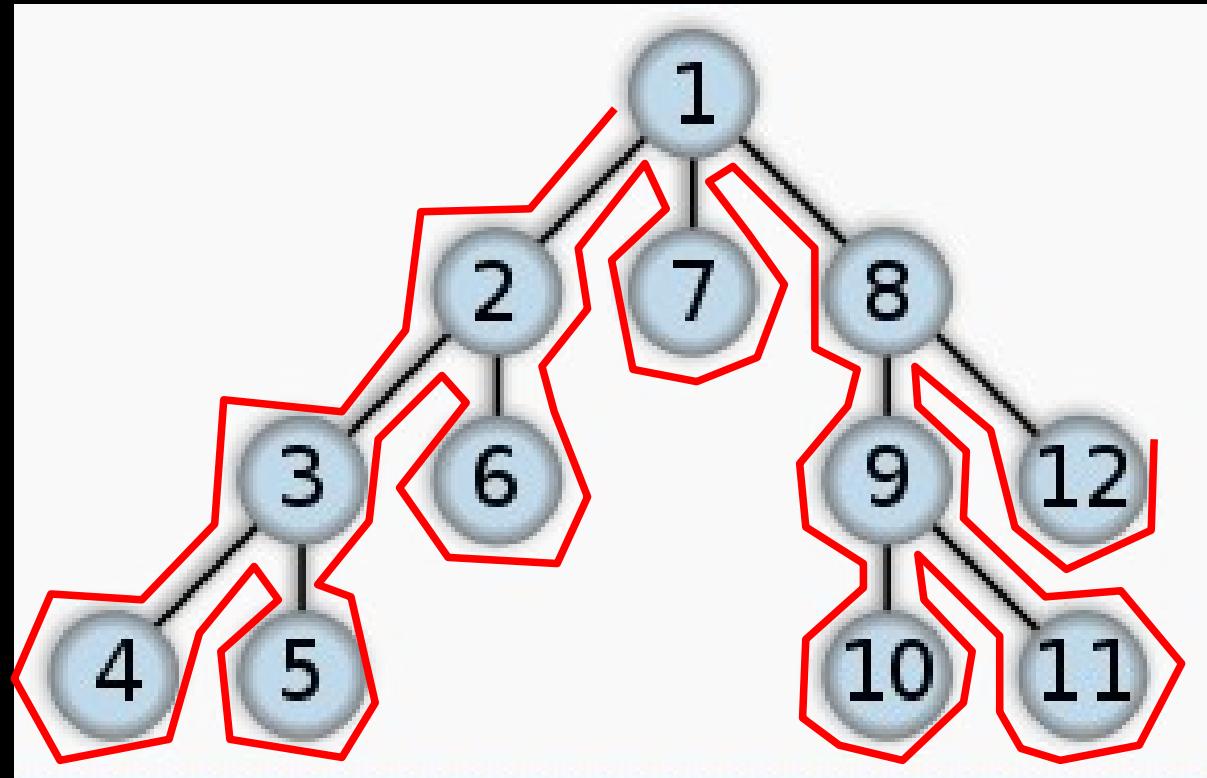


# Problem Solving

## *Deep-First Search*

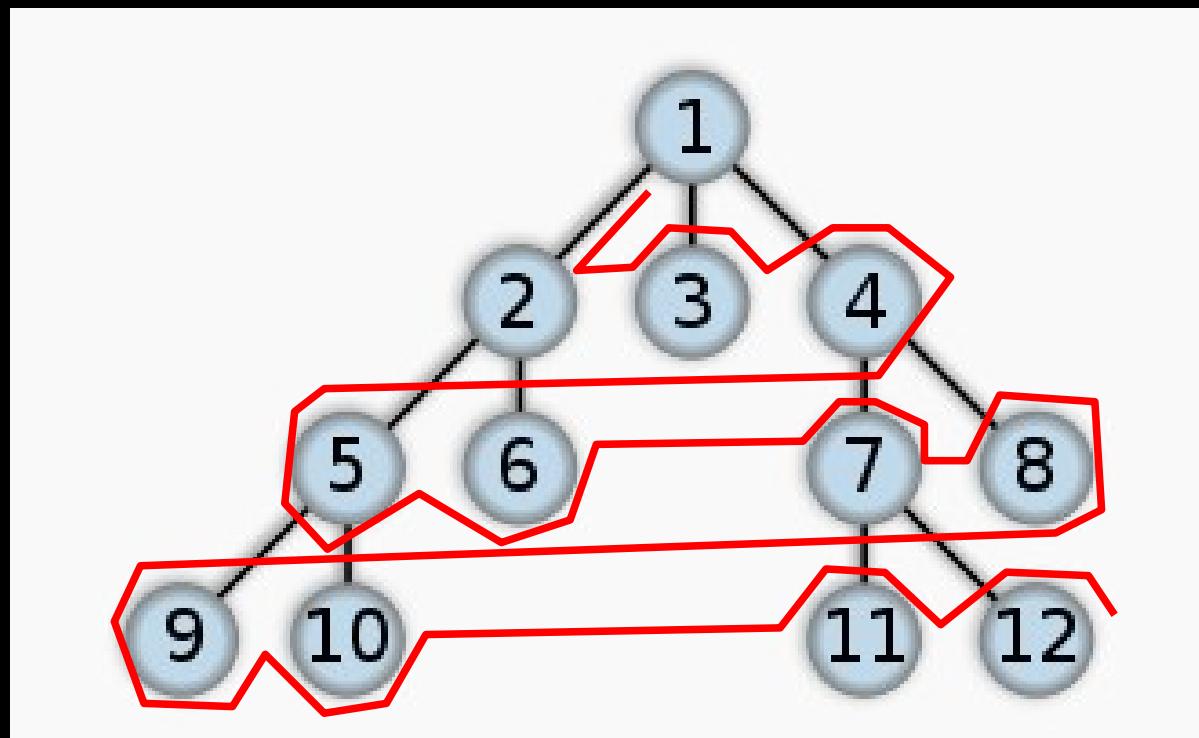
### "Cerca in Profondità"



# Problem Solving

## *Breadth-First Search*

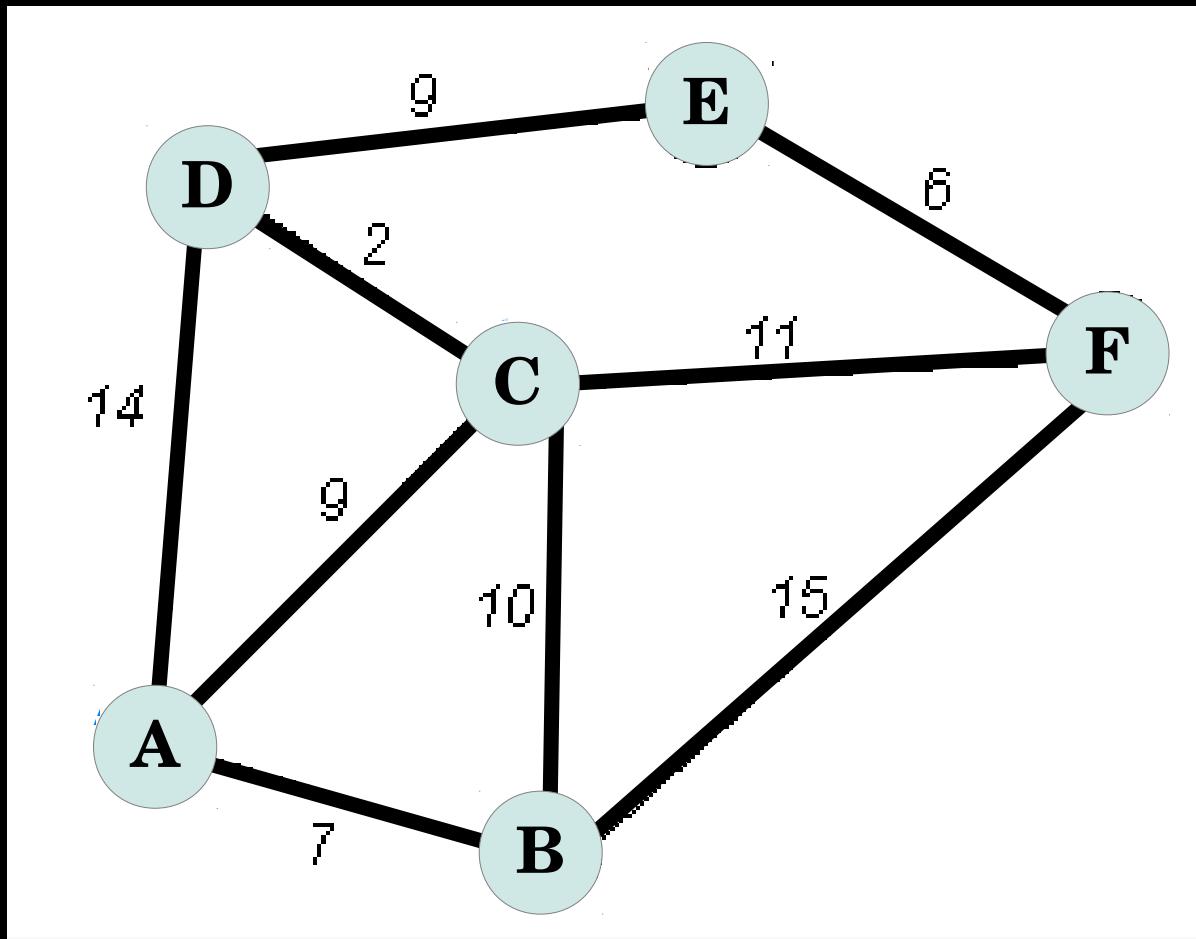
### "Cerca in Ampiezza"



# Problem Solving

## *Dijkstra's Algorithm*

"Cerca il percorso piu' breve A-F"



# Heuristic Character of Signs of Progress

When we are working intensely  
we watch eagerly for signs of progress ...

as Columbus and his companions watched for  
SIGNS of APPROACHING LAND  
on Thursday 11th October 1492...



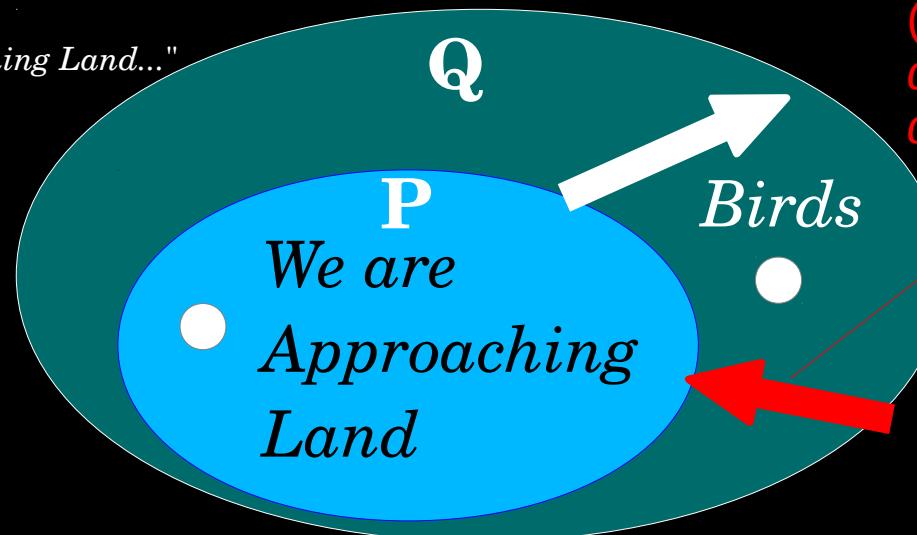
*"If we are approaching land, we often see birds  
Now we see birds.  
Therefore, PROBABLY, we are approaching land..."*

# Heuristic Character of Signs of Progress

"If we are Approaching Land, we Often see Birds"

Now we see Birds.

Therefore, PROBABLY, we are Approaching Land..."



IF WE SEE BIRDS  
ARE WE APPROACHING  
LAND???

("Fallacia  
dell'affermazione  
del conseguente")

Heuristic Reasoning  
Inductive Reasoning  
Plausible Reasoning

Demonstrative Reasoning

$(P \rightarrow Q)$  (if P is true then Q is true)  
 $(\bar{Q} \rightarrow \bar{P})$  (if Q is false then P is false)

$(P \rightarrow Q)$  (if P is true then Q is true)  
 $(Q \rightarrow P?)$  (if Q is true then P is  
**MORE CREDIBLE**)

Demonstrative Syllogism

Heuristic Syllogism

If you take a Heuristic Conclusion as Certain you may be Fooled and Disappointed;  
BUT if you Neglet Heuristic Conclusions altogether you will make NO PROGRESS at all.

**THE MOST IMPORTANT SIGNS OF PROGRESS ARE HEURISTIC**

# Heuristic Character of Signs of Progress

*"L'istinto e' una cosa meravigliosa.  
Non puo' essere spiegato,  
né dev'essere ignorato."*

Agatha Christie

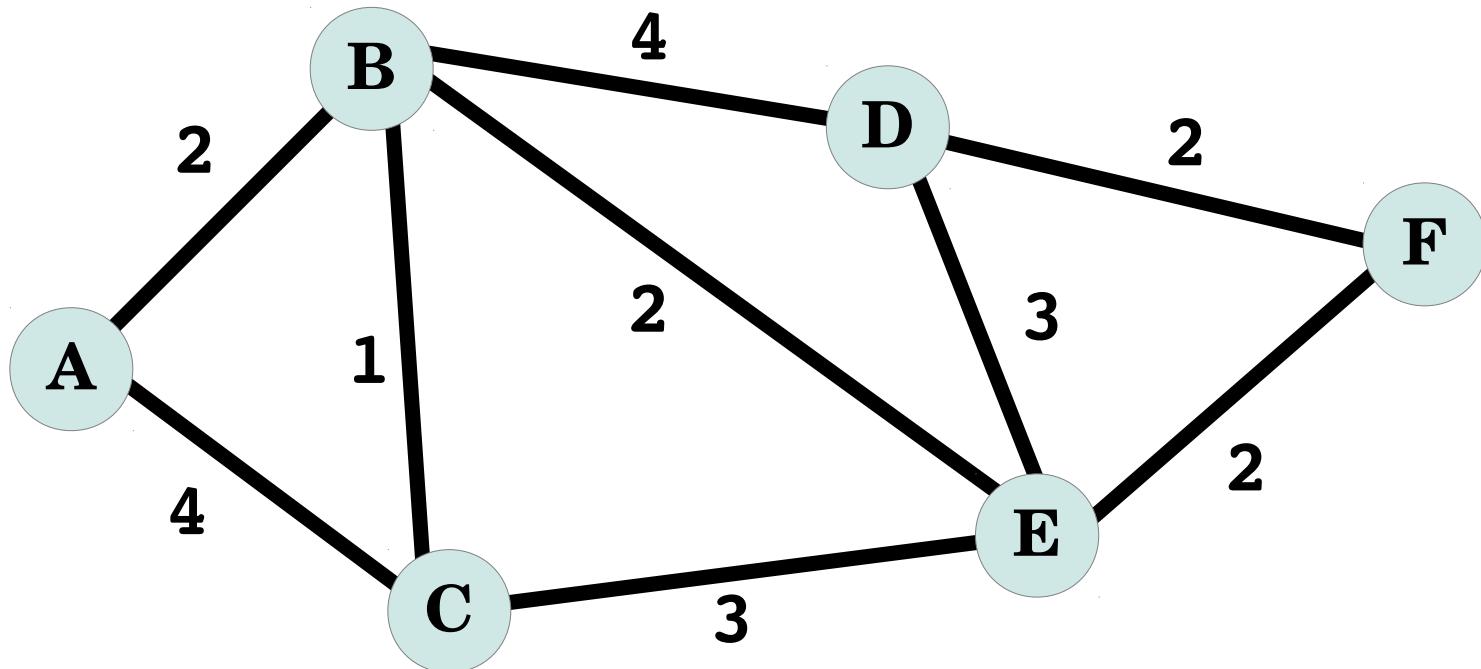
*"Un indizio e' un indizio,  
due indizi sono una coincidenza,  
ma tre indizi fanno una prova."*

Agatha Christie

## **Funzione Euristica**

*In ogni momento ci dice se  
stiamo andando  
nella "giusta" direzione  
(es. bussola)*

# Problem Solving

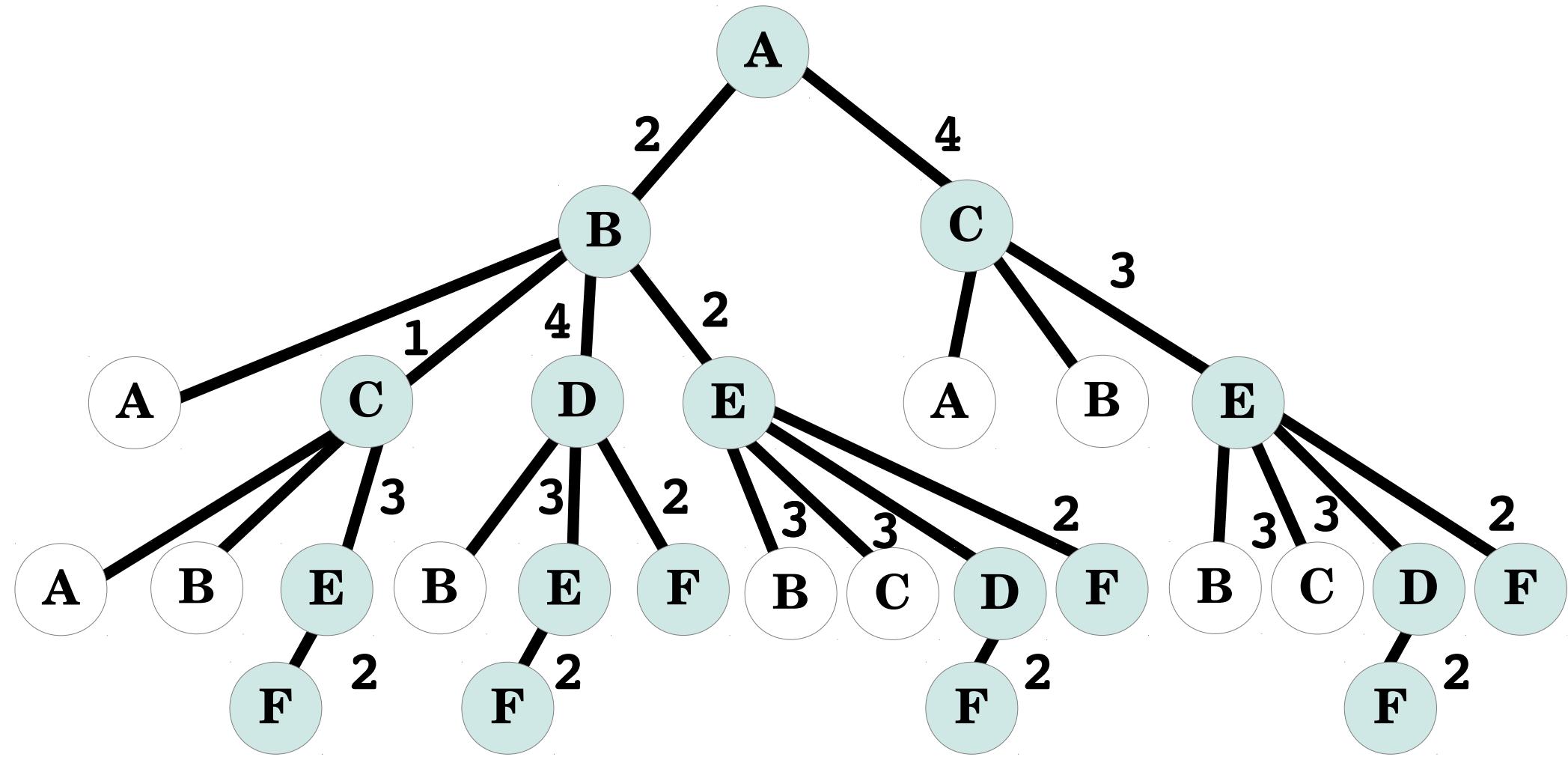


Trovare il percorso a *energia minima* da A a F

# Problem Solving

## *Deep-First Search*

### "Cerca in Profondità"



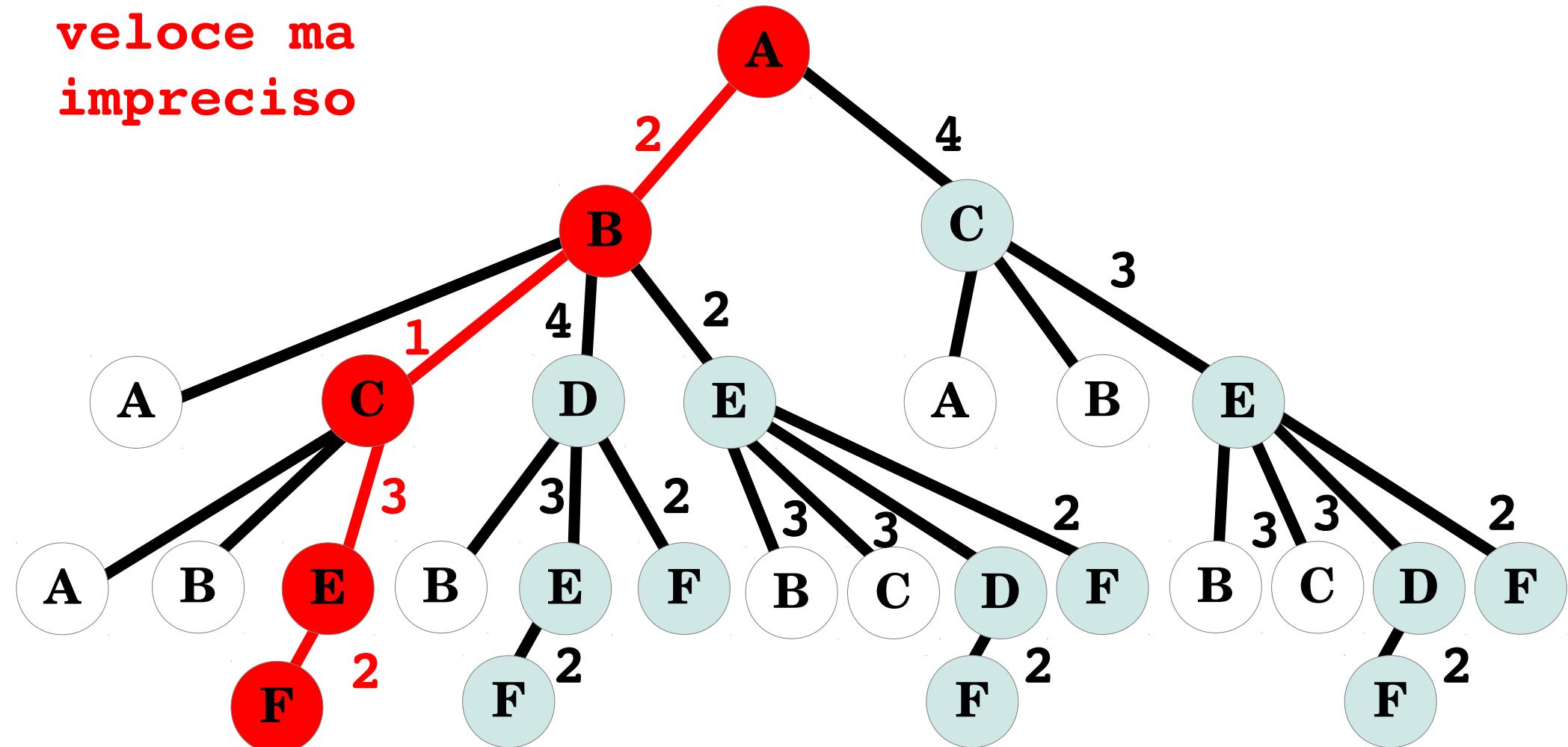
# Problem Solving

## *Deep-First Search*

### "Cerca in Profondità"

$$ABCEF = 2 + 1 + 3 + 2 = 8$$

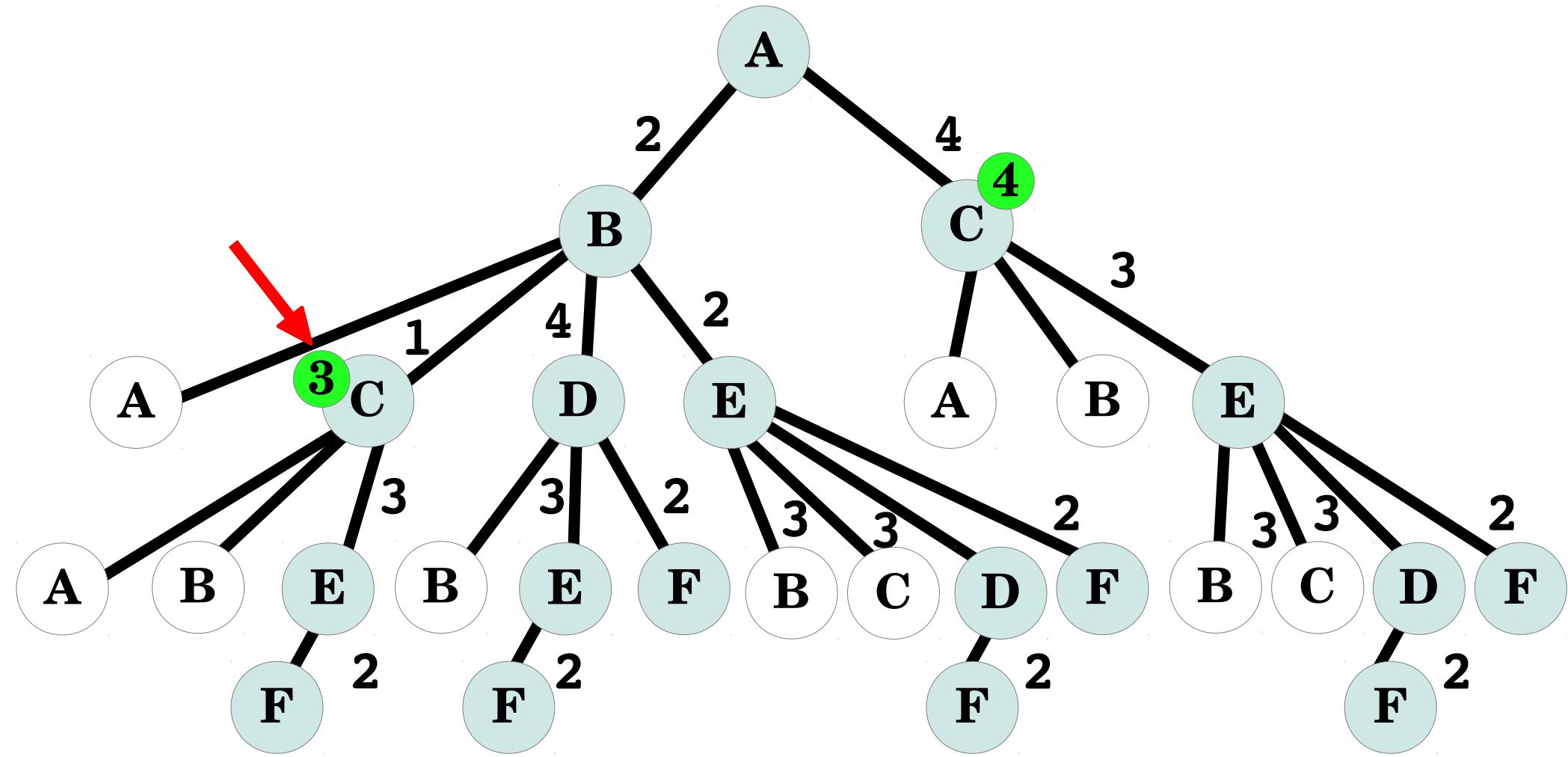
veloce ma  
impreciso



# Problem Solving

## *Breadth-First Search*

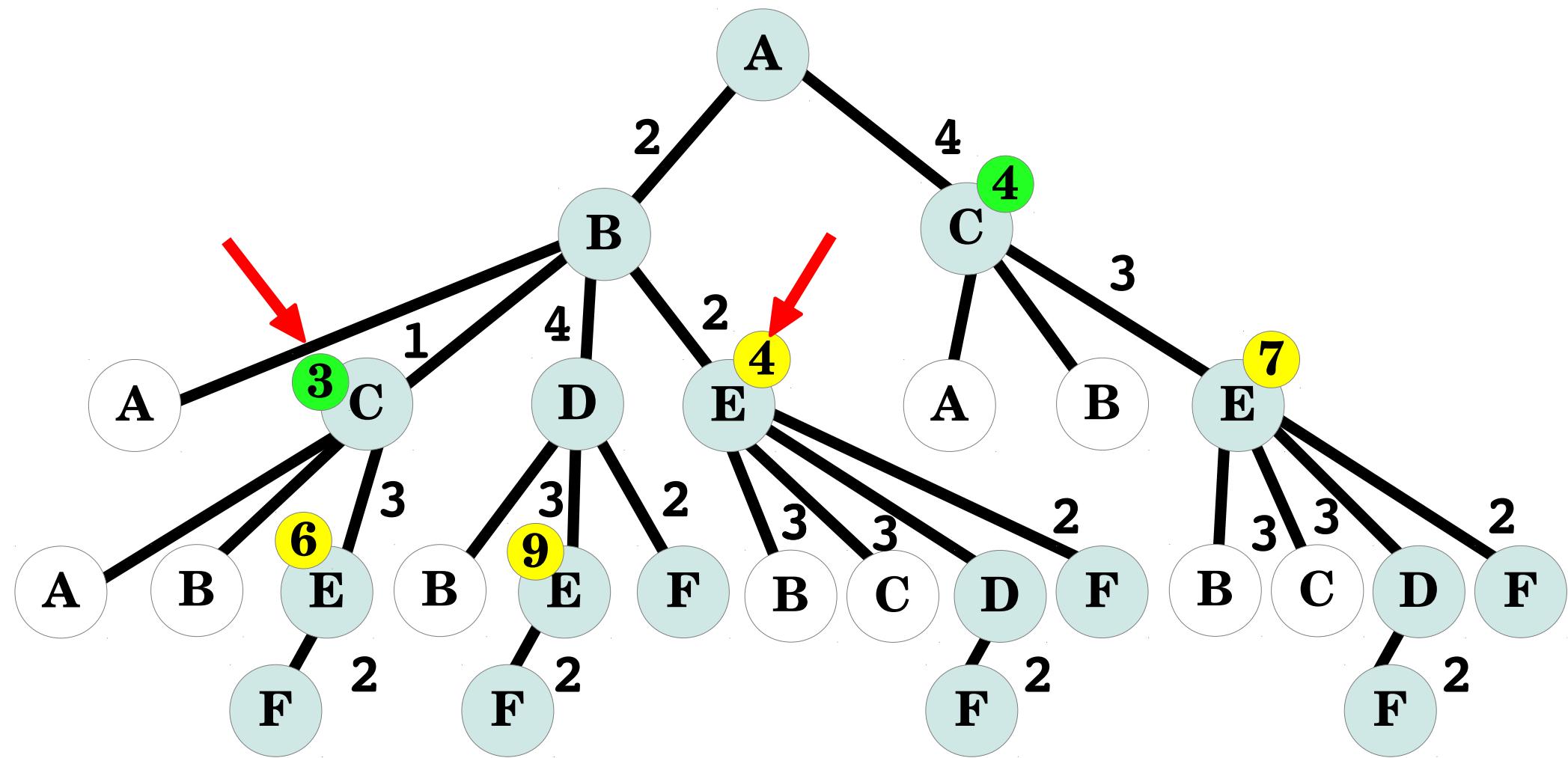
### "Cerca in Ampiezza"



# Problem Solving

## Breadth-First Search

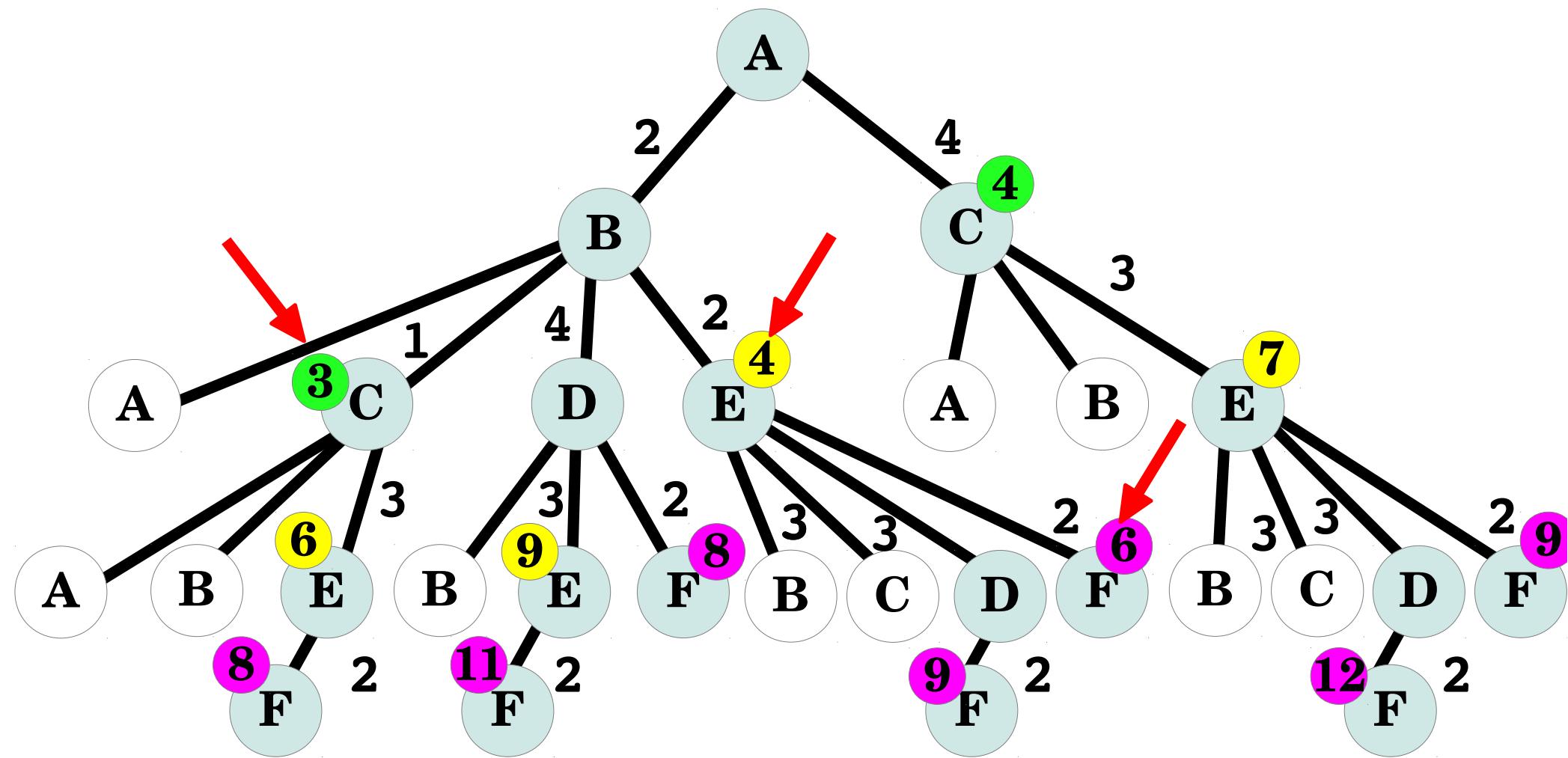
### "Cerca in Ampiezza"



# Problem Solving

## Breadth-First Search

### "Cerca in Ampiezza"



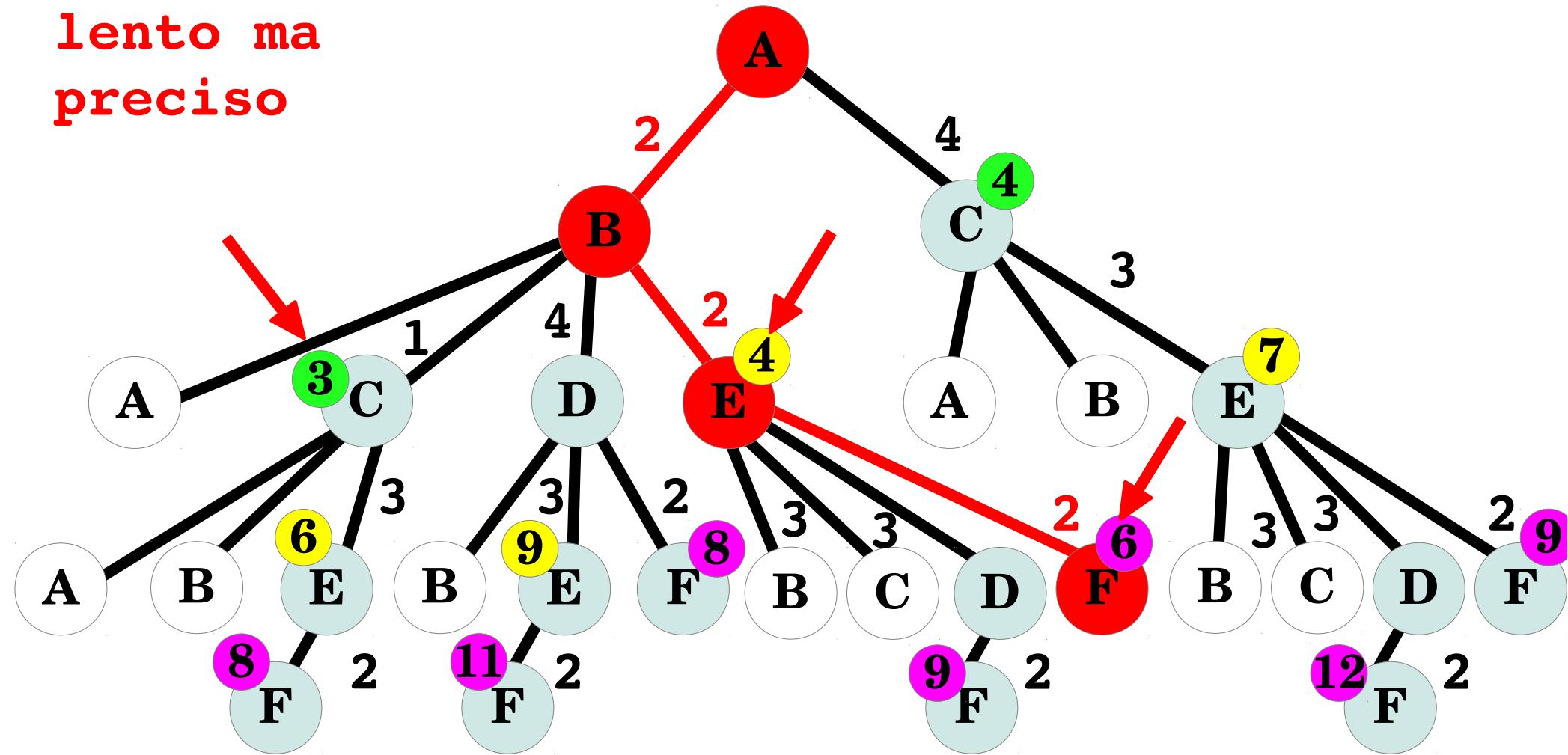
# Problem Solving

## Breadth-First Search

### "Cerca in Ampiezza"

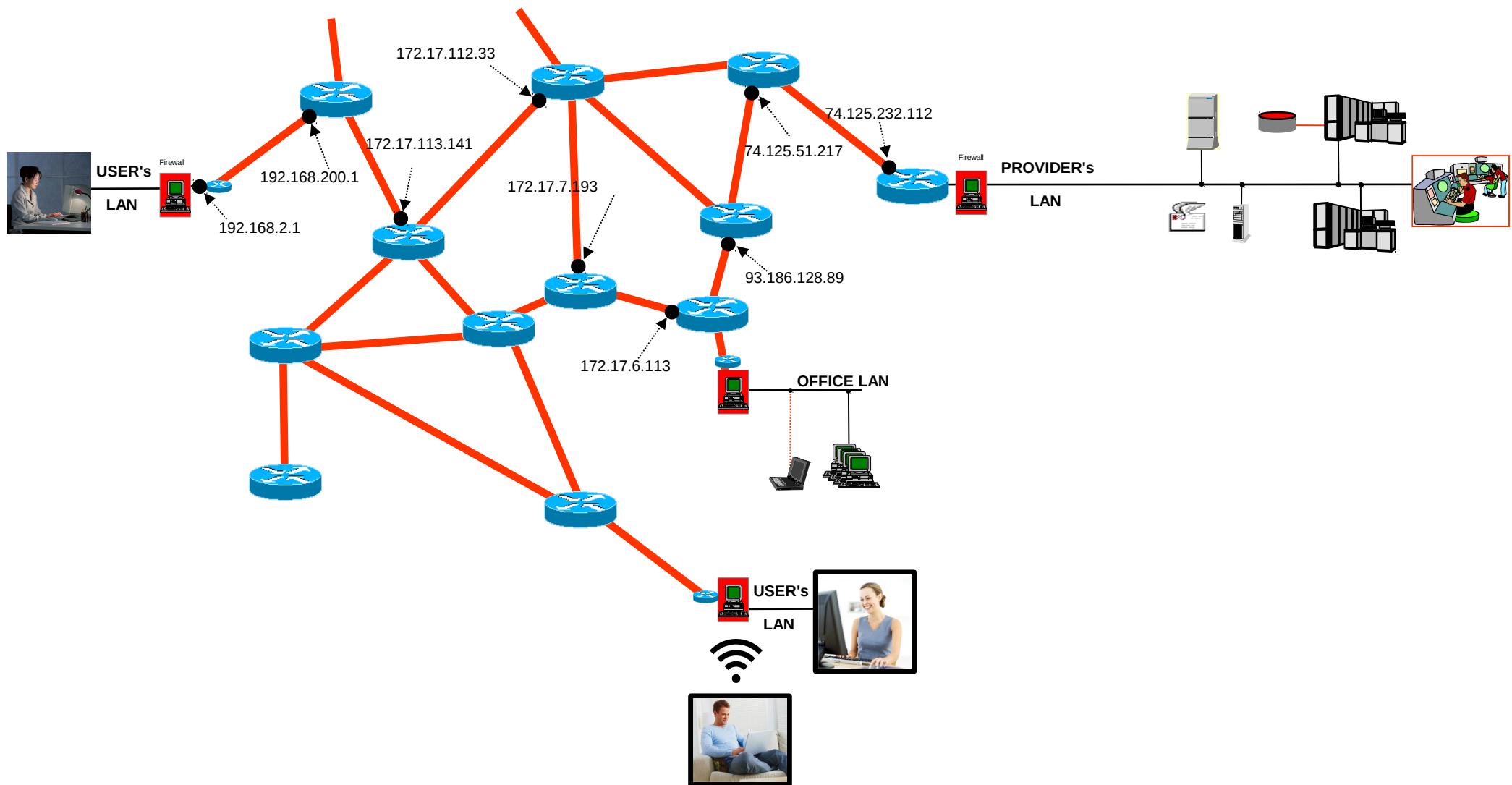
$$ABEF = 2+2+2=6$$

lento ma  
preciso



# Internet

**Nel mondo reale tracciare l'albero di tutti i percorsi diventa molto difficile...**



# Problem Solving

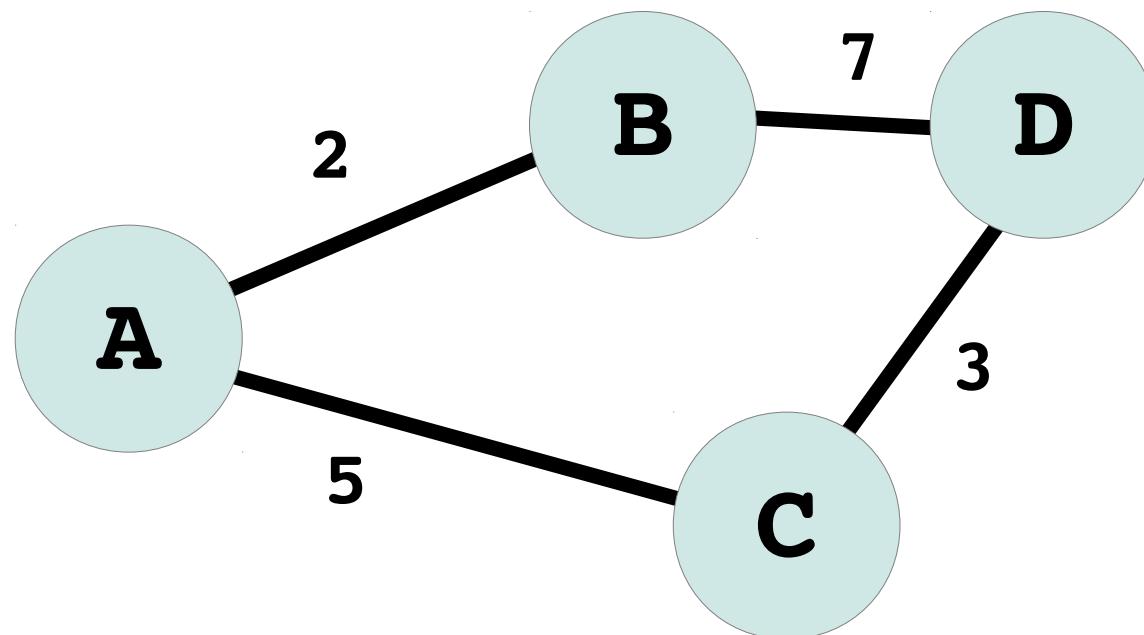
## 1959: *Dijkstra's Algorithm*

Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.



# Problem Solving

## 1959: Dijkstra's Algorithm

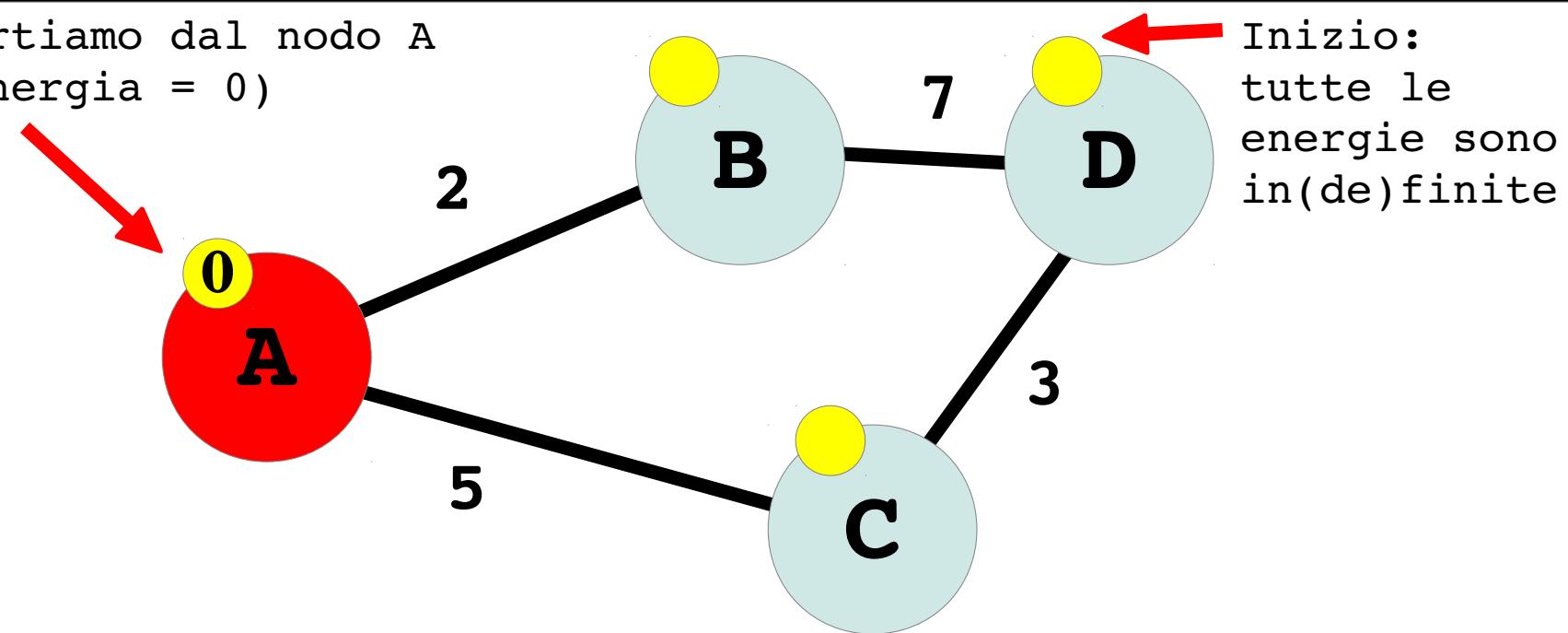
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Partiamo dal nodo A  
(energia = 0)



# Problem Solving

## 1959: Dijkstra's Algorithm

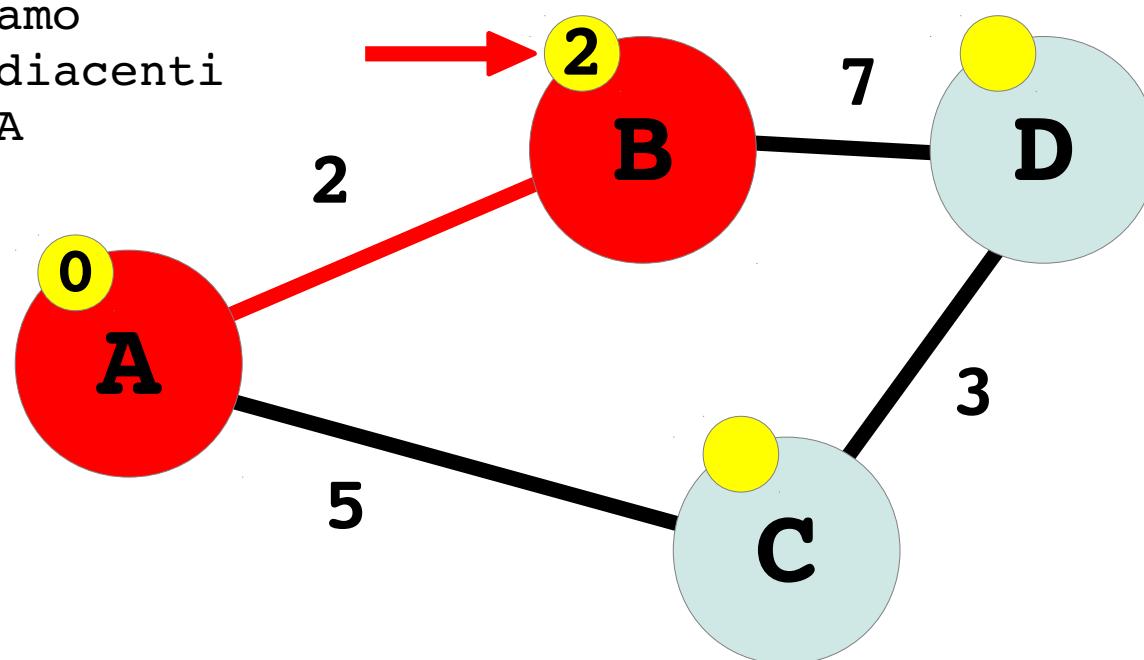
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Analizziamo  
i nodi adiacenti  
al nodo A  
(B,C)



# Problem Solving

## 1959: Dijkstra's Algorithm

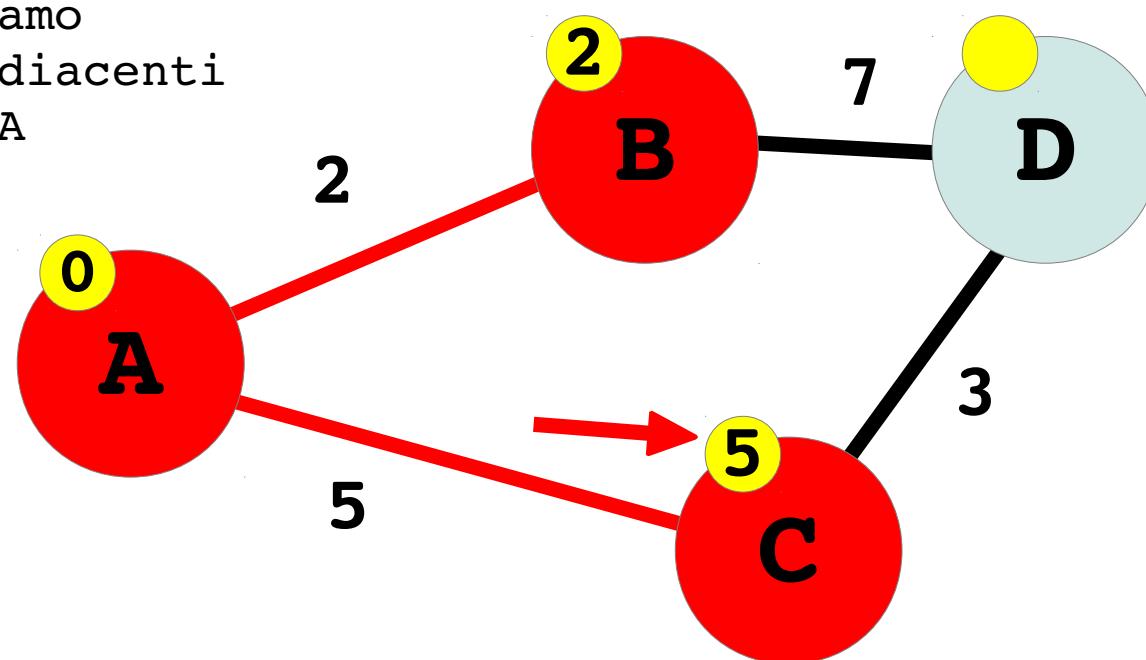
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Analizziamo  
i nodi adiacenti  
al nodo A  
(B,C)



# Problem Solving

## 1959: Dijkstra's Algorithm

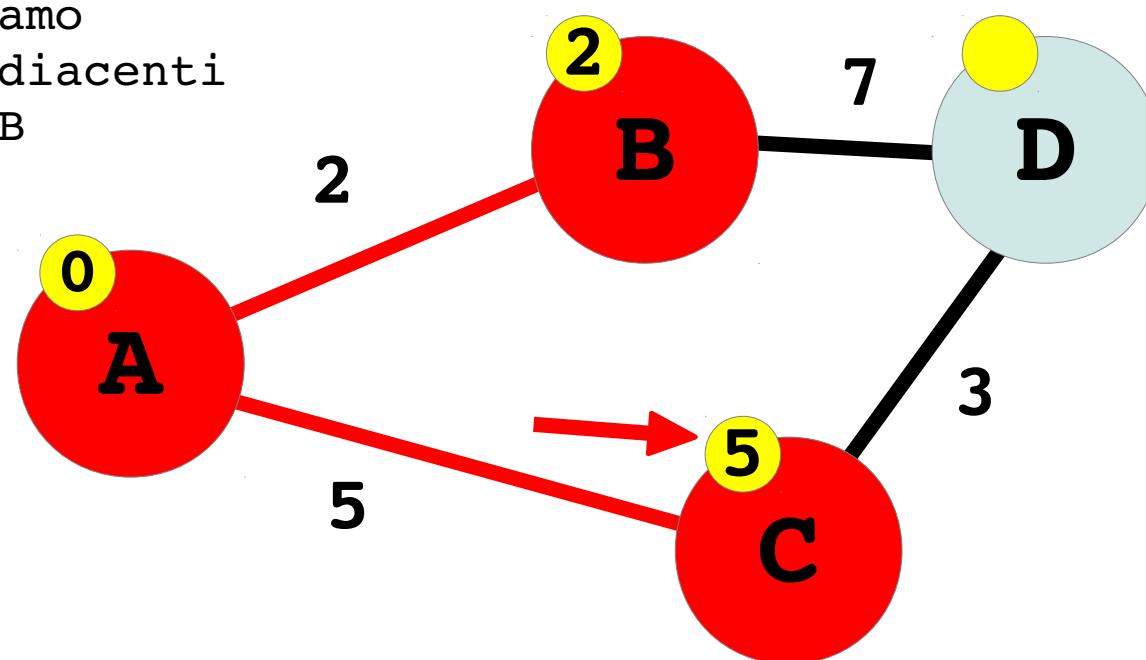
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Analizziamo  
i nodi adiacenti  
al nodo B  
(D)



# Problem Solving

## 1959: Dijkstra's Algorithm

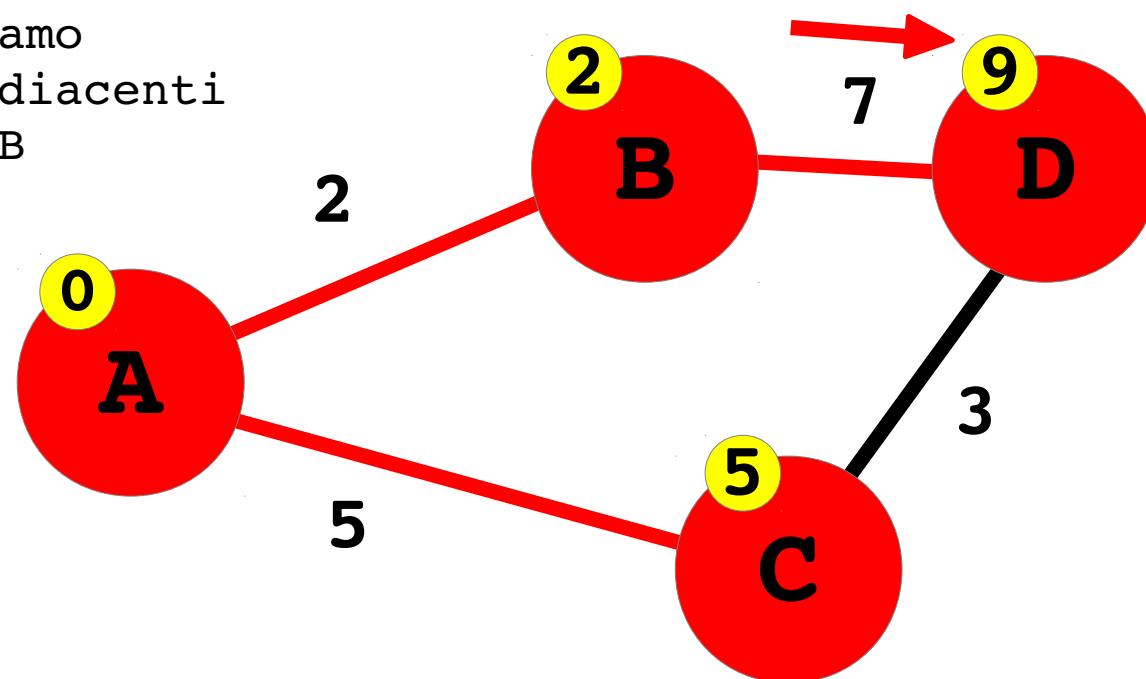
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Analizziamo  
i nodi adiacenti  
al nodo B  
(D)



# Problem Solving

## 1959: Dijkstra's Algorithm

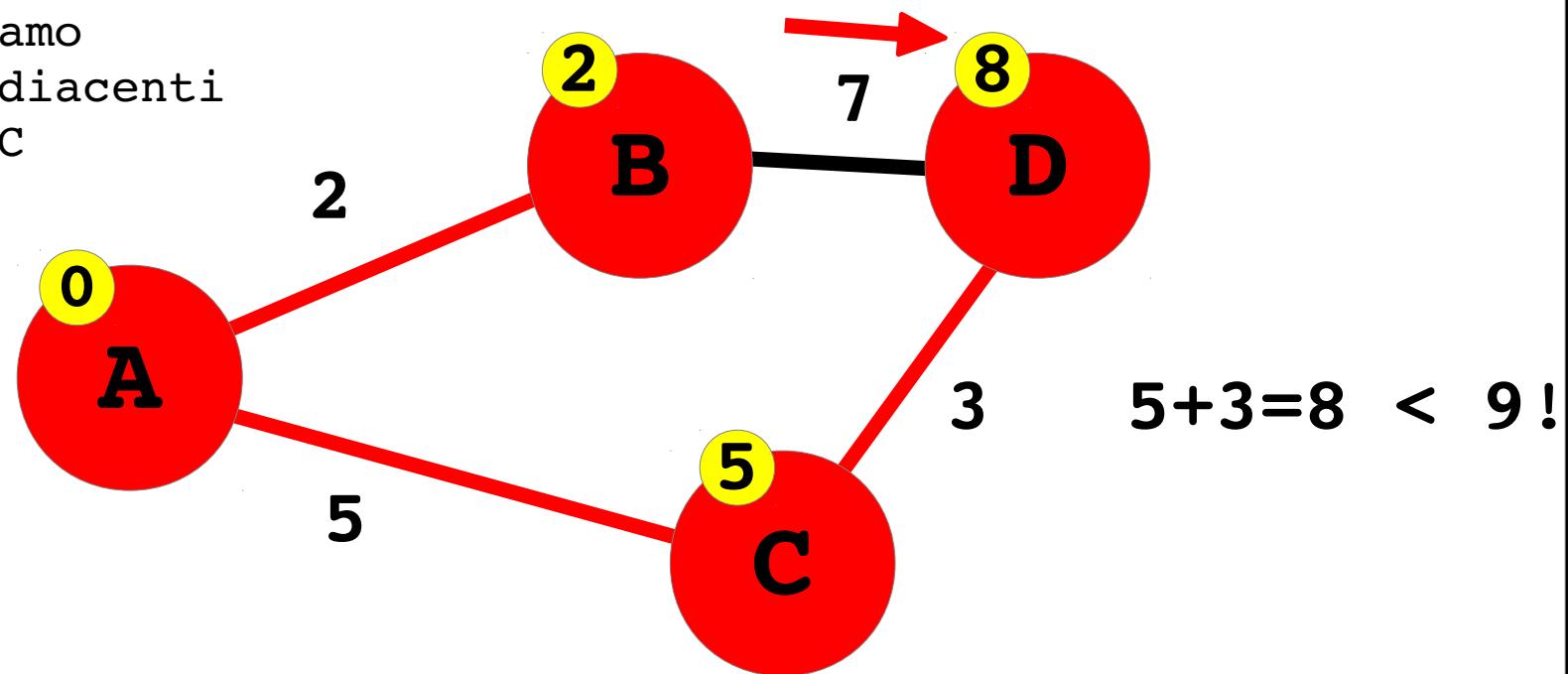
Ogni nodo memorizza l'energia minima necessaria per raggiungerlo.

Da ogni nodo si calcola l'energia minima per arrivare a tutti i suoi nodi adiacenti.

L'energia per arrivare da un nodo X a un nodo Y e' data dalla somma di  
- energia per arrivare al nodo X di partenza (memorizzata nel nodo X)  
- energia per arrivare al nodo Y di arrivo (arco di connessione)

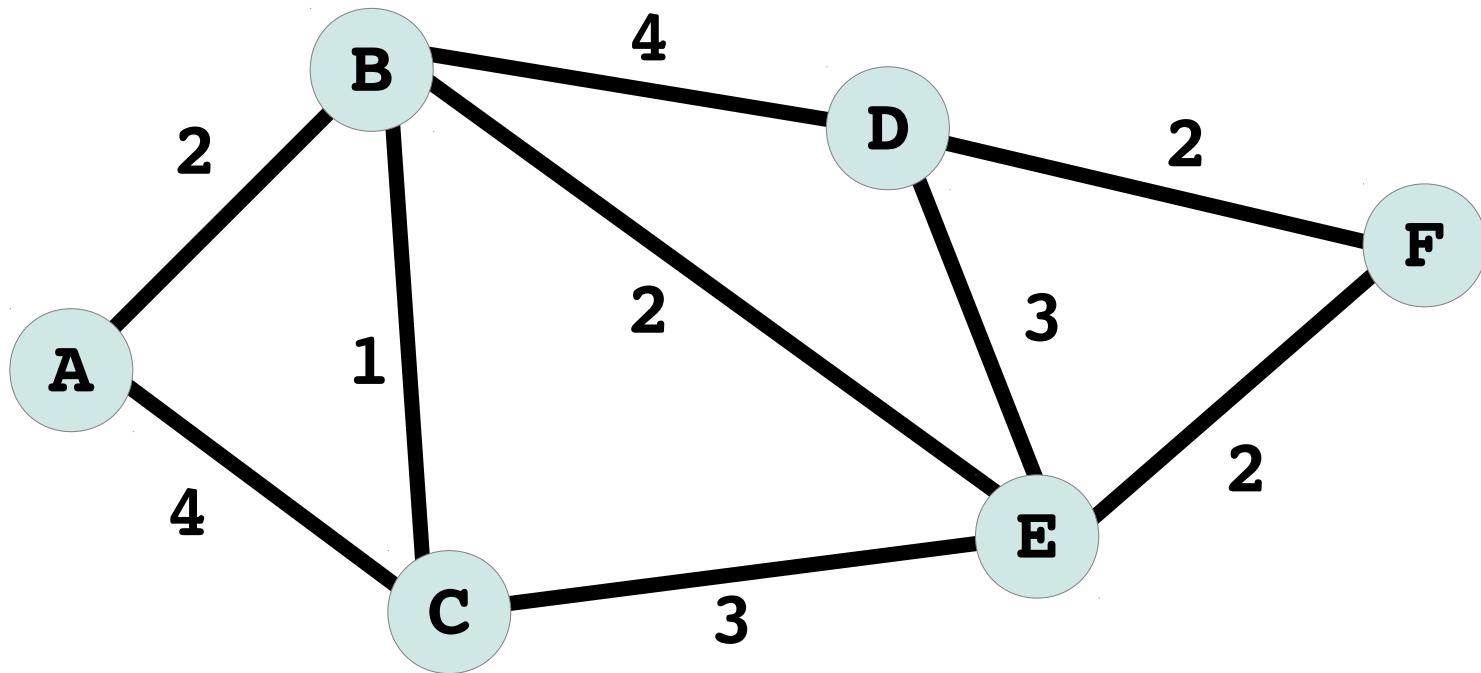
Se la somma e' inferiore a quella gia' memorizzata nel nodo Y di arrivo  
allora si memorizza la nuova energia minima e il nuovo percorso.

Analizziamo  
i nodi adiacenti  
al nodo C  
(D)



# Problem Solving

## 1959: *Dijkstra's Algorithm*

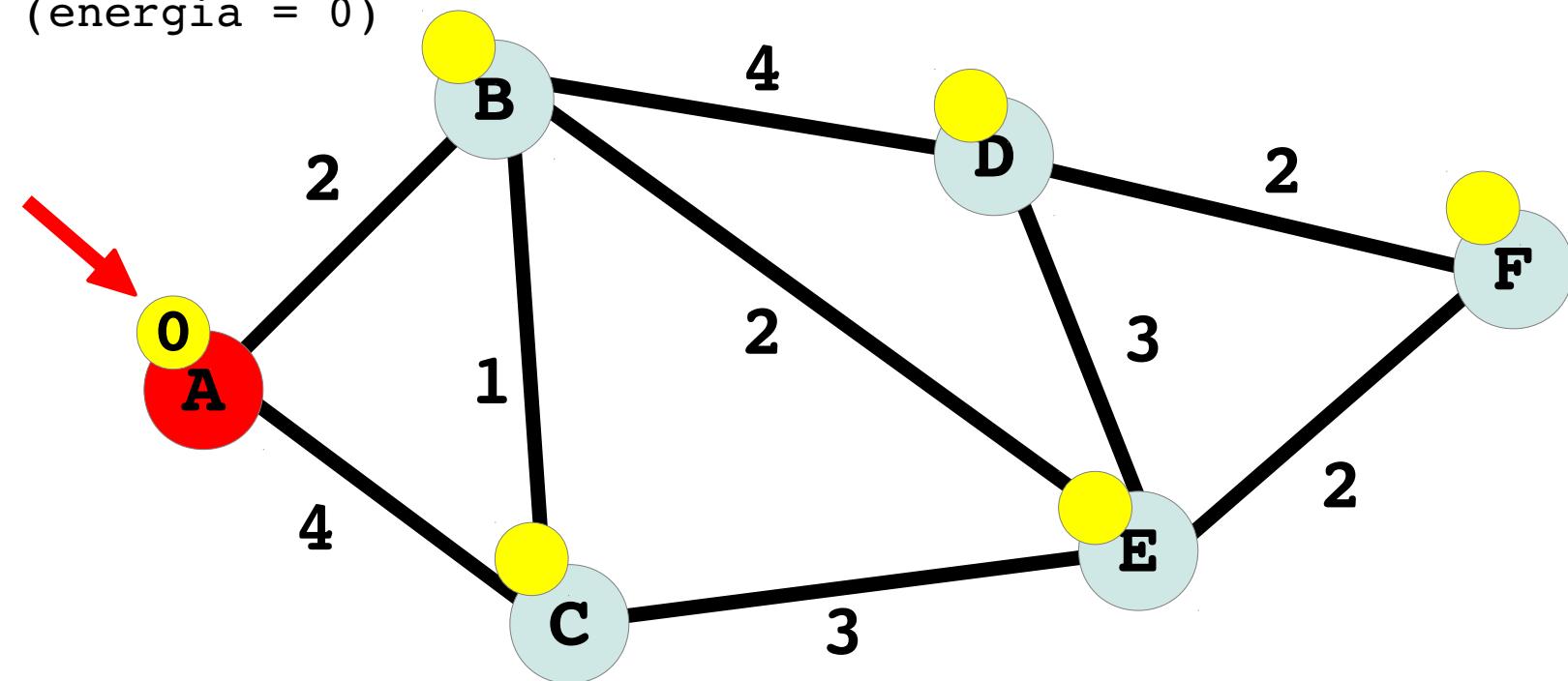


Trovare il percorso a *energia minima* da A a F

# Problem Solving

## 1959: Dijkstra's Algorithm

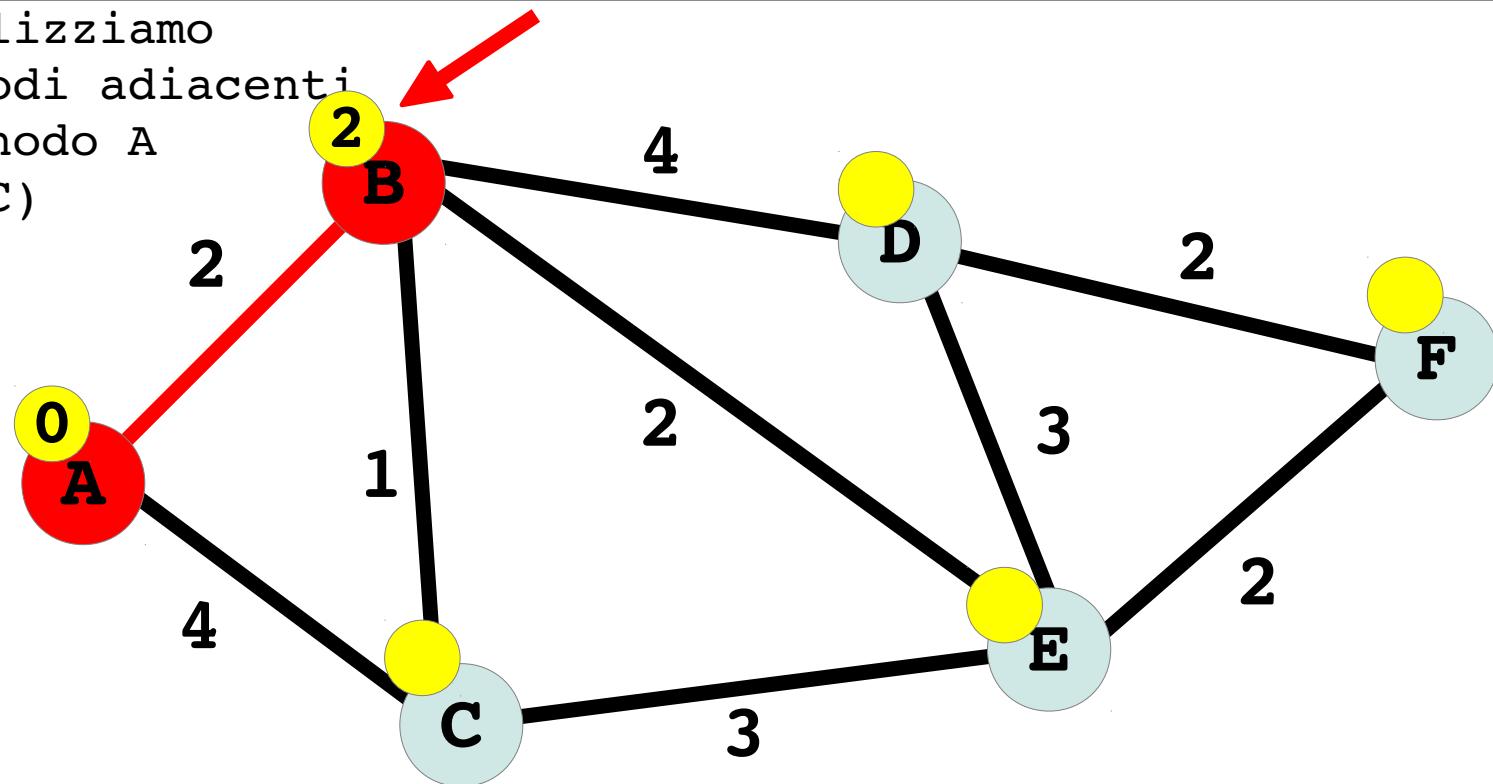
Partiamo dal nodo A  
(energia = 0)



# Problem Solving

## 1959: *Dijkstra's Algorithm*

Analizziamo  
i nodi adiacenti  
al nodo A  
(B, C)



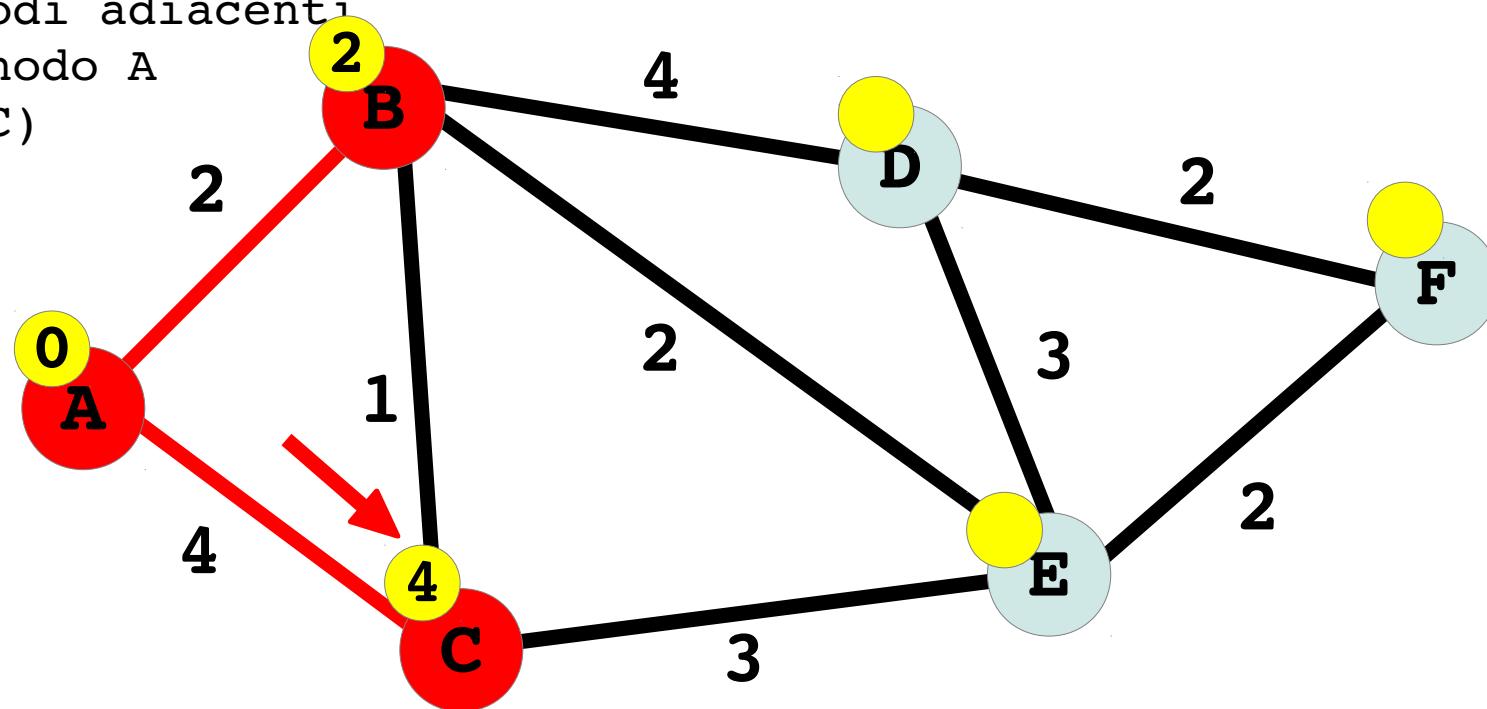
B:  $2 < \text{in(de)finito}$  ;  
B memorizza 2 e si aggiorna percorso

# Problem Solving

## 1959: *Dijkstra's Algorithm*

Analizziamo

i nodi adiacenti  
al nodo A  
(B, C)



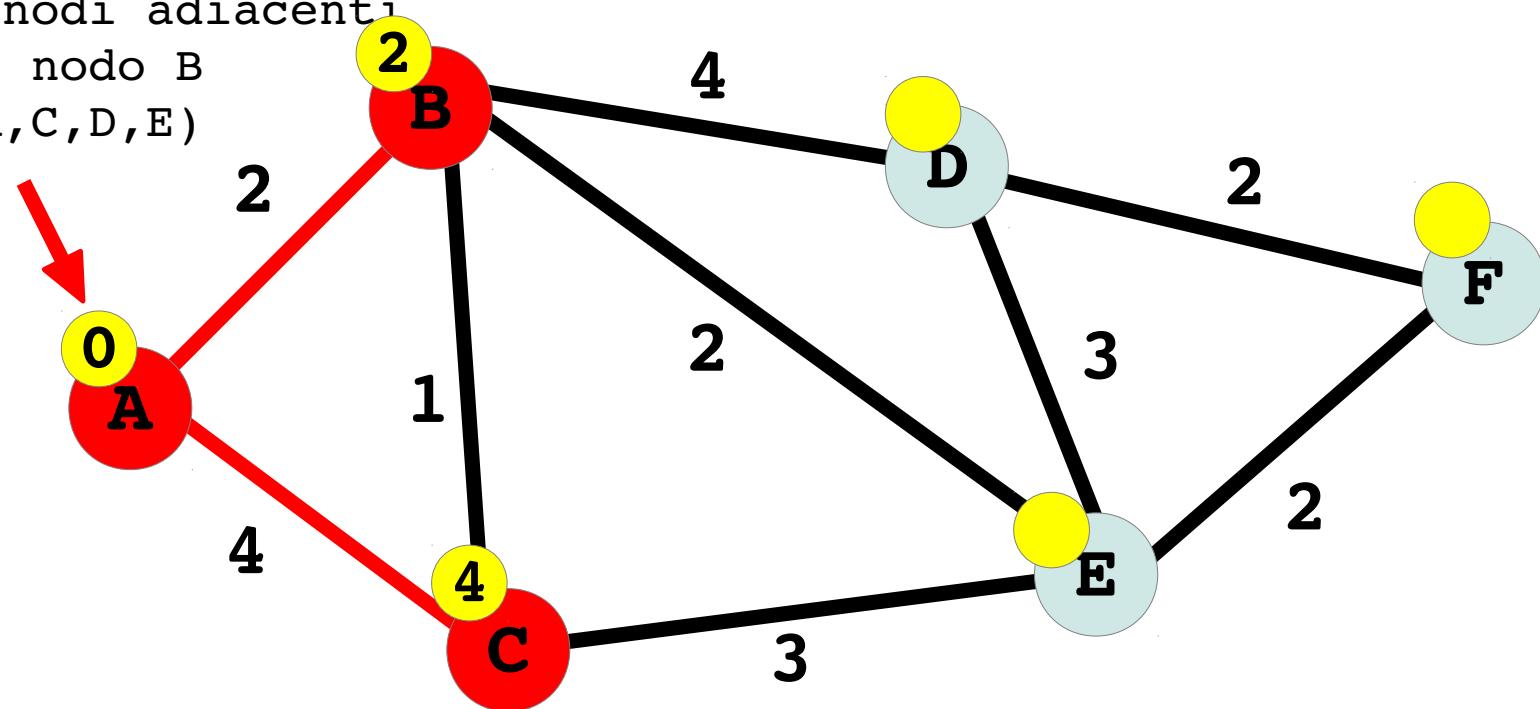
C:  $4 < \text{in(de)finito}$  ;  
C memorizza 4 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo B  
(A, C, D, E)



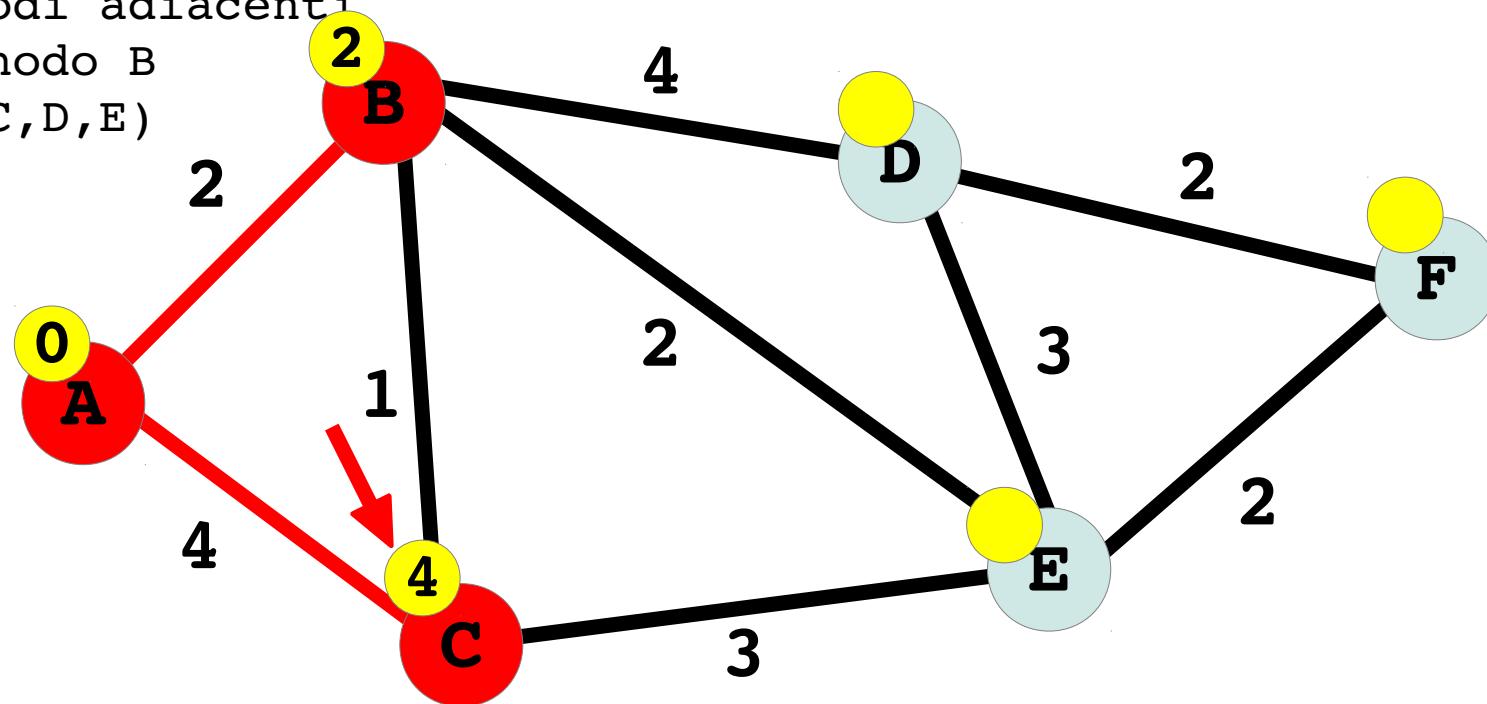
A:  $2+2=4 > 0$  ;  
A invariato

# Problem Solving

## 1959: *Dijkstra's Algorithm*

Analizziamo

i nodi adiacenti  
al nodo B  
(A, C, D, E)



$$C: 2+1=3 < 4 ;$$

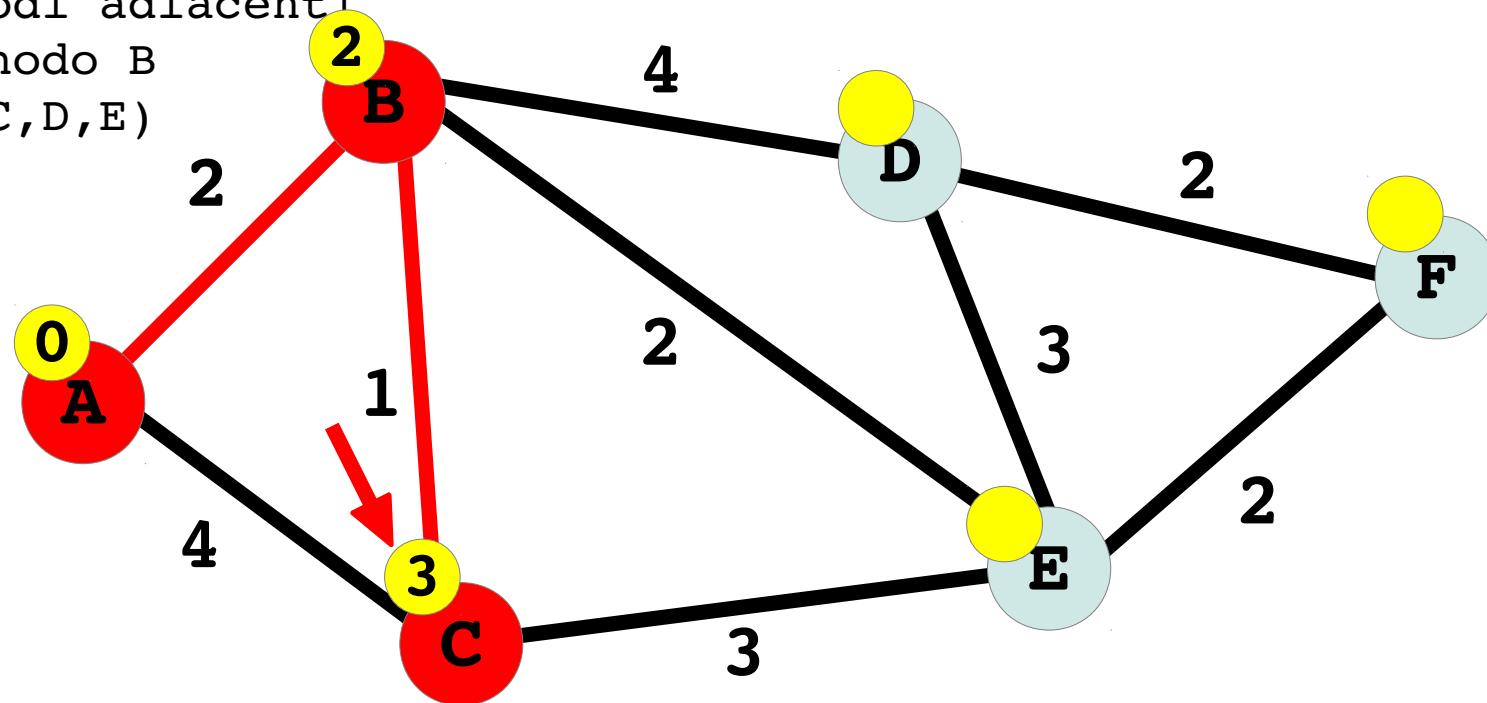
C memorizza 3 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo B  
(A, C, D, E)



$$C: 2+1=3 < 4 ;$$

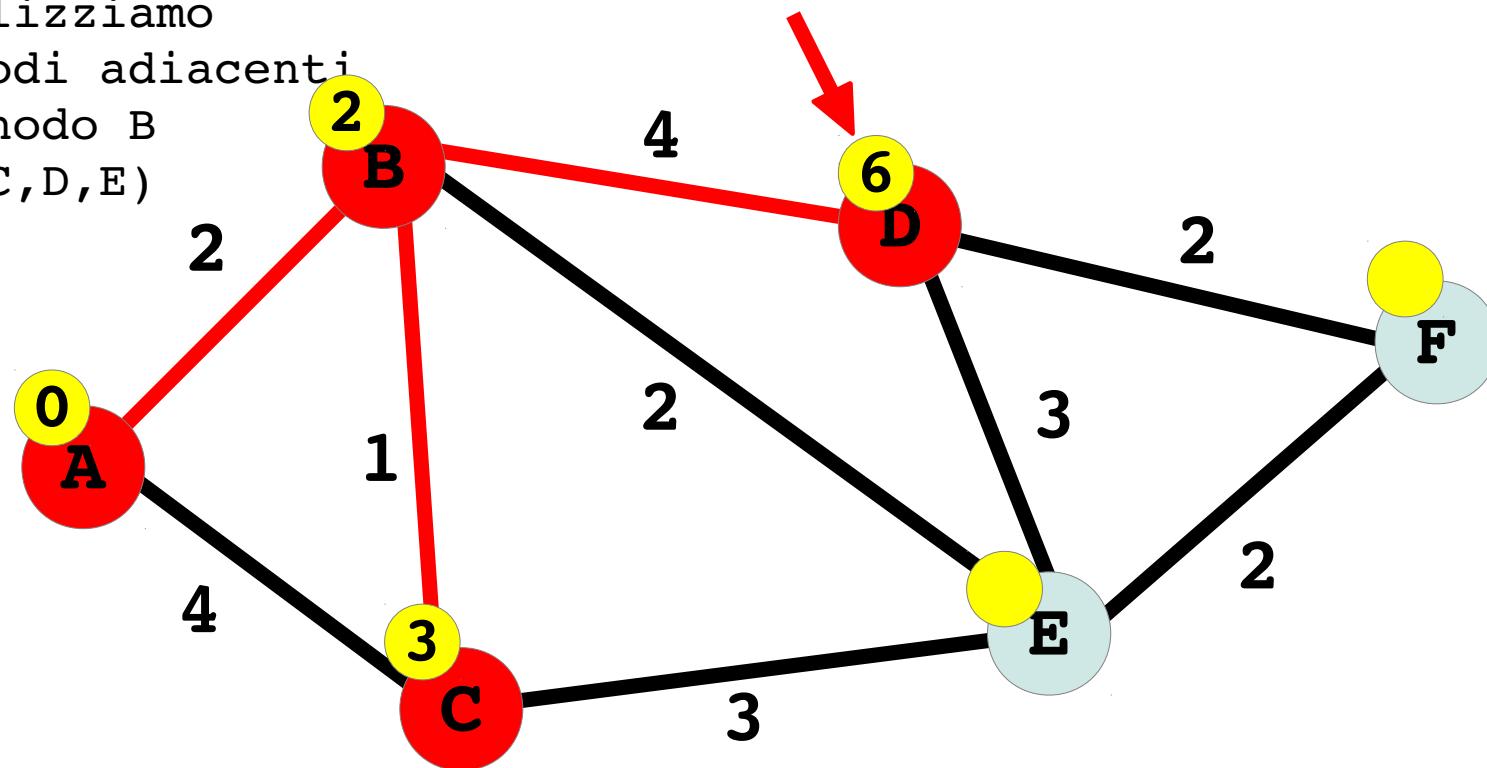
C memorizza 3 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo B  
(A, C, D, E)



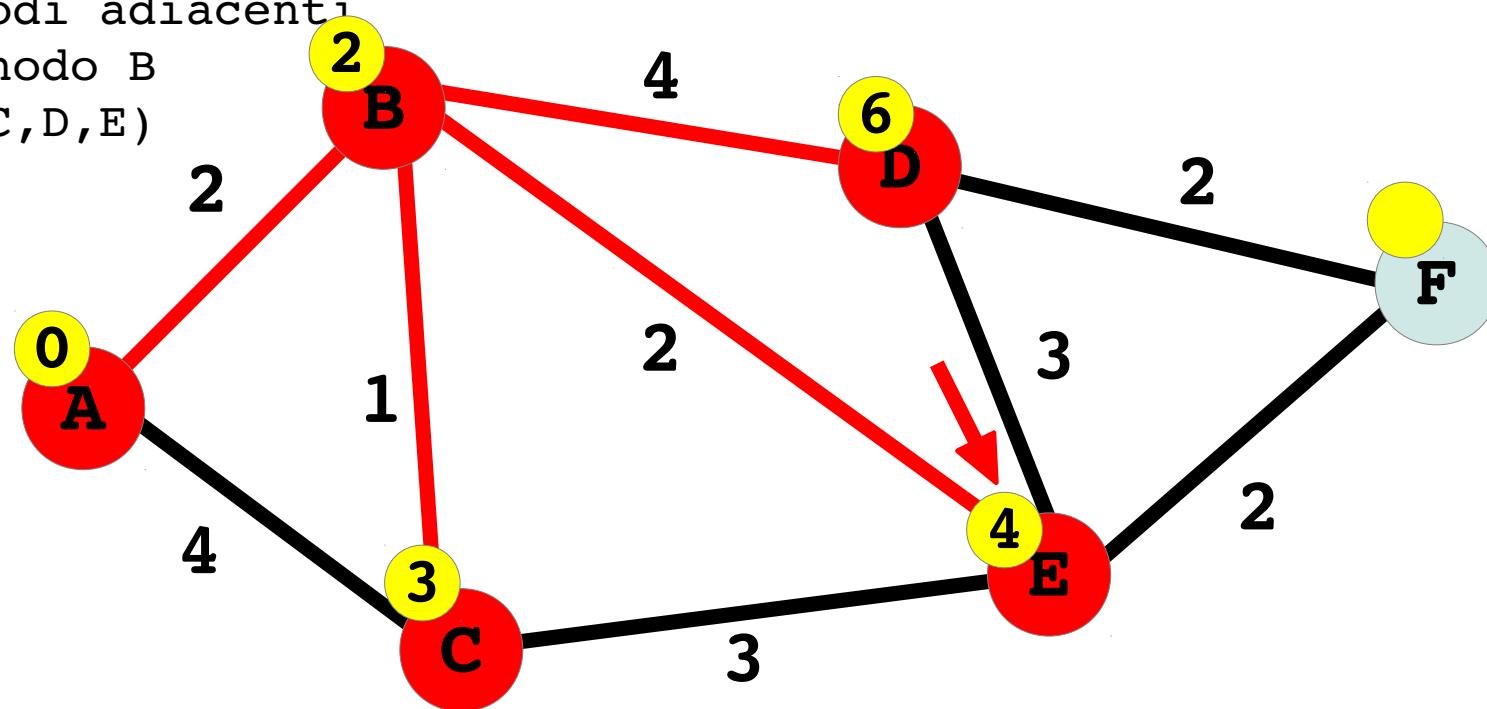
D:  $2+4=6 < \text{in(de)finito}$  ;  
D memorizza 6 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo B  
(A, C, D, E)



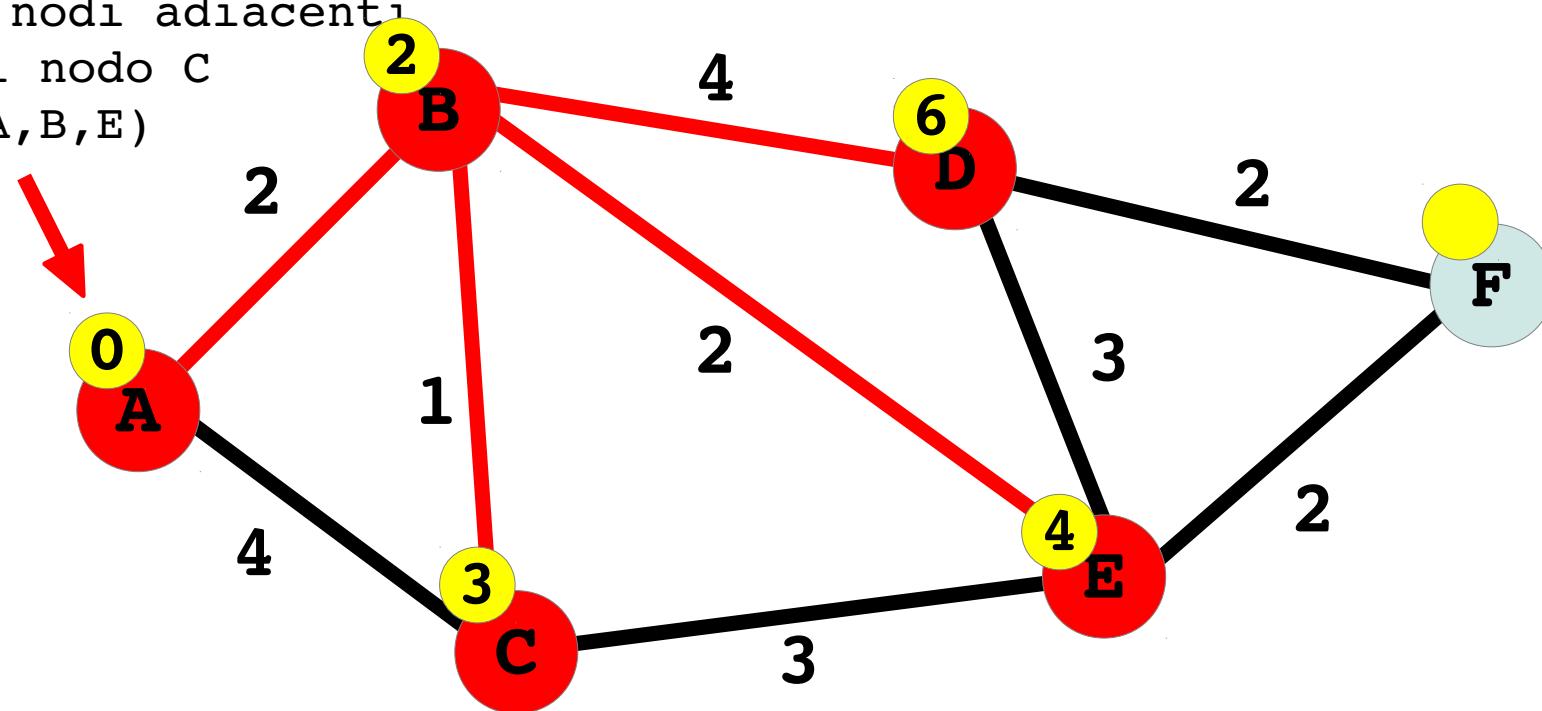
E:  $2+2=4 <$  in(de)finito ;  
E memorizza 4 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo C  
(A, B, E)



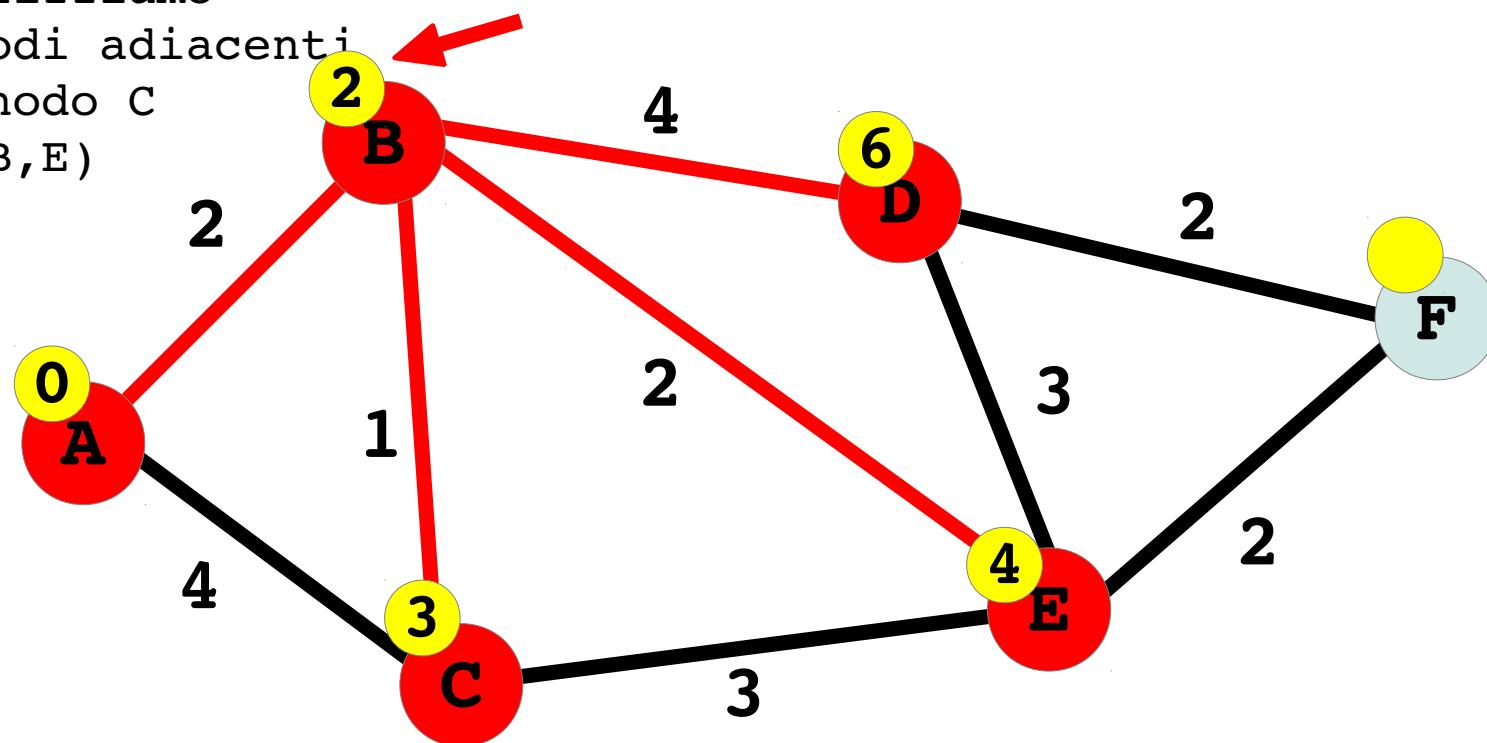
$$A: 3+4=7 > 0 ; \\ A \text{ invariato}$$

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo C  
(A, B, E)



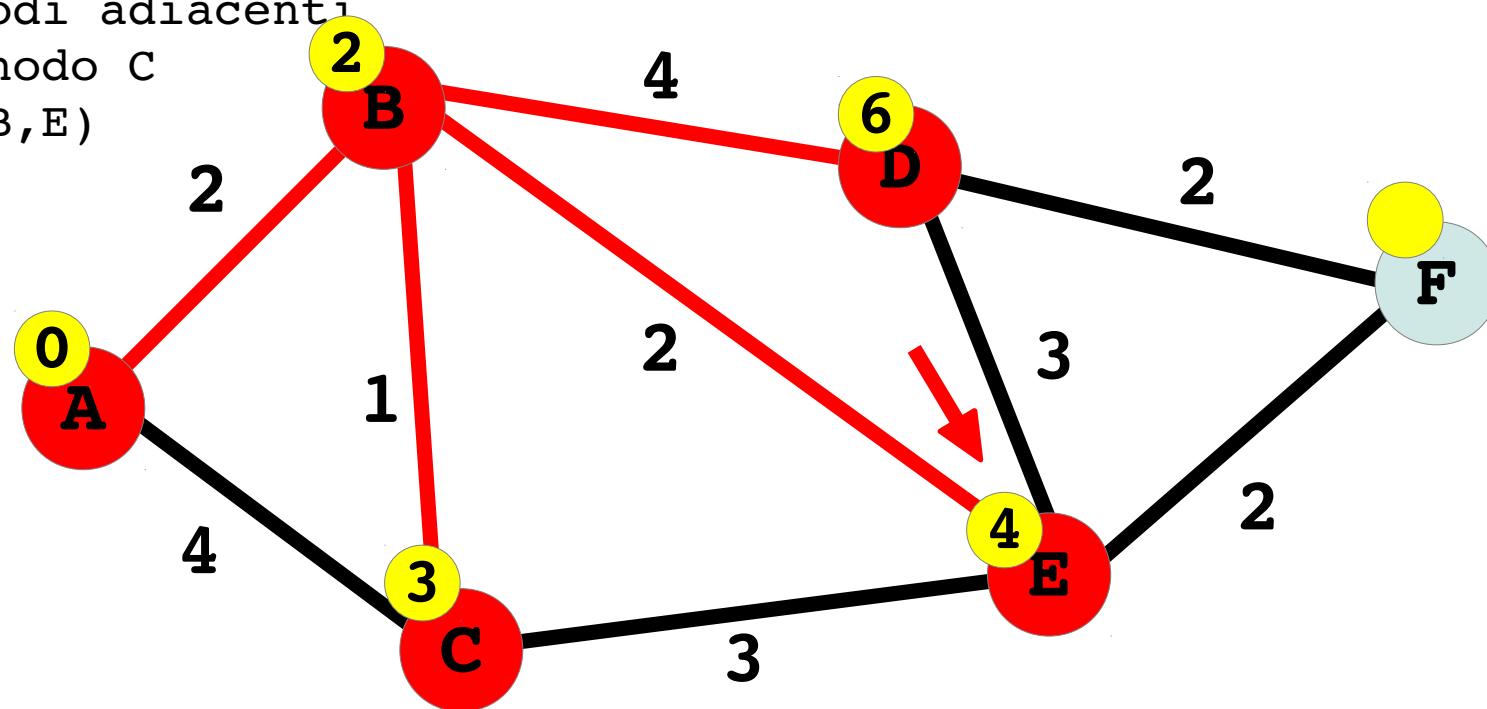
B:  $3+1=4 > 2$  ;  
B invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

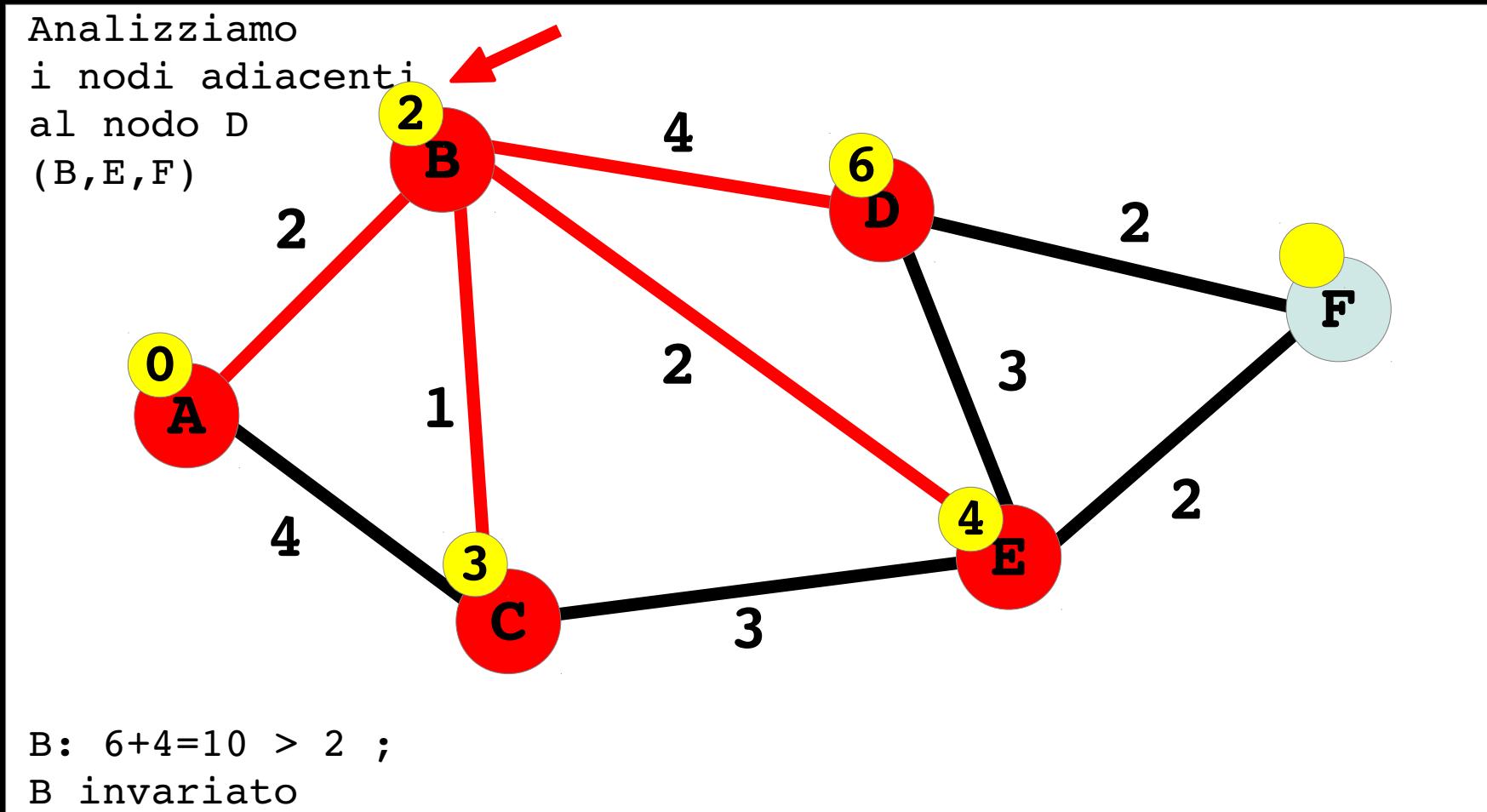
i nodi adiacenti  
al nodo C  
(A, B, E)



E:  $3+3=6 > 4$  ;  
E invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

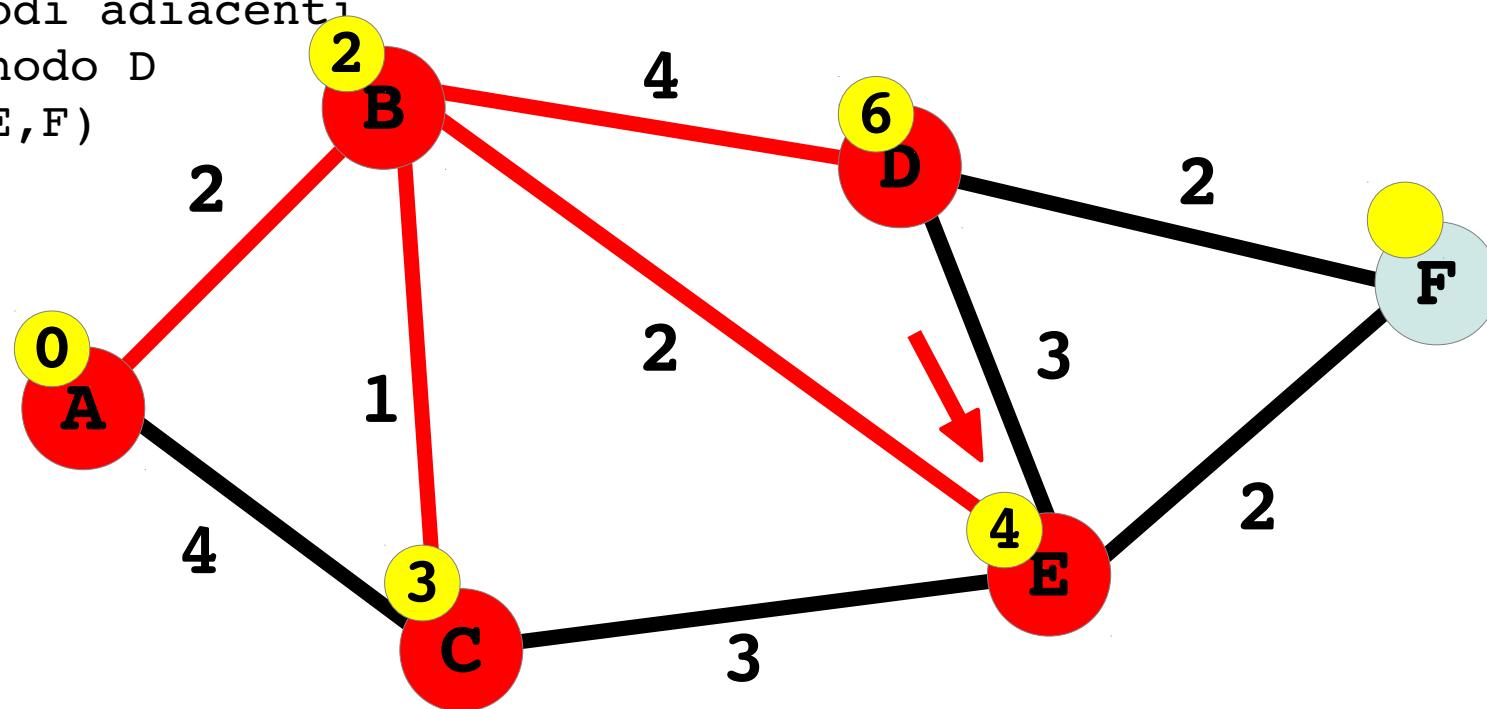


# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo D  
(B, E, F)



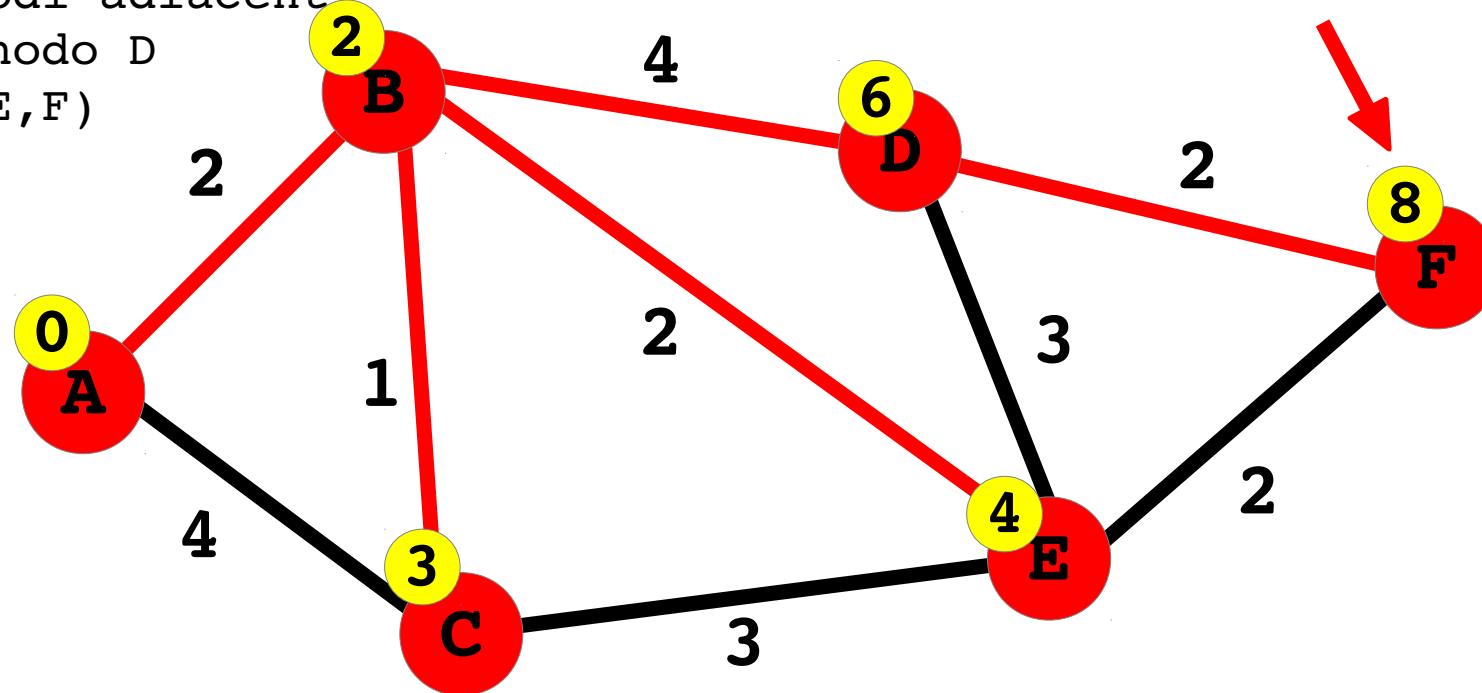
E:  $6+3=9 > 4$  ;  
E invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo D  
(B, E, F)



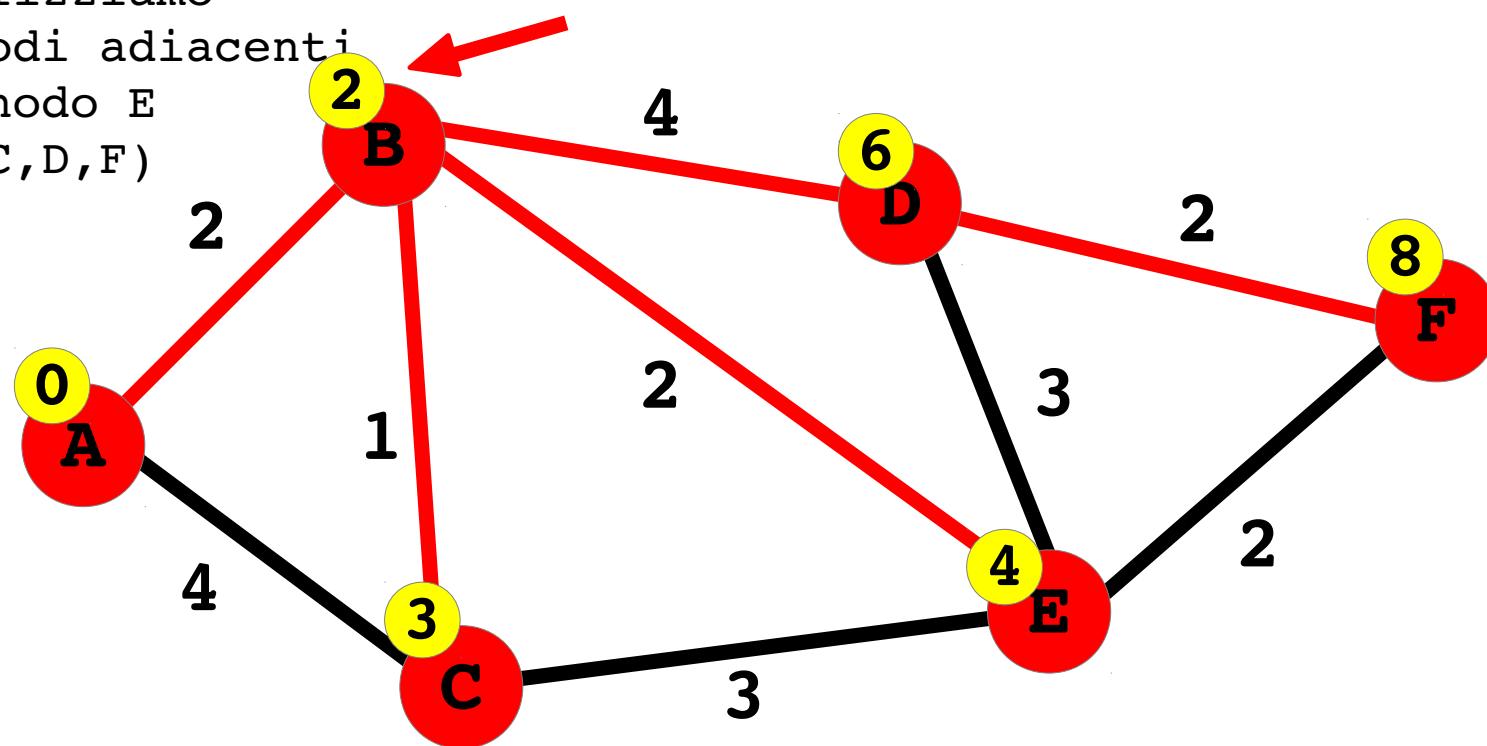
F:  $6+2=8 < \text{in(de)finito}$  ;  
F memorizza 8 e si aggiorna percorso

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo E  
(B, C, D, F)



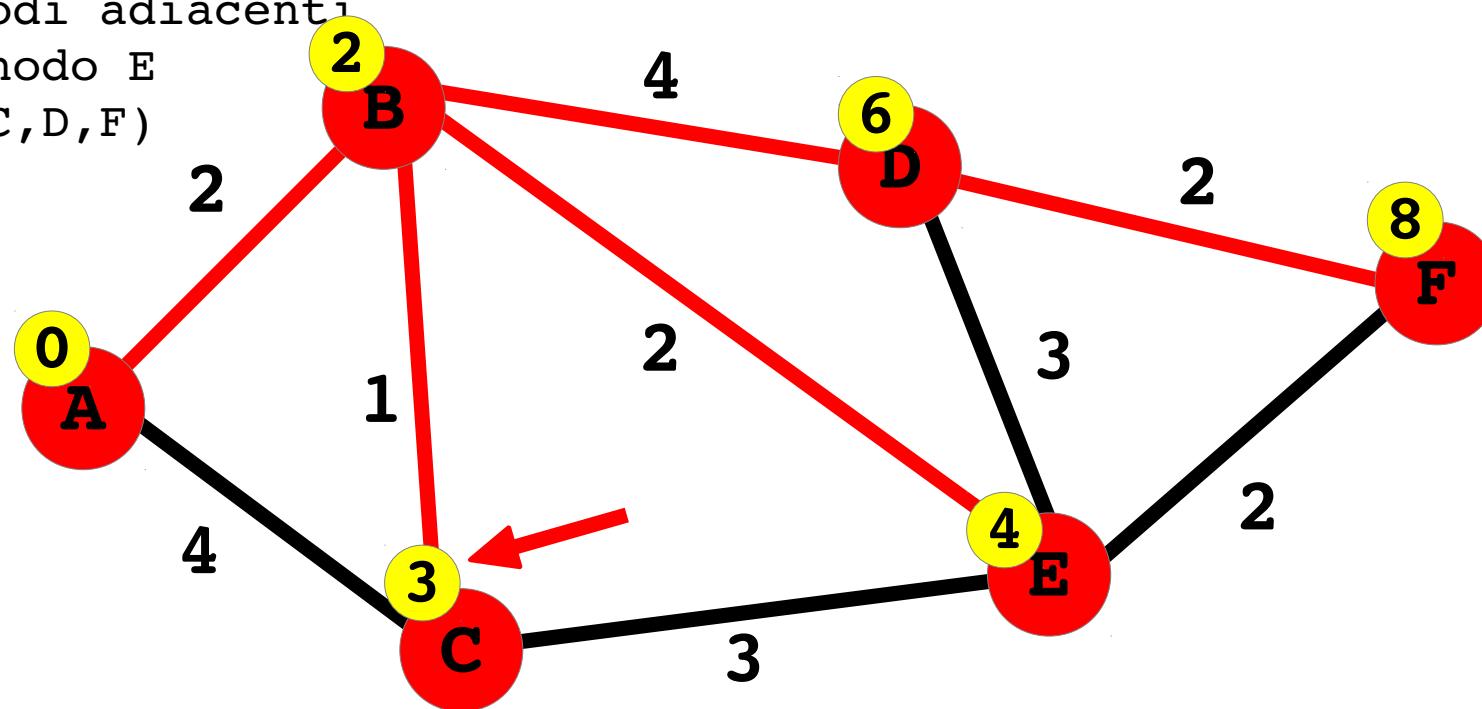
B:  $4+2=6 > 2$  ;  
B invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo E  
(B, C, D, F)



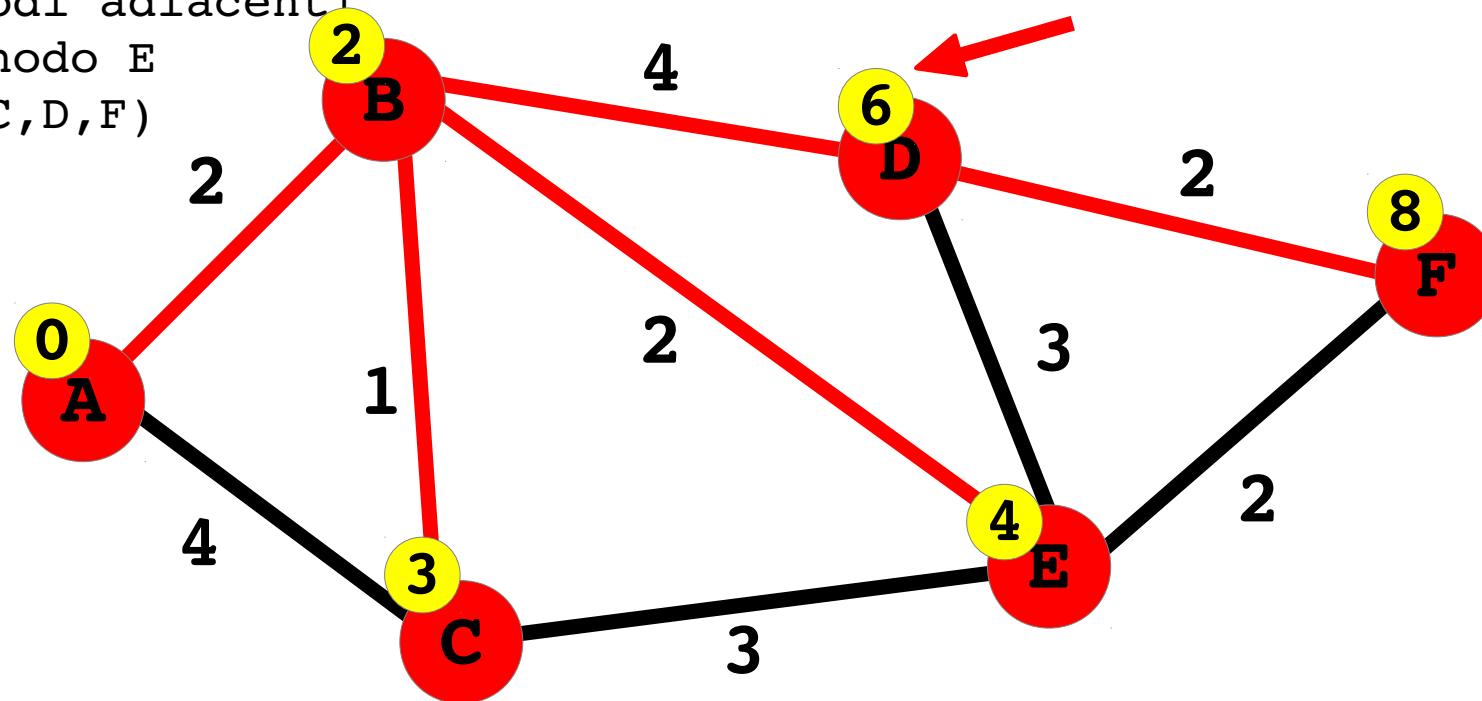
C:  $4+3=7 > 3$  ;  
C invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo E  
(B, C, D, F)



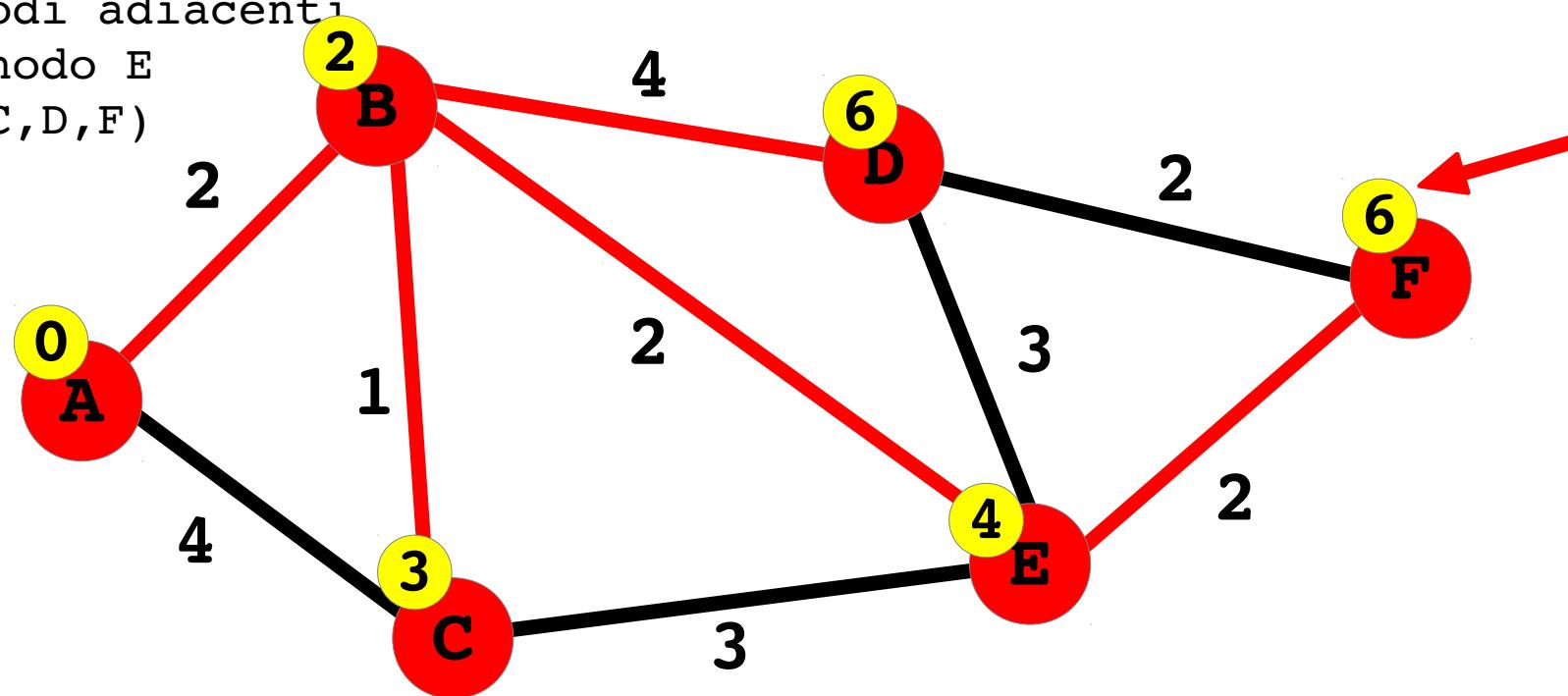
D:  $4+3=7 > 6$  ;  
D invariato

# Problem Solving

## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti  
al nodo E  
(B, C, D, F)

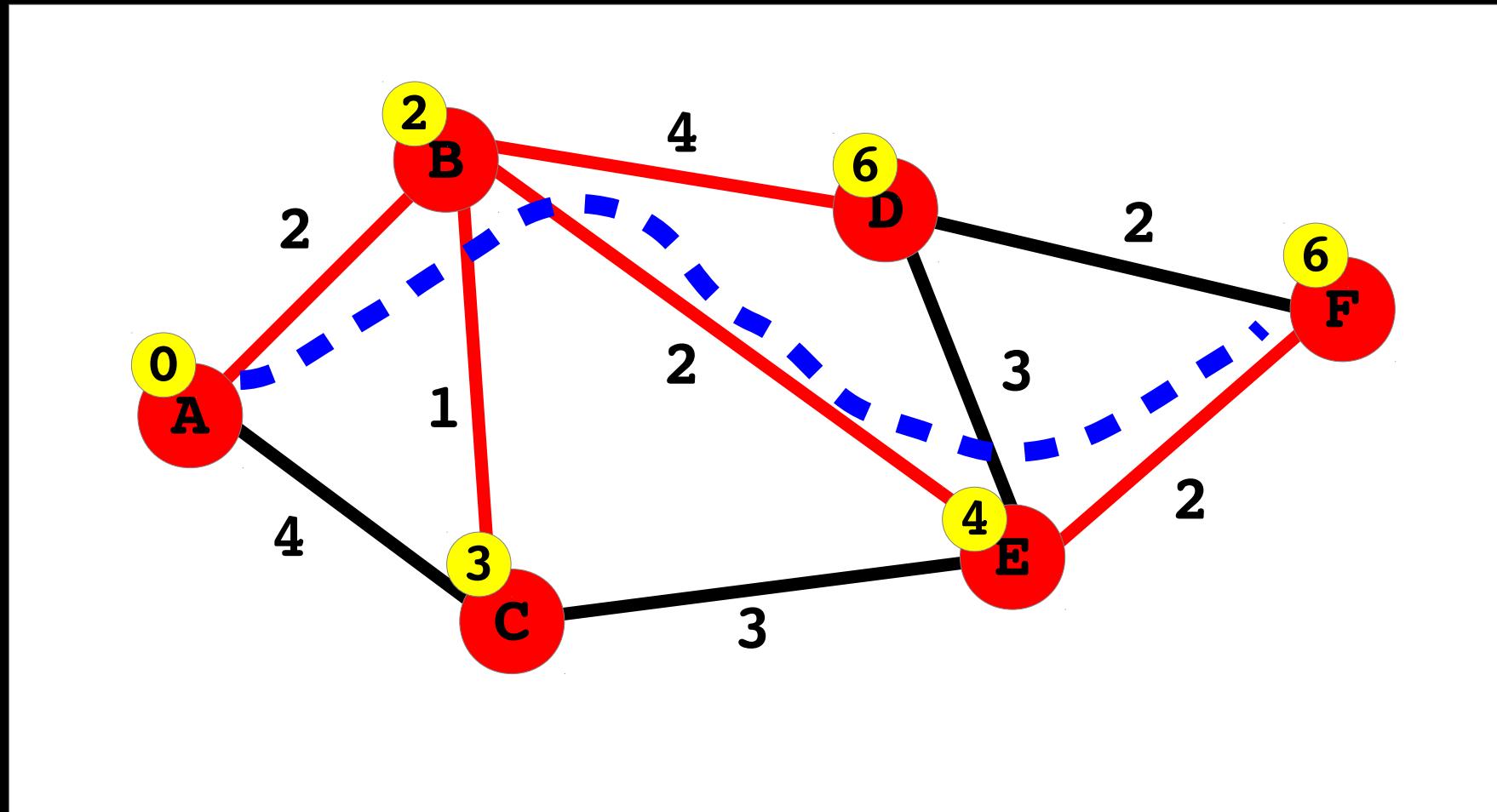


$$F: 4+2=6 < 8 ;$$

F memorizza 6 e si aggiorna percorso

# Problem Solving

## 1959: *Dijkstra's Algorithm*



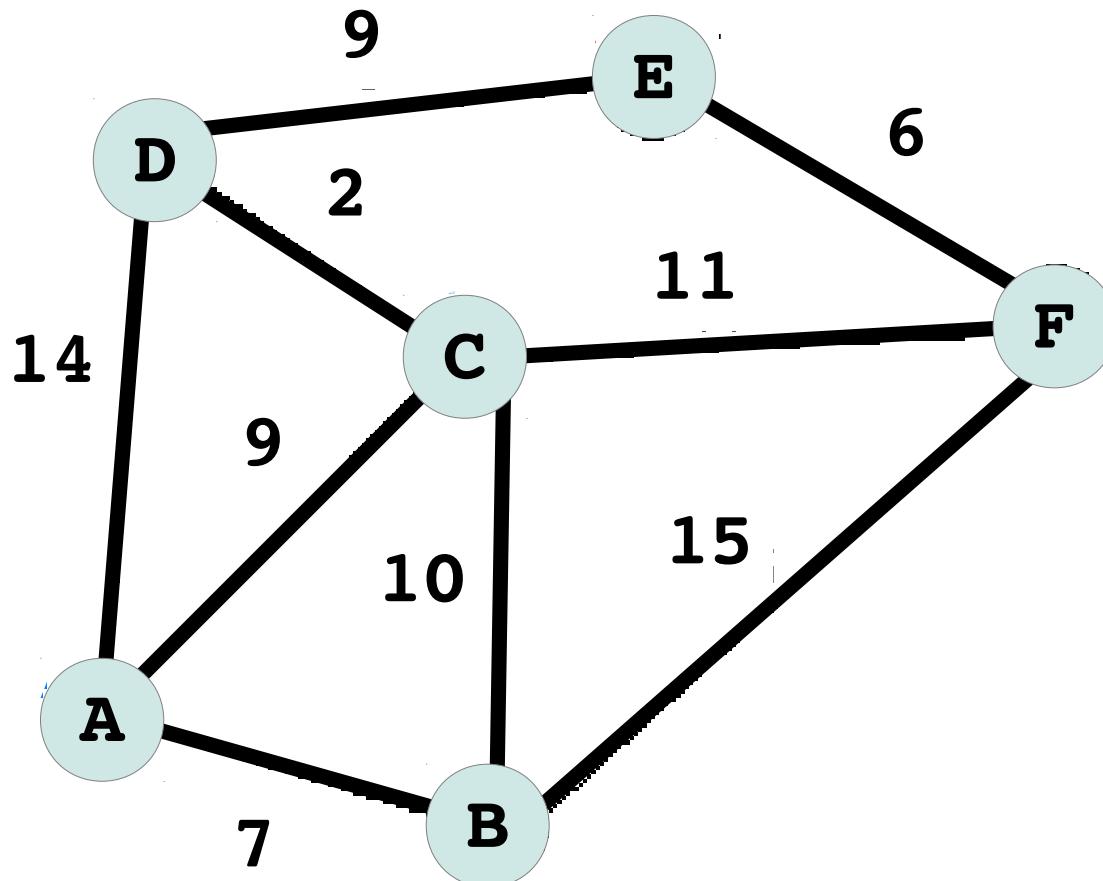
Problema: come descrivere la soluzione al computer?

Quali strutture dati? Quali variabili?

Il pensiero computazionale ci aiuta ad affrontare questo problema ...

# Problem Solving

## 1959: *Dijkstra's Algorithm*



Trovare il percorso a *energia minima* da A a F

# Problem Solving

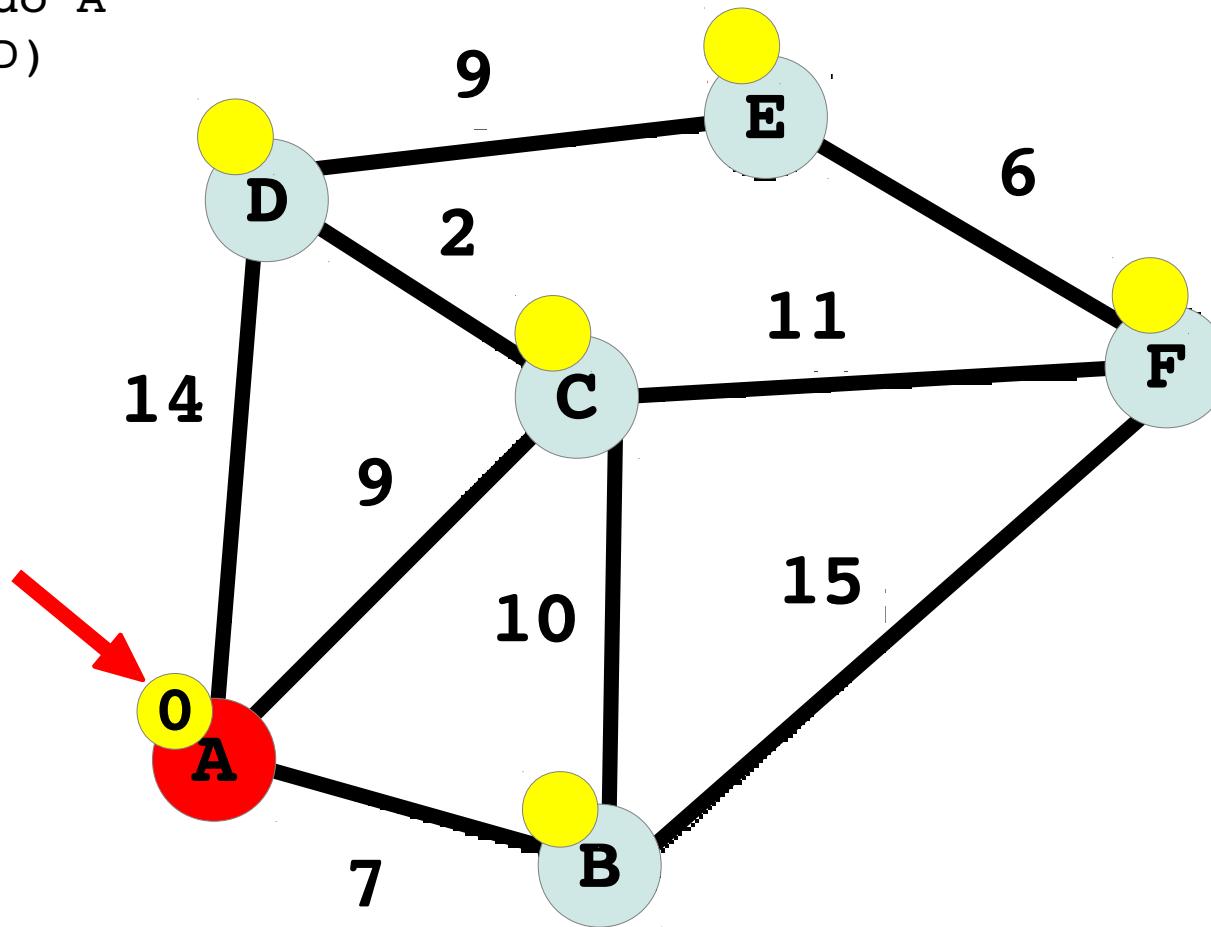
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo A

(B, C, D)



# Problem Solving

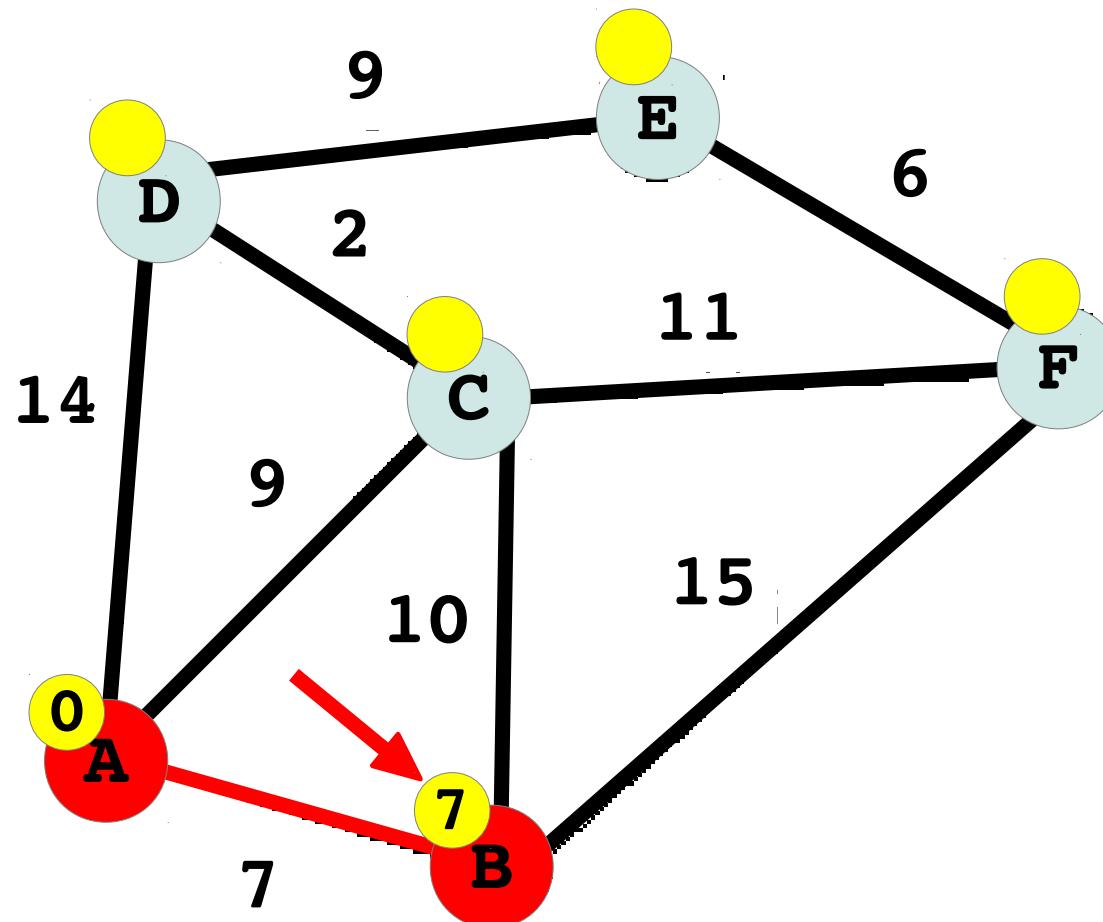
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo A

(B, C, D)



B:  $0+7=7 <$  infinito

B cambia

# Problem Solving

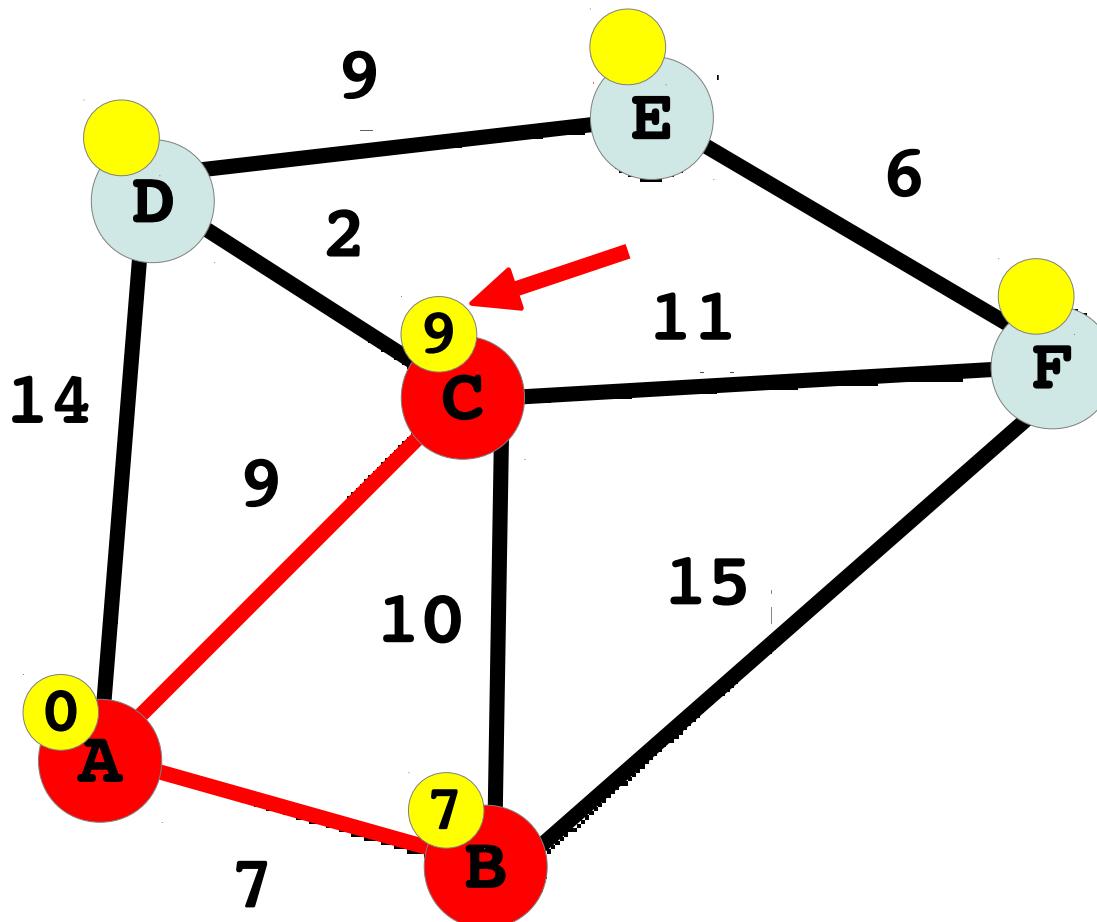
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo A

(B, C, D)



$$C: 0+9=9 < \text{in(de)finito}$$

C cambia

# Problem Solving

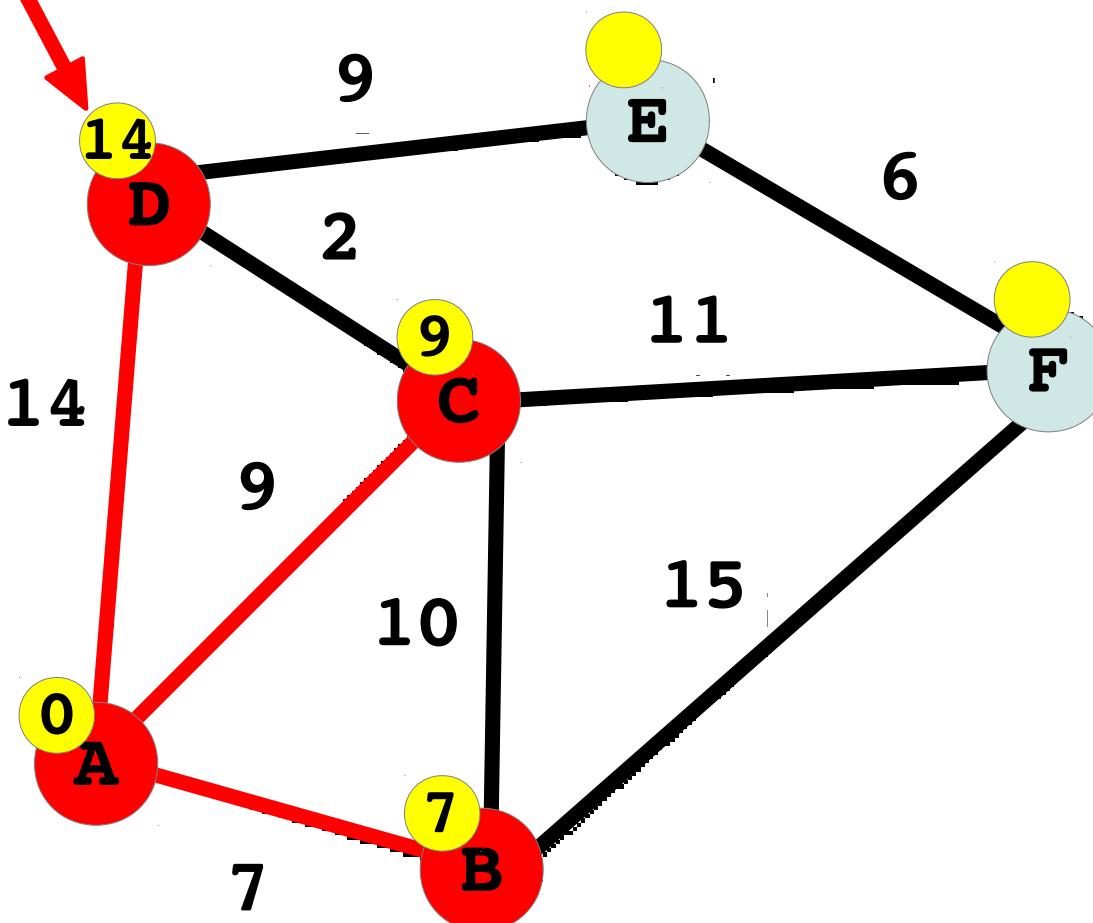
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo A

(B, C, D)



$$D: 0+14=14 < \text{in(de)finito}$$

D cambia

# Problem Solving

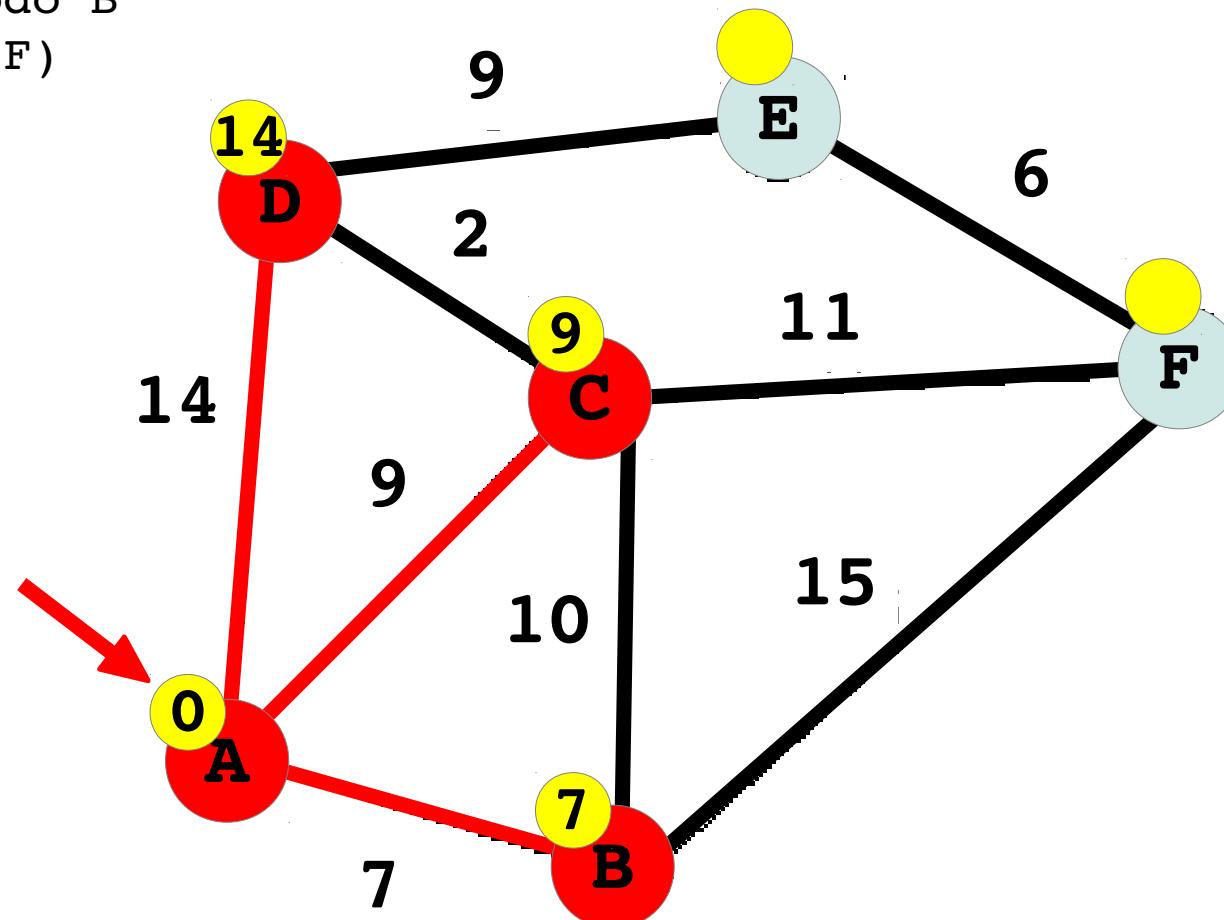
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo B

(A, C, F)



$$A: 7+7=14 > 0$$

A non cambia

# Problem Solving

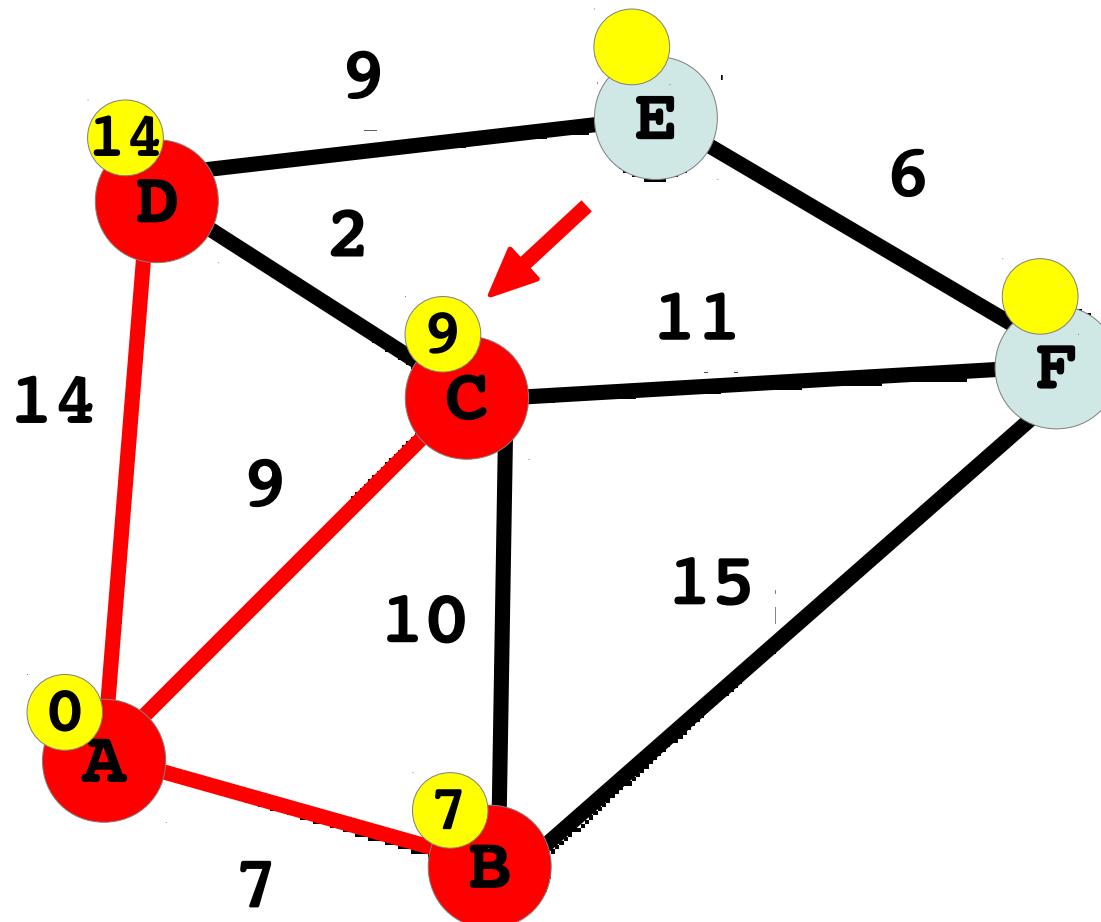
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo B

(A, C, F)



$$C: 7+10=17 > 9$$

C non cambia

# Problem Solving

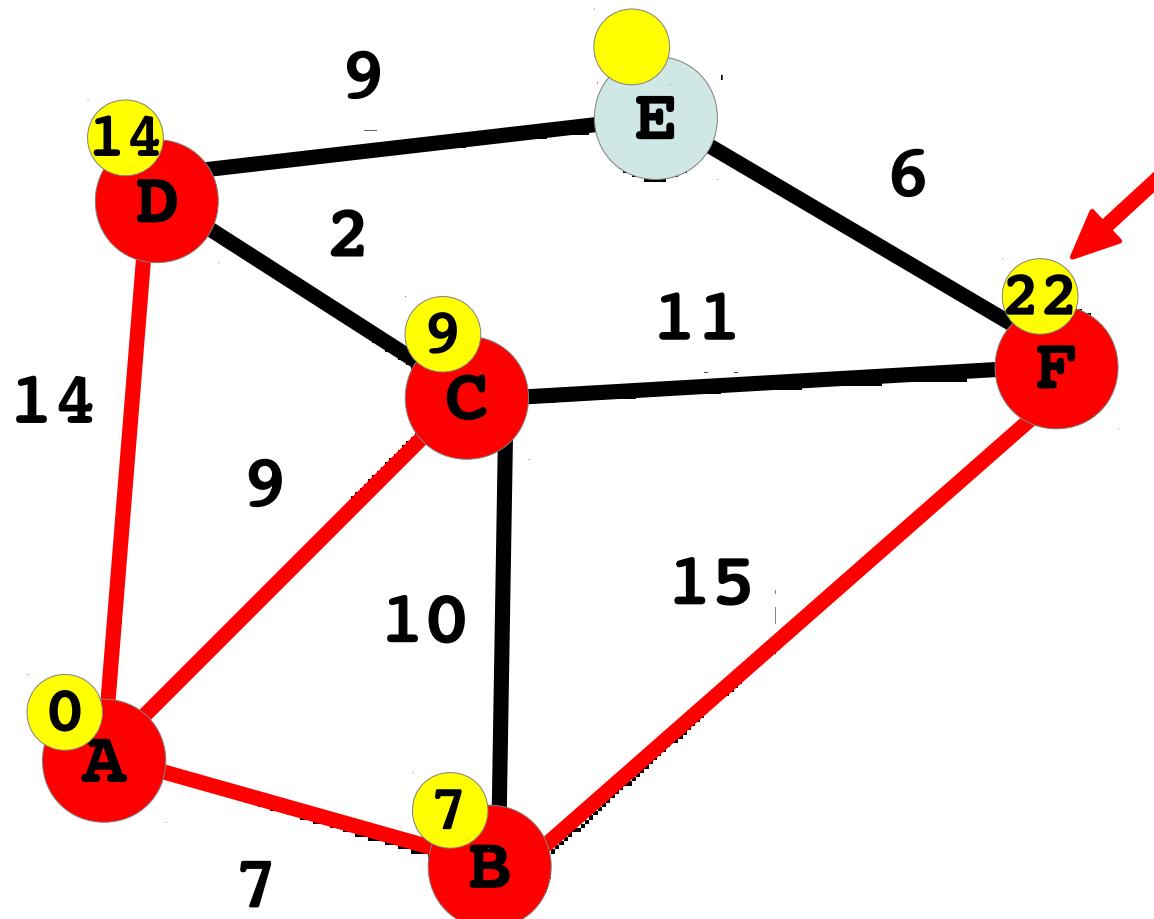
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo B

(A, C, F)



F:  $7+15=22 < \text{in(de)finito}$   
F cambia

# Problem Solving

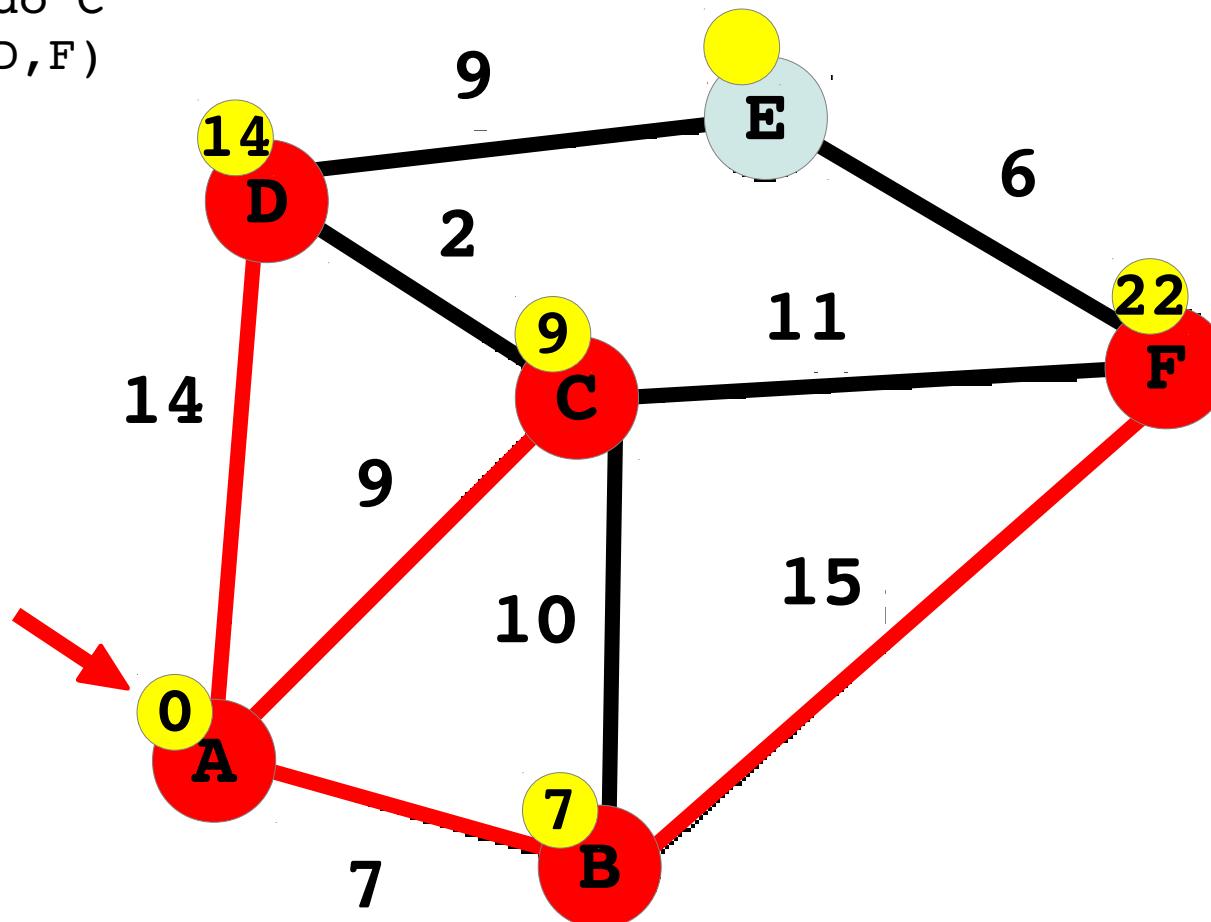
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo C

(A, B, D, F)



$$A: 9+9=18 > 0$$

A non cambia

# Problem Solving

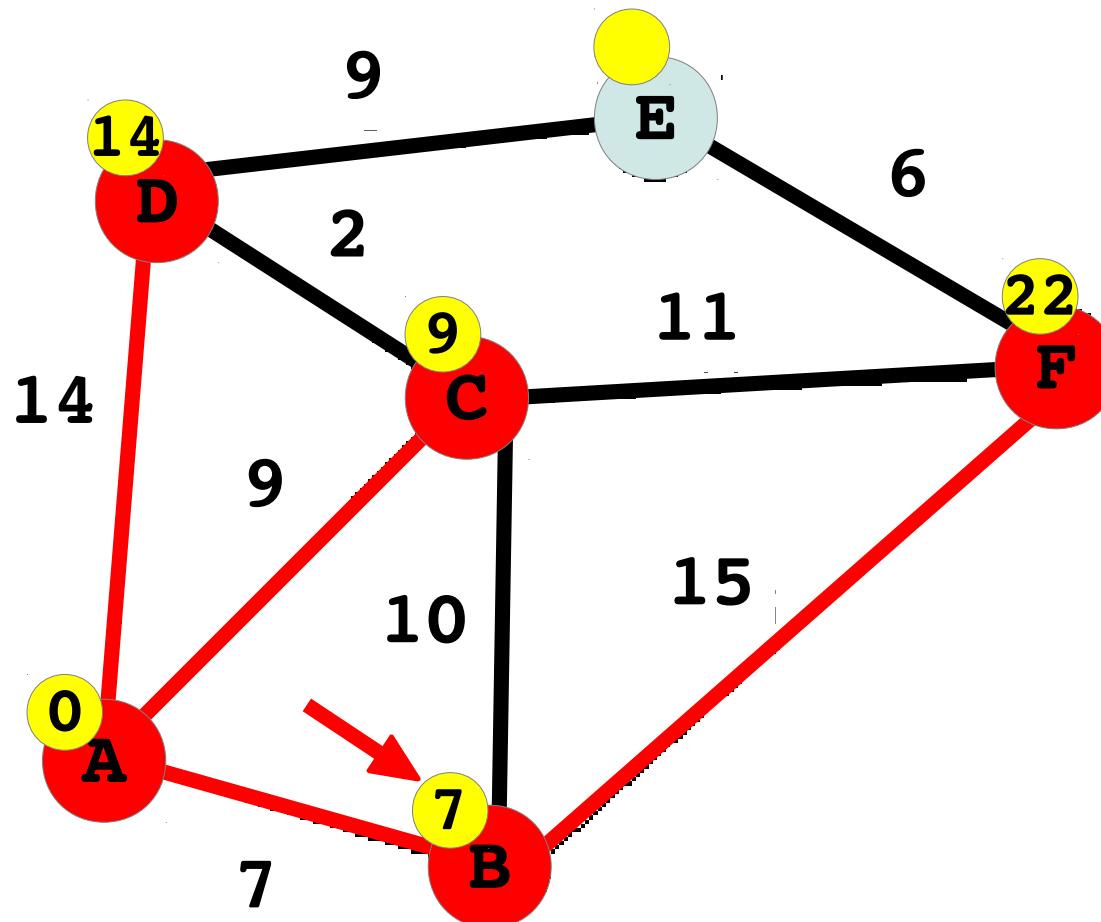
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo C

(A, B, D, F)



$$B: 9+10=19 > 7$$

B non cambia

# Problem Solving

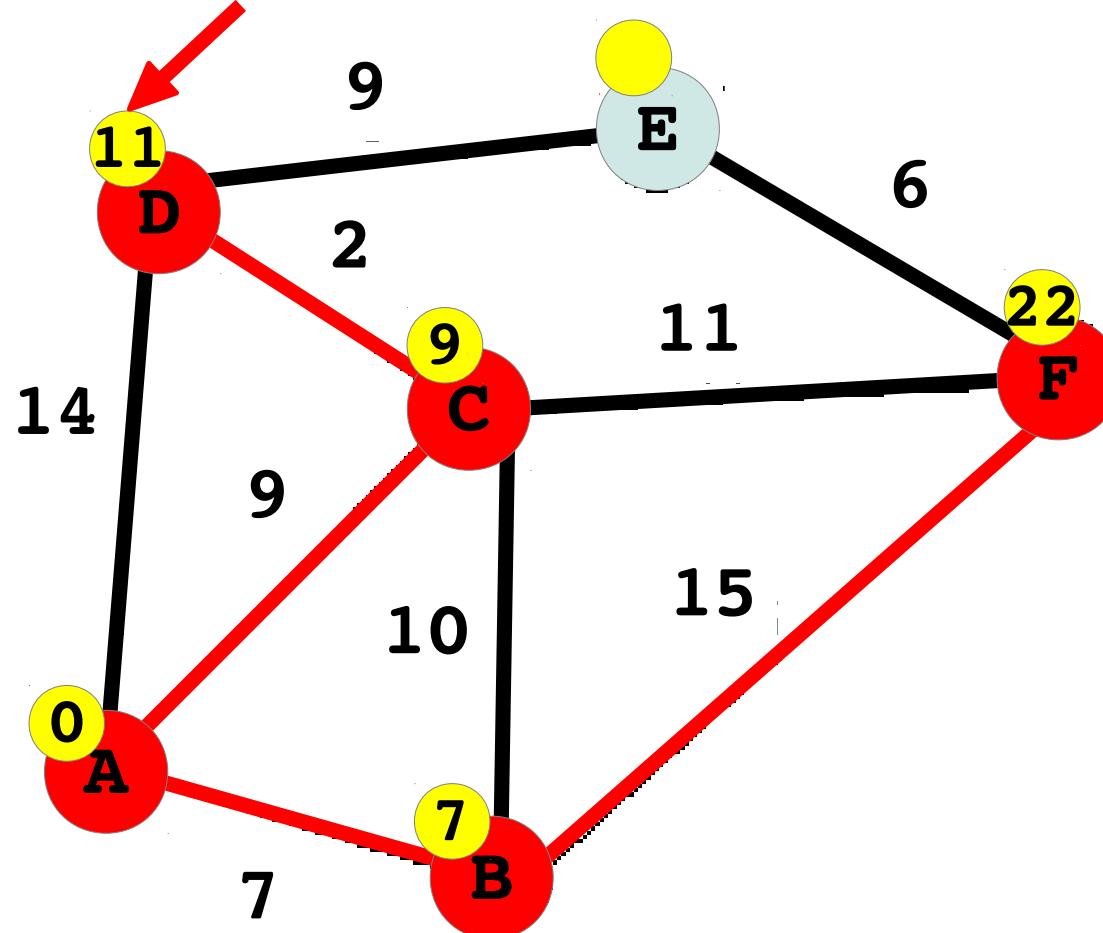
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo C

(A, B, D, F)



$$D: 9+2=11 < 14$$

D cambia

# Problem Solving

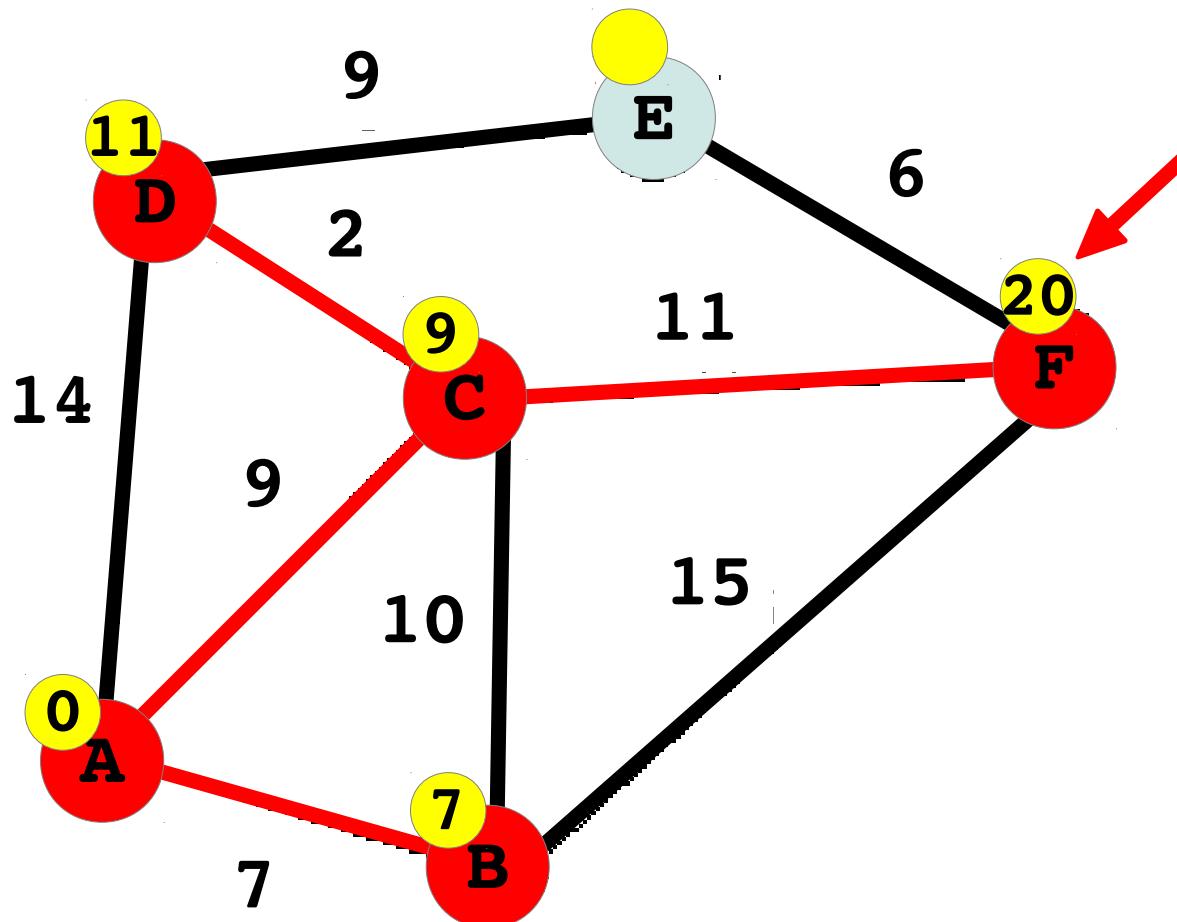
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo C

(A, B, D, F)



$$F: 9 + 11 = 20 < 22$$

F cambia

# Problem Solving

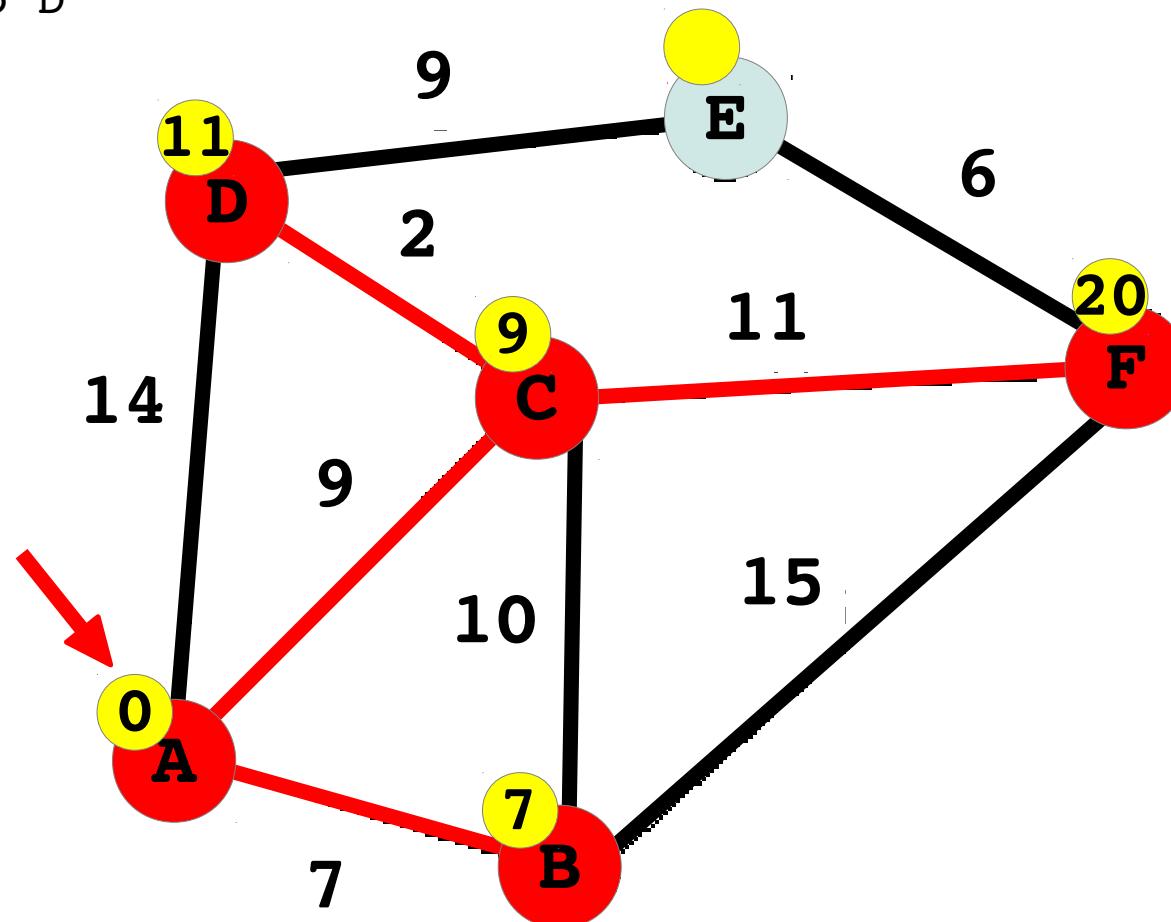
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo D

(A, C, E)



$$A: 11+14=25 > 0$$

A non cambia

# Problem Solving

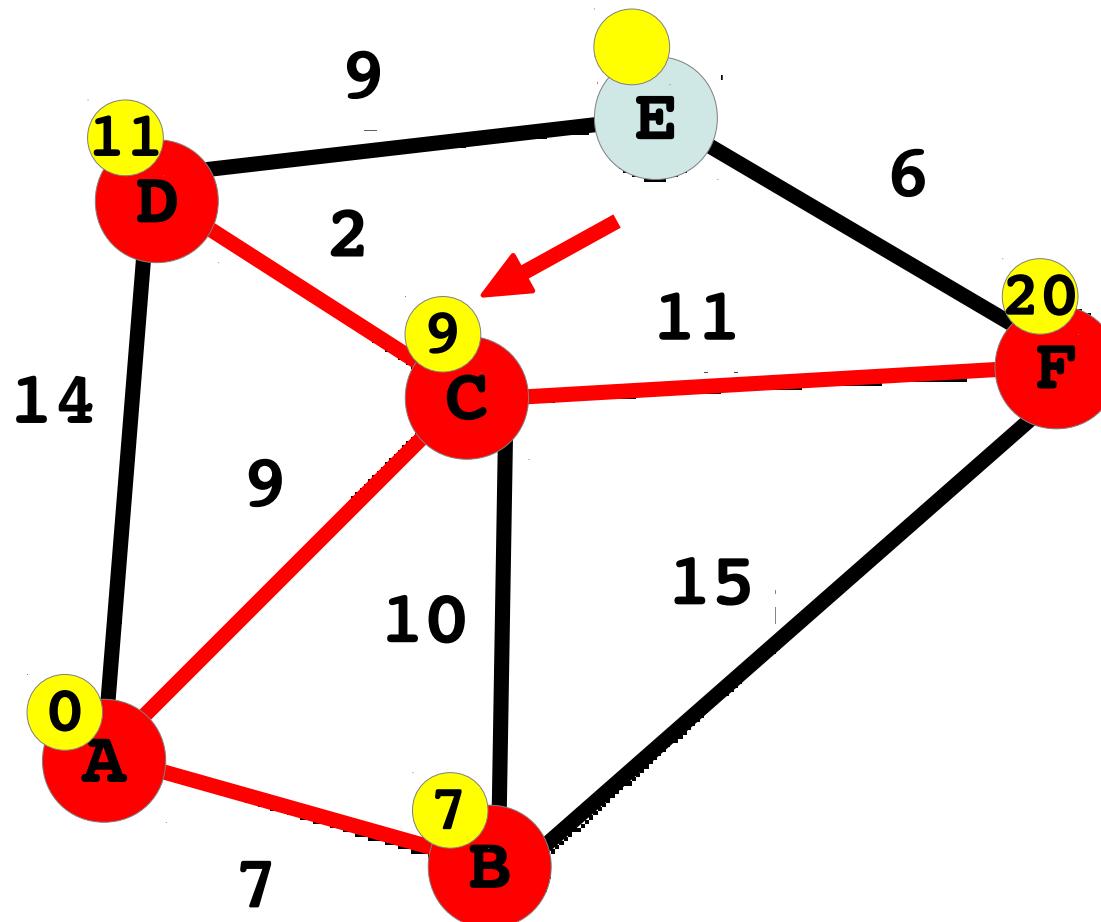
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo D

(A, C, E)



$$C: 11+2=13 > 9$$

C non cambia

# Problem Solving

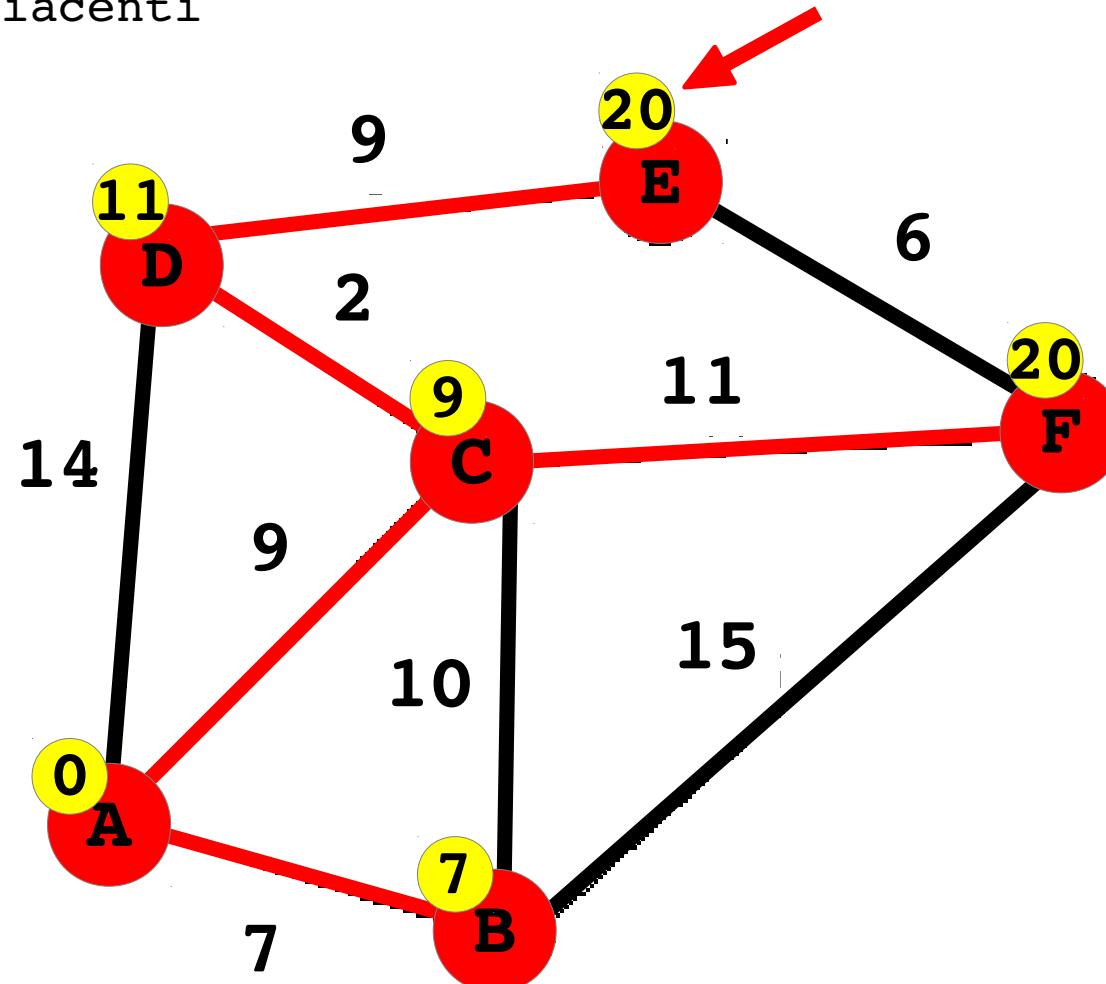
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo D

(A, C, E)



E:  $11+9=20 < \text{in(de)finito}$   
E cambia

# Problem Solving

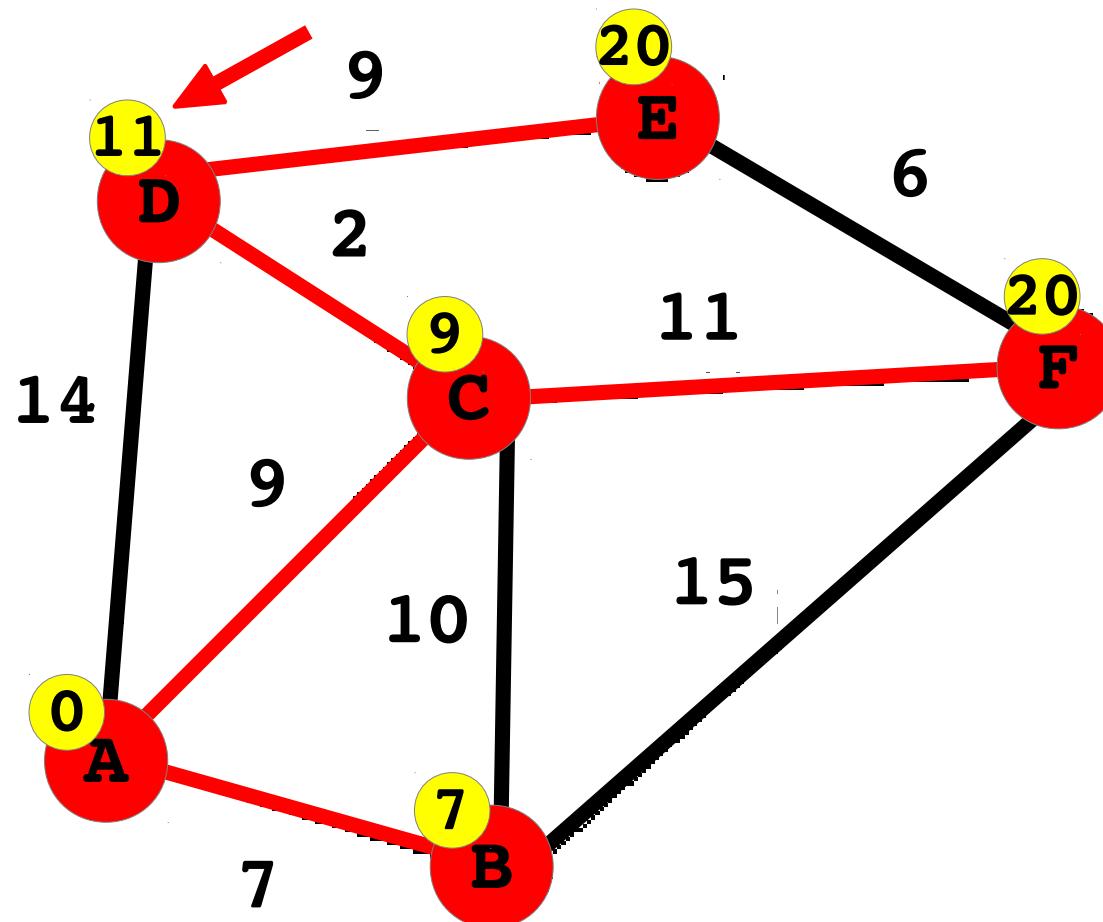
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo E

(D, F)



$$D: 20+9=29 > 11$$

D non cambia

# Problem Solving

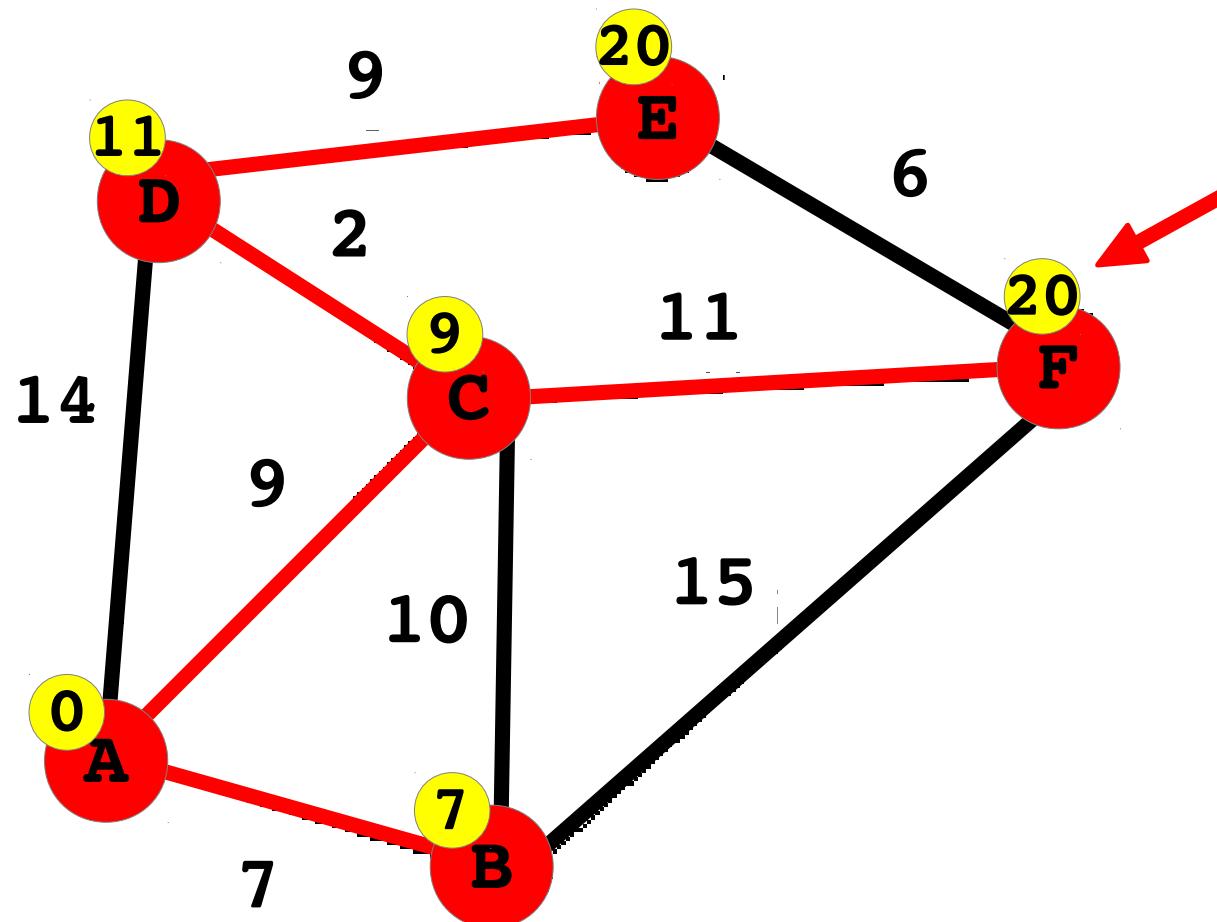
## 1959: Dijkstra's Algorithm

Analizziamo

i nodi adiacenti

al nodo E

(D, F)

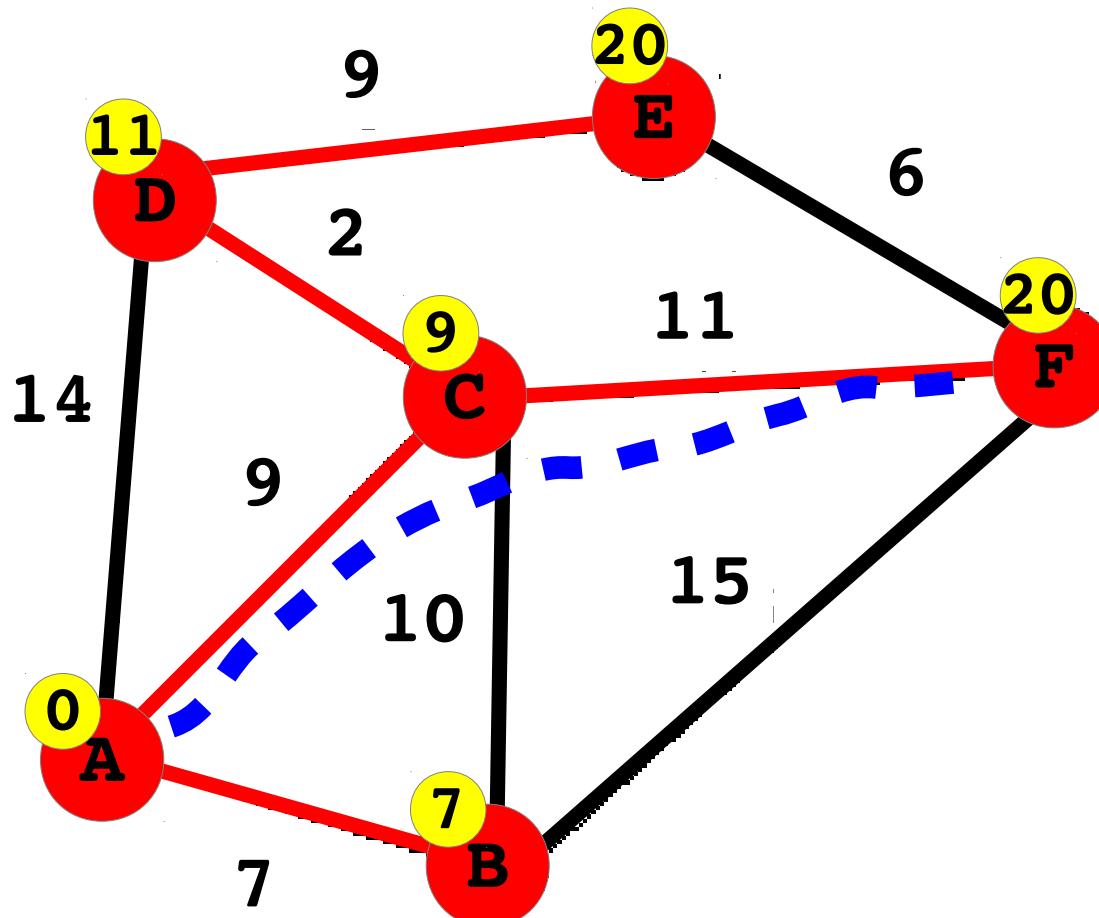


$$F: 20+6=26 > 20$$

F non cambia

# Problem Solving

## 1959: *Dijkstra's Algorithm*



Come descrivere la soluzione al computer?  
Quali strutture dati? Quali variabili?  
Pensiero computazionale!