Algoritmi greedy

Progettazione di Algoritmi a.a. 2023-24
Matricole congrue a 1
Docente: Annalisa De Bonis

1

Scelta greedy

Un algoritmo greedy è un algoritmo che effettua ad ogni passo la scelta che in quel momento sembra la migliore (localmente ottima) nella speranza di ottenere una soluzione globalmente ottima.

Domanda. Questo approccio porta sempre ad una soluzione ottima?
Risposta. Non sempre ma per molti problemi sì.
Esempio:
Voglio andare in auto dalla città a alla città b. La distanza tra a e b e di 1500 km. Lungo la strada ci sono n punti in cui posso fermarmi a fare rifornimento di carburante (a e b compresi). Voglio minimizzare il numero di volte in cui devo fermarmi per fare rifornimento considerando che con un pieno posso percorrere 450 km e che alla partenza il serbatoio è vuoto.
Strategia greedy: faccio il pieno in a e arrivo al più lontano punto di rifornimento che riesco a raggiungere (distanza da a ≤ 450 km). Da questo punto raggiungo il punto di rifornimento più lontano che si trova ad al più 450 km da esso, e così via fino a che non arrivo in b. Si può dimostrare che questa strategia fornisce la soluzione ottima.

2

Cammini minimi

- Si vuole andare da Napoli a Milano in auto percorrendo il minor numero di chilometri
- Si dispone di una mappa stradale su cui sono evidenziate le intersezioni tra le strade ed è indicata la distanza tra ciascuna coppia di intersezioni adiacenti
- Come si può individuare il percorso più breve da Napoli a Milano?

3

Cammini minimi

- Esempi di applicazioni dei cammini minimi in una rete
- Trovare il cammino di **tempo minimo** in una rete
- Se i pesi esprimono l'inaffidabilità delle connessioni in una rete, trovare il collegamento che è **più sicuro**

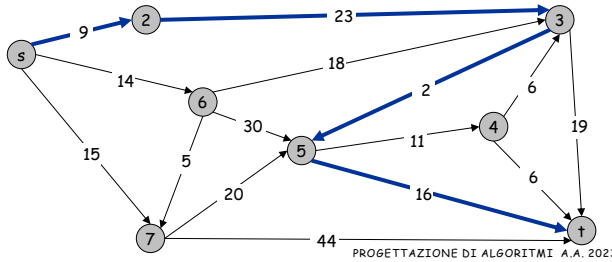
4

Il problema dei cammini minimi

- Input:
  - Grafo direzionato G = (V, E).
  - Per ogni arco e, valore ℓ_e (costo, peso o lunghezza dell'arco e)
  - s = sorgente
- Def. Per ogni percorso direzionato P, $\ell(P)$ = somma delle lunghezze degli archi in P.

Il problema dei cammini minimi: trova i percorsi direzionati più corti da s verso tutti gli altri nodi.

NB: Se il grafo non è direzionato possiamo sostituire ogni arco (u,v) con i due archi direzionati (u,v) e (v,u)



Costo del percorso da s a t
s-2-3-5-t = 9 + 23 + 2 + 16 = 50

5

Varianti del problema dei cammini minimi

- Single Source Shortest Paths: determinare il cammino minimo da un dato vertice sorgente s ad ogni altro vertice
- Single Destination Shortest Paths: determinare i cammini minimi ad un dato vertice destinazione t da tutti gli altri vertici
  - Si riduce a Single Source Shortest Path invertendo le direzioni degli archi
- Single-Pair Shortest Path: per una data coppia di vertici u e v determinare un cammino minimo da un dato vertice u a v
  - i migliori algoritmi noti per questo problema hanno lo stesso tempo di esecuzione asintotico dei migliori algoritmi per Single Source Shortest Path.
- All Pairs Shortest Paths: per ogni coppia di vertici u e v, determinare un cammino minimo da u a v

6

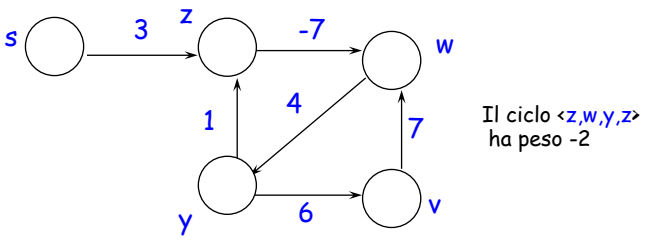
Cammini minimi

- Soluzione inefficiente:
  - si considerano tutti i percorsi possibili e se ne calcola la lunghezza
  - il numero di percorsi potrebbe essere esponenziale (ad esempio, nel grafo completo)
  - l'algoritmo non termina in presenza di cicli
- Si noti che l'algoritmo di visita BFS è un algoritmo per Single Source Shortest Paths nel caso in cui tutti gli archi hanno lo stesso peso

7

Cicli negativi

- Se esiste un ciclo negativo lungo un percorso da s a v, allora non è possibile definire il cammino minimo da s a v



Il ciclo <z,w,y,z> ha peso -2

- Attraversando il ciclo <z,w,y,z> un numero arbitrario di volte possiamo trovare percorsi da s a v di peso arbitariamente piccolo

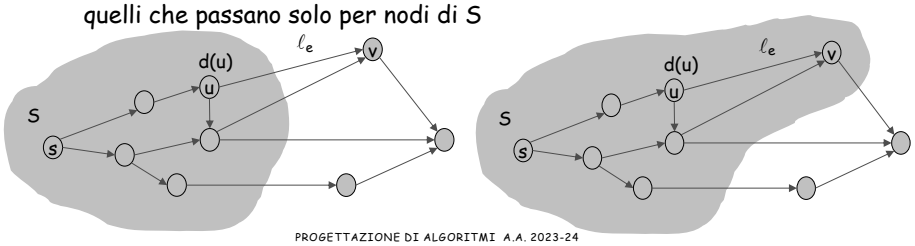
8

Algoritmo di Dijkstra

in G solo archi con lunghezze maggiori o uguali di zero

Algoritmo di Dijkstra (1959).

- Ad ogni passo mantiene l'insieme S dei nodi esplorati, cioé di quei nodi u per cui è già stata calcolata la distanza minima d(u) da s.
- Inizializzazione S = { s }, d(s) = 0.
- Ad ogni passo, sceglie tra i nodi non ancora in S ma adiacenti a qualche nodo di S, quello che può essere raggiunto nel modo più economico possibile (scelta greedy)
- In altra parole sceglie v che minimizza $d'(v) = \min_{e=(u,v)\,:\,u\in S} d(u) + \ell_e$, aggiunge v a S e pone d(v) = d'(v).
- d'(v) rappresenta la lunghezza del percorso più corto da s a v tra quelli che passano solo per nodi di S



9

Algoritmo di Dijkstra

in G solo archi con lunghezze maggiori o uguali di zero

```
Dijkstra's Algorithm (G, ℓ), s)
Let S be the set of explored nodes
    For each u ∈ S, we store a distance d(u)
Initially S = {s} and d(s) = 0
While S ≠ V
    Select a node v ∉ S with at least one edge from S for which
       d'(v) = min_{e=(u,v):u∈S} d(u) + ℓ_e is as small as possible
    Add v to S and define d(v) = d'(v)
EndWhile
```

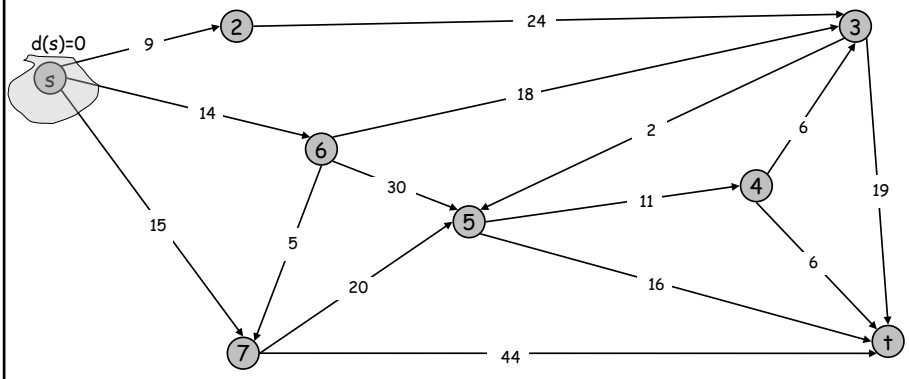
Questa versione dell'algoritmo di Dijkstra assume che tutti i nodi siano raggiungibili da s. Se non si puo` fare questa assunzione allora si puo` modificare l'algoritmo inserire un if per interrompe il ciclo se ad una certa iterazione non ci sono nodi in cui entrano archi provenienti da nodi di S .

Esercizio: Ad ogni passo l'algoritmo di Dijkstra aggiunge un certo nodo v ad S. Dimostrare per induzione che il percorso più corto da s a v computato dall'algoritmo passa solo attraverso nodi già inseriti in S.

10

Algoritmo di Dijkstra

S={s} computo d'(2)=9 , d'(6)=14, d'(7)=15 → scelgo 2 e pongo d(2)=9, shortest path = s-2

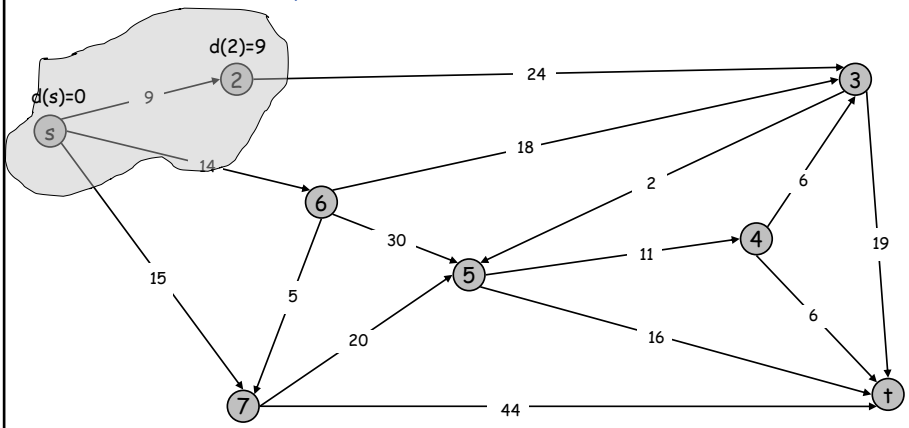


11

11

Algoritmo di Dijkstra

S={s,2} computo d'(6)=14, d'(7)=15, d'(3)=9+24=33 → scelgo 6 e pongo d(6)=14 , shortest path = s-6

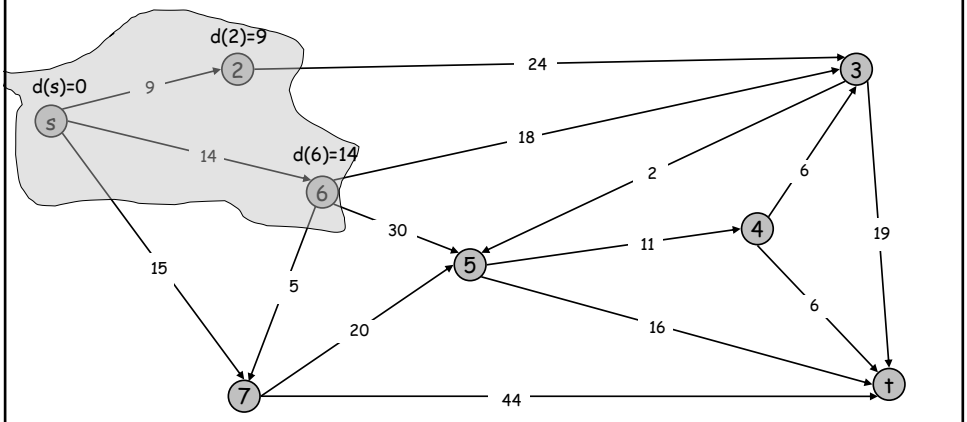


12

12

Algoritmo di Dijkstra

S={s,2,6} computo d'(7)=min{15,14+5}=15, d'(5)=14+30=44, d'(3)=min{9+24, 14+18} =32 → scelgo 7 e pongo d(7)=15, shortest path = s-7

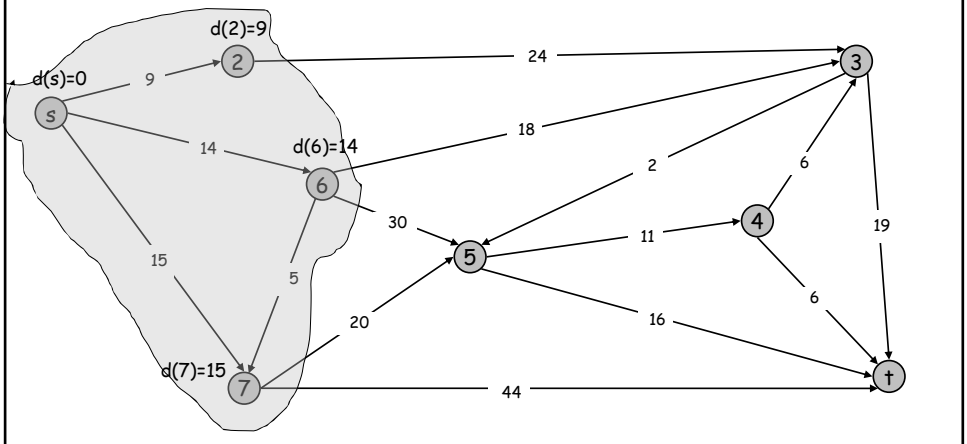


13

13

Algoritmo di Dijkstra

S={s,2,6,7} computo d'(5)=min{14+30,15+20}=35, d'(3)=min{9+24, 14+18} =32, d'(t)=15+44=59 → scelgo 3 e pongo d(3)=32 , shortest path = s-6-3

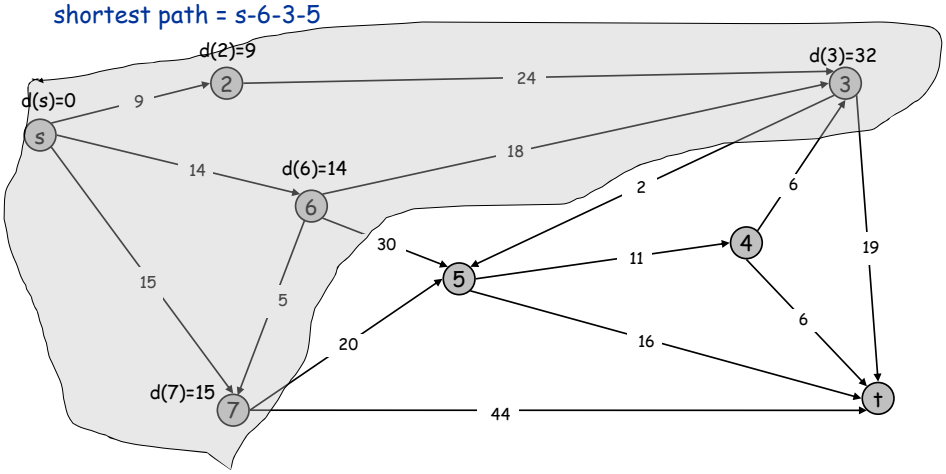


14

14

Algoritmo di Dijkstra

S={s,2,6,7,3} computo d'(5)=min{14+30,15+20,32+2}=34,
d'(t)=min{15+44, 32+19}=51 → scelgo 5 e pongo d(5)=34 ,
shortest path = s-6-3-5

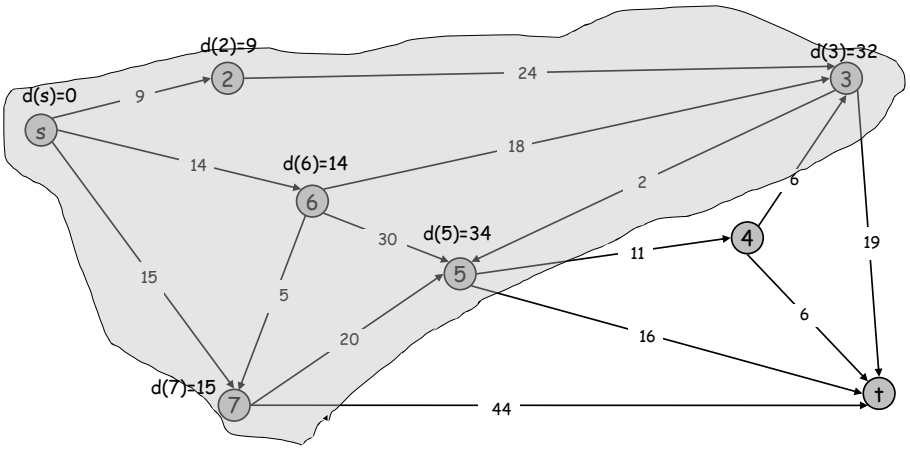


15

15

Algoritmo di Dijkstra

S={s,2,6,7,3,5} computo d'(4)=34+11=45, d'(t)=min{15+44, 32+19, 34+16}=50
→ scelgo 4 e pongo d(4)=45 , shortest path = s-6-3-5-4

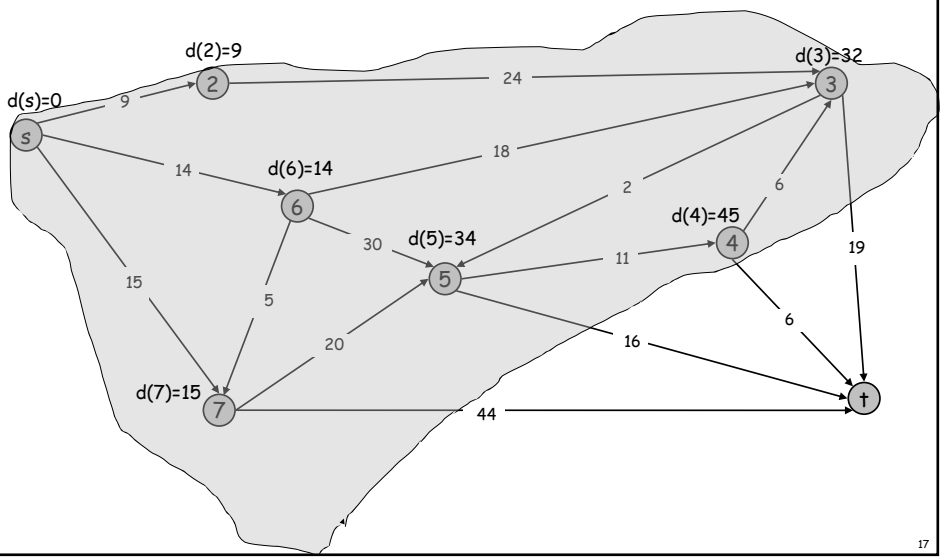


16

16

Algoritmo di Dijkstra

S={s,2,6,7,3,5,4} computo d'(t)=min{15+44, 32+19, 34+16,45+6}=50
→ scelgo t e pongo d(t)=50 , shortest path = s-6-3-5-t

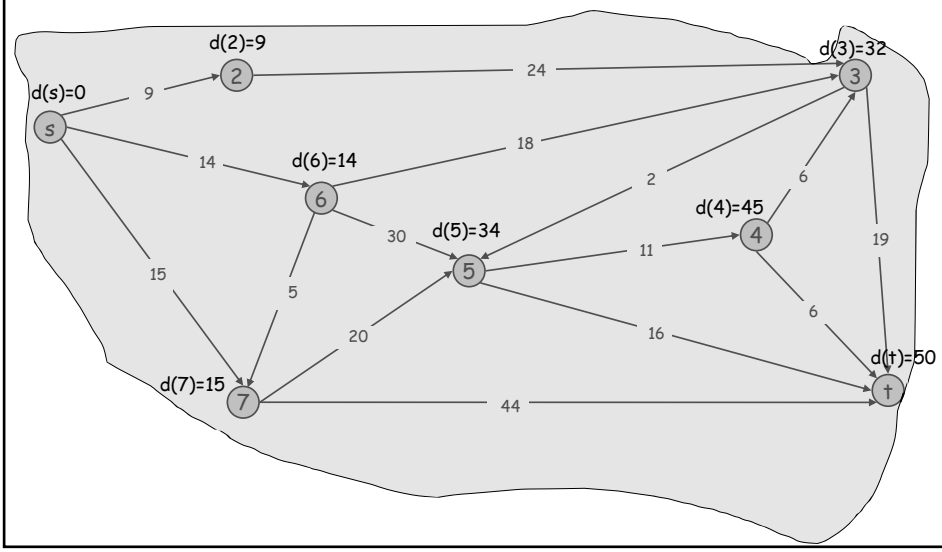


17

17

Algoritmo di Dijkstra

S={s,2,6,7,3,5,4,t}



18

18

Algortimo di Dijkstra: Correttezza

Teorema. Sia G un grafo in cui per ogni arco e è definita una lunghezza $\ell_e \geq 0$ (è fondamentale che ℓ_e **non sia negativa**). Per ogni nodo u ∈ S il valore d(u) calcolato dall'algoritmo di Dijkstra è la lunghezza del percorso più corto da s a u.

Dim. (per induzione sulla cardinalità |S| di S)

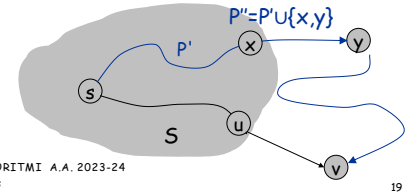
Base: |S| = 1. In questo caso S={s} e d(s)=0 per cui la tesi vale banalmente.

Ipotesi induttiva: Assumiamo vera la tesi per |S| = k ≥ 1.

- Sia v il prossimo nodo inserito in S dall'algoritmo e sia (u,v) l'arco attraverso il quale è stato raggiunto v, cioè quello per cui si ottiene

$$d'(v) = \min_{e=(u,v):\, u \in S} d(u) + \ell_e ,$$

- Consideriamo il percorso di lunghezza d'(v), cioé quello formato dal percorso più corto da s ad u più l'arco (u,v)
- Consideriamo un **qualsiasi** altro percorso P da s a v. Dimostriamo che P non è più corto di d'(v)
- Sia (x,y) il primo arco di P che esce da S.
- Sia P' il sottocammino di P fino a x.
- P' più l'arco (x,y) ha già lunghezza non più piccola di d'(v) altrimenti l'algoritmo non avrebbe scelto v.



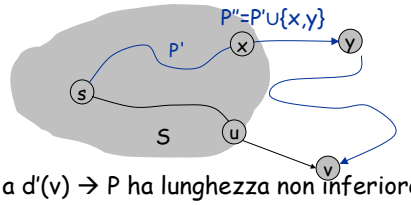
19

Algortimo di Dijkstra: Correttezza

- Dimostriamo che il percorso P'' formato da P' più l'arco (x,y) ha già lunghezza non più piccola di d'(v).

1. ℓ(P') + ℓ((x,y)) ≥ d(x) + ℓ((x,y)) siccome d(x) è per ipotesi induttiva uguale alla lunghezza del percorso più corto da s a x
2. d(x) + ℓ((x,y)) ≥ d'(y) siccome d'(y)= min $_{(u,y):\, u \in S}$ d(u)+ ℓ((u,y))
3. d'(y) ≥ d'(v) altrimenti alla k-esima iterazione l'algoritmo non avrebbe inserito v in S

- 1,2,3 → ℓ(P''))) ≥ d'(v)



- P'' ha lunghezza non inferiore a d'(v) → P ha lunghezza non inferiore a d'(v)
  - **l'implicazione è vera perchè P'' è il sottocammino di P cha va da s a y e il sottocammino di P che va da y a v non può avere costo negativo.**

20