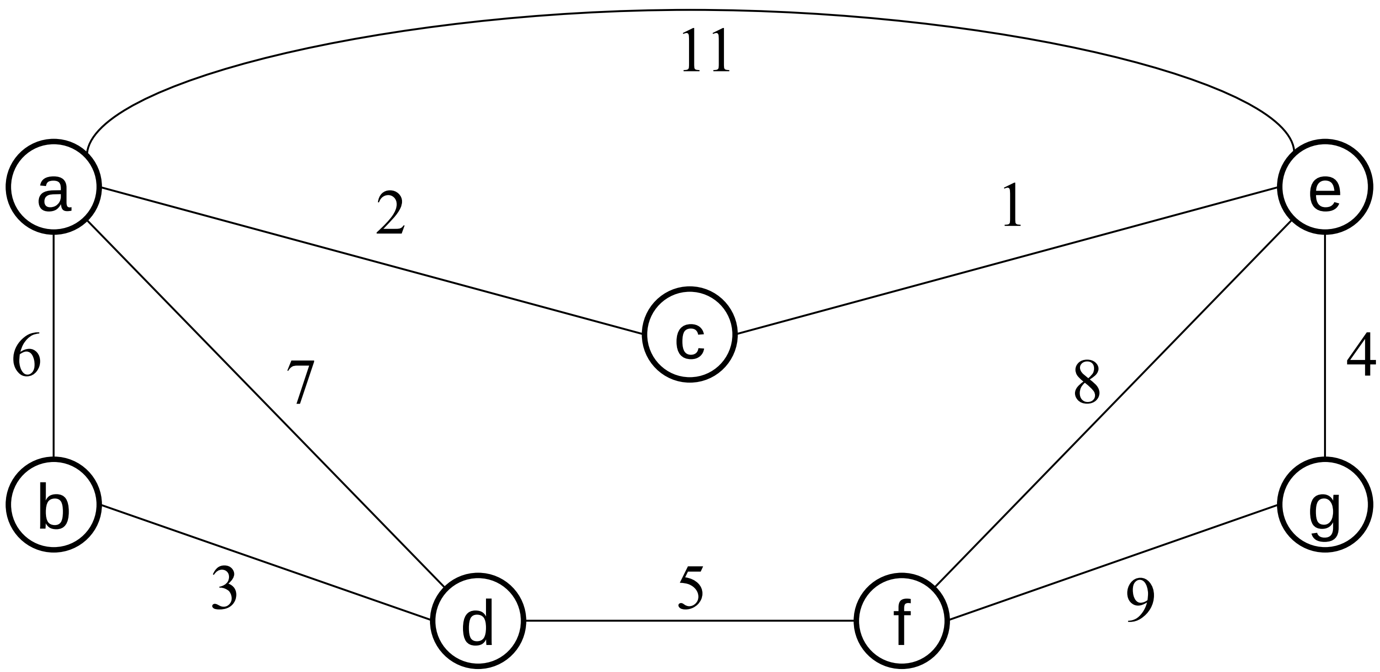
Algoritmi e Strutture Dati

Esercitazione 9

Domenico Fabio Savo

Esercizio: algoritmo di Kruskal

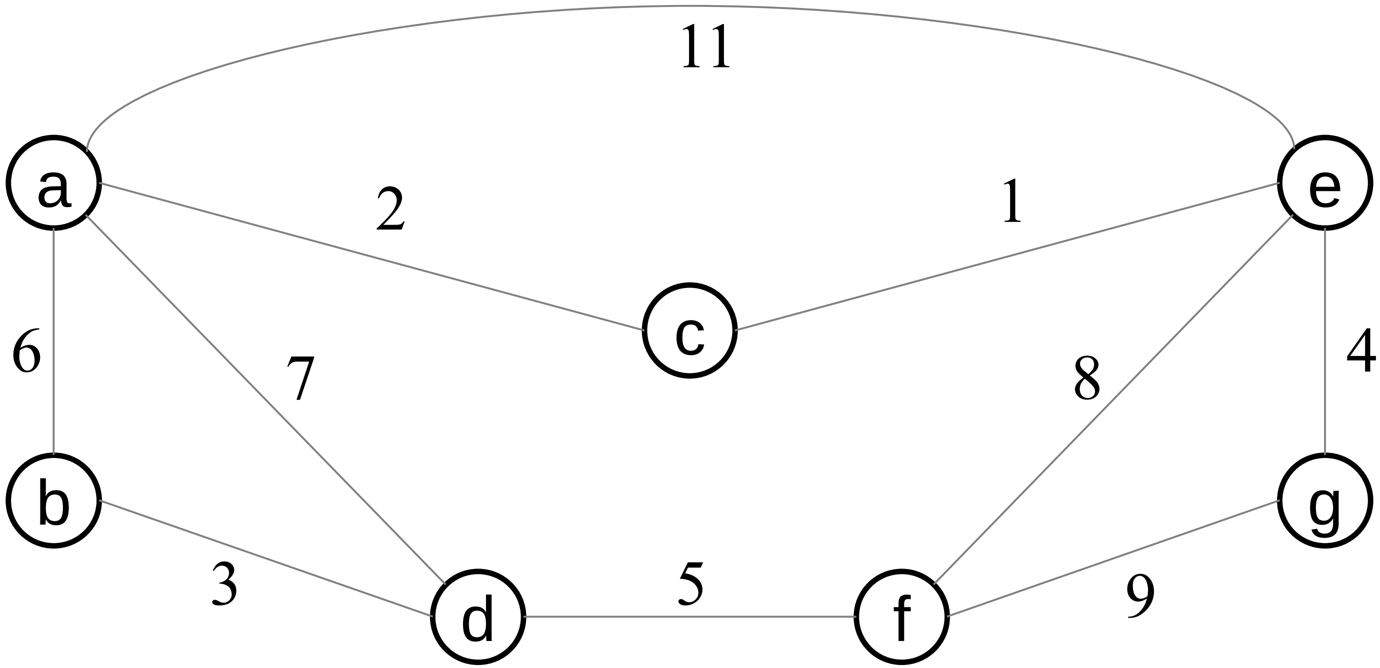
Esercizio: Calcolare il minimo albero ricoprente del grafo rappresentato in figura. Adottare l'algoritmo di **Kruskal**



Esercizio: algoritmo di Kruskal

1° step: ordinare gli archi del grafo in ordine NON decrescente

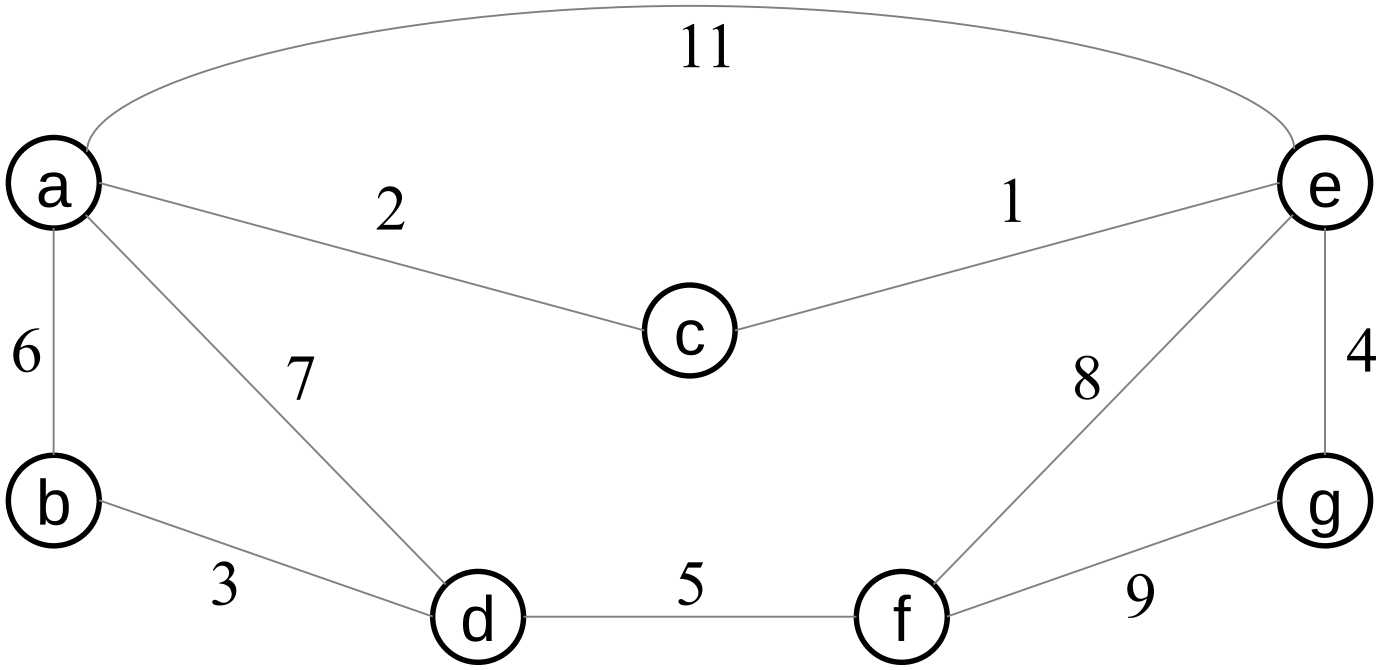
(a,e),(g,f),(e,f),(a,d),(a,b),(d,f),(e,g),(b,d),(a,c),(c,e) ⇨



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e),(g,f),(e,f),(a,d),(a,b),(d,f),(e,g),(b,d),(a,c),(c,e) ⇨

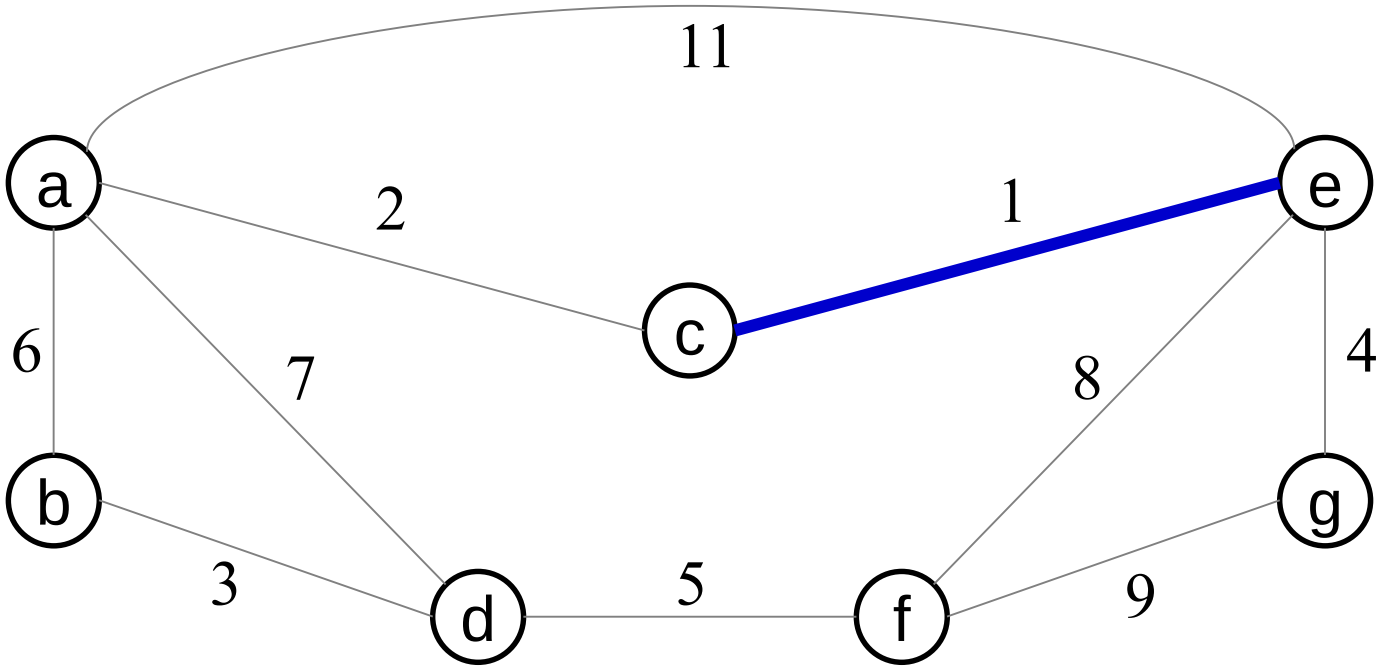


Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e),(g,f),(e,f),(a,d),(a,b),(d,f),(e,g),(b,d),(a,c)

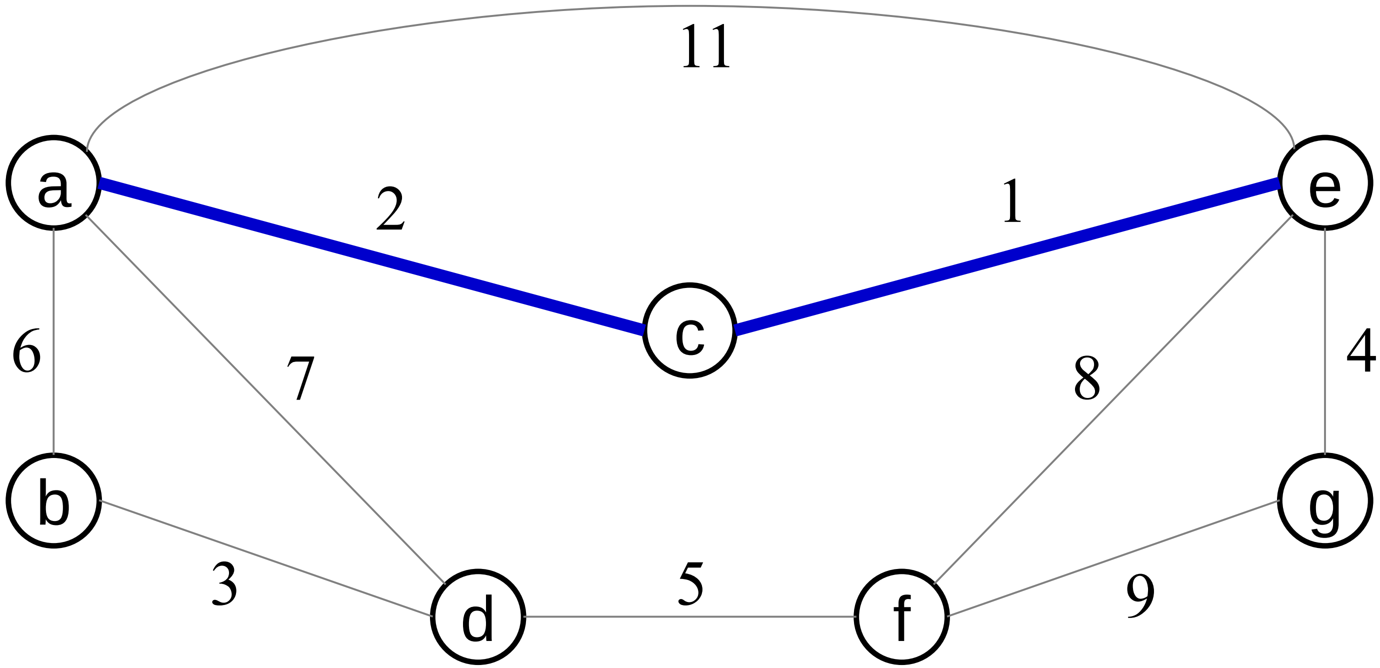
(e,c)



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

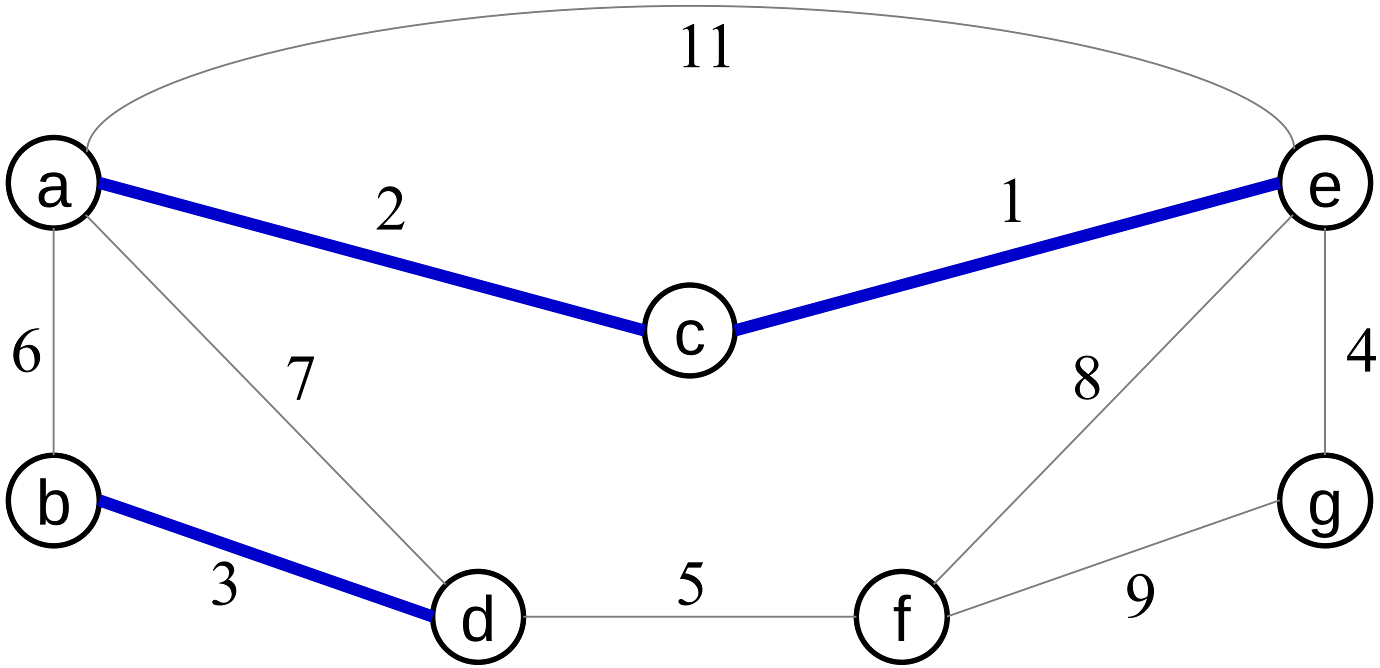
(a,e),(g,f),(e,f),(a,d),(a,b),(d,f),(e,g),(b,d) **(a,c)**



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

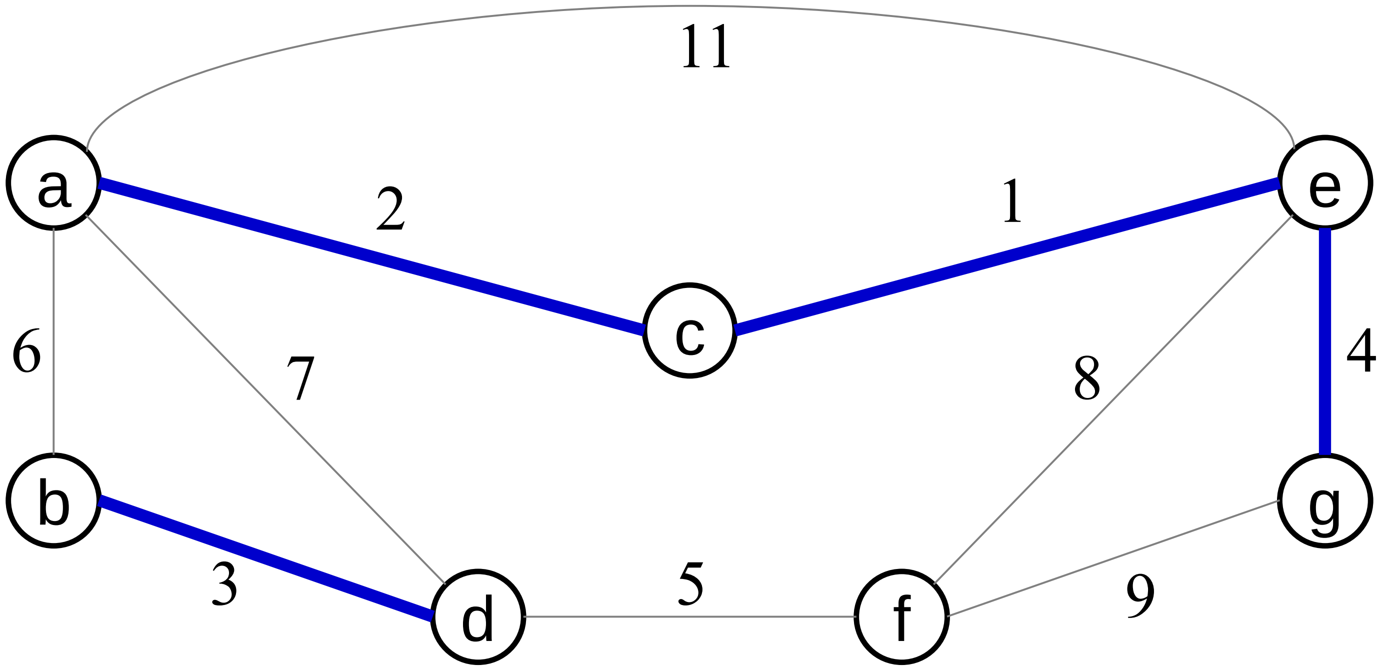
(a,e),(g,f),(e,f),(a,d),(a,b),(d,f),(e,g) **(b,d)**



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

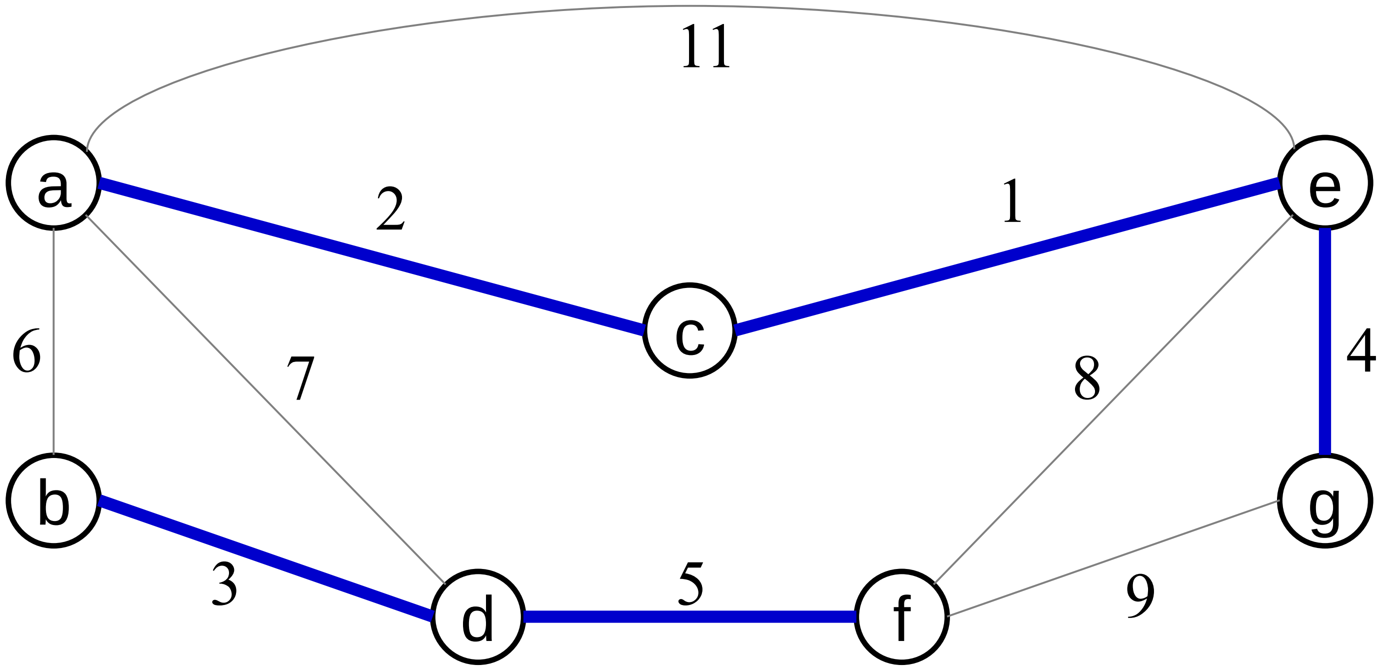
(a,e),(g,f),(e,f),(a,d),(a,b),(d,f) **(e,g)**



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

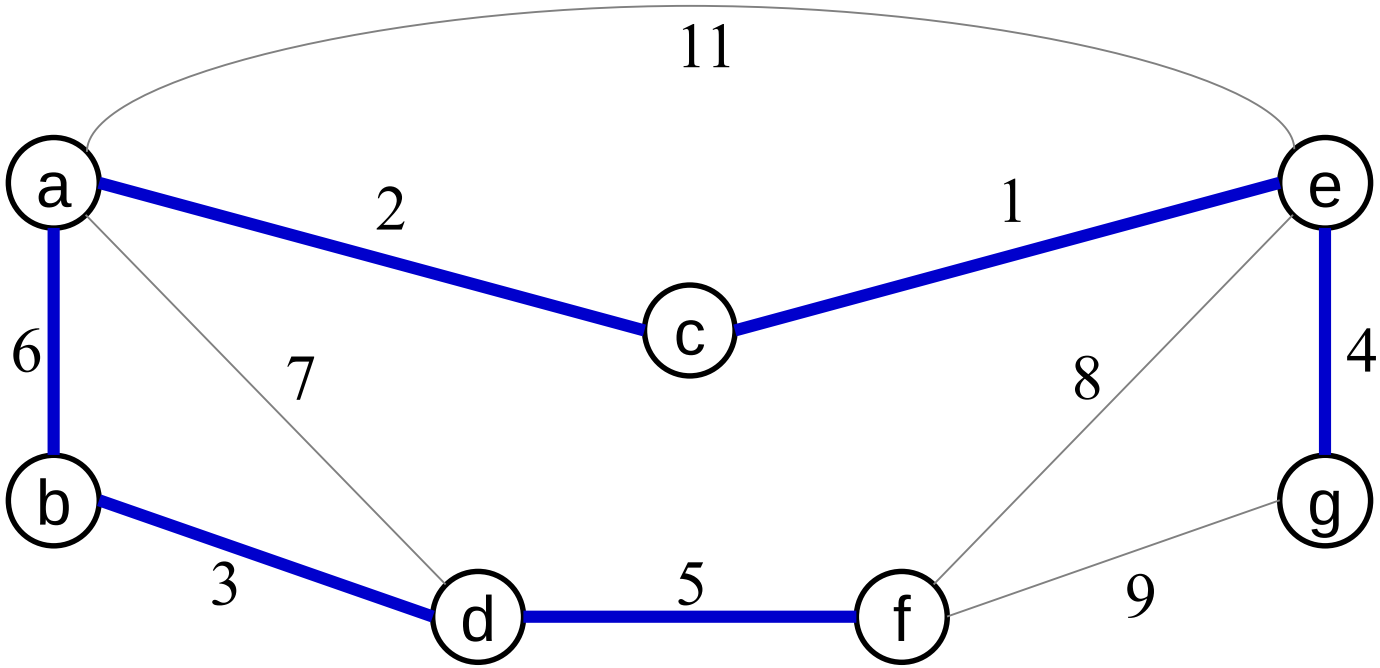
(a,e),(g,f),(e,f),(a,d),(a,b) **(d,f)**



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e),(g,f),(e,f),(a,d) **(a,b)**

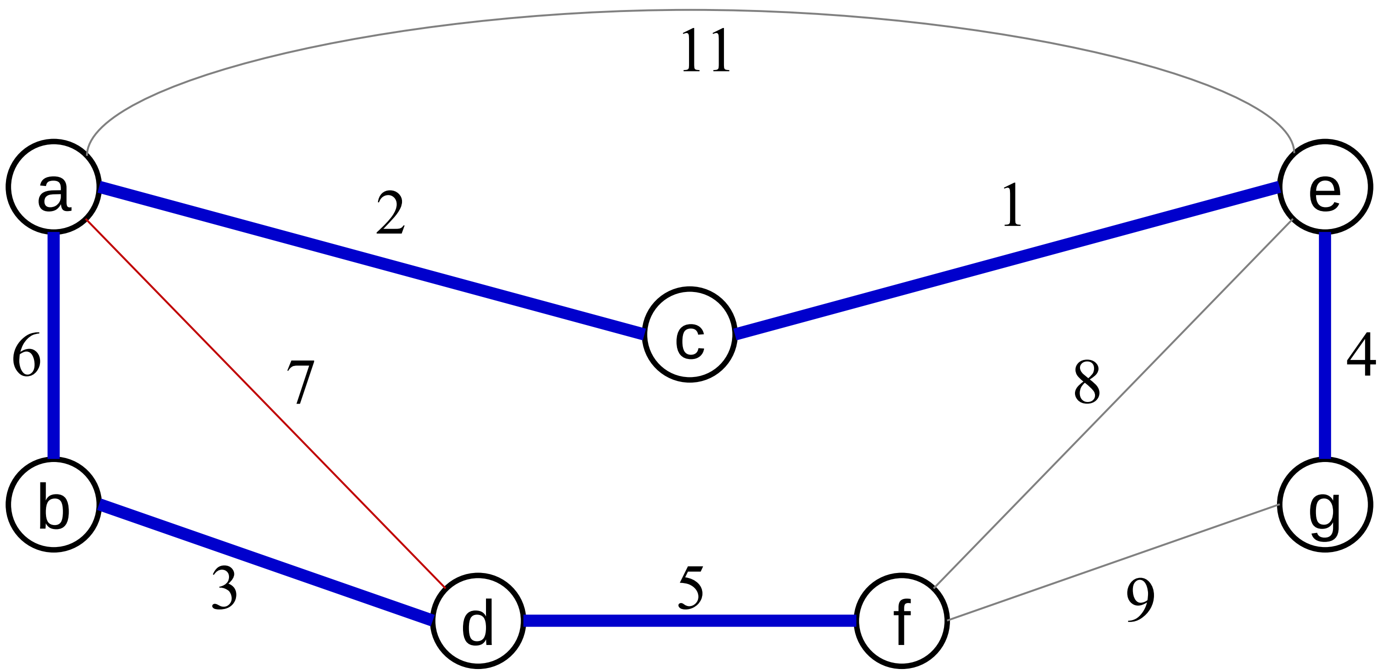


Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e),(g,f),(e,f)

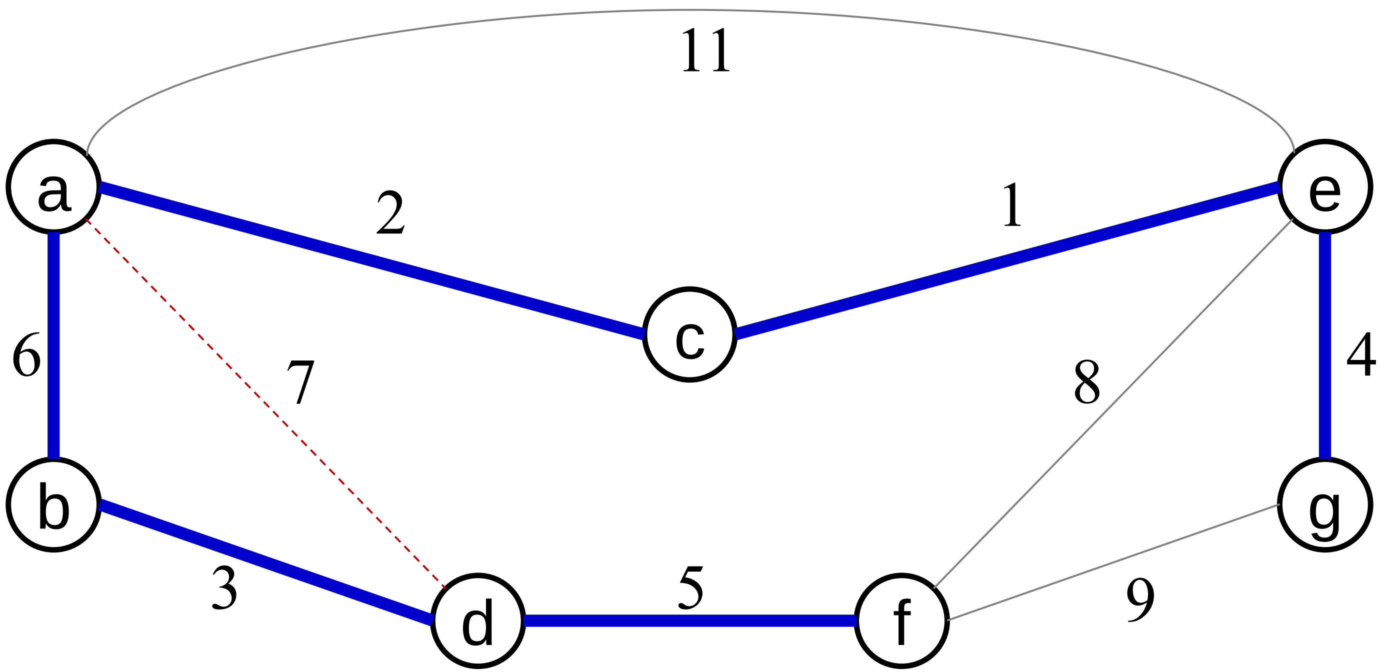
(a,d)



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

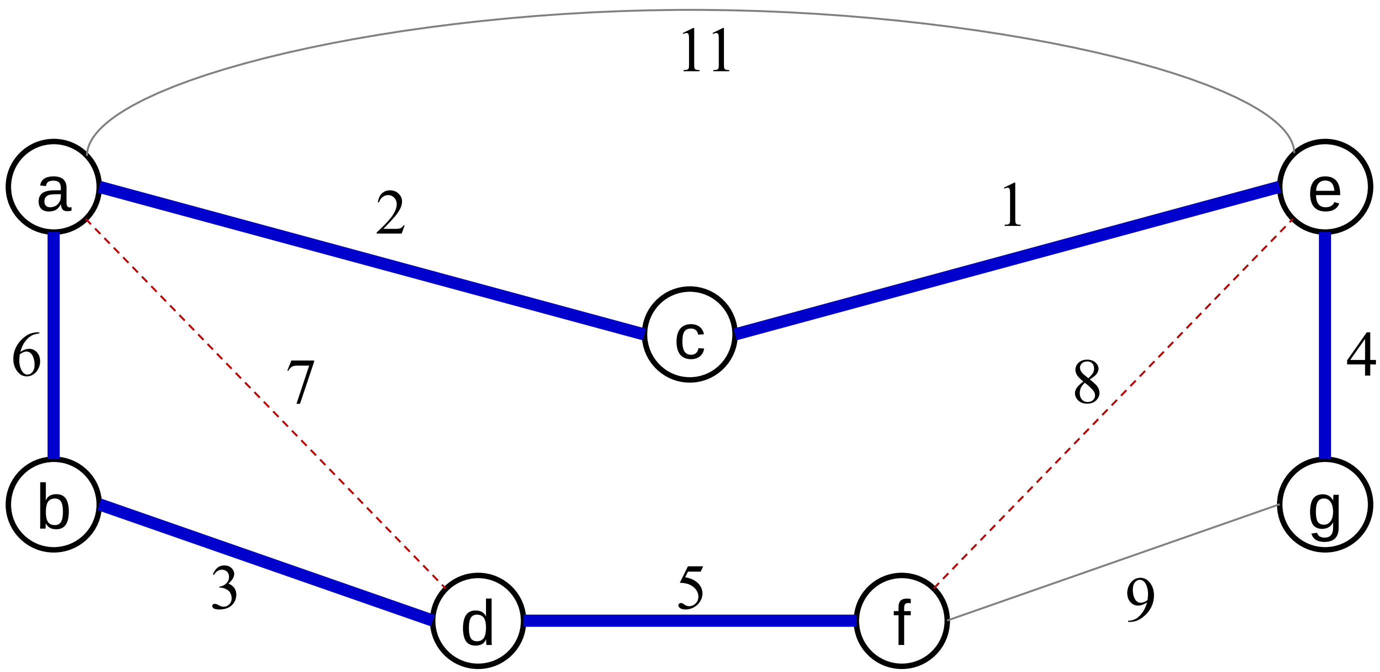
(a,e),(g,f),(e,f) **(a,d)**



Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e),(g,f) **(e,f)**

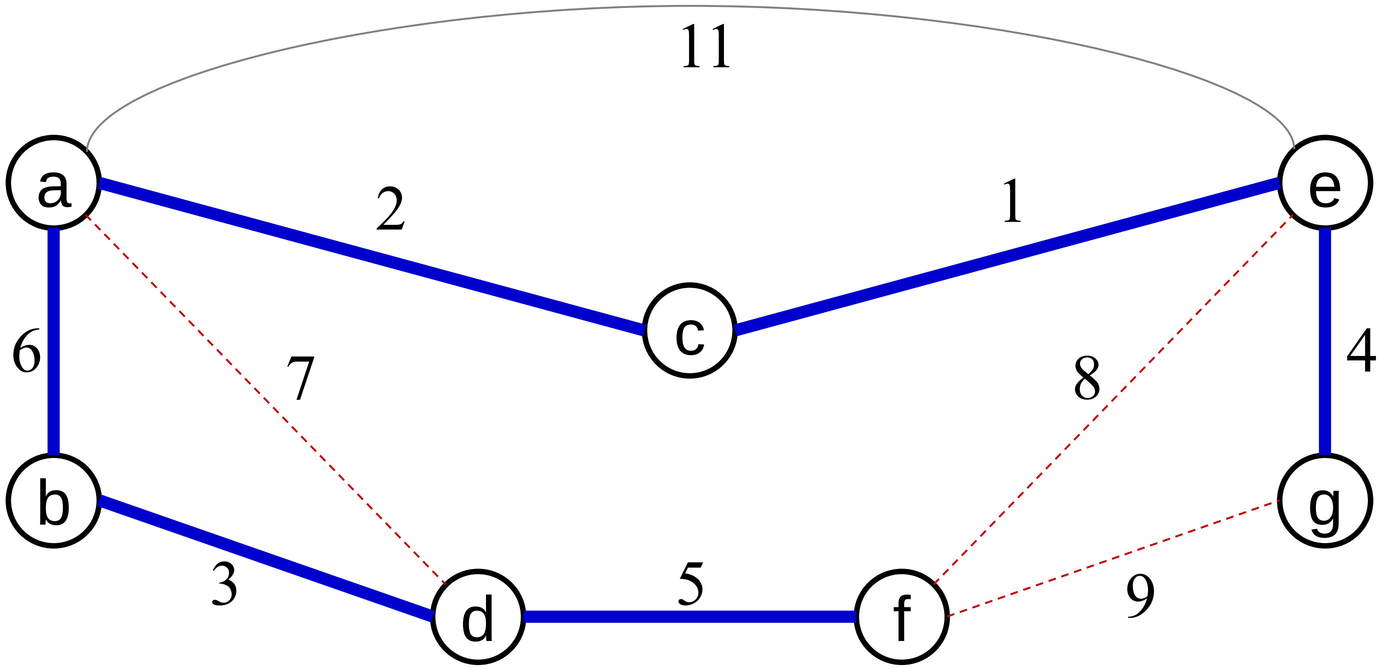


Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

(a,e)

(g,f)

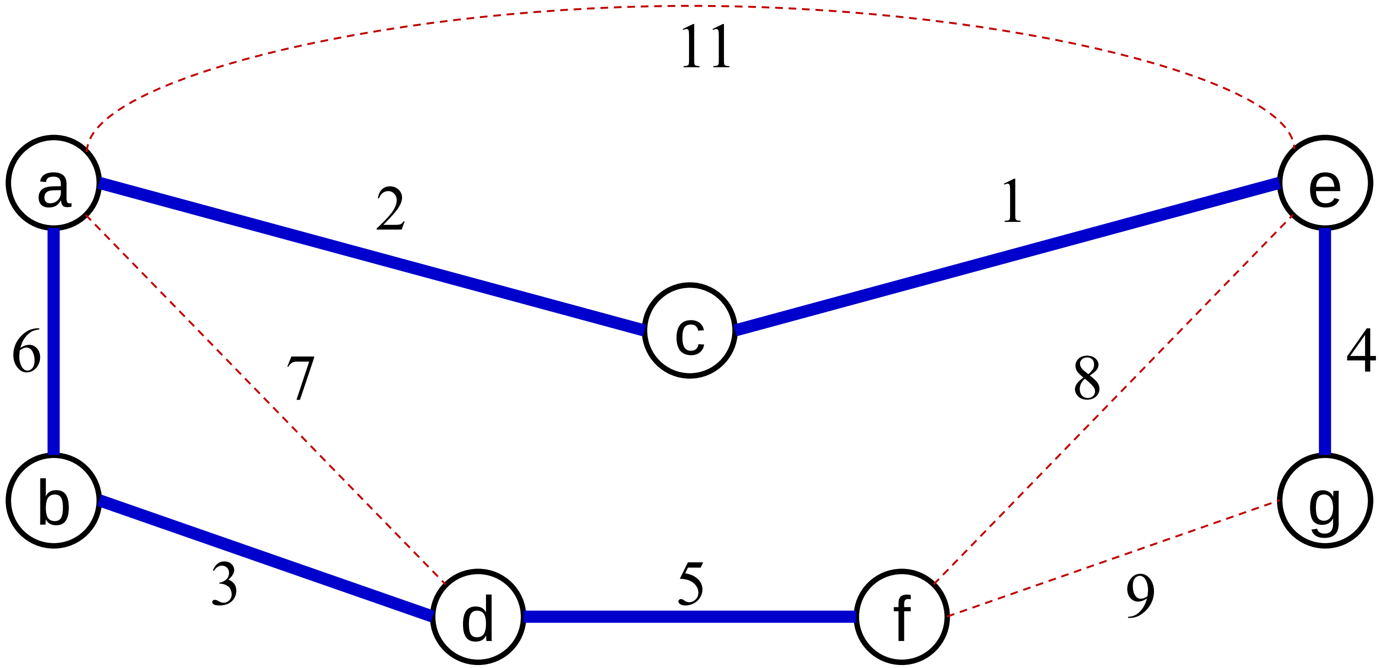


Esercizio: algoritmo di Kruskal

ciclo: per ogni arco (x,y) (estratto in modo non decrescente), se **x** e **y** non sono connessi in **T**, allora aggiungi l'arco (x,y) a **T.**

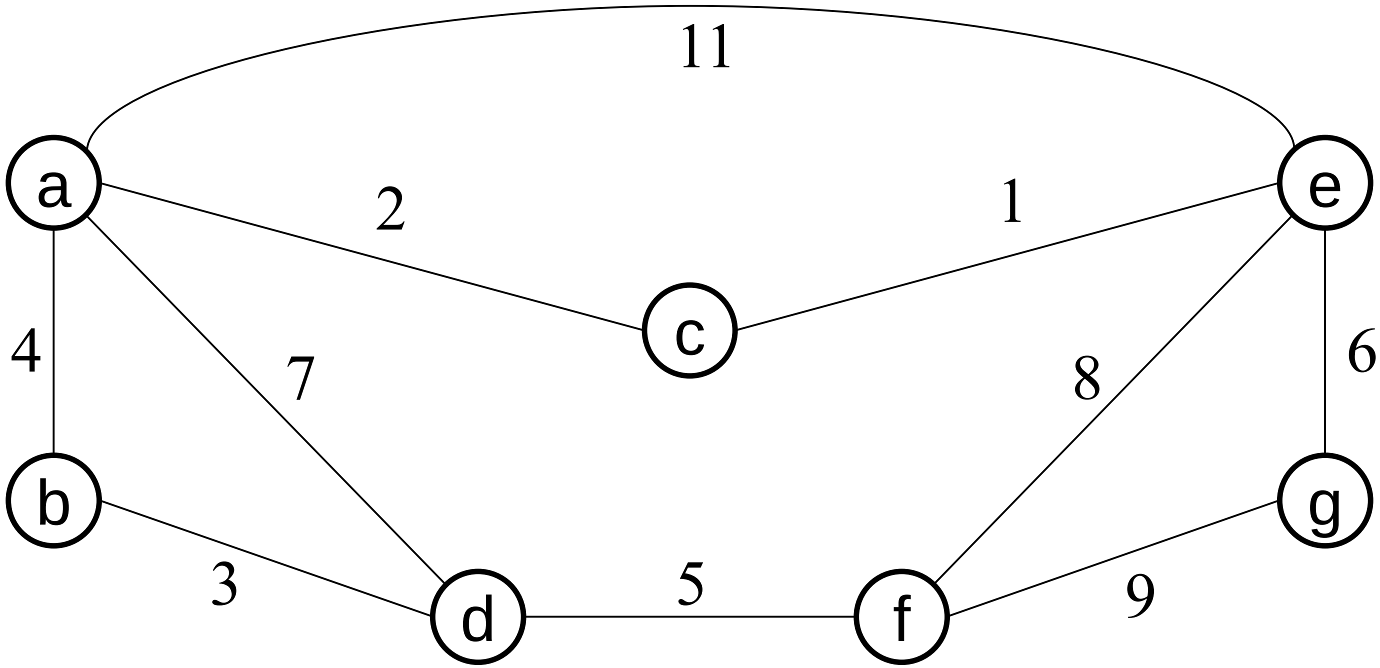
Fine!!

(a,e)



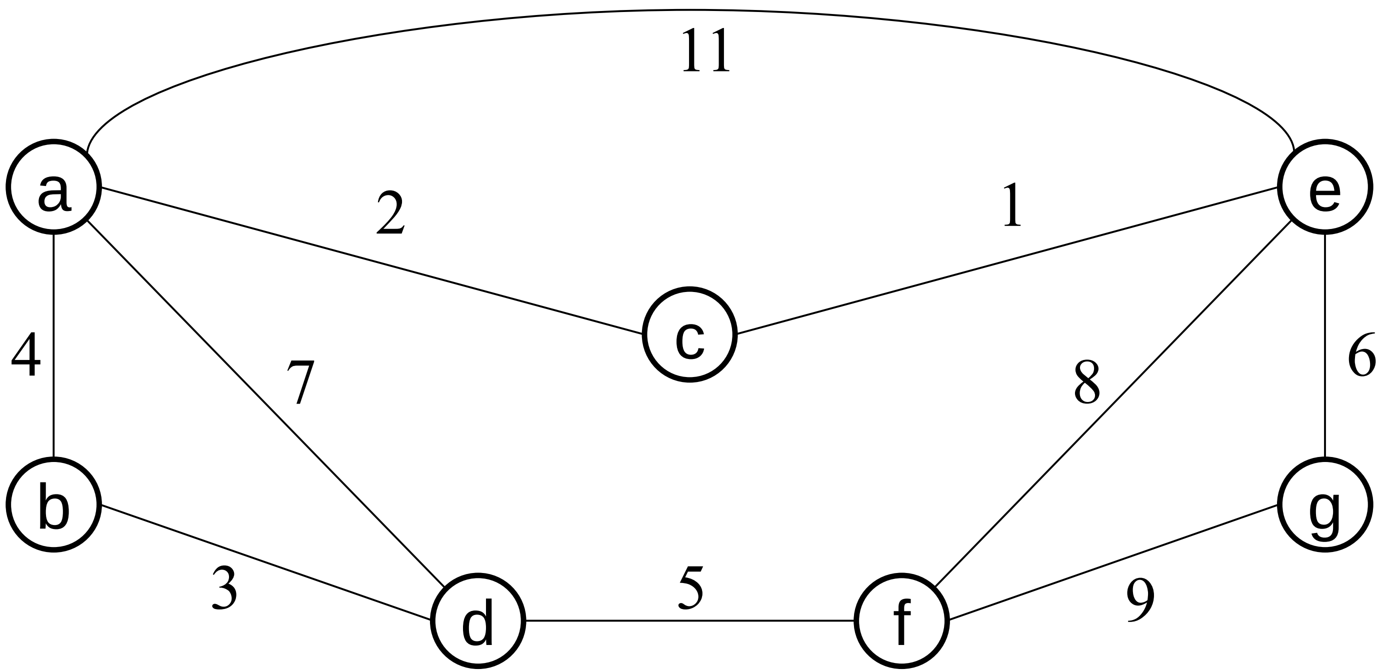
Esercizio: algoritmo di Prim

Esercizio: Calcolare il minimo albero ricoprente del grafo rappresentato in figura. Adottare l'algoritmo di **Prim** partendo dal nodo **c.**



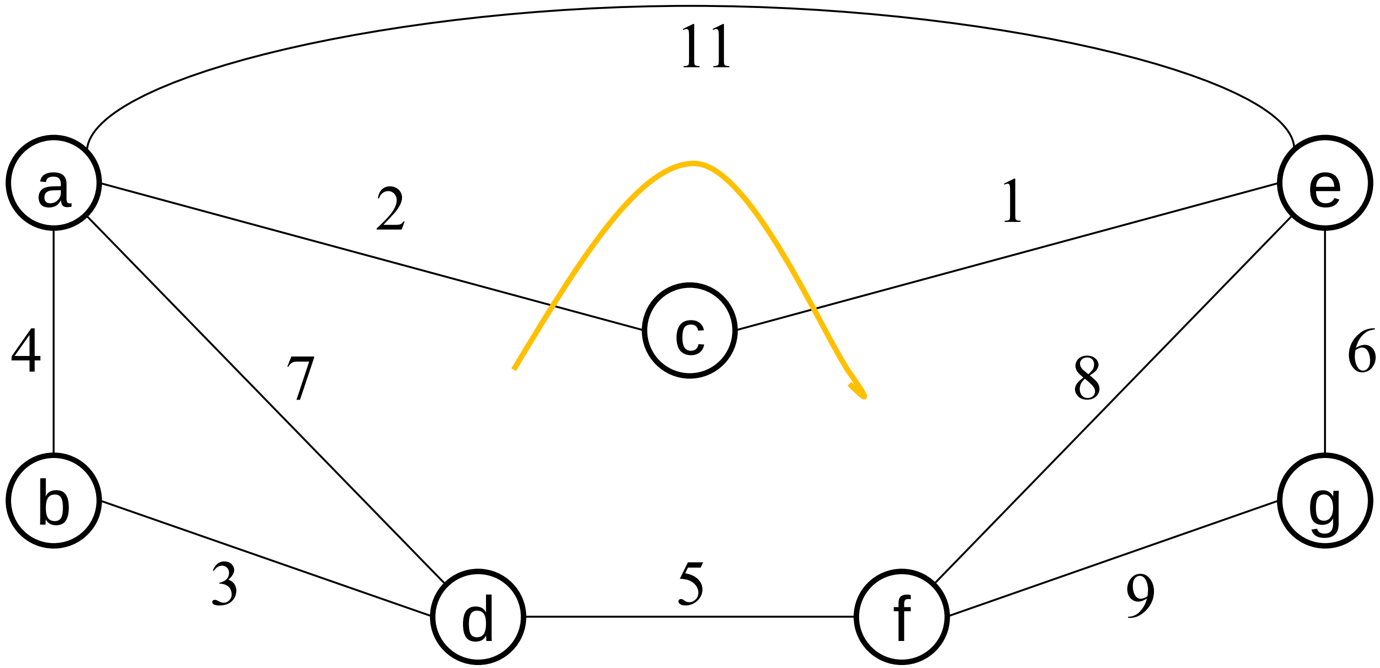
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



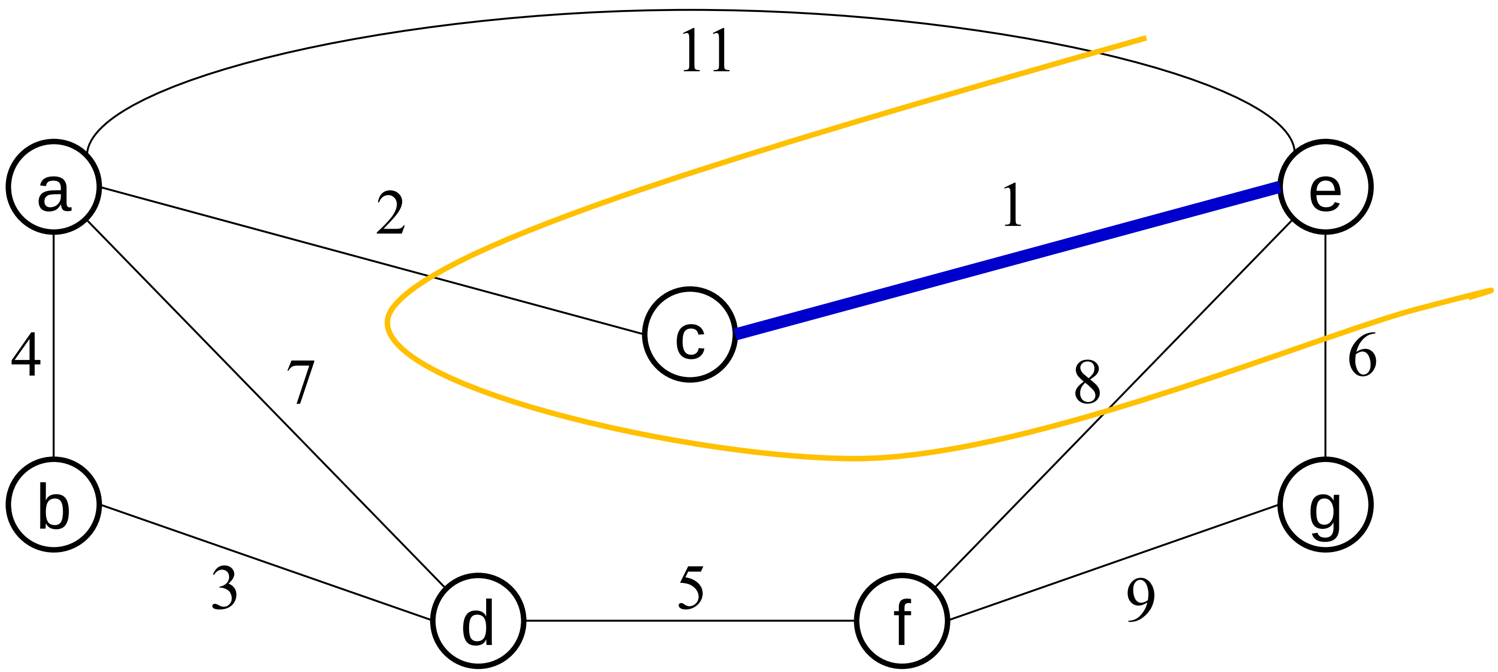
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



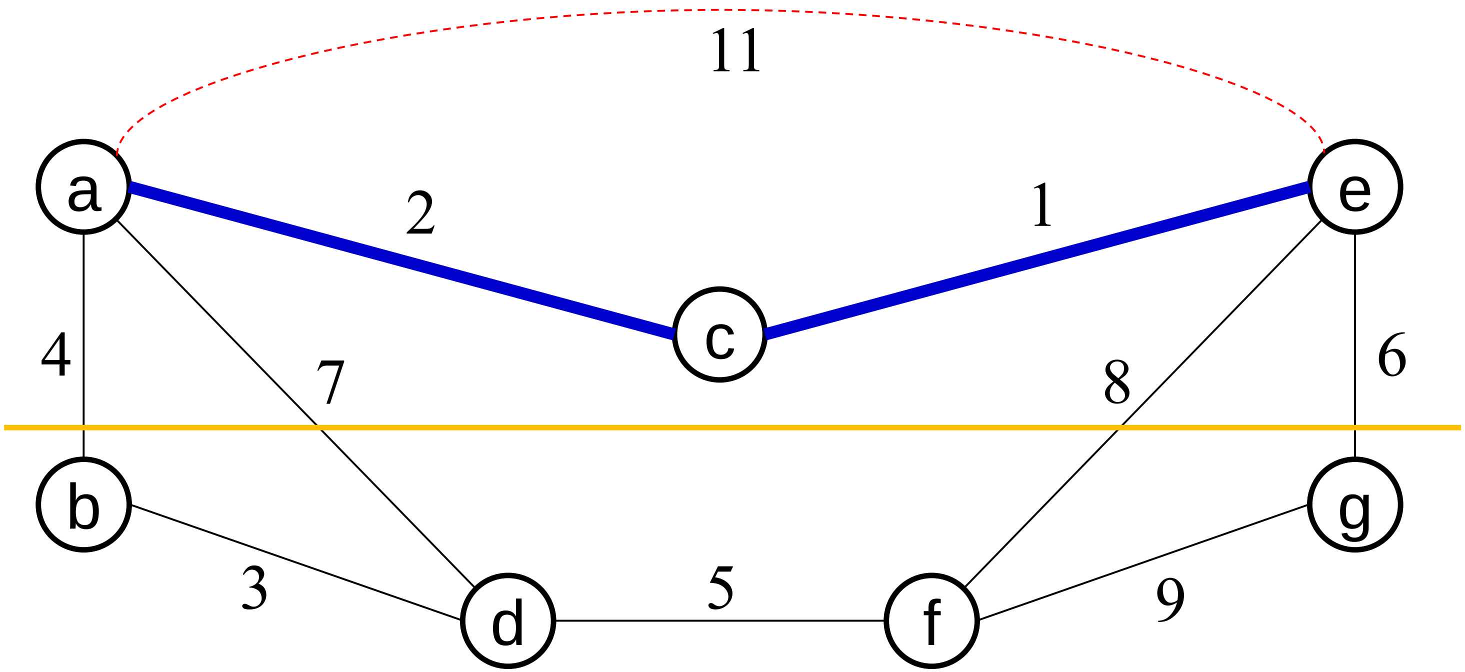
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



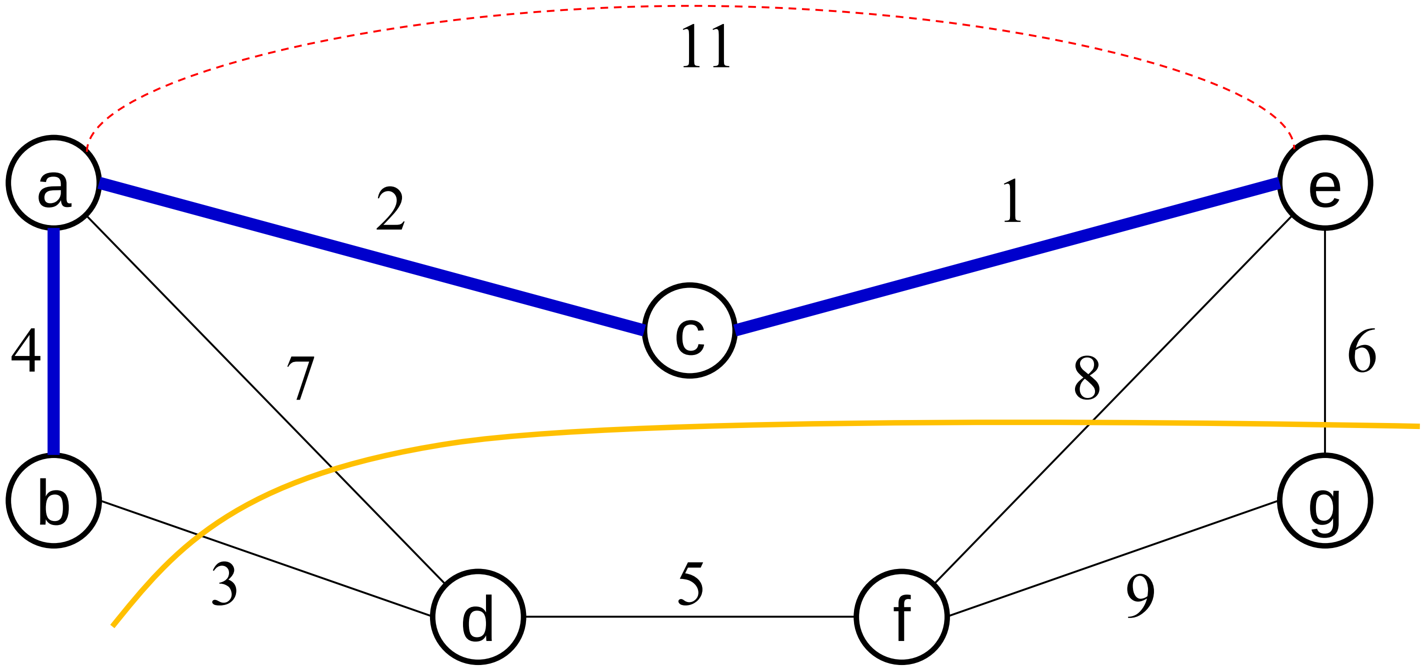
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



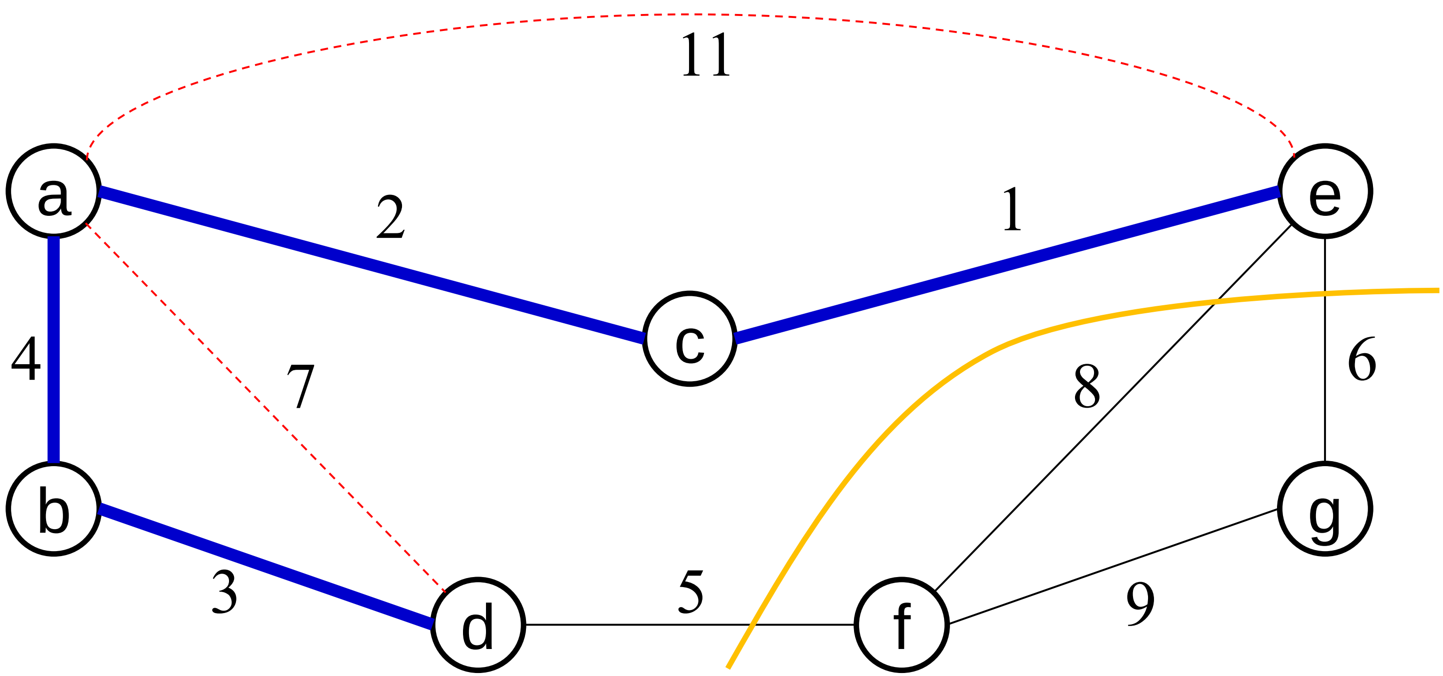
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



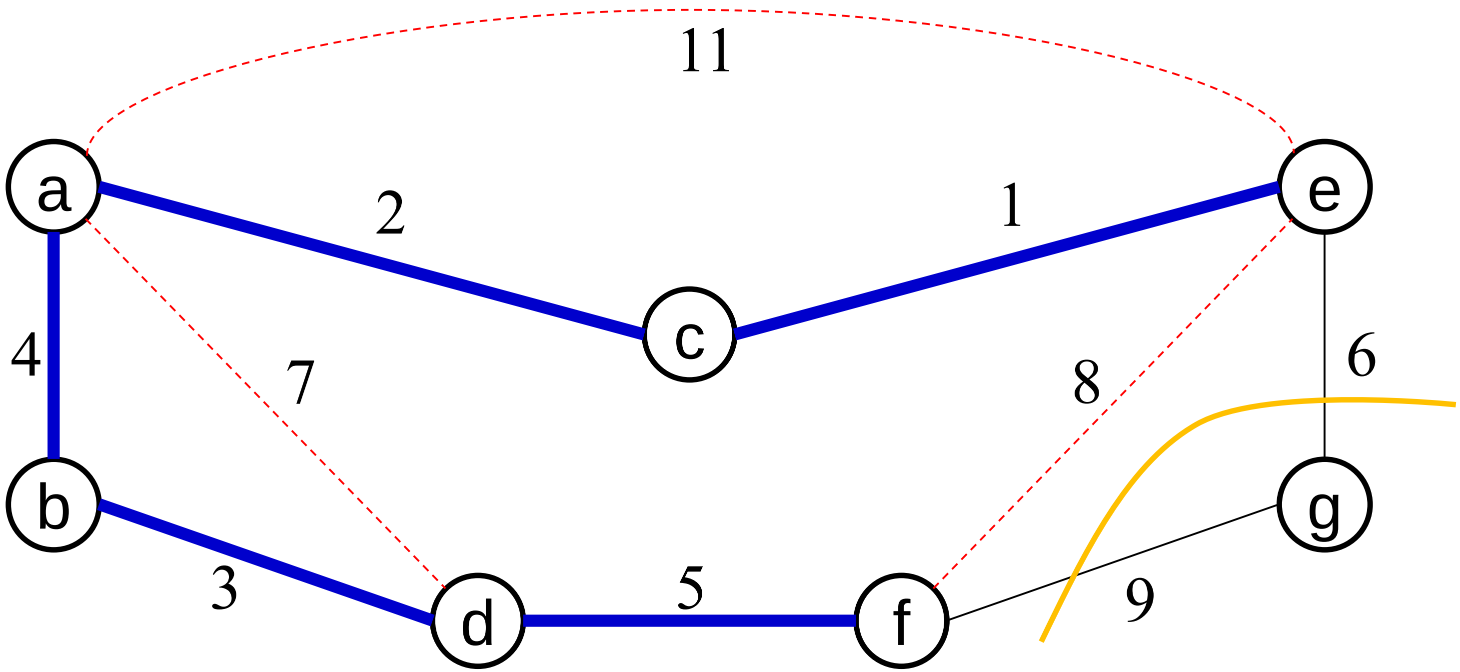
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



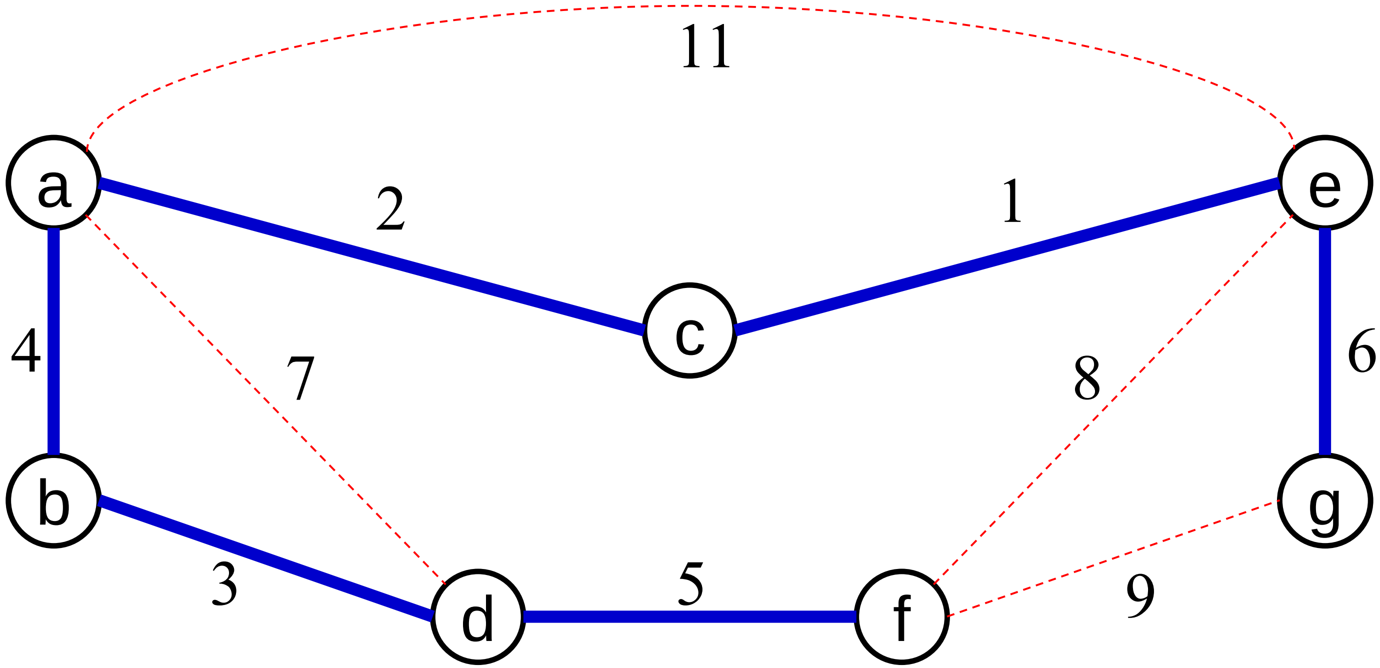
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



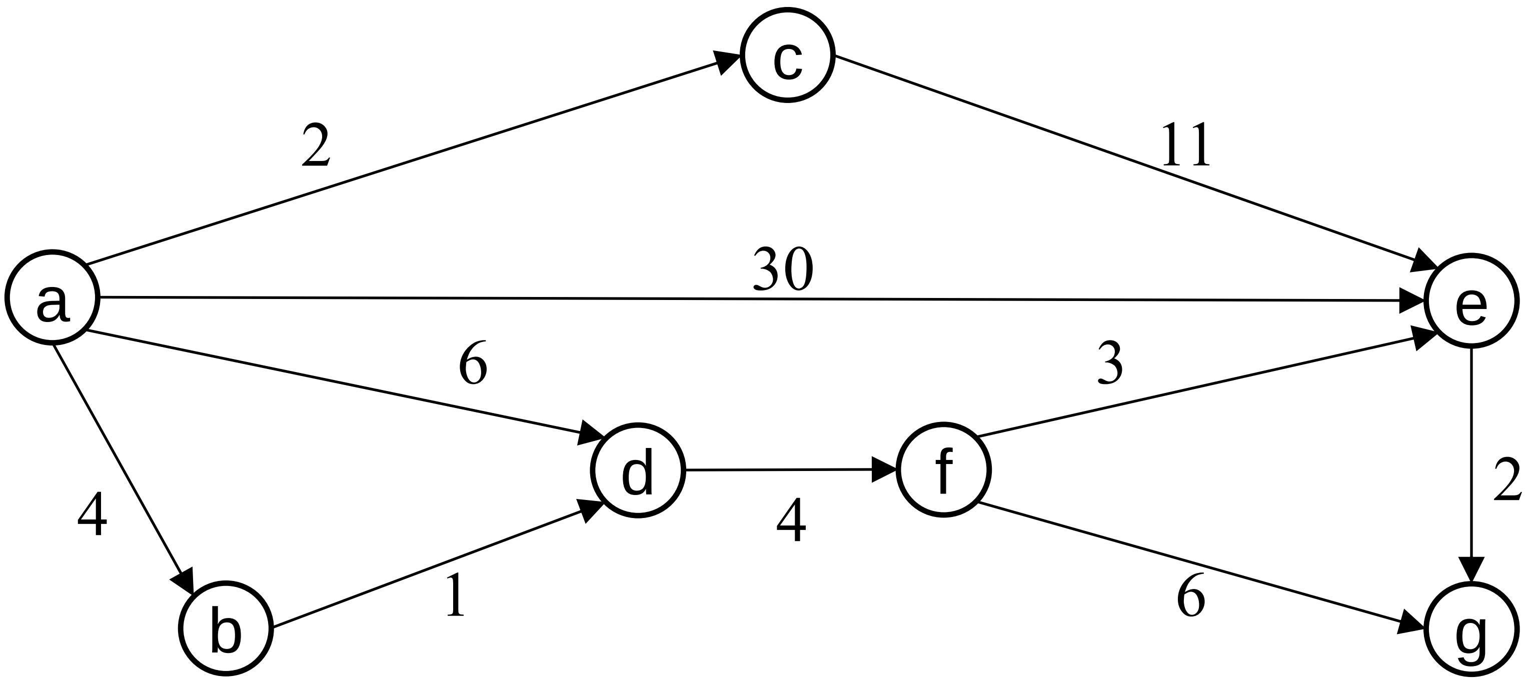
Esercizio: algoritmo di Prim

Strategia: Costruisco passo passo l'albero **T**: finché l'albero non contiene tutti i nodi del grafo, scegli tra gli **archi azzurri** quello con costo minimo ed aggiungilo a **T**



Esercizio: ordinamento topologico

Calcolare un ordinamento topologico del grafo in figura applicando l'algoritmo `ordinamentoTopologico`



Esercizio: ordinamento topologico

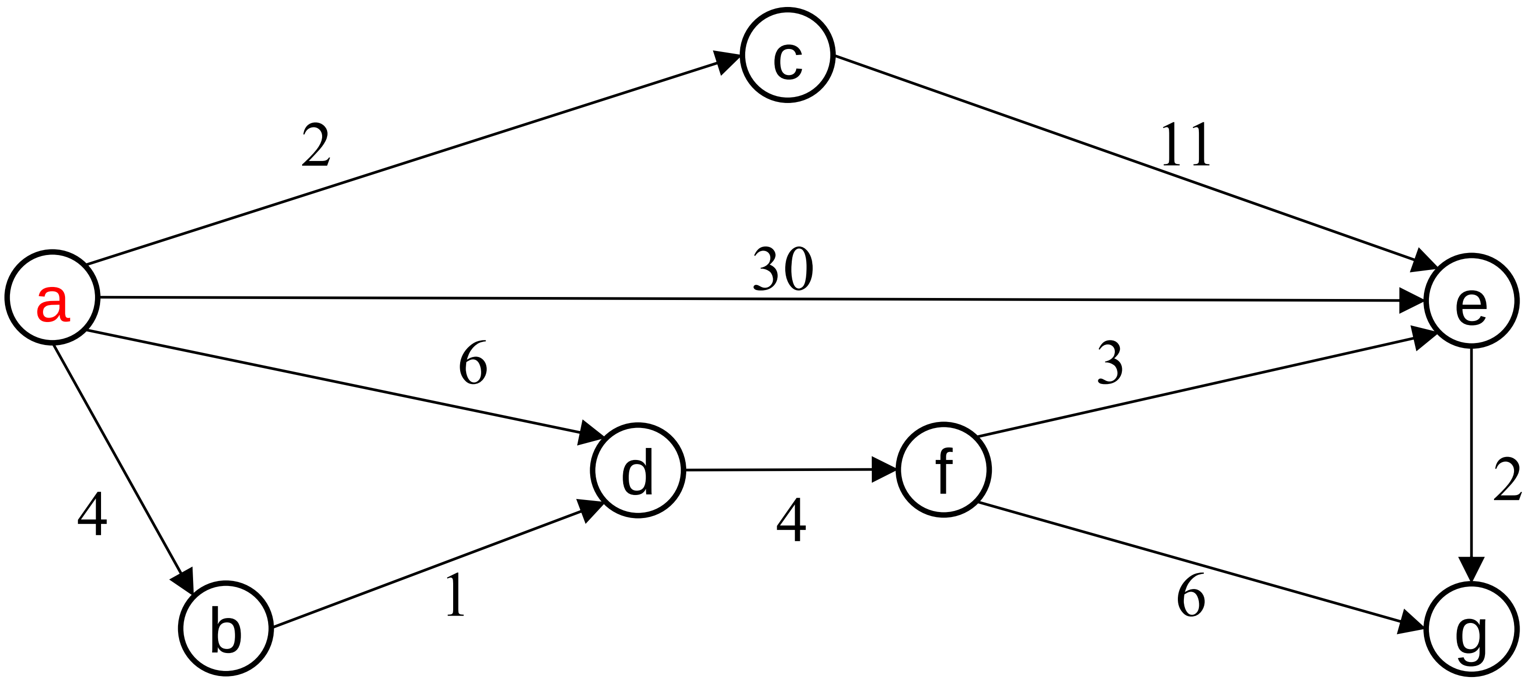
Soluzione:

L'algoritmo genera la lista di nodi *ord* applicando i seguenti passi:

1. scelgo un nodo **u** senza archi entranti, lo cancello dal grafo e lo inserisco in coda alla lista *ord*.
2. ripeto il passo 1. : scelgo un altro nodo **v** senza archi entranti, lo cancello dal grafo e lo inserisco in coda alla lista *ord*.
3. ripeto finché non ho eliminato tutti i nodi dal grafo.

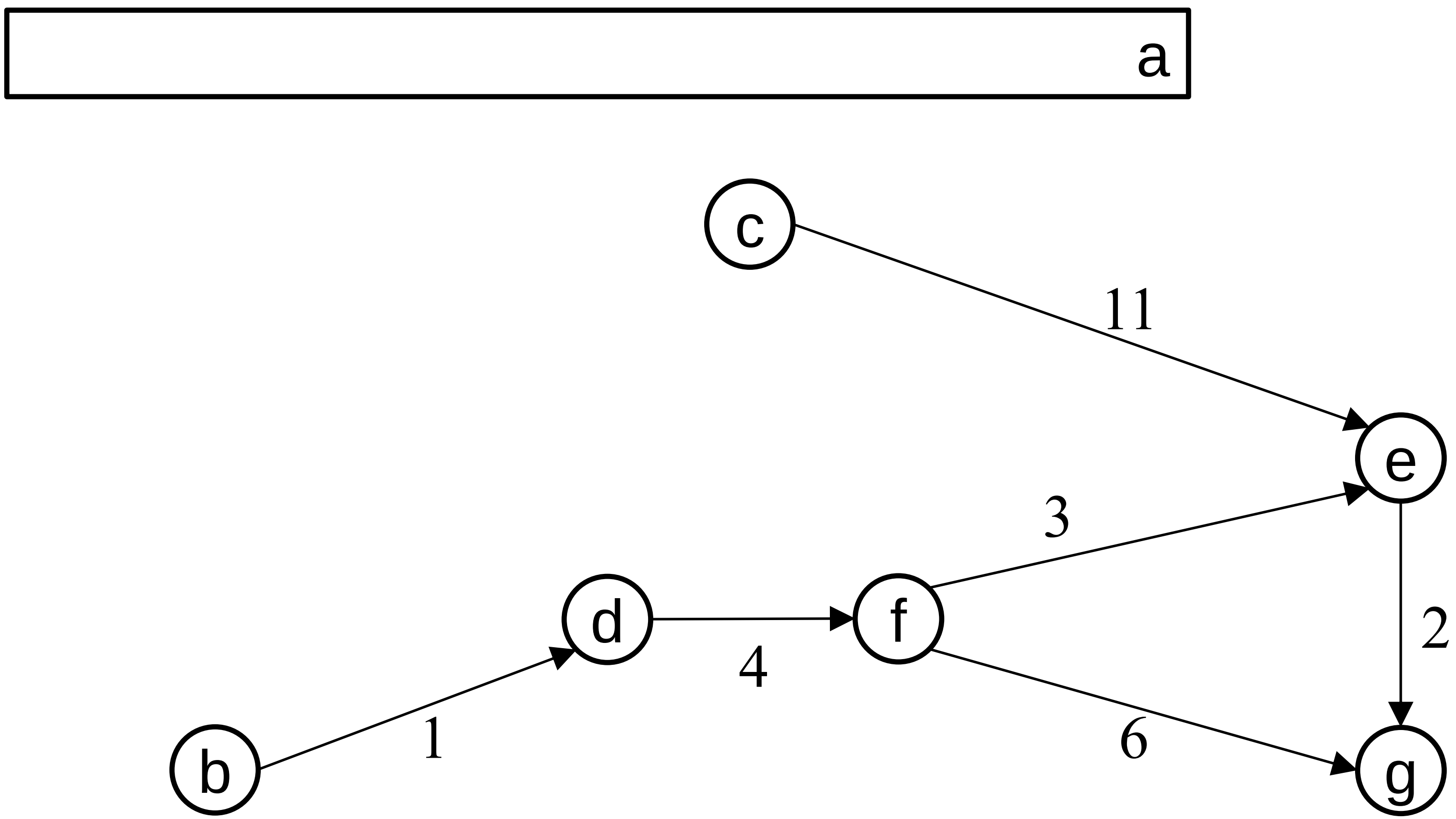
Esercizio: ordinamento topologico

L'unico nodo con soli archi uscenti è il nodo **a**.



Esercizio: ordinamento topologico

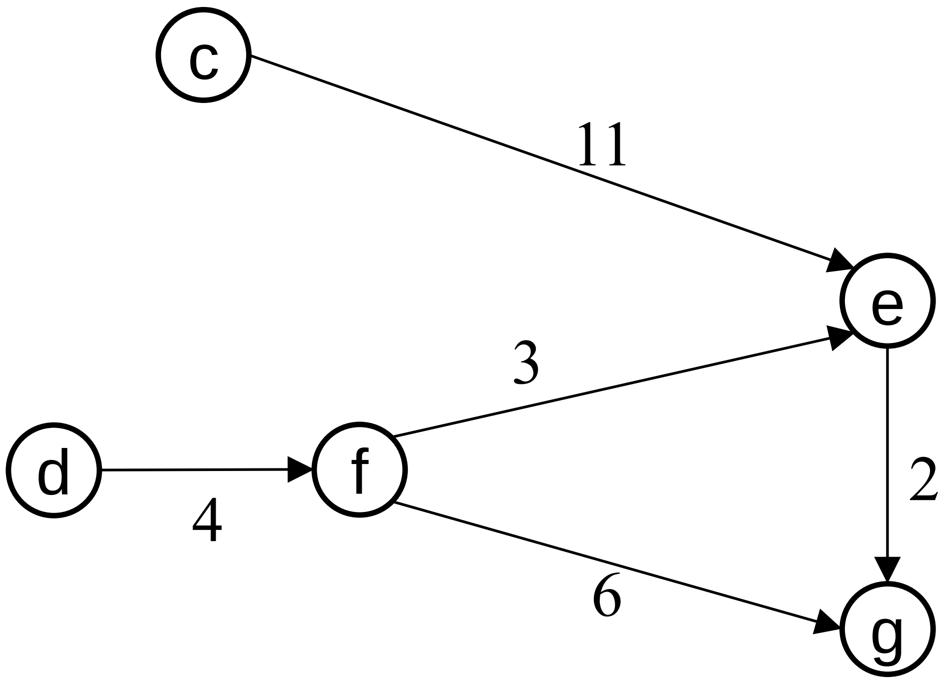
ord



Esercizio: ordinamento topologico

ord

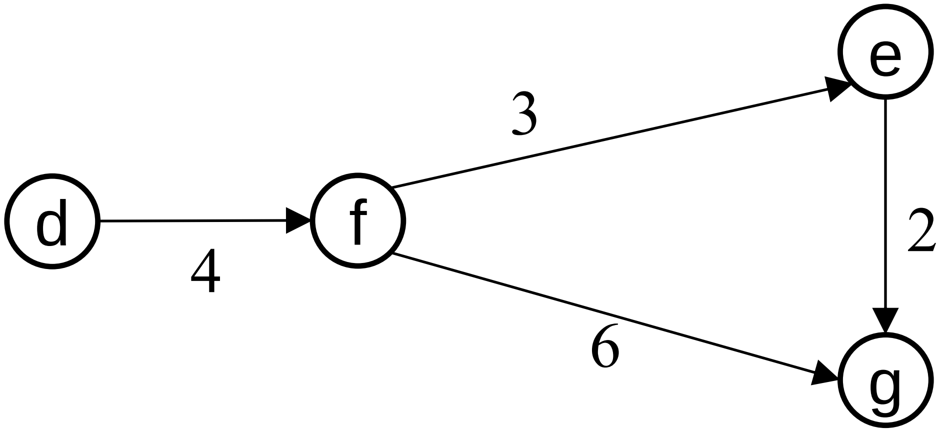
a,b



Esercizio: ordinamento topologico

ord

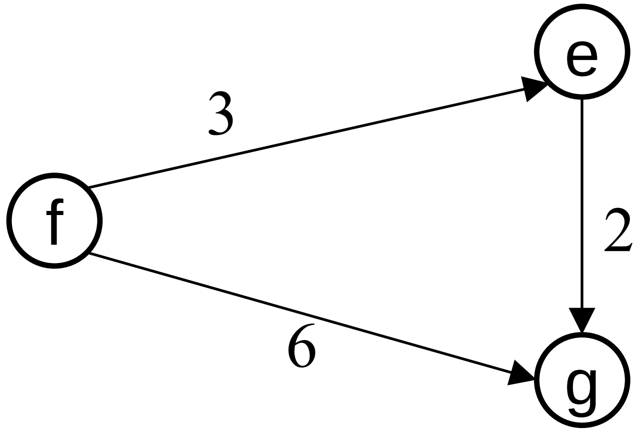
a,b,c



Esercizio: ordinamento topologico

ord

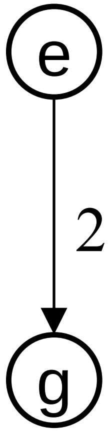
a,b,c,d



Esercizio: ordinamento topologico

ord

a,b,c,d,f



Esercizio: ordinamento topologico

ord

a,b,c,d,f,e

g

Esercizio: ordinamento topologico

ord

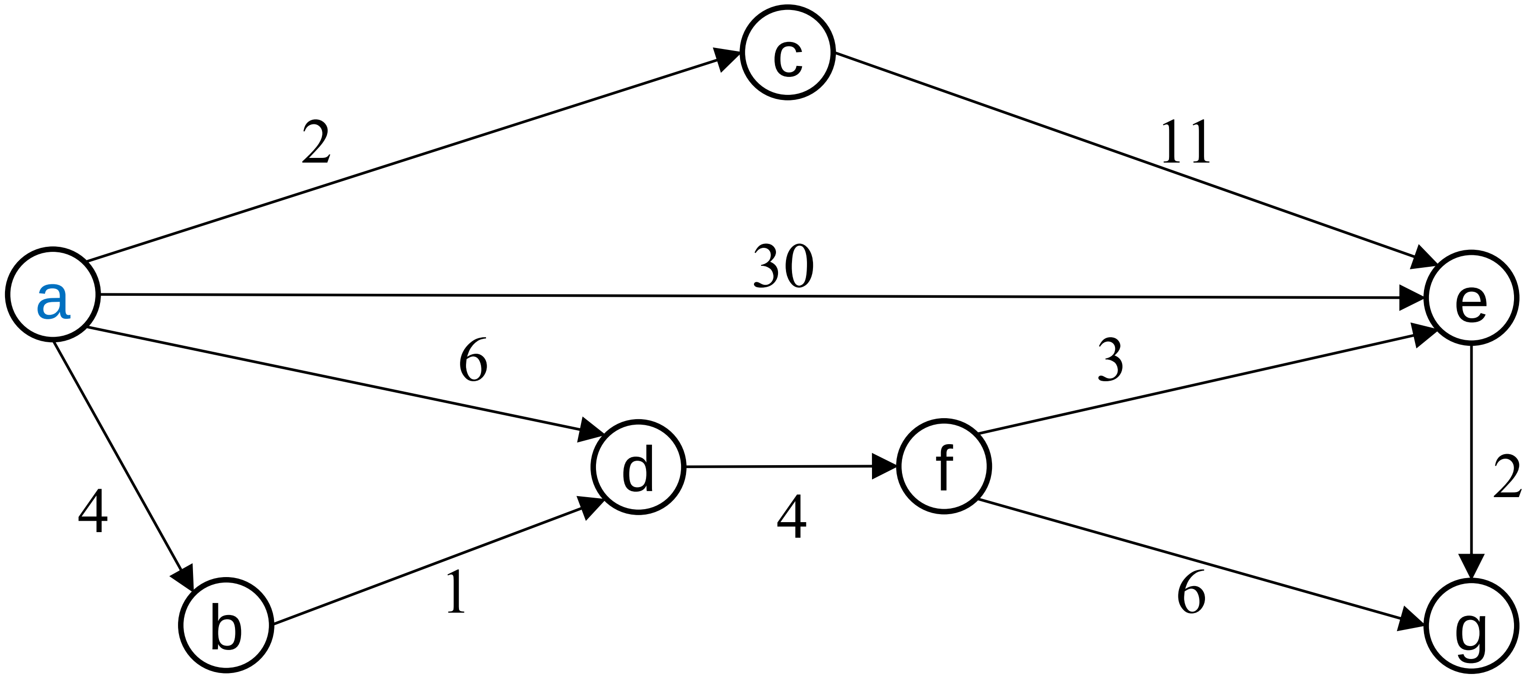
a,b,c,d,f,e,g

L'ordinamento topologico ottenuto è a,b,c,d,f,e,g.

Altri ordinamenti topologici del grafo in figura sono:
- a,b,d,c,f,e,g
- a,c,b,d,f,e,g

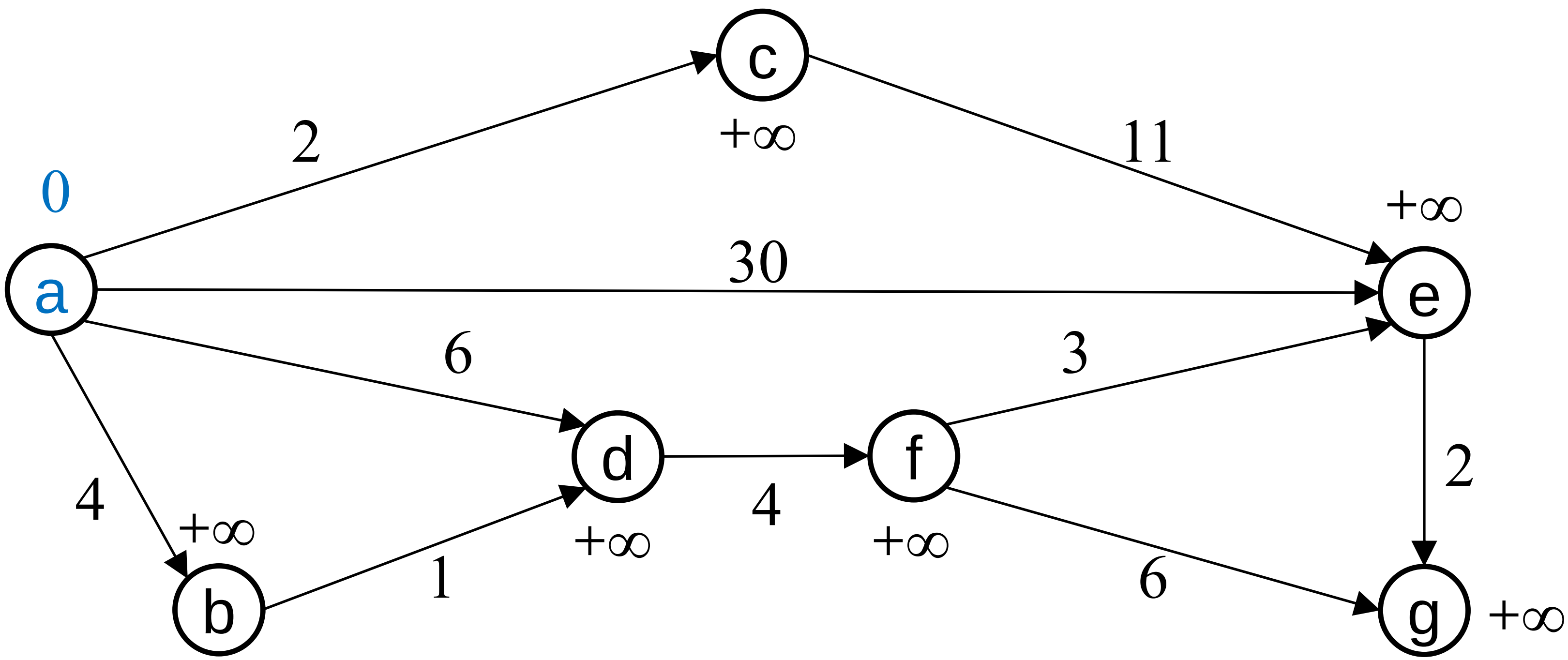
Esercizio: distanze in un grafo aciclico

Applicare l'algoritmo `distanzeAciclico` al grafo in figura utilizzando l'ordinamento topologico calcolato nel precedente esercizio. Porre il nodo **a** come nodo sorgente.



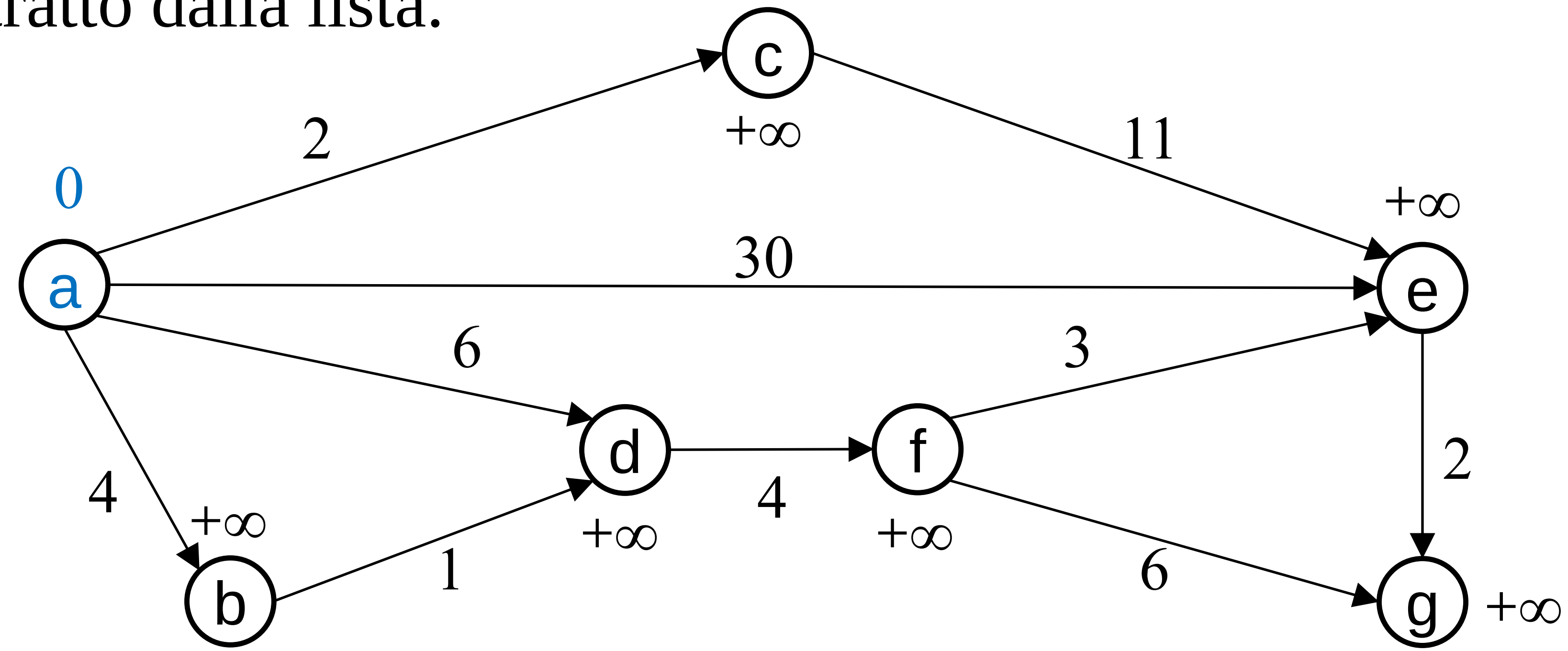
Esercizio: distanze in un grafo aciclico

Il primo passo dell'algoritmo prevede di inizializzare a $+\infty$ le distanze $\mathbf{D}_{\mathbf{sv}}$ per ogni vertice **v** di **G** diverso da **s** e a 0 la distanza $\mathbf{D}_{\mathbf{ss}}$



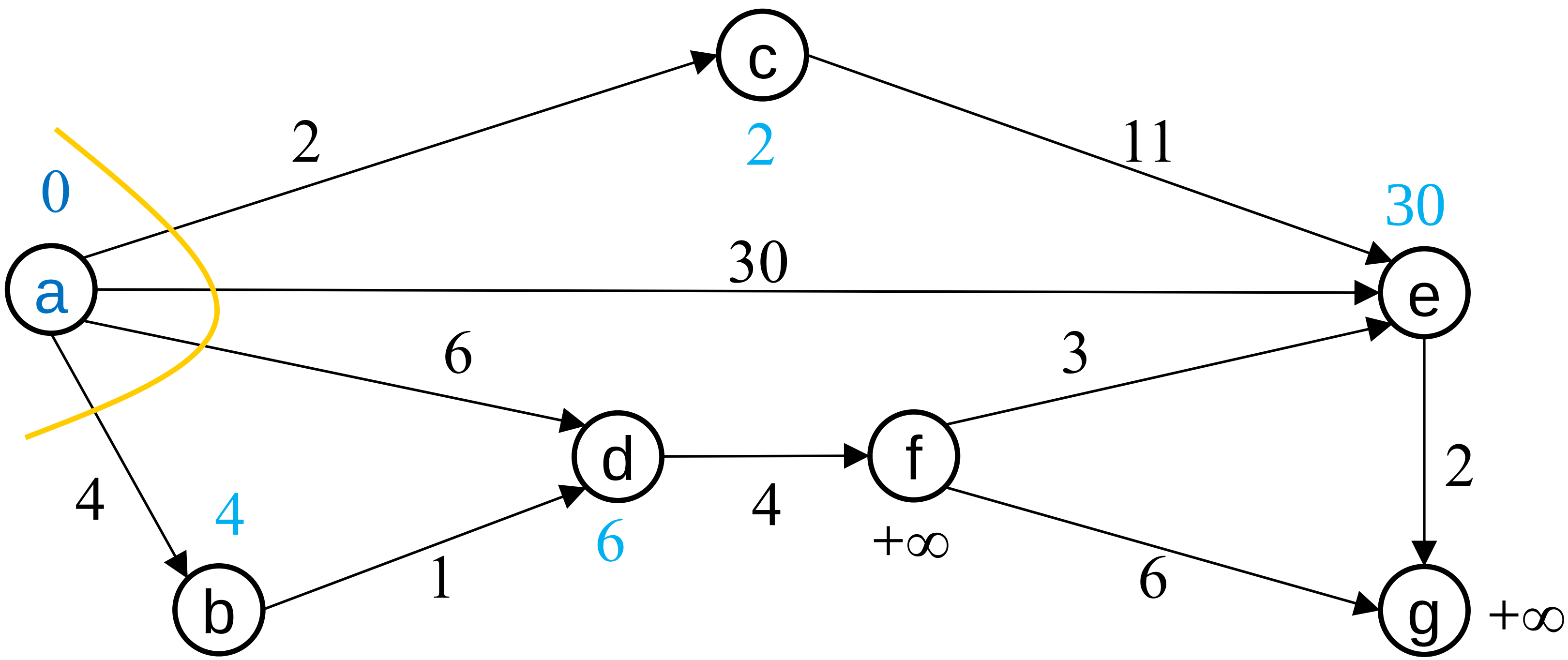
Esercizio: distanze in un grafo aciclico

Dopo aver calcolato l'ordinamento topologico contenuto nella lista *ord* (a,b,c,d,f,e,g) l'algoritmo inizia a ciclare "rilassando" tutti gli archi che hanno origine nel nodo estratto dalla lista.



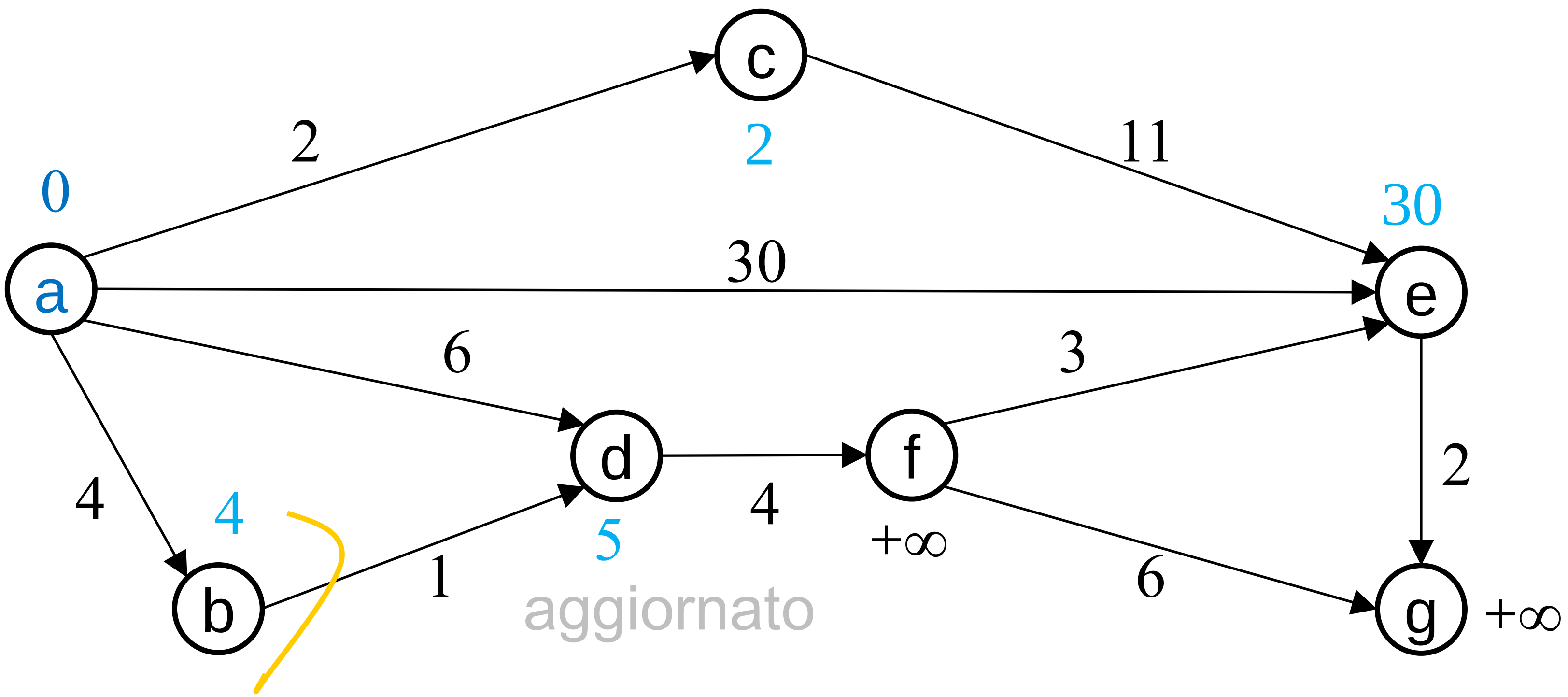
Esercizio: distanze in un grafo aciclico

a ← *ord* (b,c,d,f,e,g)



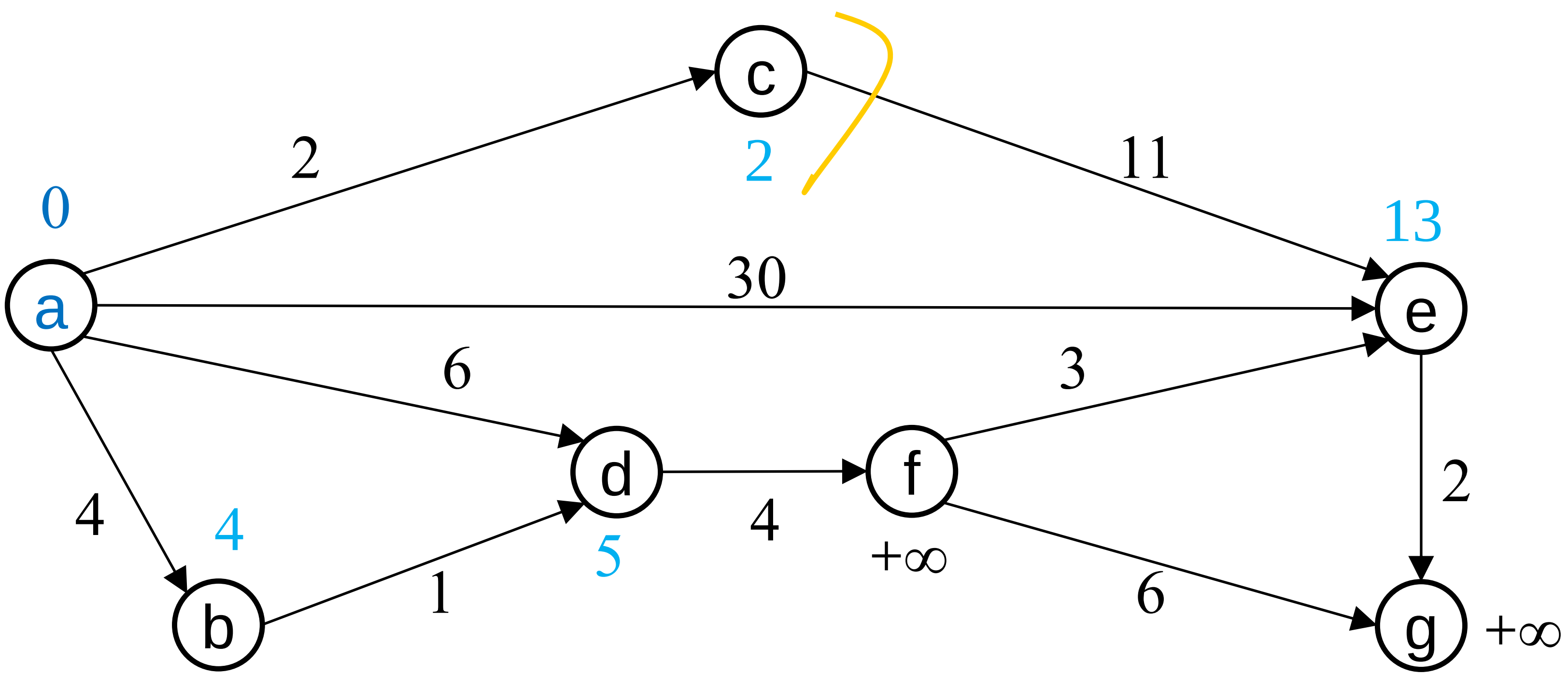
Esercizio: distanze in un grafo aciclico

$b \leftarrow ord$ (c,d,f,e,g)



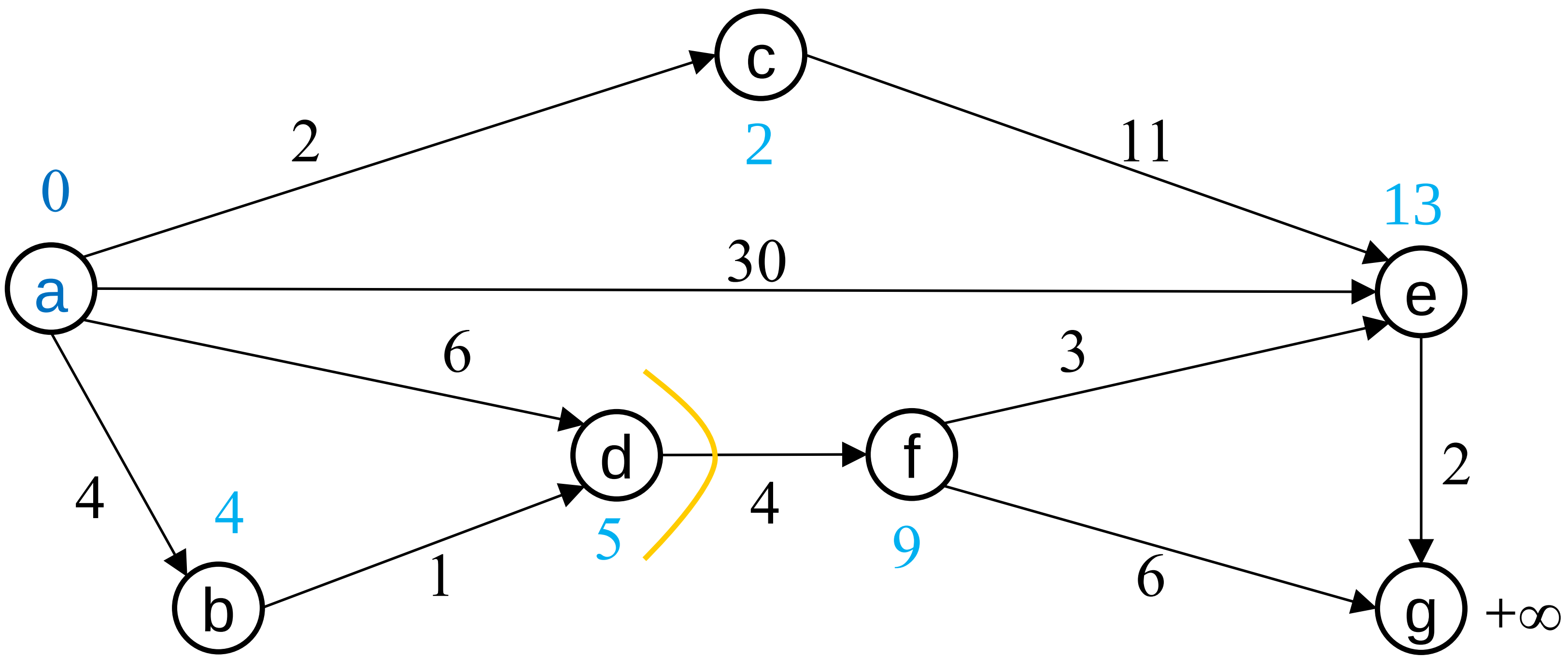
Esercizio: distanze in un grafo aciclico

$c \leftarrow ord$ (d,f,e,g)



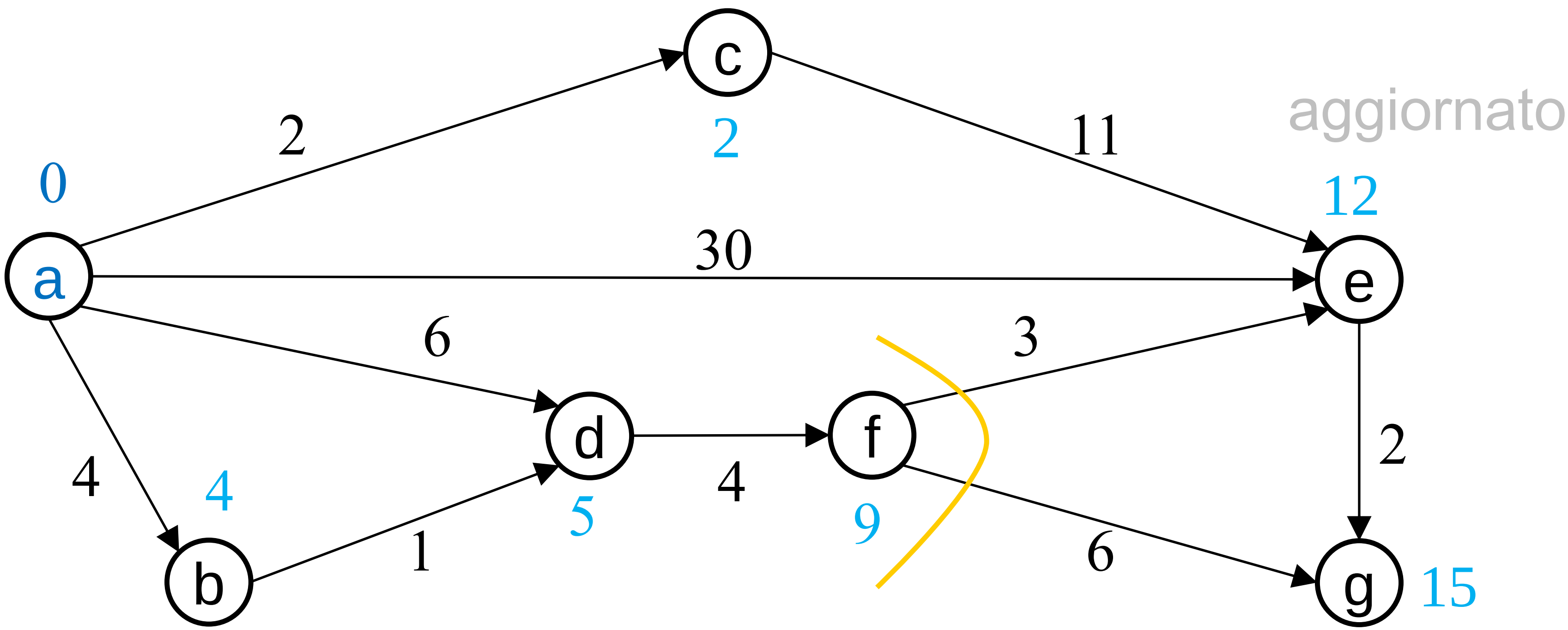
Esercizio: distanze in un grafo aciclico

$d \leftarrow ord$ (f,e,g)



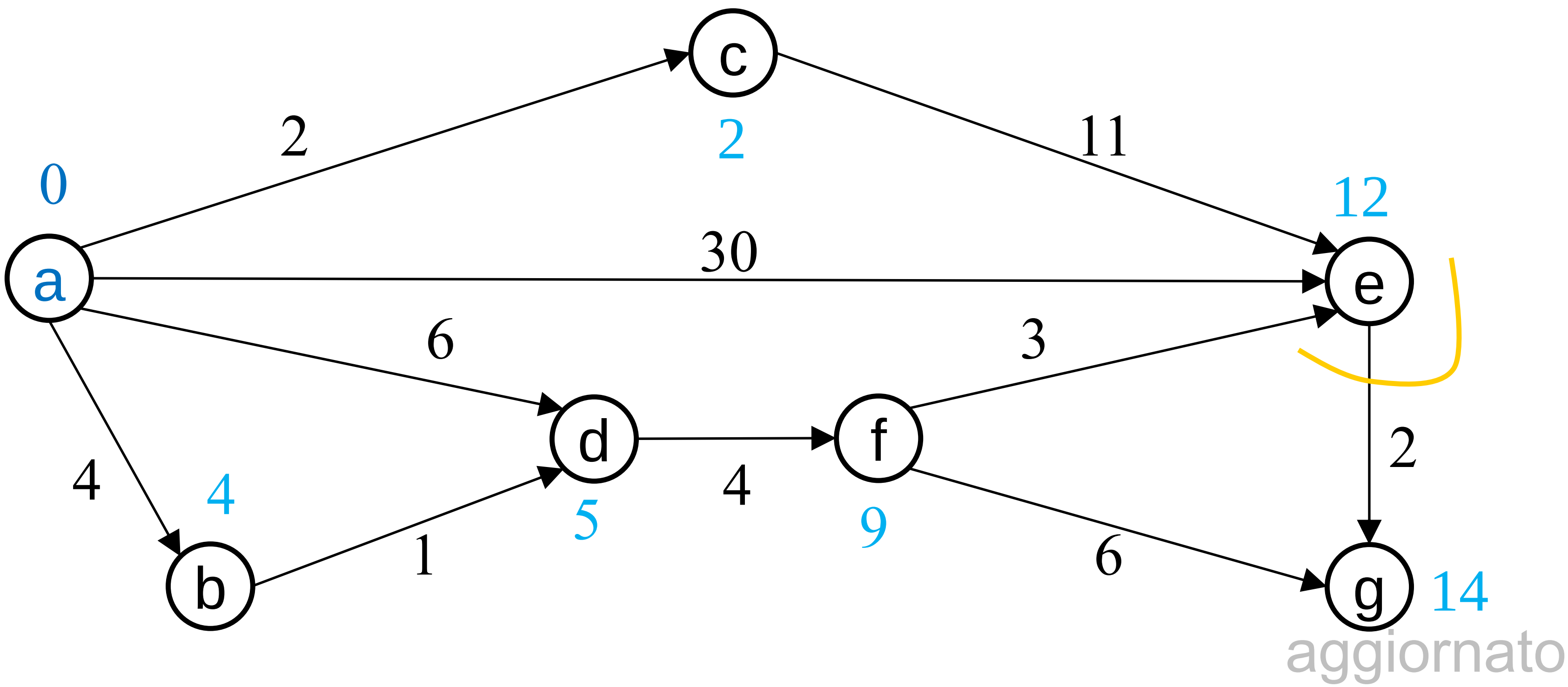
Esercizio: distanze in un grafo aciclico

$f \leftarrow ord$ (e,g)



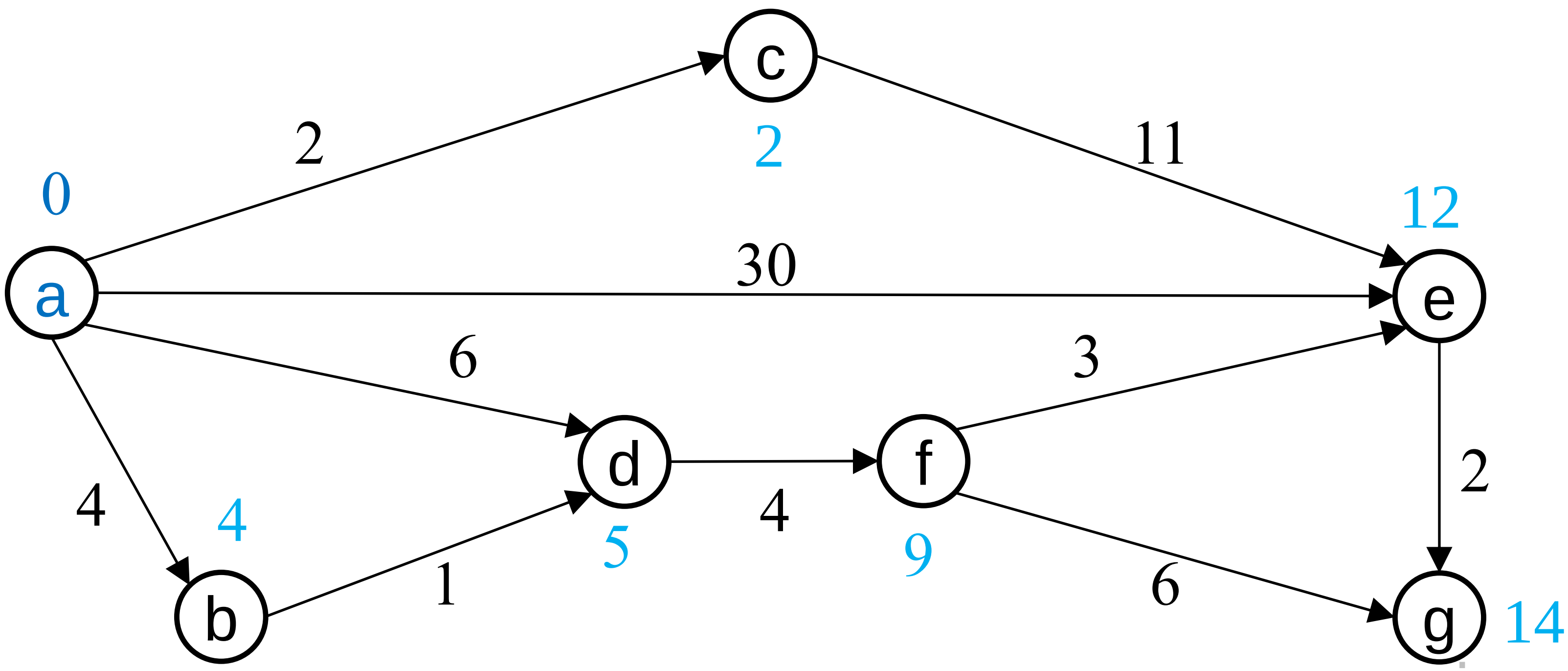
Esercizio: distanze in un grafo aciclico

$e \leftarrow ord$ (g)



Esercizio: distanze in un grafo aciclico

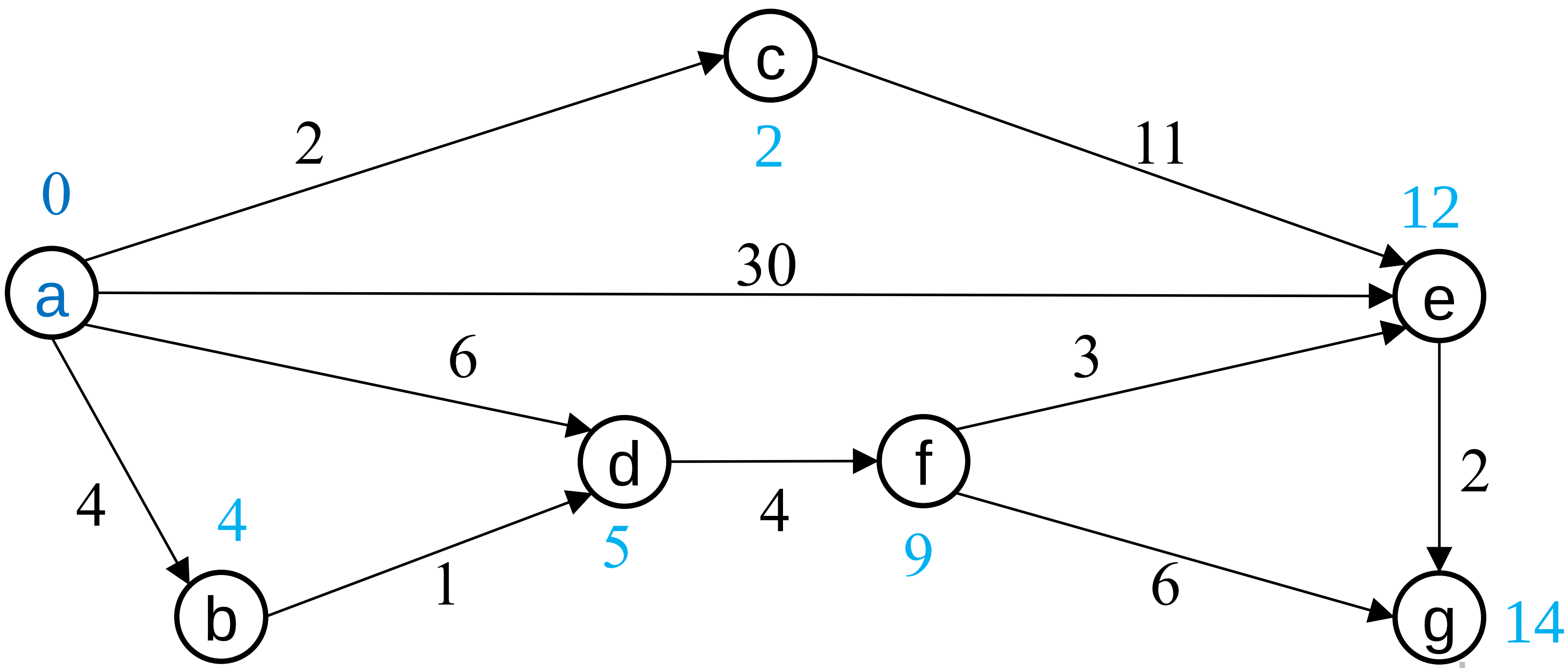
$g \leftarrow ord()$



aggiornato

Esercizio: distanze in un grafo aciclico

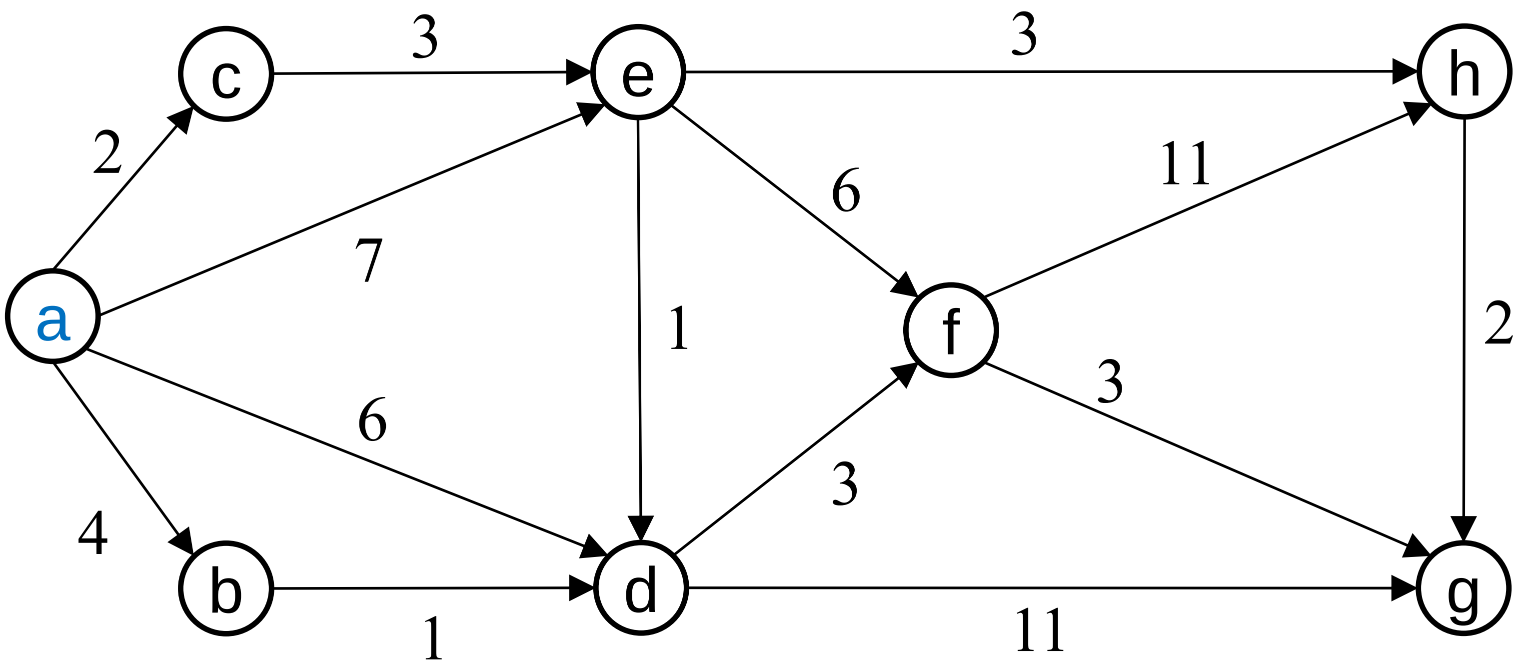
$\leftarrow$ *ord* ()



aggiornato

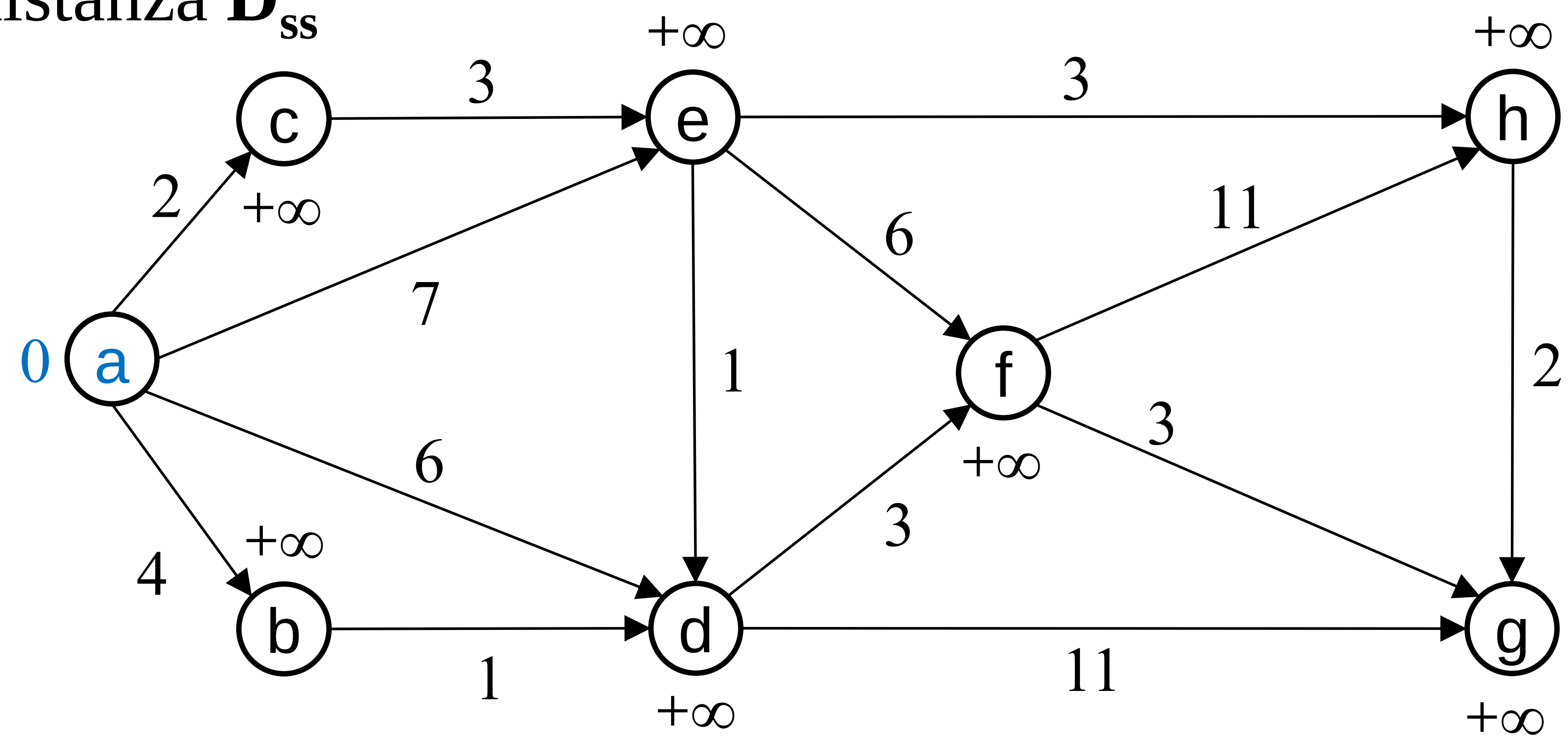
Esercizio: l'algoritmo di Dijkstra

Applicare l'algoritmo di Dijkstra per calcolare l'albero dei cammini minimi del grafo rappresentato in figura. Porre il nodo **a** come nodo sorgente.



Esercizio: l'algoritmo di Dijkstra

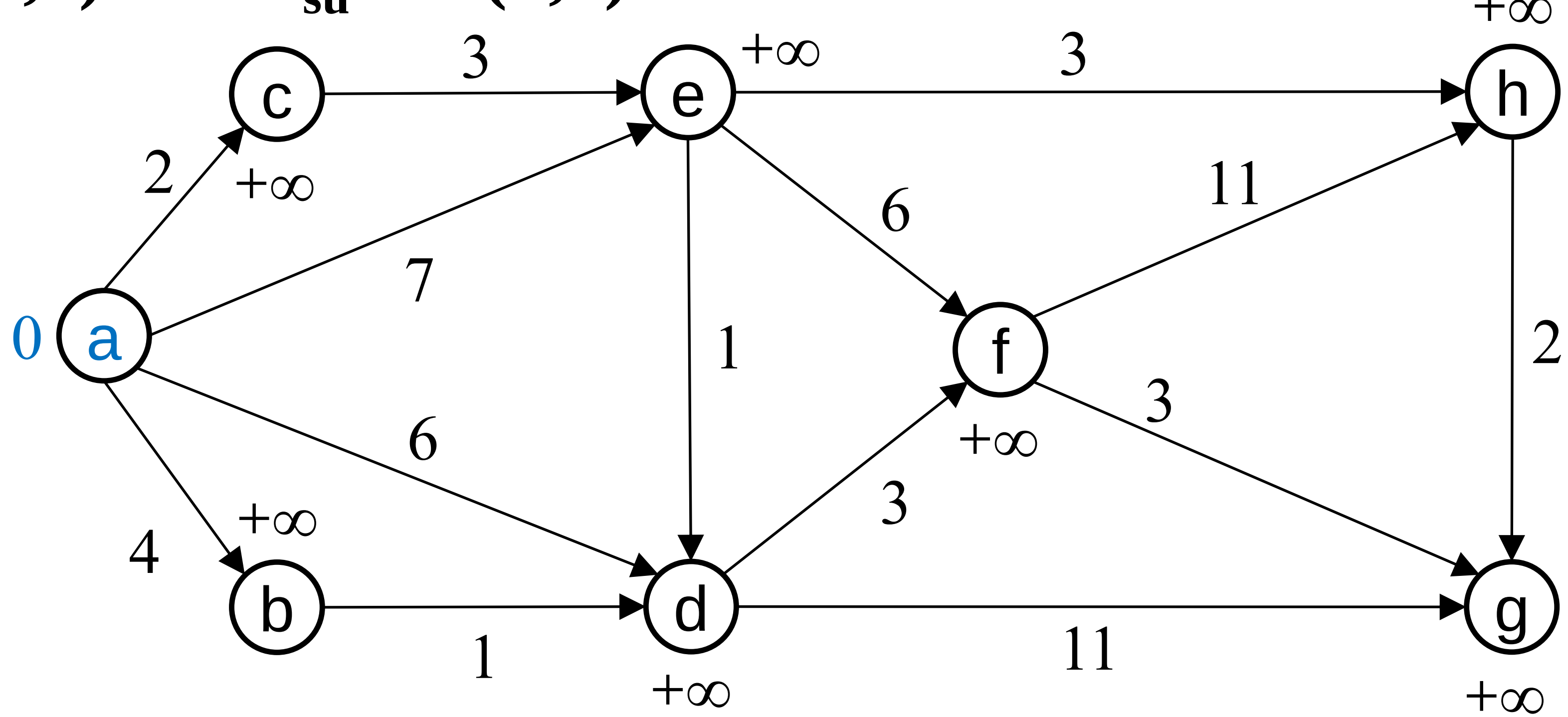
Il primo passo dell'algoritmo prevede di inizializzare a $+\infty$ le distanze $\mathbf{D}_{\mathbf{sv}}$ per ogni vertice **v** di **G** diverso da **s** e a 0 la distanza $\mathbf{D}_{\mathbf{ss}}$



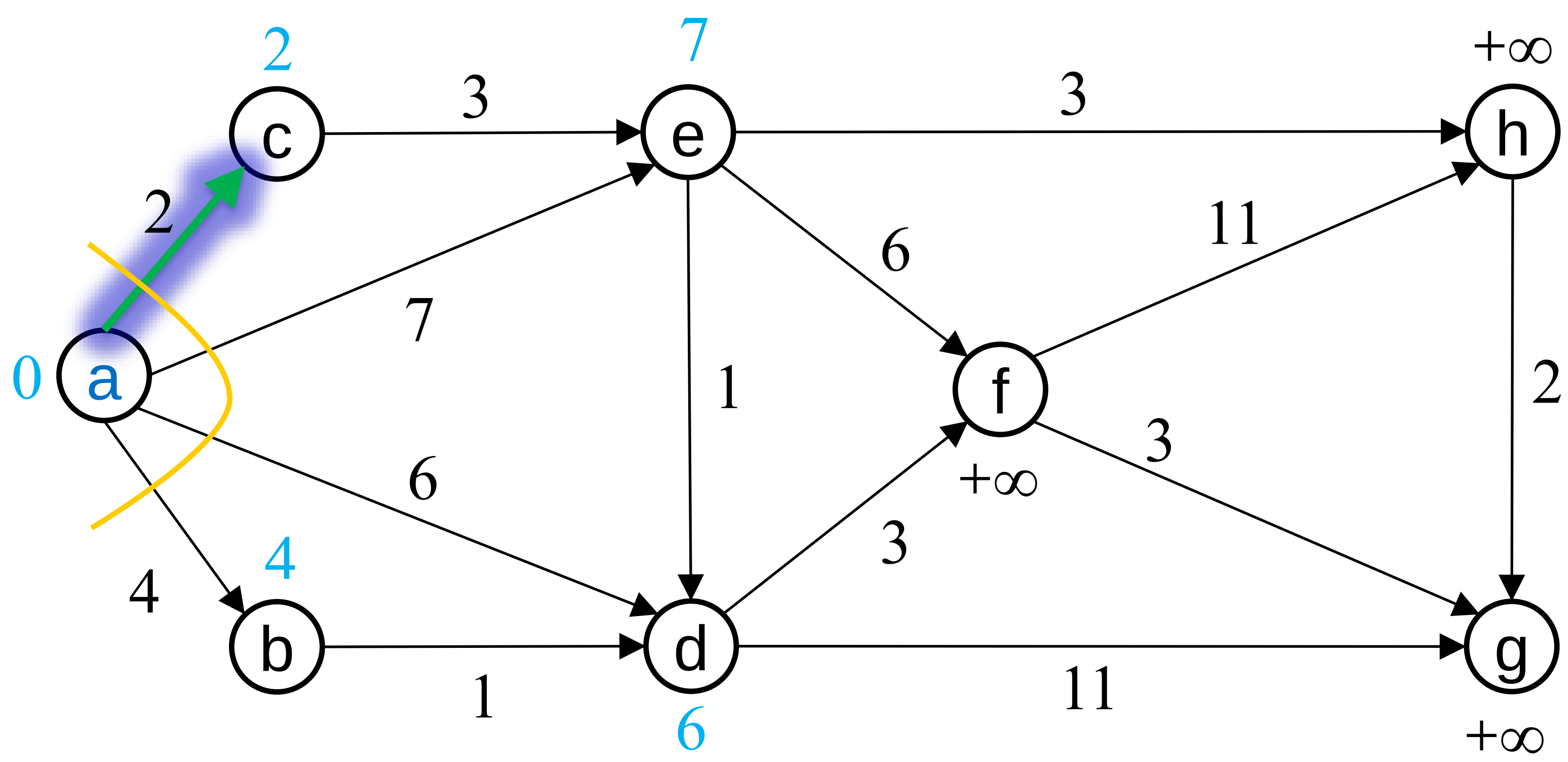
Esercizio: l'algoritmo di Dijkstra

Inizialmente l'albero **T** conterrà solo il nodo sorgente **a**

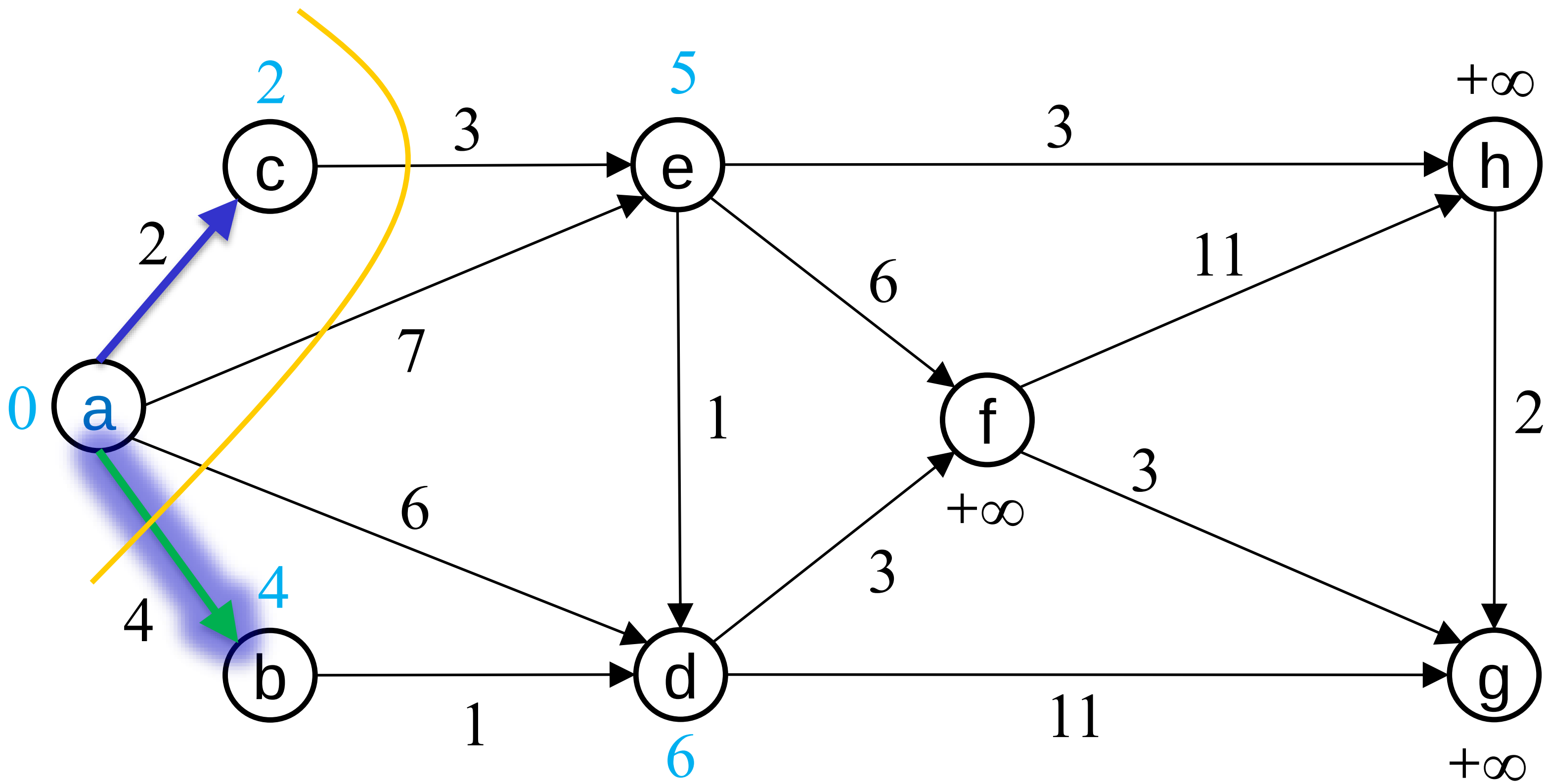
L'algoritmo procederà rilassando di volta in volta l'arco **(u,v)** con $\mathbf{D_{su}}$ **+ w(u,v)** minimo.



Esercizio: l'algoritmo di Dijkstra

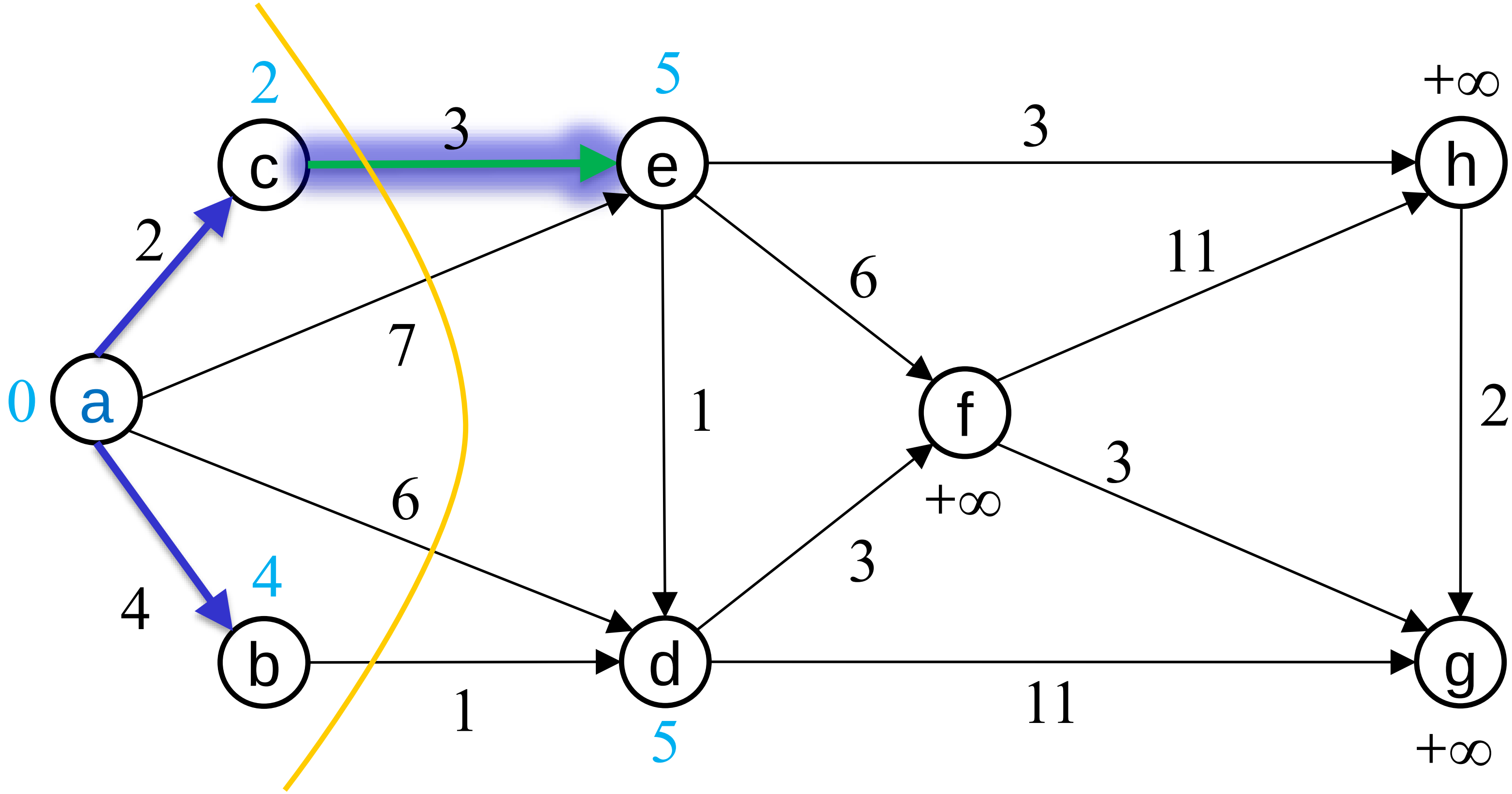


Esercizio: l'algoritmo di Dijkstra

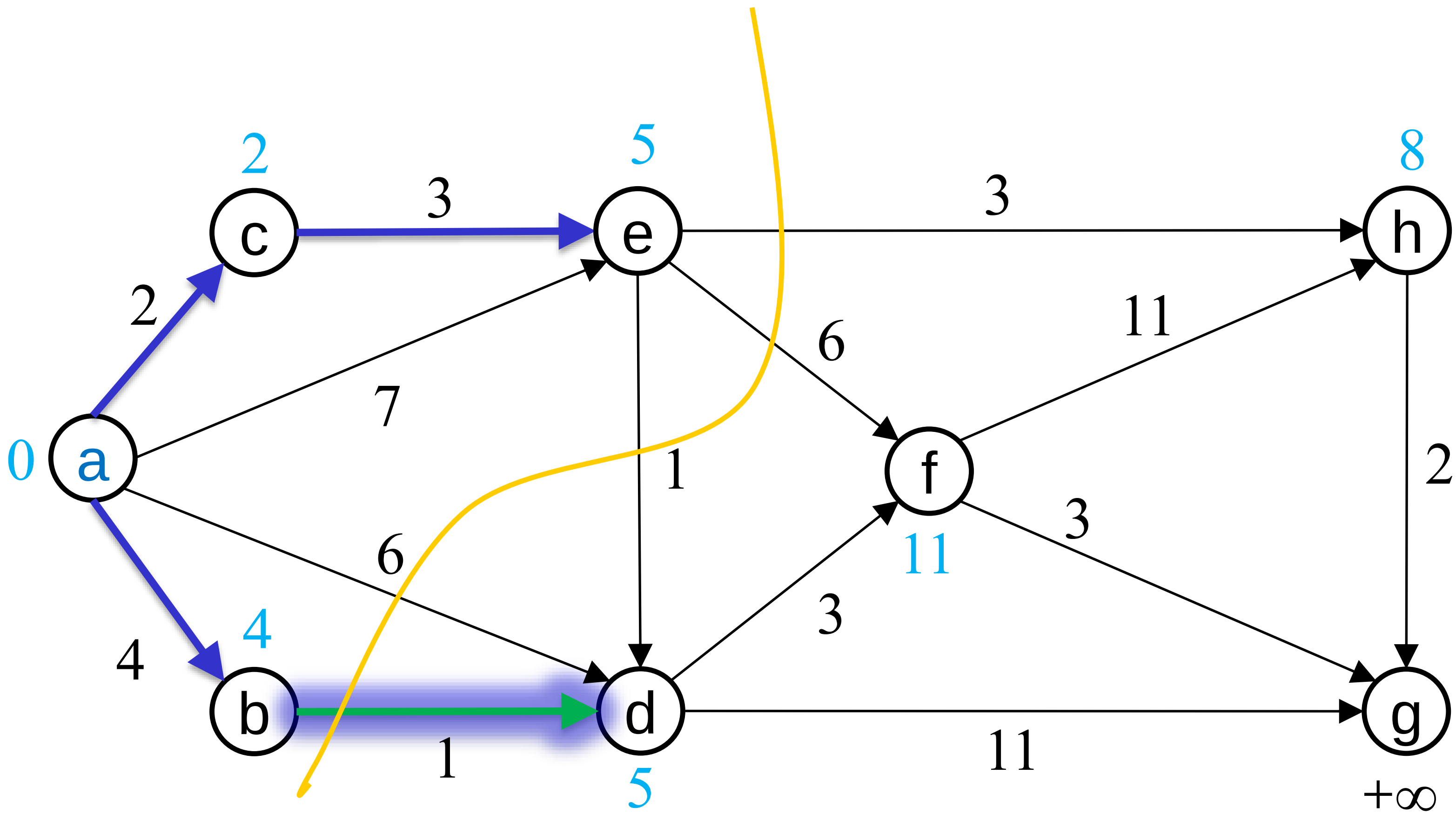


Esercizio: l'algoritmo di Dijkstra

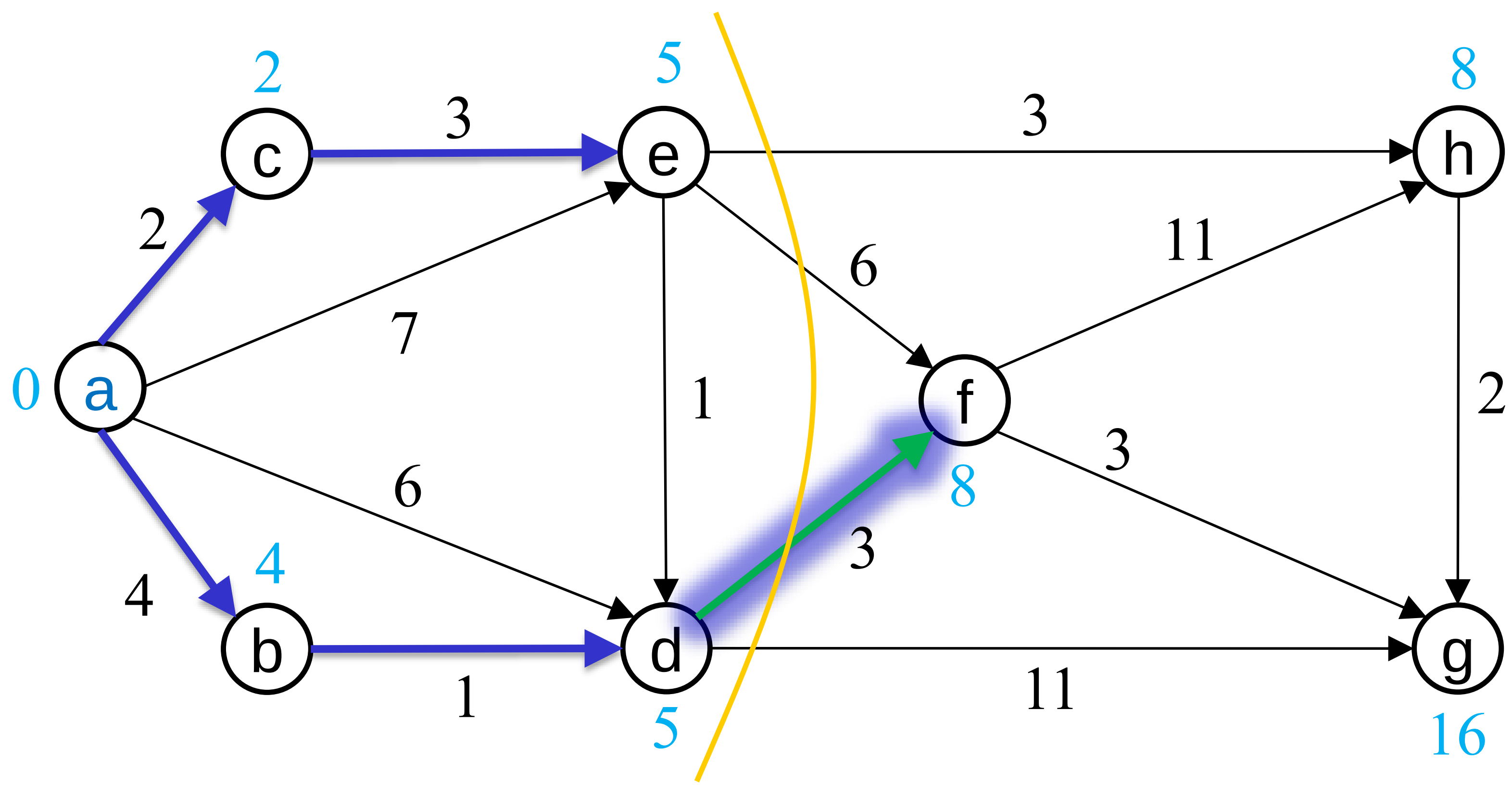
NOTA: posso scegliere l'arco (c,e) oppure l'arco (b,d).
Scelgo arbitrariamente l'arco (c,e),



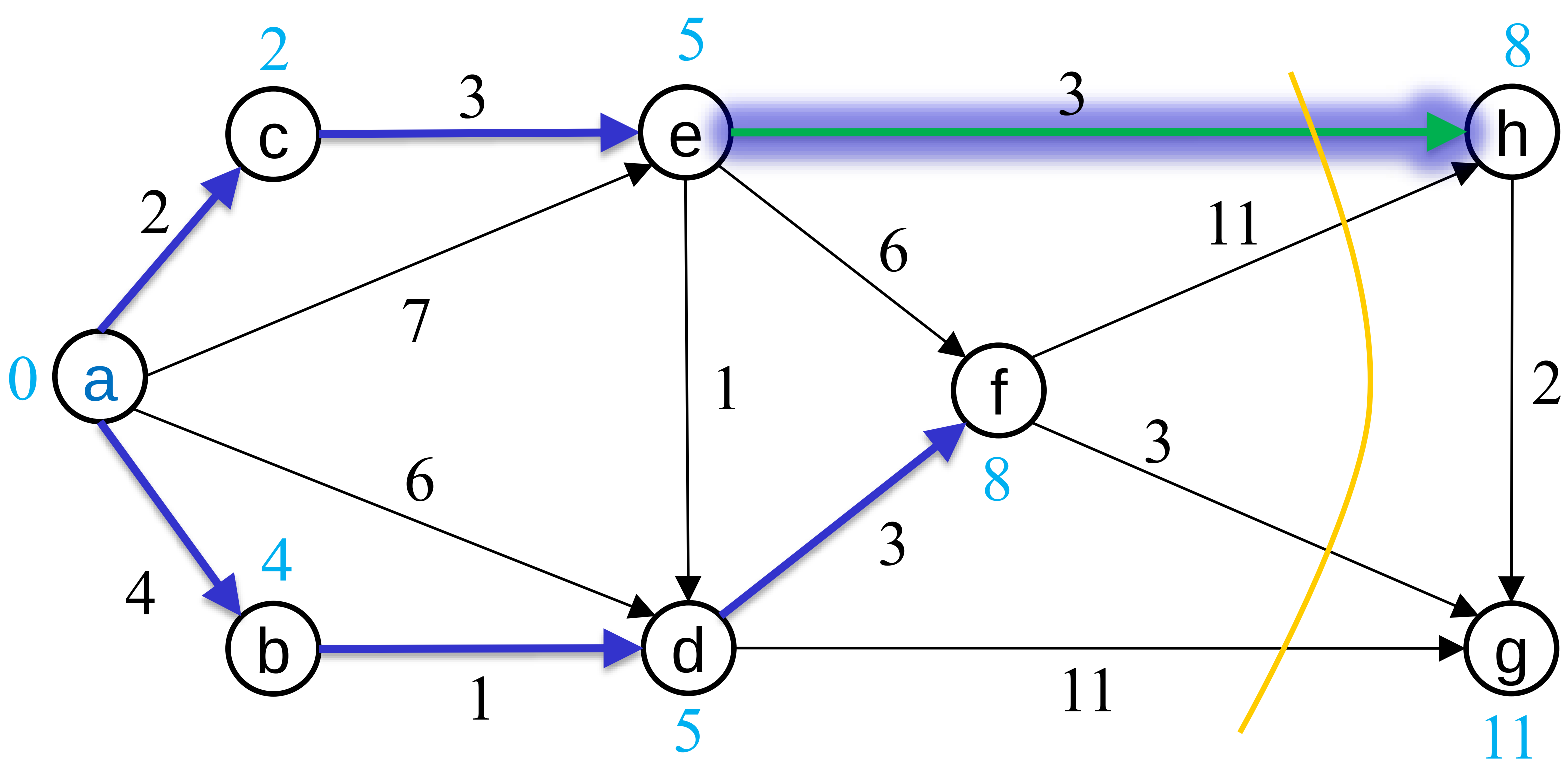
Esercizio: l'algoritmo di Dijkstra



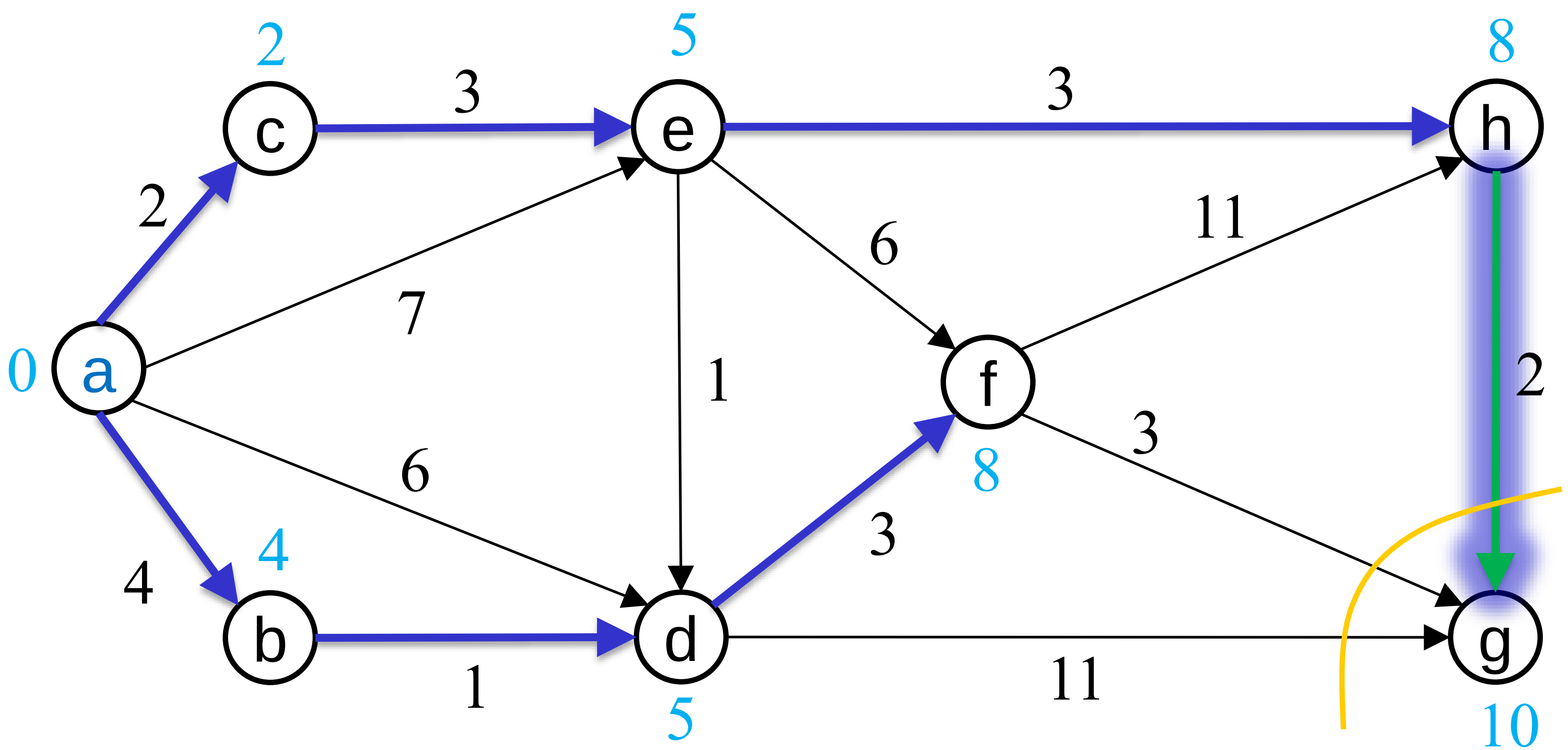
Esercizio: l'algoritmo di Dijkstra



Esercizio: l'algoritmo di Dijkstra

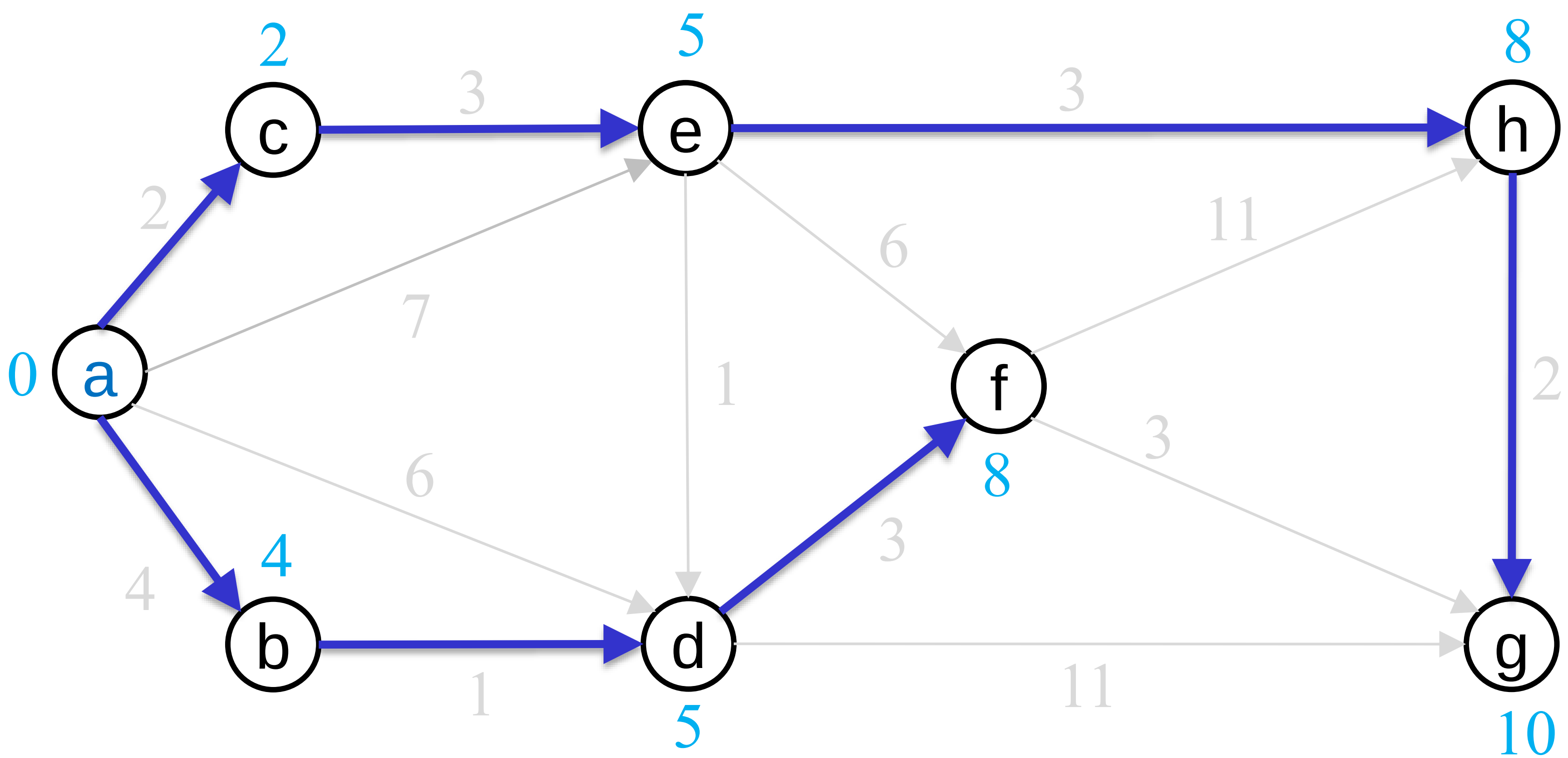


Esercizio: l'algoritmo di Dijkstra



Esercizio: l'algoritmo di Dijkstra

L'algoritmo termina restituendo il seguente albero dei cammini minimi



Esercizio: il castello dei mostri

Un castello è rappresentato come un grafo non orientato **G=(V, E):** ogni vertice **v** rappresenta una stanza, ed ogni arco **(u,v)** indica un corridoio che collega la stanza **u** alla stanza **v**.
Il castello ha **n** stanze, numerate con gli interi da **1** a **n**. Un array di interi **M[n]** indica la presenza o meno di un **mostro** in ciascuna stanza: se **M[s] = 0** la stanza **s** è libera, se invece **M[s] = 1**, nella stanza c'è un mostro.
Scrivere un algoritmo che, preso in input il grafo **G**, l'array **M[n]** e i nodi **s**, **d**, restituisca il minimo numero di corridoi che è necessario attraversare per andare da **s** a **d**, **senza incontrare mostri**, se questo è possibile. Nel caso in cui ciò non è possibile, l'algoritmo deve restituire $+\infty$. (Nota: se **s=d**, l'algoritmo restituisce 0. Se **s** e **d** sono adiacenti, l'algoritmo restituisce **1**).

Esercizio: il castello dei mostri

Soluzione:

L'esercizio può essere risolto utilizzando l'algoritmo per la visita in ampiezza opportunamente modificata per evitare le stanze occupate da mostri e per tenere traccia del numero di corridoi attraversati.

Un possibile algoritmo può quindi essere il seguente.

Esercizio: il castello dei mostri

Soluzione:

```
algoritmo castello(grafo G, array M, nodo s, nodo d) → intero oppure +∞
    associa ad ogni nodo di G un valore dist inizialmente pari a +∞;
    coda C;
    s.dist ← 0;
    C.enqueue(p);
    while ( not C.isEmpty() ) do
        u ← C.dequeue();
        if(u = s) then return u.dist;
        for each ( arco (u, v) in G) do
            if(v.dist = +∞ && M[v] = 0) then
                v.dist ← u.dist +1;
                C.enqueue(v);
    return +∞;
```