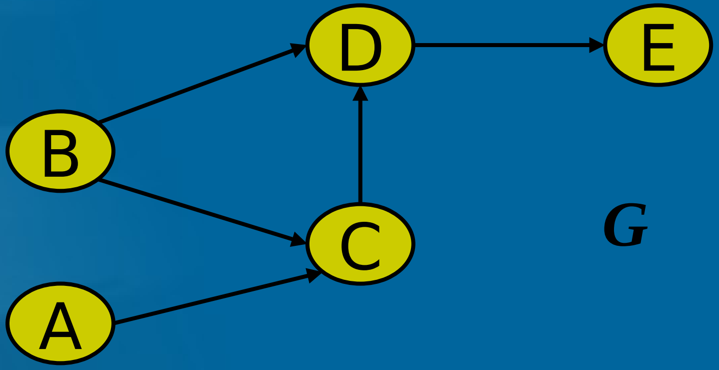
Medicina e Tecnologie TD
Lezione 11

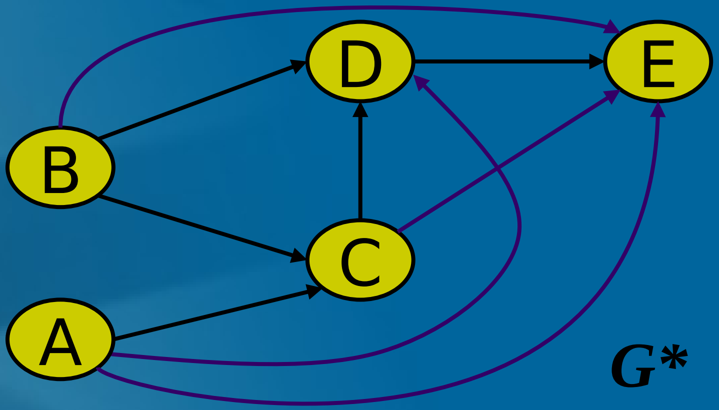
Pierangelo Veltri
pierangelo.veltri@unical.it

Chiusura Transitiva

- Dato un grafo orientato G, la chiusura transitiva calcola un grafo ***G* tale che***
  - ***G**** ha gli stessi nodi di ***G***
  - se ***G*** ha un cammino da ***u*** a ***v*** ($u \neq v$), ***G**** ha arco da ***u*** a ***v***



G



G*

Metodi

- Dare un peso unitari a tutti gi archi e calcolare il cammino minimo per tutti i nodi (Dijkstra)
- Applicare l'algoritmo calcola tutti i cammini (algo floyd-warshall) dando peso unitario a tutti gli archi:
  - Alla fine un cammino i,j esiste se nella matrice dei cammini di,j <=n-1. altrimenti se = infinito non sono connessi

Algoritmi Greedy

- Un algoritmo greedy è un paradigma algoritmico, dove l'algoritmo cerca una soluzione ammissibile da un punto di vista globale attraverso la scelta della soluzione più appetibile (definita in precedenza dal programmatore) per quel determinato programma a ogni passo locale.

- Quando applicabili, questi algoritmi consentono di trovare soluzioni ottimali per determinati problemi in un tempo polinomiale, mentre negli altri non è garantita la convergenza all'ottimo globale.

- In particolare questi algoritmi cercano di mantenere una proprietà di sottostruttura ottima, quindi cercano di risolvere i sottoproblemi in maniera "avida" (da cui la traduzione letterale algoritmi avidi in italiano) considerando una parte definita migliore nell'input per risolvere tutti i problemi.

Algoritmo di Dijkstra

- L'algoritmo di Dijkstra è un algoritmo utilizzato per cercare i cammini minimi in un grafo con o senza ordinamento, **ciclico** e con **pesi non negativi** sugli archi.

- Fu inventato nel 1956 dall'informatico olandese Edsger Dijkstra che lo pubblicò successivamente nel 1959.

- Tale algoritmo trova applicazione in molteplici contesti quale l'ottimizzazione nella realizzazione di reti (idriche, telecomunicazioni, stradali, circuitali, ecc.) o l'organizzazione e la valutazione di percorsi runtime nel campo della robotica.

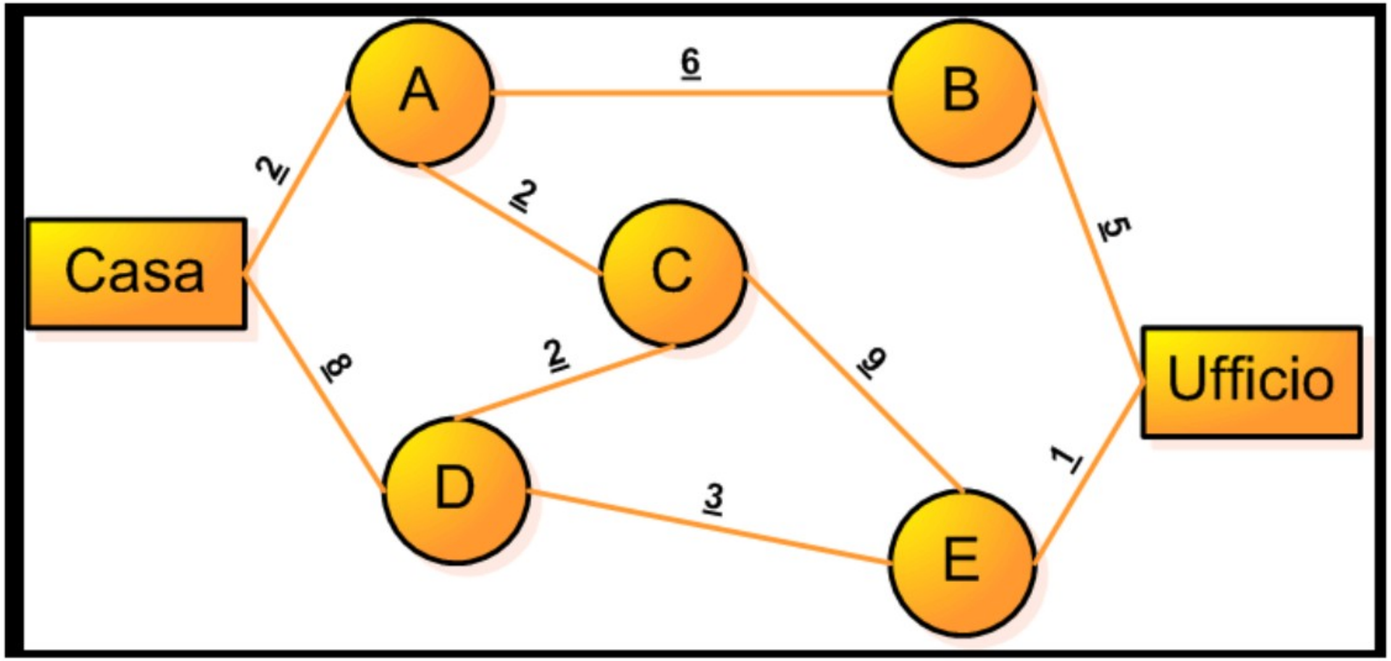
Algoritmo di Dijkstra

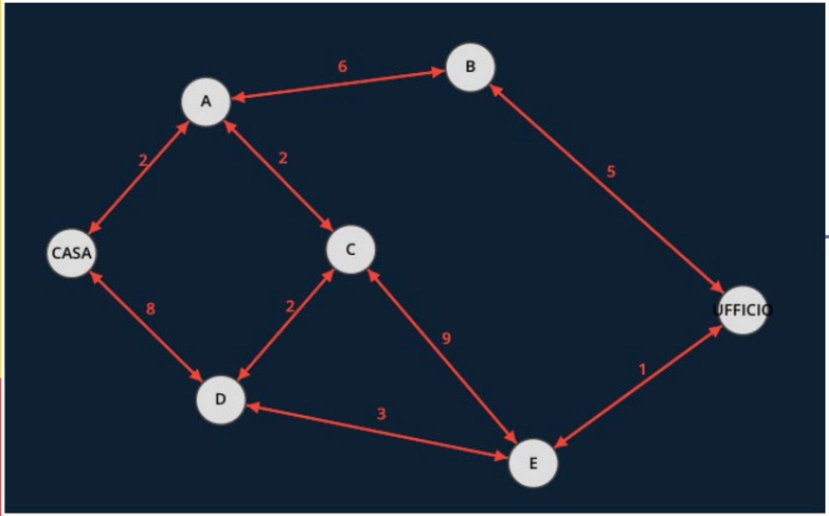
- Alla base di questi problemi c'è lo scopo di trovare il percorso minimo (più corto, più veloce, più economico...) tra due punti, uno di partenza e uno di arrivo

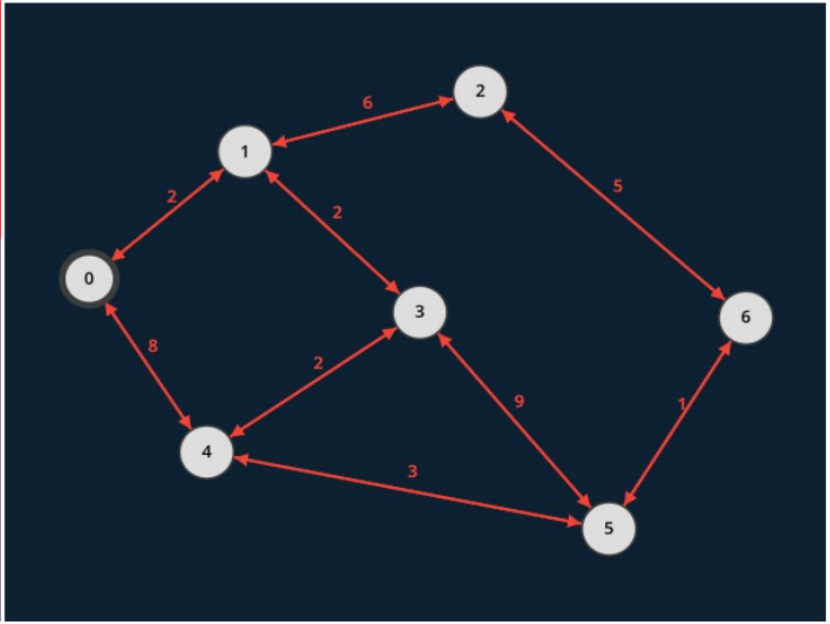
- è possibile ottenere non solo il percorso minimo tra un punto di partenza e uno di arrivo ma l'albero dei cammini minimi, *cioè tutti i percorsi minimi tra un punto di partenza e tutti gli altri punti della rete*.

Esempi

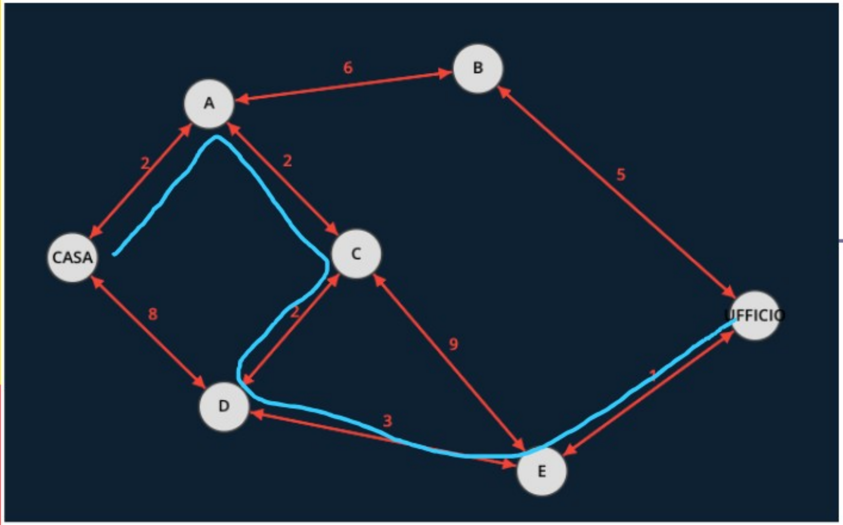
- Un esempio di cammino da casa ad ufficio = (CASA,A)(A,C)(C,E)(E,UFFICIO)
- Non è minimo!
- (CASA,A)(A,C)(C,D)(D,E)(E,UFFICIO) è il cammino minimo (distanza minima 10)



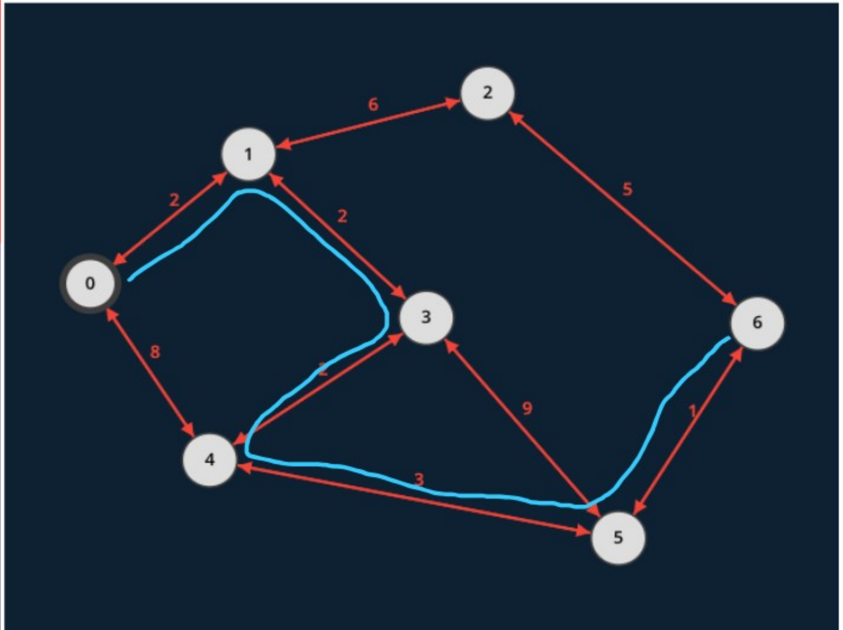




Rappresentano lo stesso grafo!



•(CASA,A)(A,C)(C,D)(D,E)(E,UFFICIO) è il cammino minimo
(distanza minima 10)

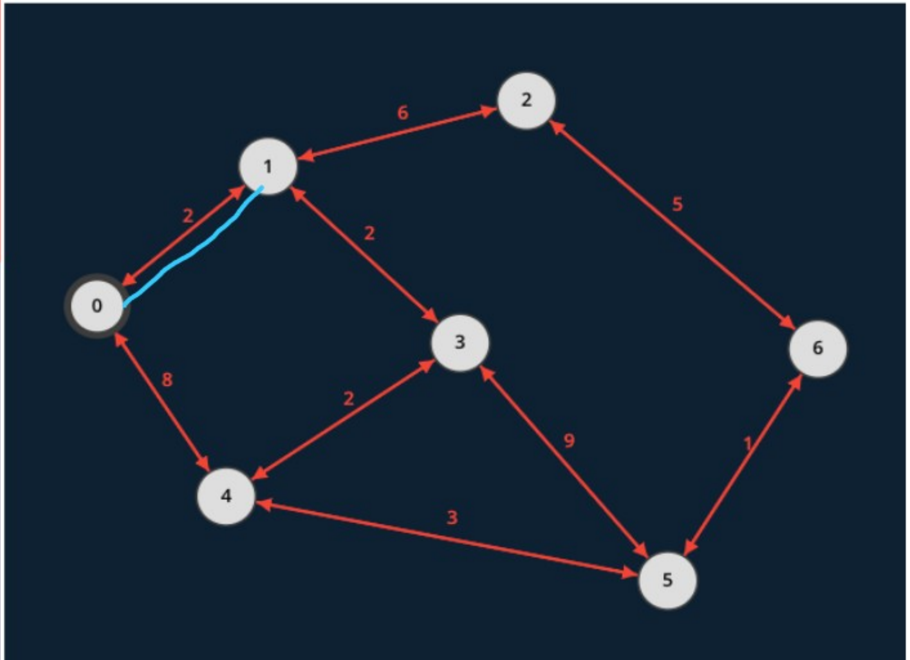


•(0,1)(1,3)(3,4)(4,5)(5,6) è il cammino minimo
(distanza minima 10)

Cos’è pred in dijkstra(G,0)?

pred = [0, 0, 1, 1, 3, 4, 5]

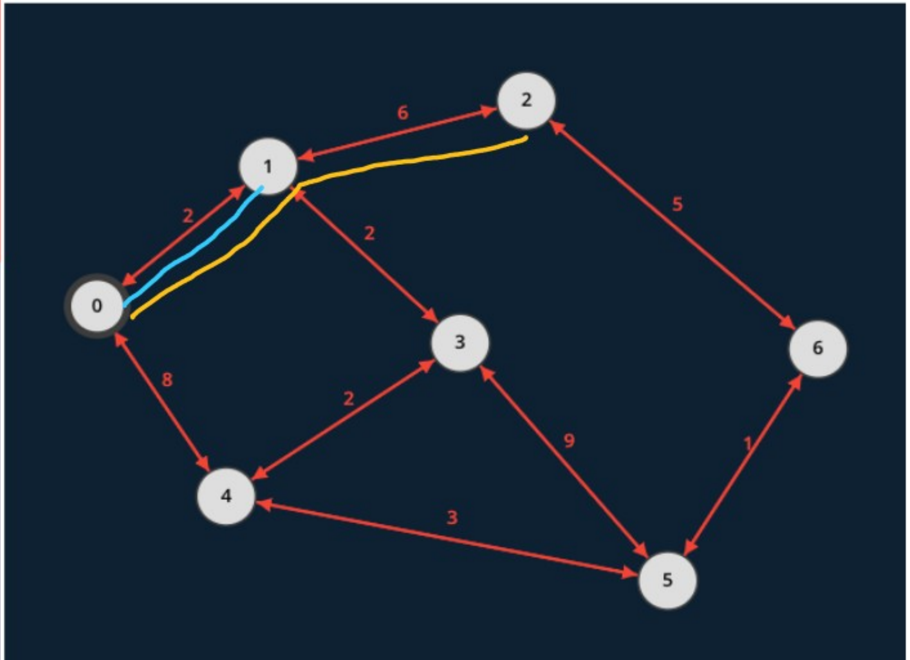
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 1 è (0,1)



Cos'è pred in dijkstra(G,0)?

pred = [0, 0, 1, 1, 3, 4, 5]

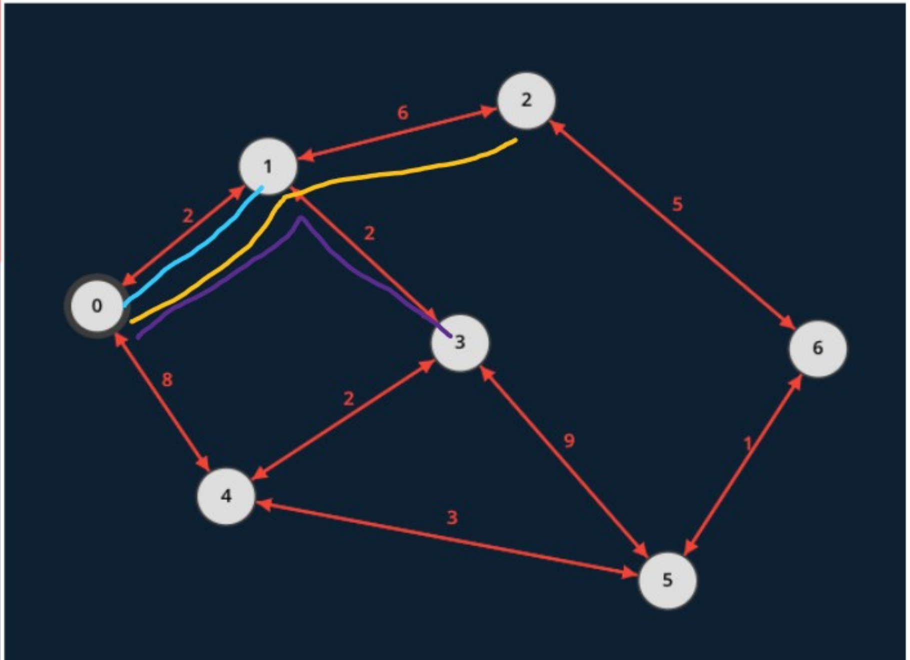
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 1 è (0,1)
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 2 è (0,1)(1,2)



Cos'è pred in dijkstra(G,0)?

pred = [0, 0, 1, 1, 3, 4, 5]

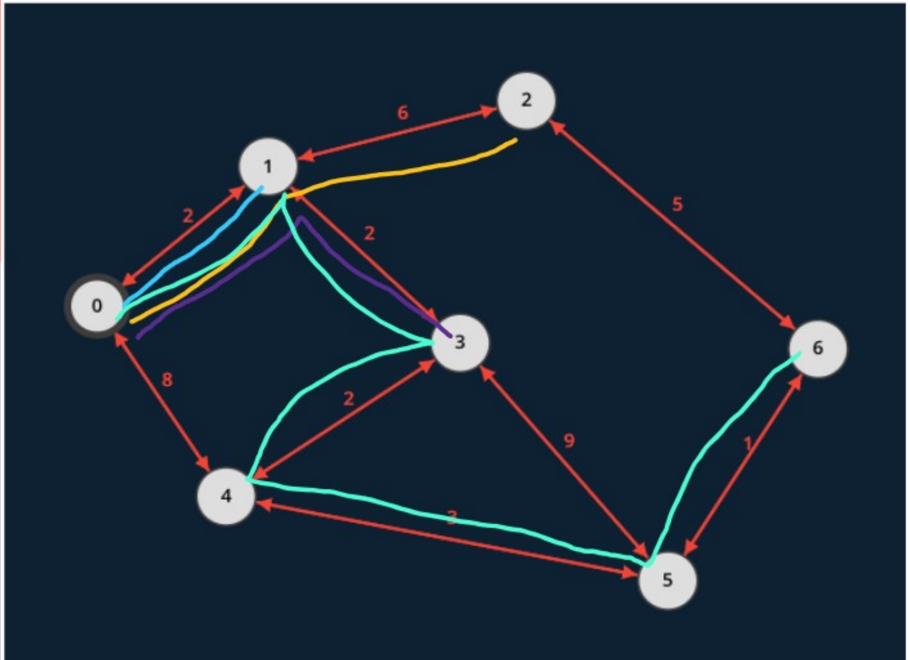
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 1 è (0,1)
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 2 è (0,1)(1,2)
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 3 è (0,1)(1,3)



Cos'è pred in dijkstra(G,0)?

pred = `[0, 0, 1, 1, 3, 4, 5]`

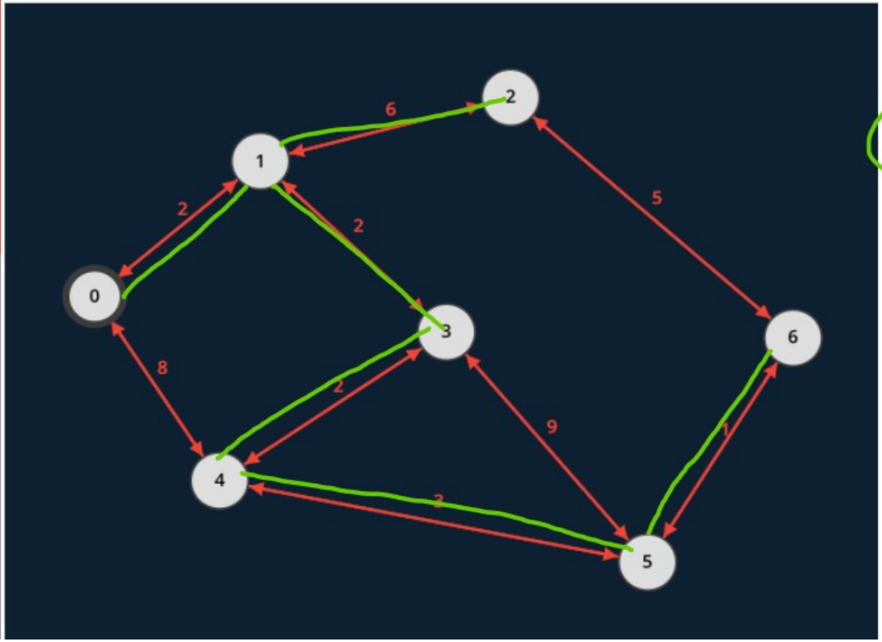
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 1 è (0,1)
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 2 è (0,1)(1,2)
- Il cammimo minimo per andare dal nodo 0 al nodo al nodo 3 è (0,1)(1,3)
- …. dal nodo 0 al nodo al nodo 6 è (0,1)(1,3)(3,4)(4,5)(5,6)

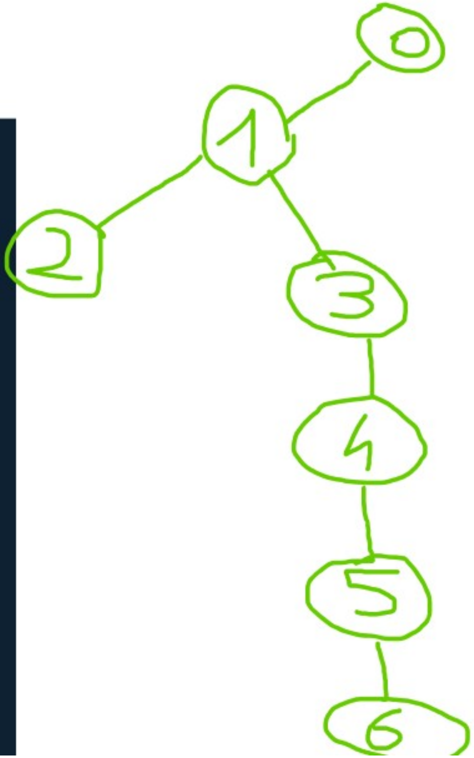


Cos’è pred in dijkstra(G,0)?

pred = `[0, 0, 1, 1, 3, 4, 5]`

- Albero dei cammini minimi dal nodo 0 (cioè CASA)



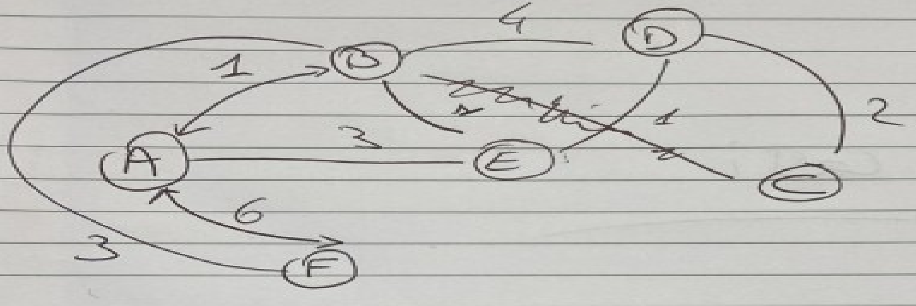


Minimo albero ricoprente

- Un albero ricoprente (anche detto di copertura, di connessione o di supporto) di un grafo, connesso e con archi non orientati, è un albero che contiene tutti i nodi del grafo e contiene soltanto un sottoinsieme degli archi, cioè solo quelli necessari per connettere tra loro tutti i nodi con uno e un solo cammino.
- Sia dato un grafo G non orientato pesato e supponiamo che esso abbia almeno un albero ricoprente.
- Un minimo albero ricoprente di G è l'albero ricoprente tale che la somma dei pesi dei suoi archi sia minima.

Ese

$G = (\{A, B, C, D, E, F\}, \{(A,B)=1, (A,F)$
$A \; ... \;)$



	LISTA	B	C	D	E	F
0	A	1, A	∞	∞	3, A	6, A
1	B			5, B	2, B	1+3=4, B
	ABE			3, E		
3	ABED		5, D			

Aggiungo e vedo ADIACENTI

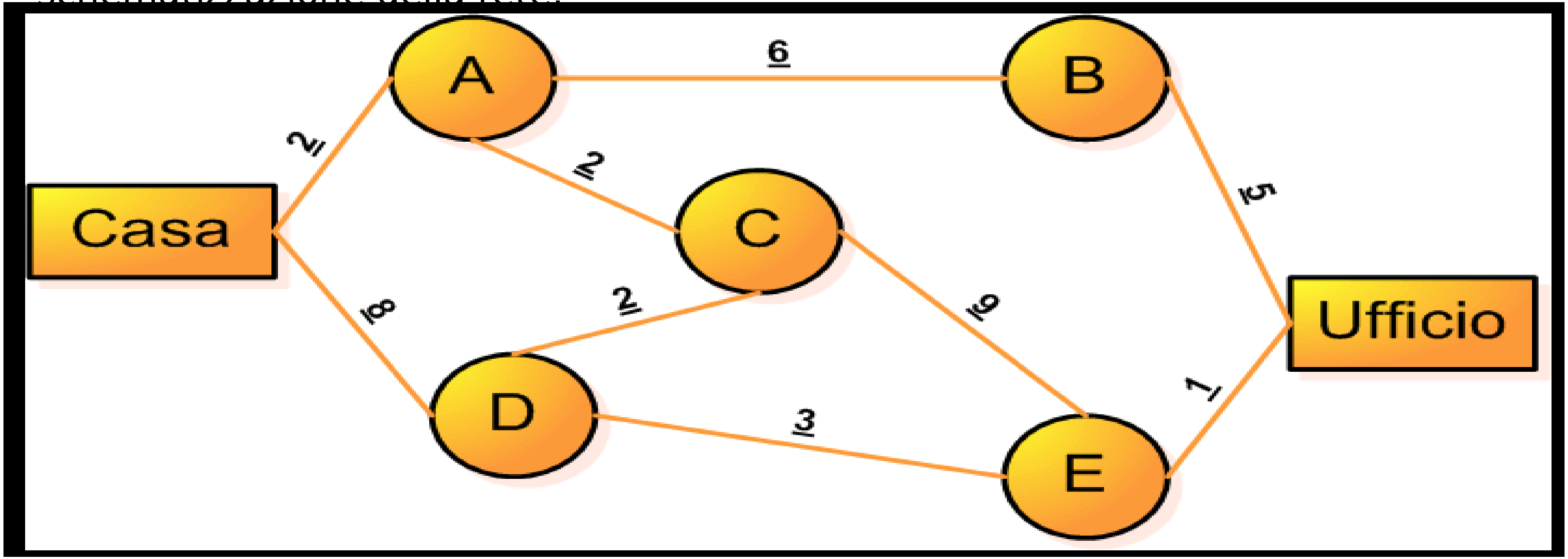
4	ABEDF
5	ABEDFC

Animazione per Algo Dijkstra

- https://www.youtube.com/watch?v=wtdtkJgcYUM
-

Algoritmo di Dijkstra

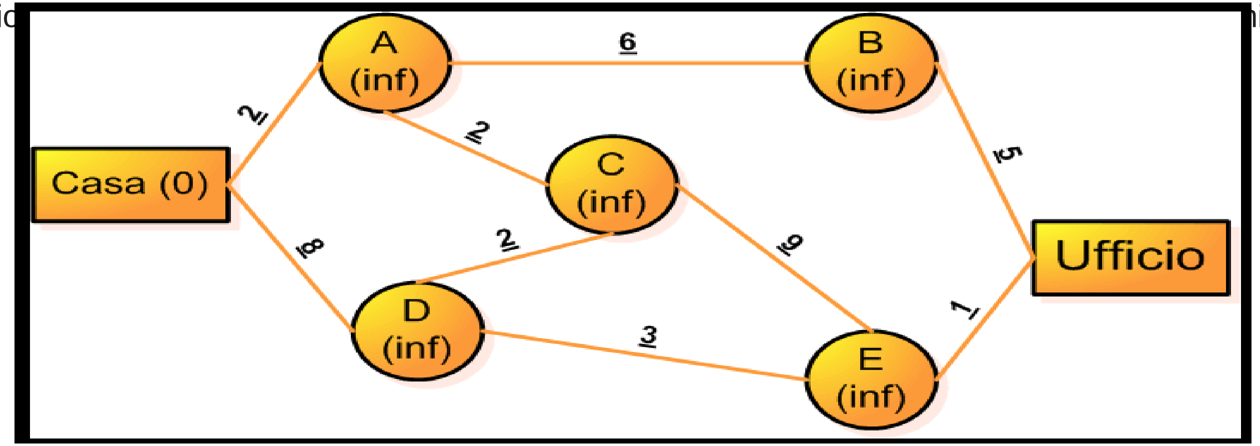
- Si consideri un problema in cui si vuole calcolare il percorso minimo tra casa e il posto di lavoro. Si schematizzino tutti i possibili percorsi e il relativo tempo di percorrenza (supponendo di voler calcolare il percorso più breve in fatto di tempo di percorrenza). I nodi A, B, C, D, E indicano le cittadine per cui è possibile passare. Ecco una schematizzazione della rete:



Algoritmo di Dijkstra

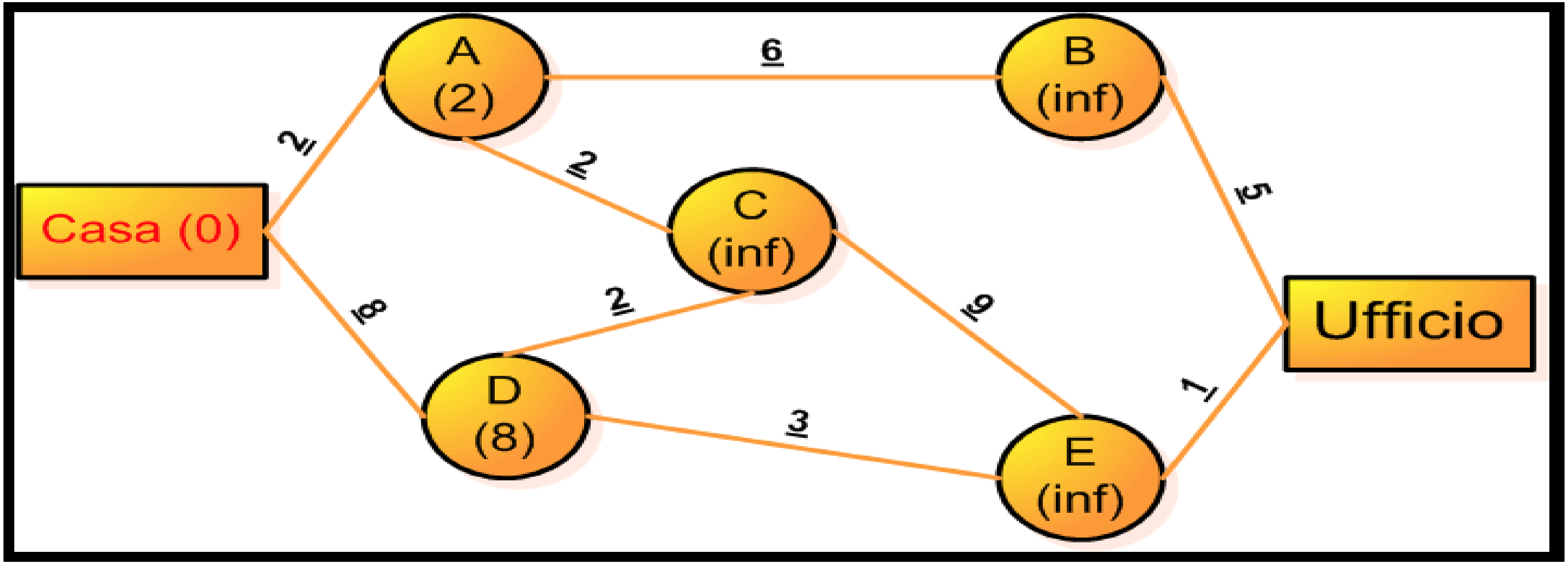
Bisogna ora assegnare a ogni nodo un valore, chiamato “potenziale”, seguendo alcune regole:

- Ogni nodo ha potenziale inizialmente pari ad infinito
- Il nodo di partenza (in questo caso “casa”) ha potenziale 0 (ovvero dista zero da sé stesso);
- Ogni volta si sceglie il nodo con potenziale minore e lo si rende definitivo (colorando il potenziale di rosso) e si aggiornano i nodi adiacenti;
- Il potenziale di un nodo è dato dalla somma del potenziale del nodo precedente + il costo del collegamento;
- Non si aggiornano i potenziali dei nodi resi definitivi;
- I potenziali definitivi indicano la distanza di quel nodo da quello di partenza;
- Quando si aggio ... inimo).



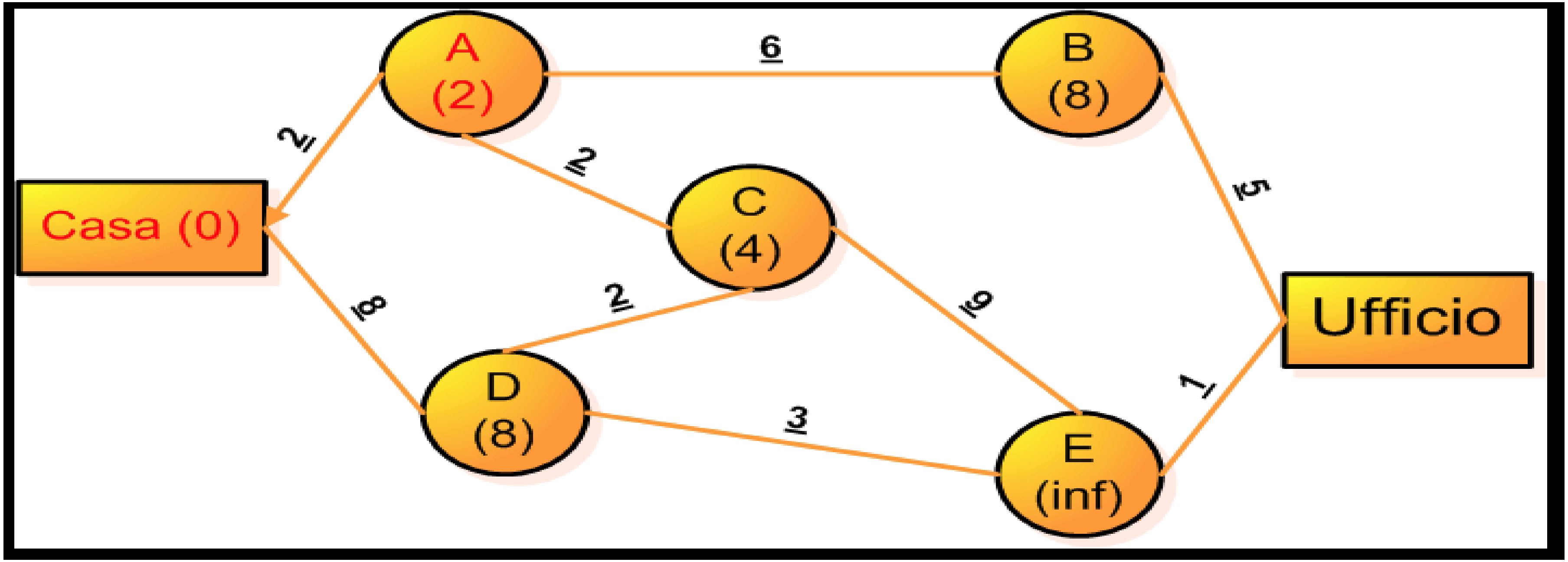
Algoritmo di Dijkstra

Seguendo le regole appena fissate si consideri il nodo con potenziale minore (“casa”) e lo si renda definitivo (colorandolo di rosso) e si aggiornino tutti i nodi adiacenti sommando l'attuale valore del potenziale (ovvero zero) al costo del percorso. Si aggiornino i potenziali perché si aveva, nel caso di A, potenziale infinito mentre ora il potenziale è 2. Ricordando che il potenziale minore è sempre preferibile. Ecco come si è aggiornata la rete:



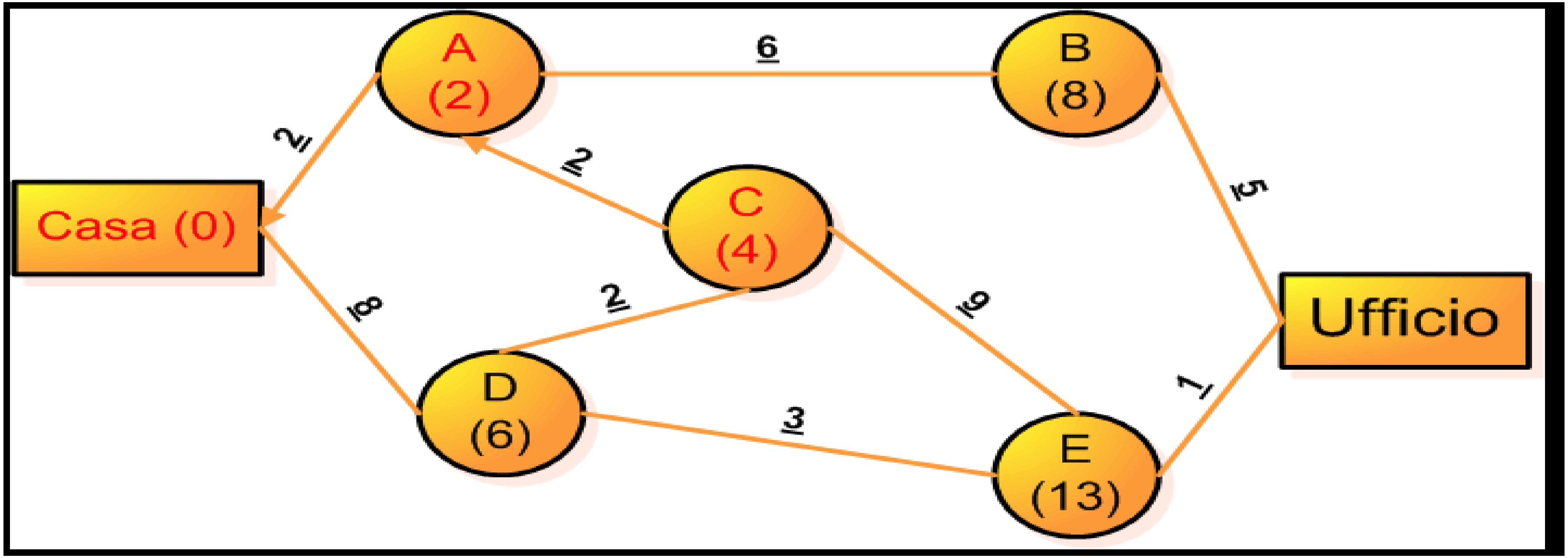
Algoritmo di Dijkstra

Bisogna ora considerare il nodo non definitivo (ovvero quelli scritti in nero) con potenziale minore (il nodo A). Lo si rende definitivo e si aggiornano i potenziali dei nodi adiacenti B e C. Si indichi con una freccia da dove proviene il potenziale dei nodi resi definitivi.



Algoritmo di Dijkstra

Il nodo con potenziale minore ora è C. lo si rende definitivo e si aggiornano quelli adiacenti.

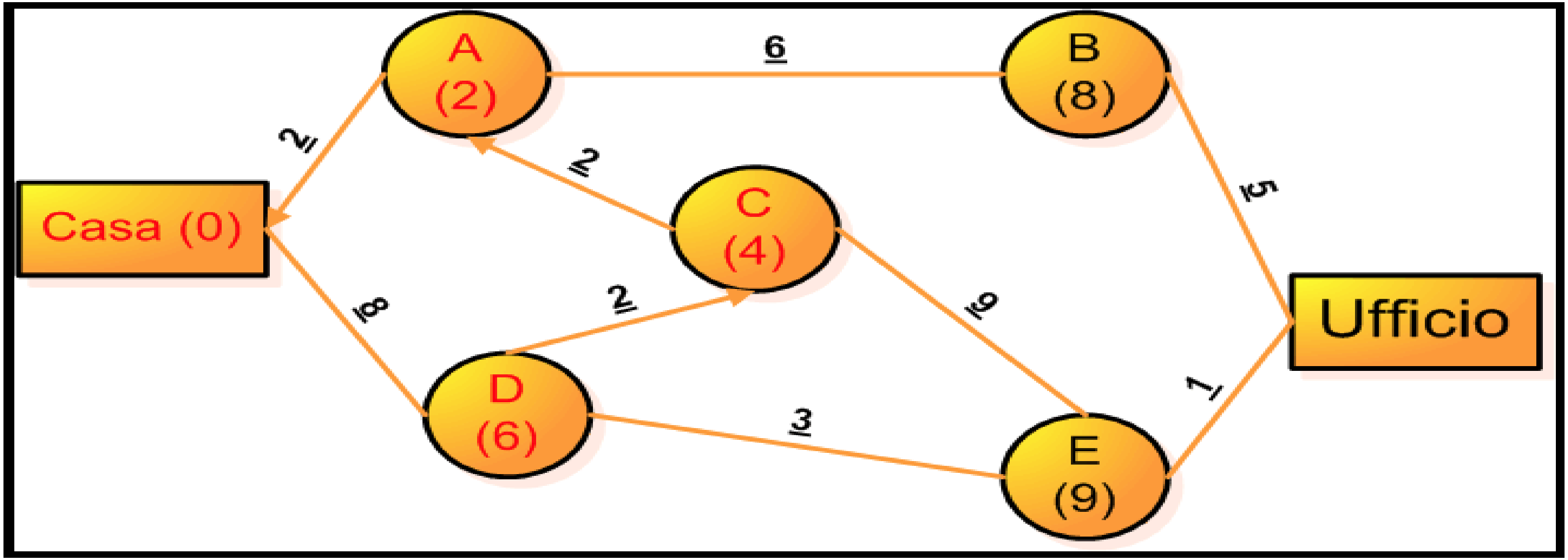


Algoritmo di Dijkstra

Va notato come il nodo D abbia ora potenziale 6 in quanto 6 è minore di 8 e quindi lo si aggiorna.

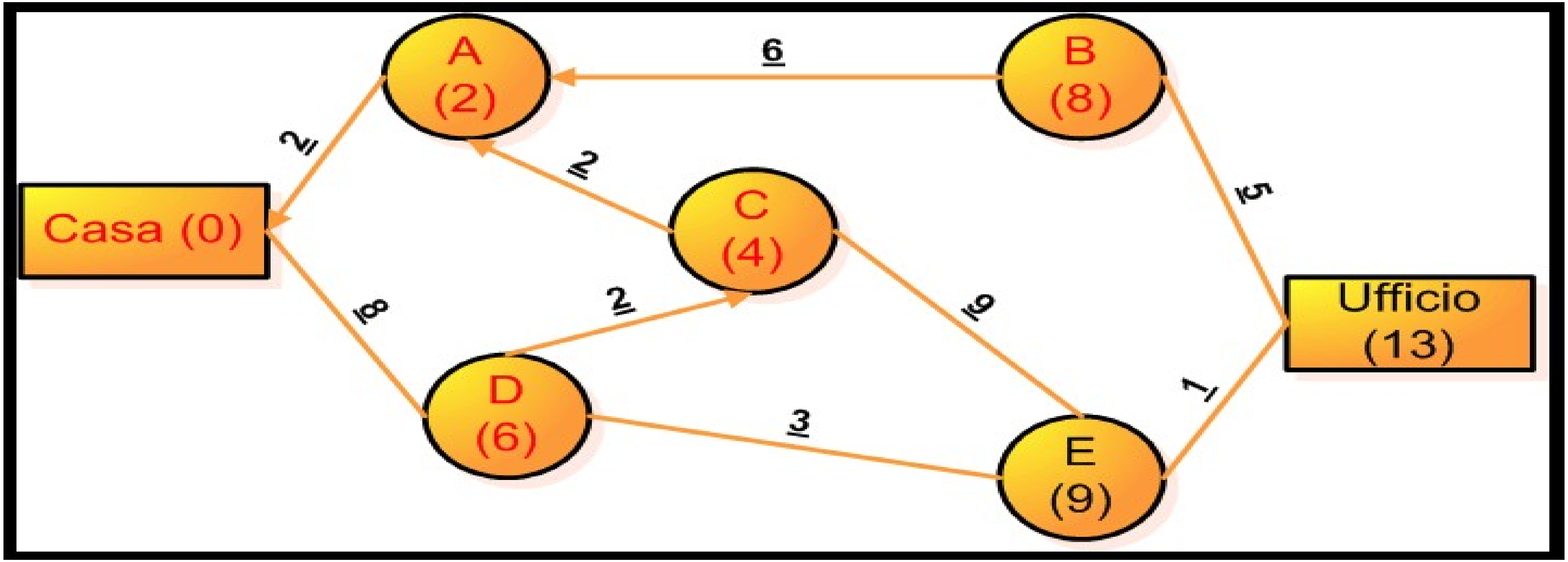
Se si fosse ottenuto un valore maggiore di quello che già c'era si sarebbe dovuto lasciare invariato.

Si renda definitivo il nodo D e si aggiorni il grafico:



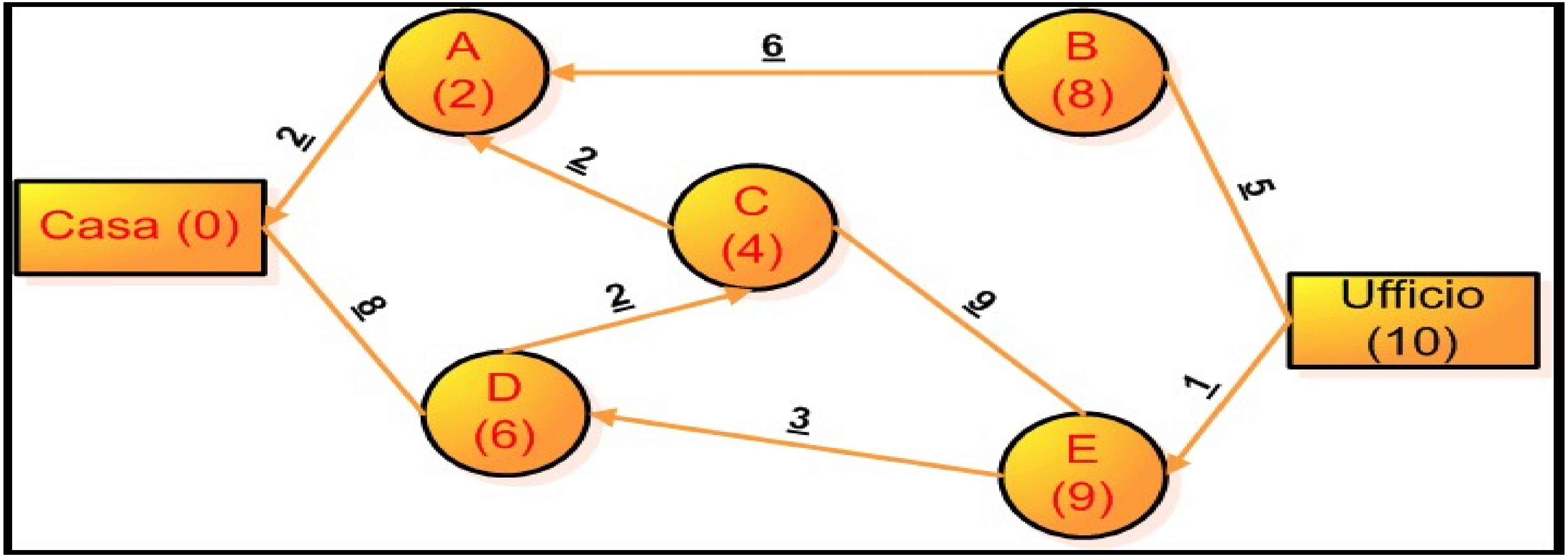
Algoritmo di Dijkstra

Il nodo con potenziale minore restante è B e lo si rende definitivo aggiornando di conseguenza il grafico:



Algoritmo di Dijkstra

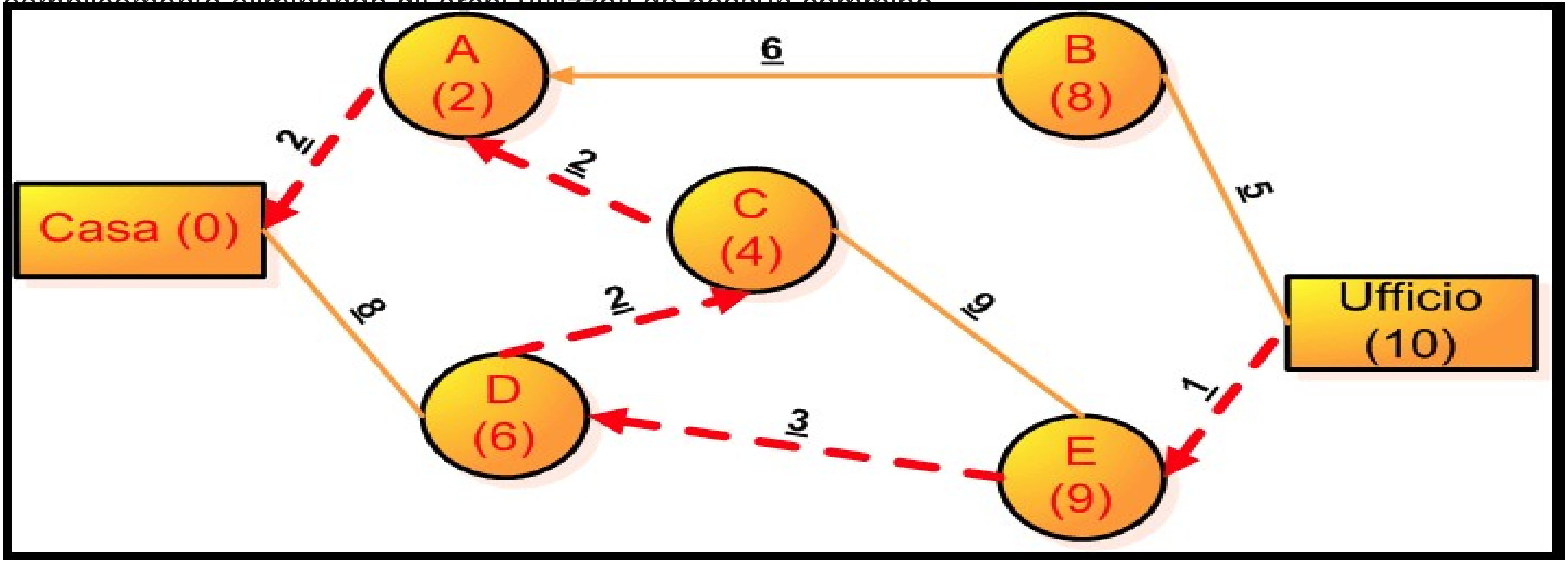
Restano da considerare il nodo E e da aggiornare “ufficio”.



Algoritmo di Dijkstra

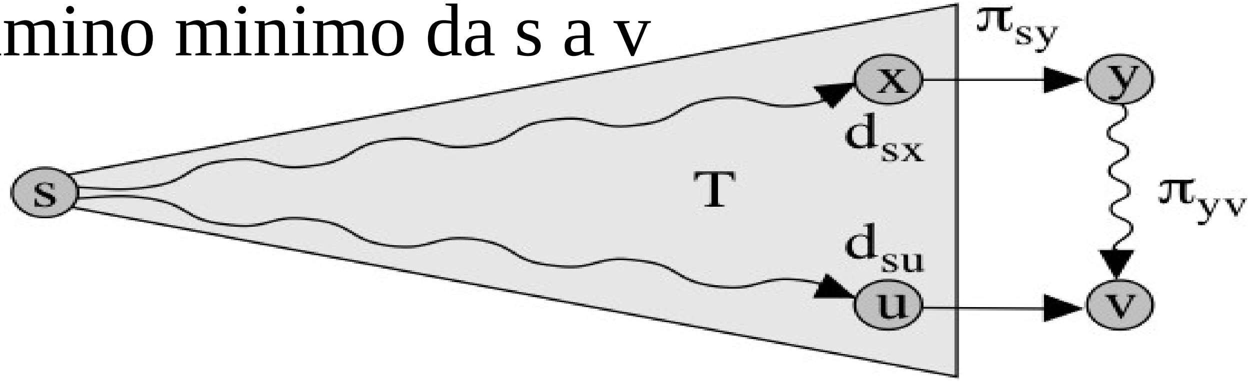
Seguendo all'indietro le frecce si ottiene il percorso minimo da casa a ufficio che misura (come indicato dal potenziale) “10”.

Bisogna notare come questo algoritmo ci dia non solo la distanza minima tra il punto di partenza e quello di arrivo ma la distanza minima di tutti i nodi da quello di partenza, da cui si può realizzare l'albero dei cammini minimi semplicemente eliminando gli archi utilizzati da nessun cammino.



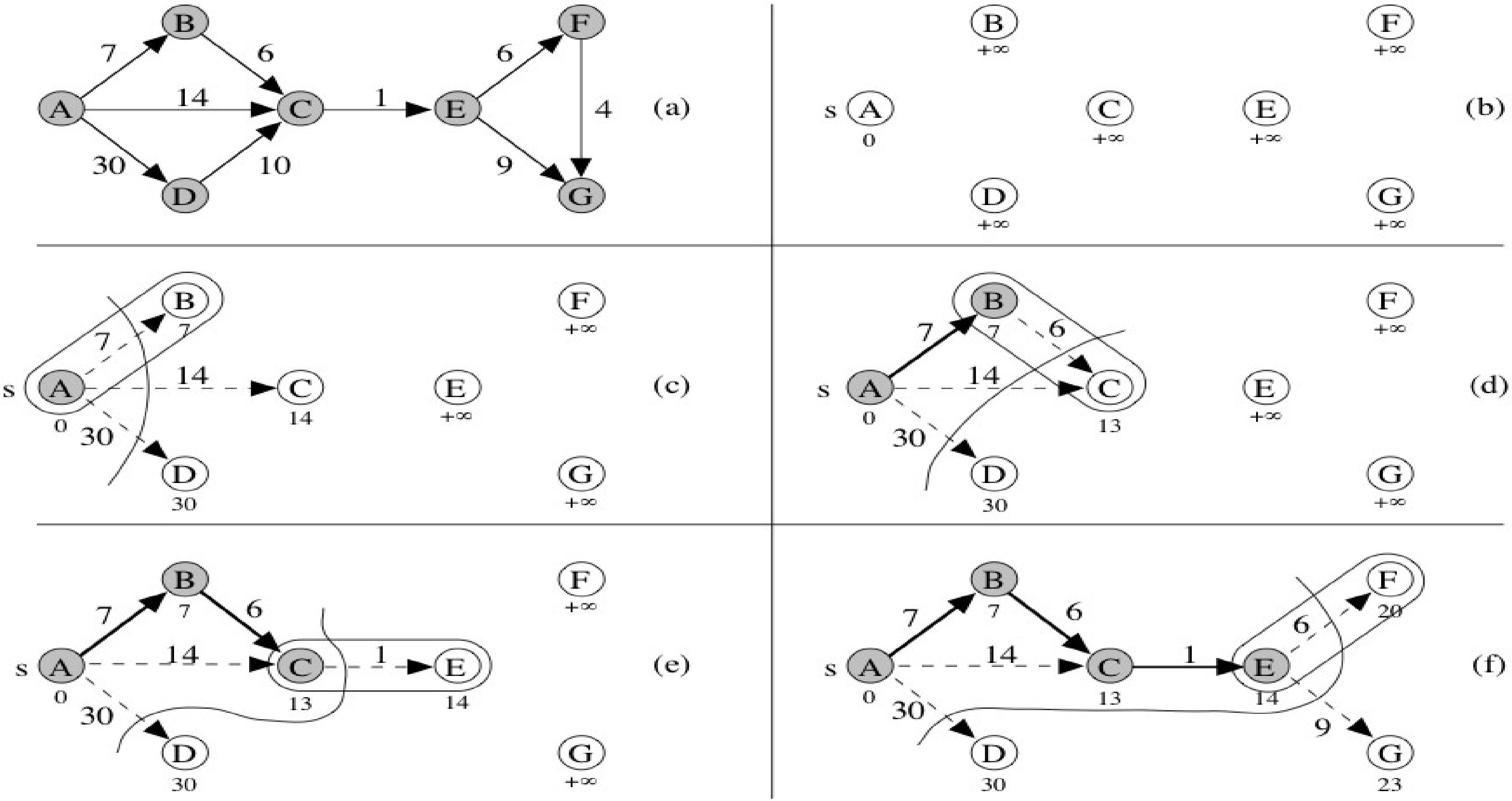
Algoritmo di Dijkstra (ulteriori intuizioni)

Se T è un albero dei cammini minimi radicato in s che non include tutti i vertici raggiungibili da s, l'arco (u,v) tale che u T e v T che minimizza la quantità d_{su}+w(u,v) appartiene a un cammino minimo da s a v

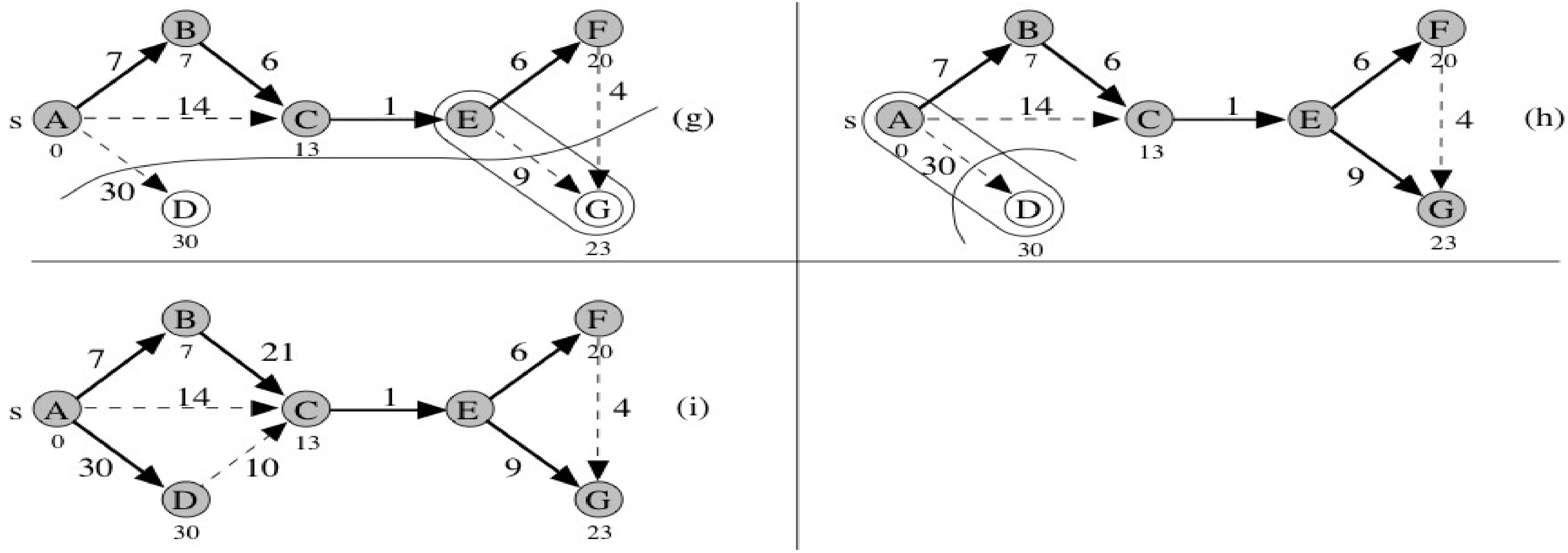


Scegli un arco (u,v) con u T e v T che minimizza la quantità d_{su}+w(u,v), assegna d_{sv} d_{su}+w(u,v), ed aggiungilo a T

Algoritmo di Dijkstra (ulteriori intuizioni)



Algoritmo di Dijkstra (ulteriori intuizioni)



Algoritmo di Dijkstra (ulteriori intuizioni)

Implementazione

https://www.udacity.com/blog/2021/10/implementing-dijkstras-algorithm-in-python.html

```python
def dijkstra_algorithm(graph, start_node):
    unvisited_nodes = list(graph.get_nodes())

    # We'll use this dict to save the cost of visiting each node and update it
    shortest_path = {}

    # We'll use this dict to save the shortest known path to a node found so fa
    previous_nodes = {}

    # We'll use max_value to initialize the "infinity" value of the unvisited n
    max_value = sys.maxsize
    for node in unvisited_nodes:
        shortest_path[node] = max_value
    # However, we initialize the starting node's value with 0
    shortest_path[start_node] = 0

    # The algorithm executes until we visit all nodes
    while unvisited_nodes:
        # The code block below finds the node with the lowest score
        current_min_node = None
        for node in unvisited_nodes: # Iterate over the nodes
            if current_min_node == None:
                current_min_node = node
            elif shortest_path[node] < shortest_path[current_min_node]:
                current_min_node = node

        # The code block below retrieves the current node's neighbors and updat
        neighbors = graph.get_outgoing_edges(current_min_node)
        for neighbor in neighbors:
            tentative_value = shortest_path[current_min_node] + graph.value(cur
            if tentative_value < shortest_path[neighbor]:
                shortest_path[neighbor] = tentative_value
                # We also update the best path to the current node
                previous_nodes[neighbor] = current_min_node

        # After visiting its neighbors, we mark the node as "visited"
        unvisited_nodes.remove(current_min_node)

    return previous_nodes, shortest_path
```