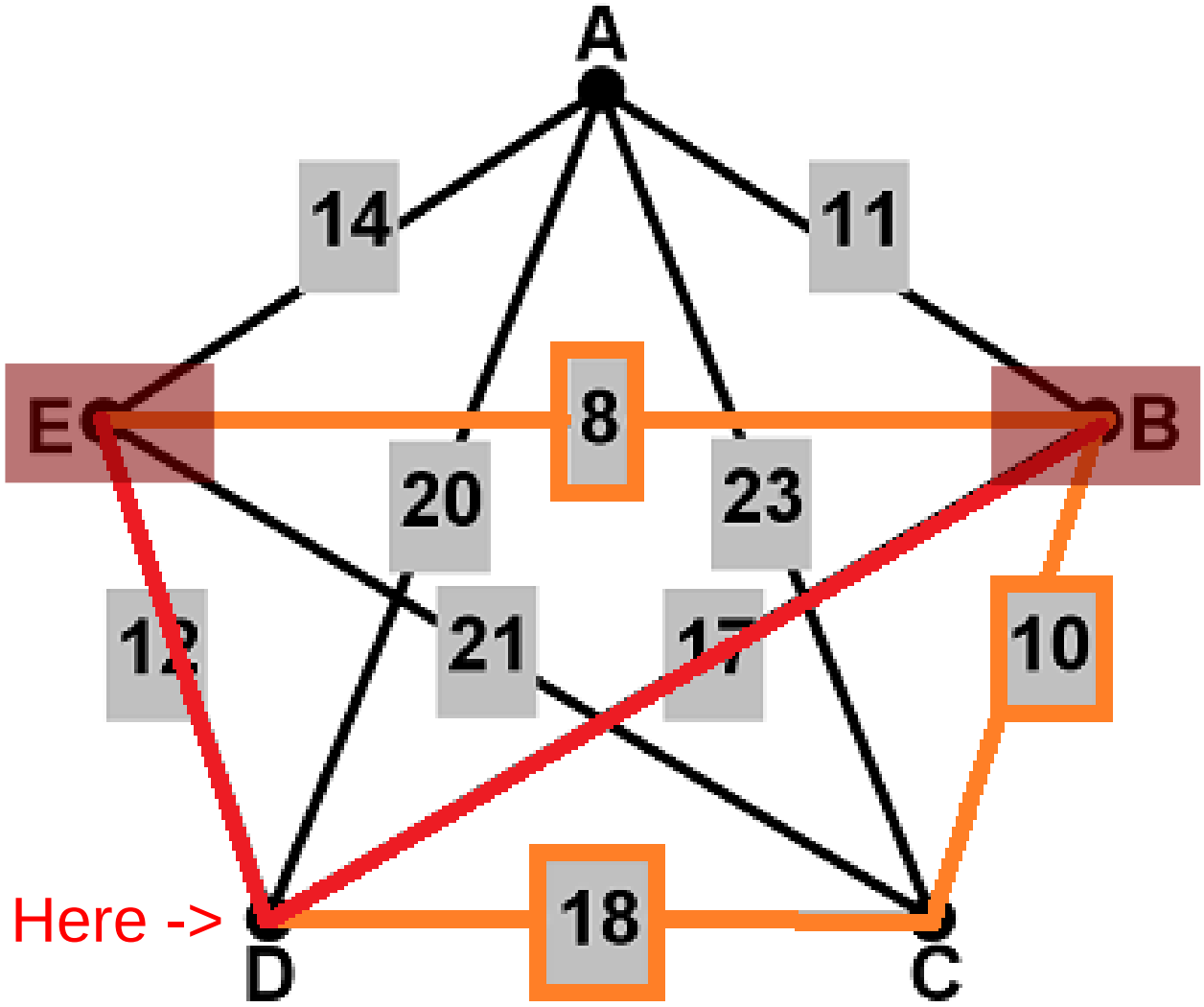
?(7.5) Next?

Cheapest edge at C that leads to a new vertex is 18 (to D).

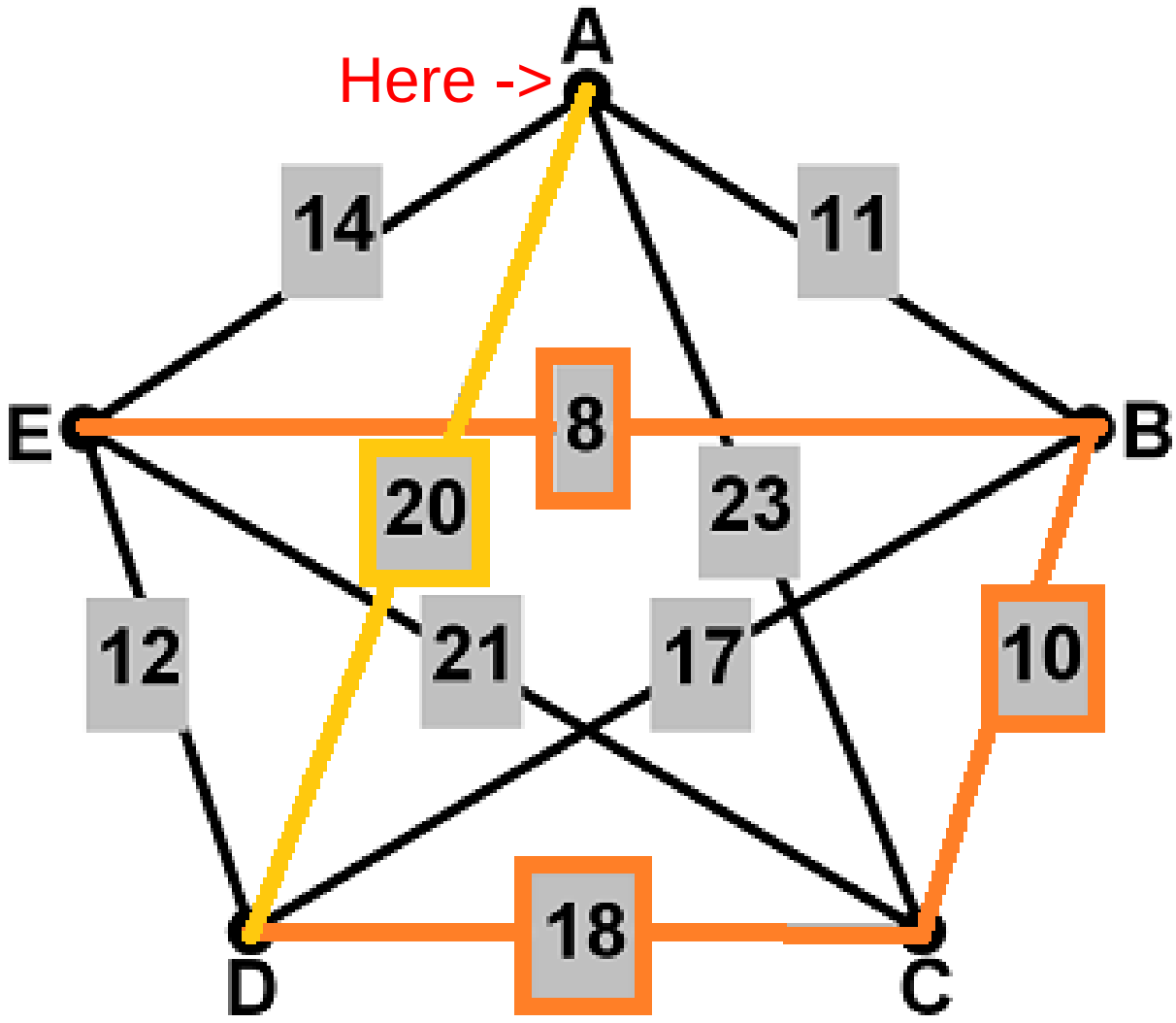
Type the next vertex.



Do not use 12 and 17 edges. They lead to previous vertices.



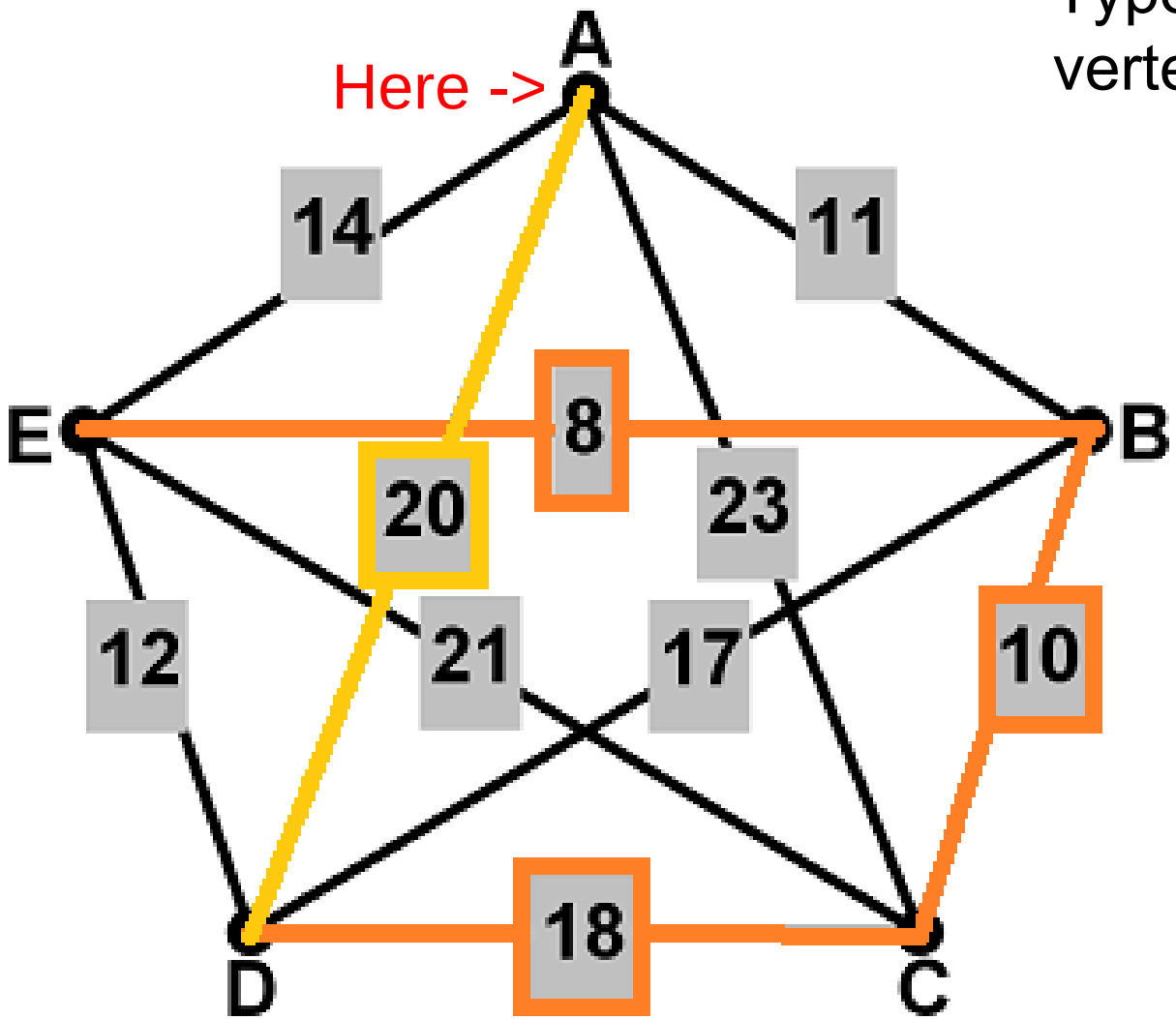
Cheapest edge at D that leads to a new vertex is 20 (to A).



?(7.6) Next?

Cheapest edge at D that leads to a new vertex is 20 (to A).

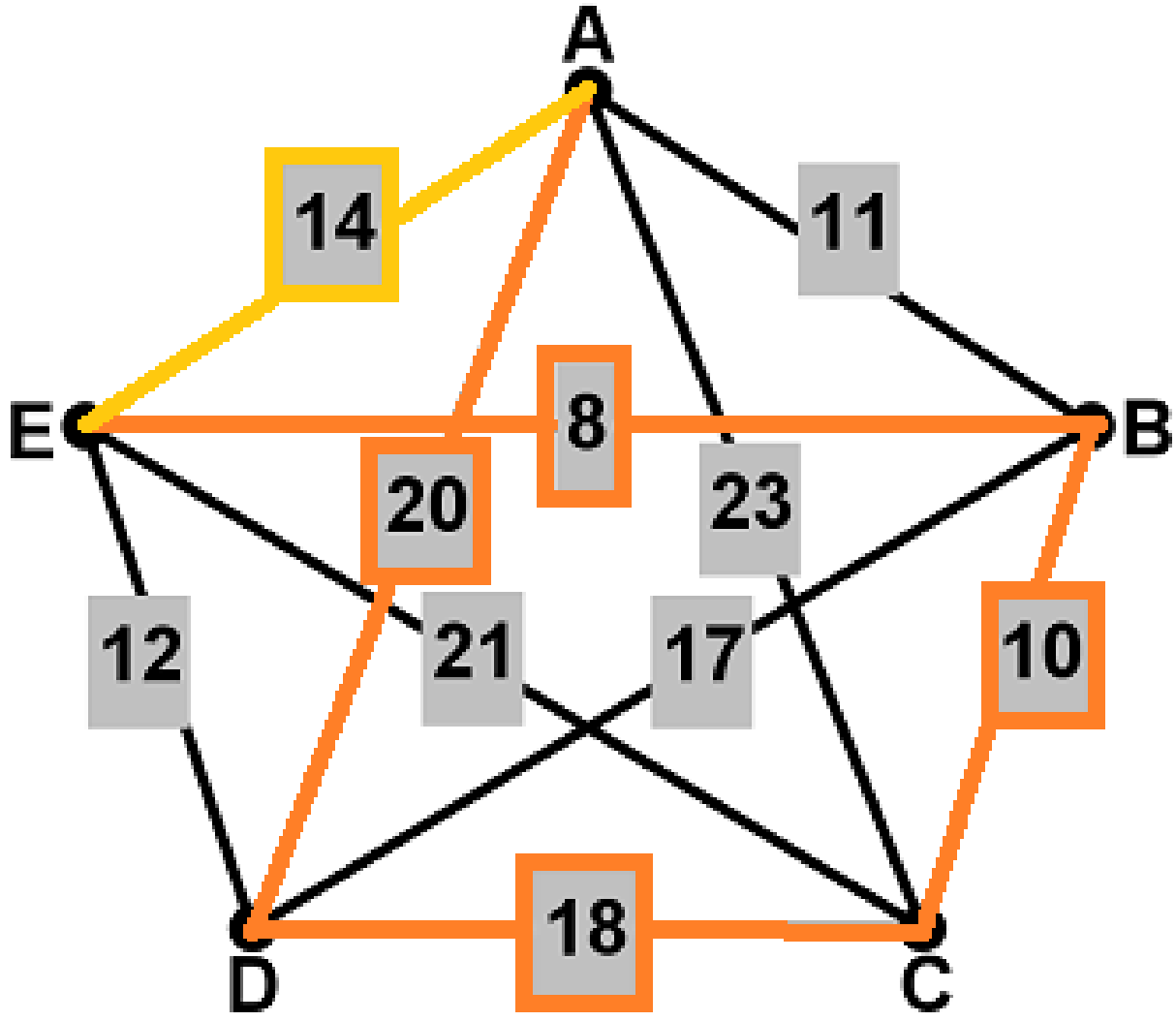
Type the next vertex.



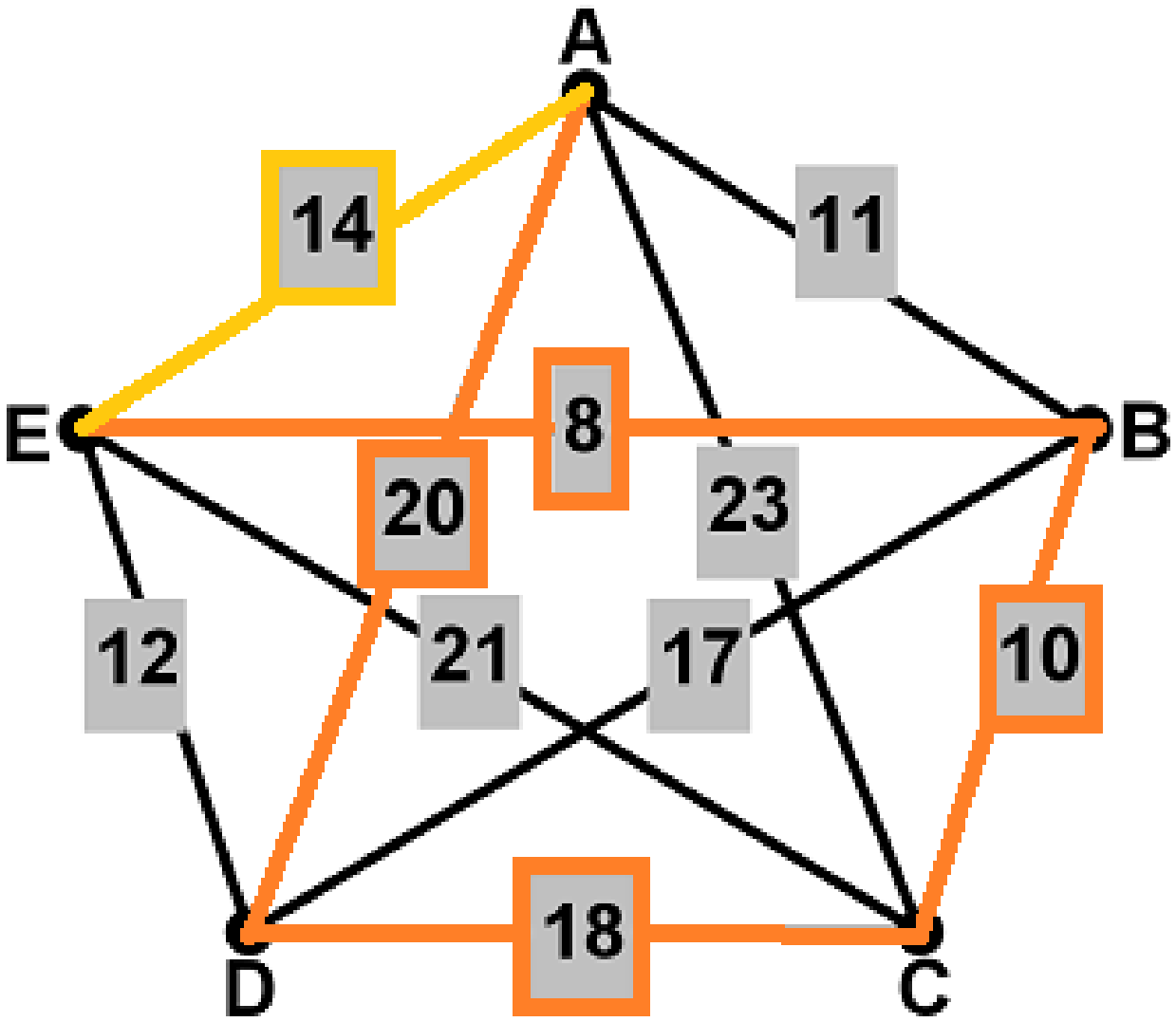
Return to Start! E, B, C, D, A, E

At A, take 14 edge back to E to finish the circuit.

Return ->



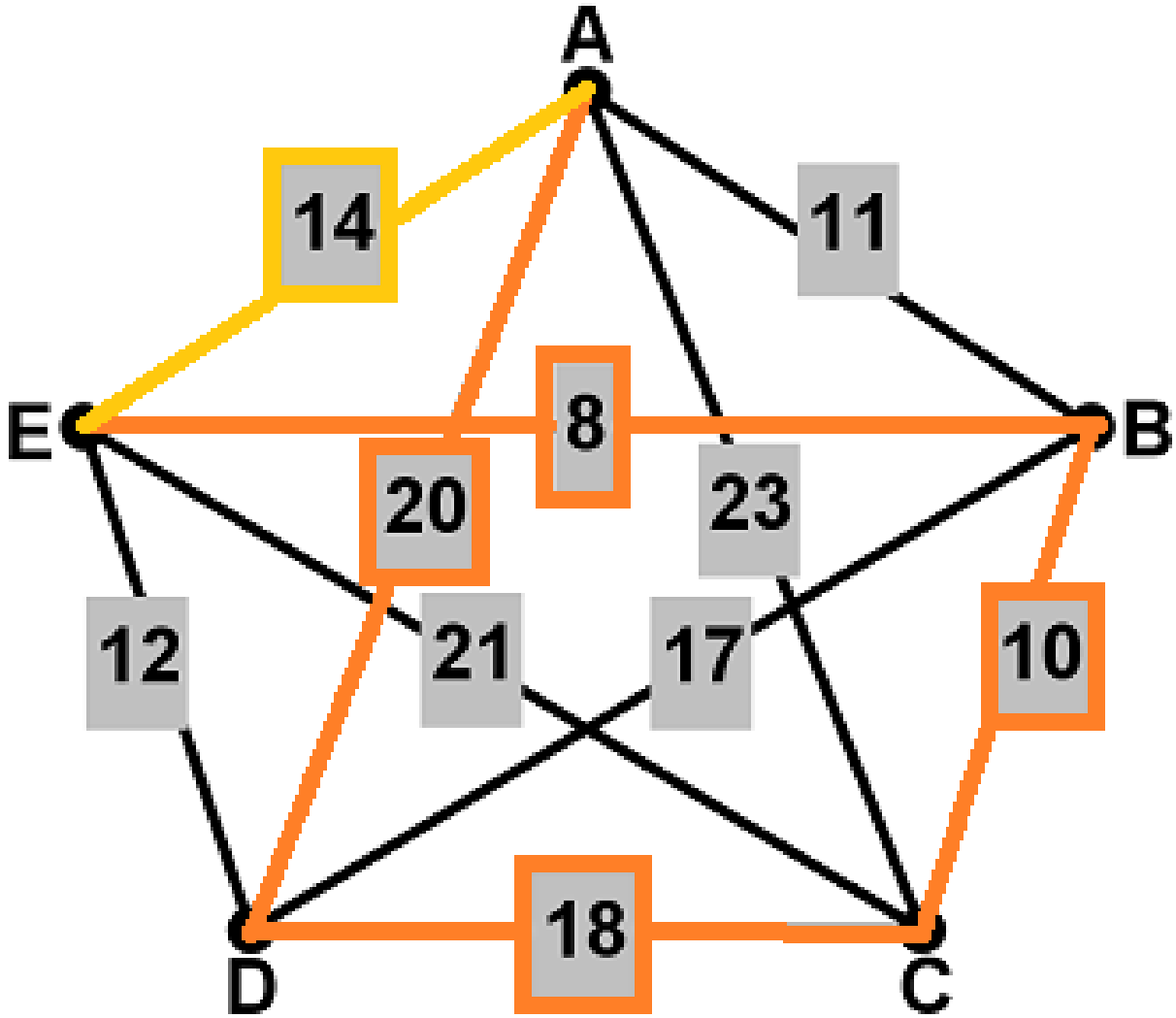
?(7.7) Total Weight



Type and send a **number**.

E, B, C, D, A, E

Total weight: $8 + 10 + 18 + 20 + 14 = 70$



?(7.8) Shortest Hamiltonian Circuit

Is E, B, C, D, A, E, the circuit Nearest Neighbor found, the shortest Hamiltonian circuit on this graph?

Yes, or **No**?

Shortest Hamiltonian Circuit

Is E, B, C, D, A, E, the circuit Nearest Neighbor found, the shortest Hamiltonian circuit on this graph?

No!

- E, B, C, D, A, E has total weight 70
- The one we found earlier, A, B, C, D, E, A, was shorter, with total weight 65.

Nearest Neighbor Algorithm Evaluation

The Nearest Neighbor Algorithm is:

- **Not Optimal:** It might not get the circuit with the lowest total weight
- **Efficient:** It gives an answer quickly

The Nearest Neighbor Algorithm is a **heuristic algorithm**. It is fast, and while its answer is not the best, it is still good.

Next Time

- Sorted Edges Algorithm