ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

Cammini minimi su grafi: singola sorgente e pesi negativi

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

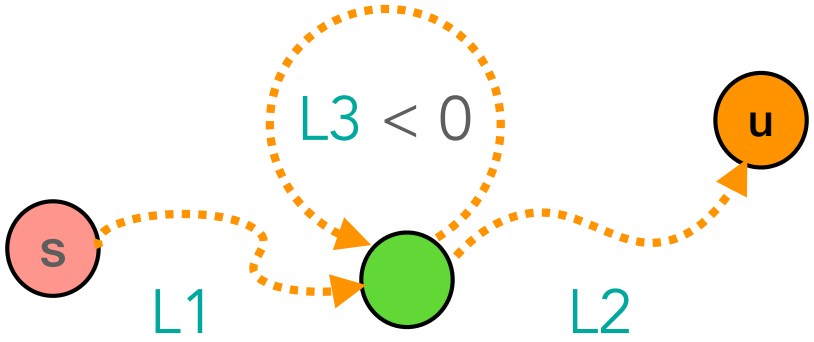
Cammini minimi da sorgente singola

DOMANDA:

cosa succede se gli archi possono avere pesi negativi?

RISPOSTA:

Se c'è un ciclo di lunghezza negativa, il problema è mal posto



Qual è la distanza di u da s?

Cammini minimi da sorgente singola

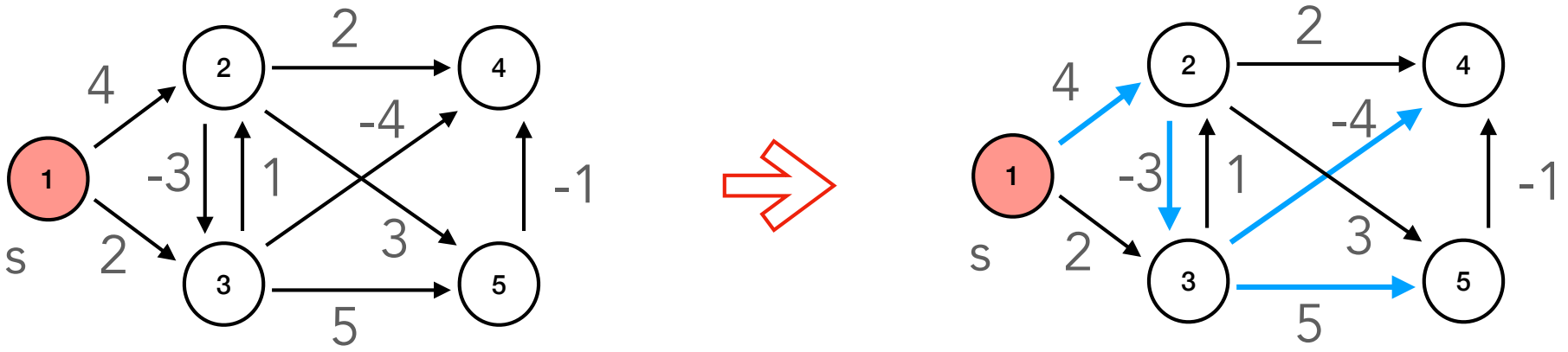
PROBLEMA:

INPUT: Grafo $G = (V, E)$ (diretto/indiretto) senza cicli negativi

funzione di costo non negativa sugli archi $c : E \rightarrow R$

un nodo sorgente $s \in V$

OUTPUT: l'albero cammini minimi radicato in s



cammini minimi

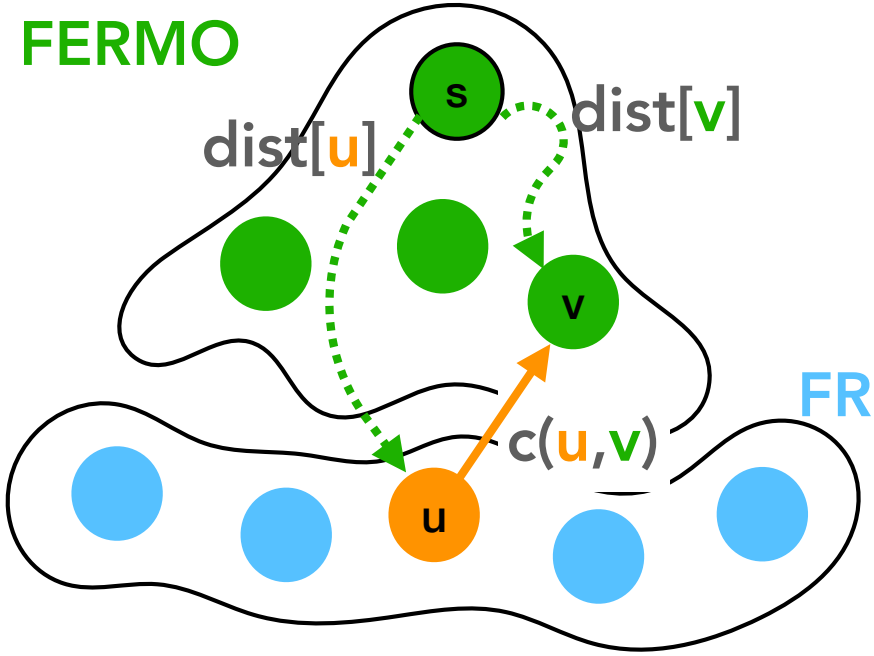
Cammini minimi da sorgente singola

DOMANDA:

Se ci sono archi con pesi negativi, ma non ci sono cicli negativi, l'algoritmo di Dijkstra funziona ancora?

RISPOSTA: NO

MONDO FERMO



Rilassamento arco verso il mondo fermo

dist[**u**] + c(**u**,**v**) < dist[**v**] ?

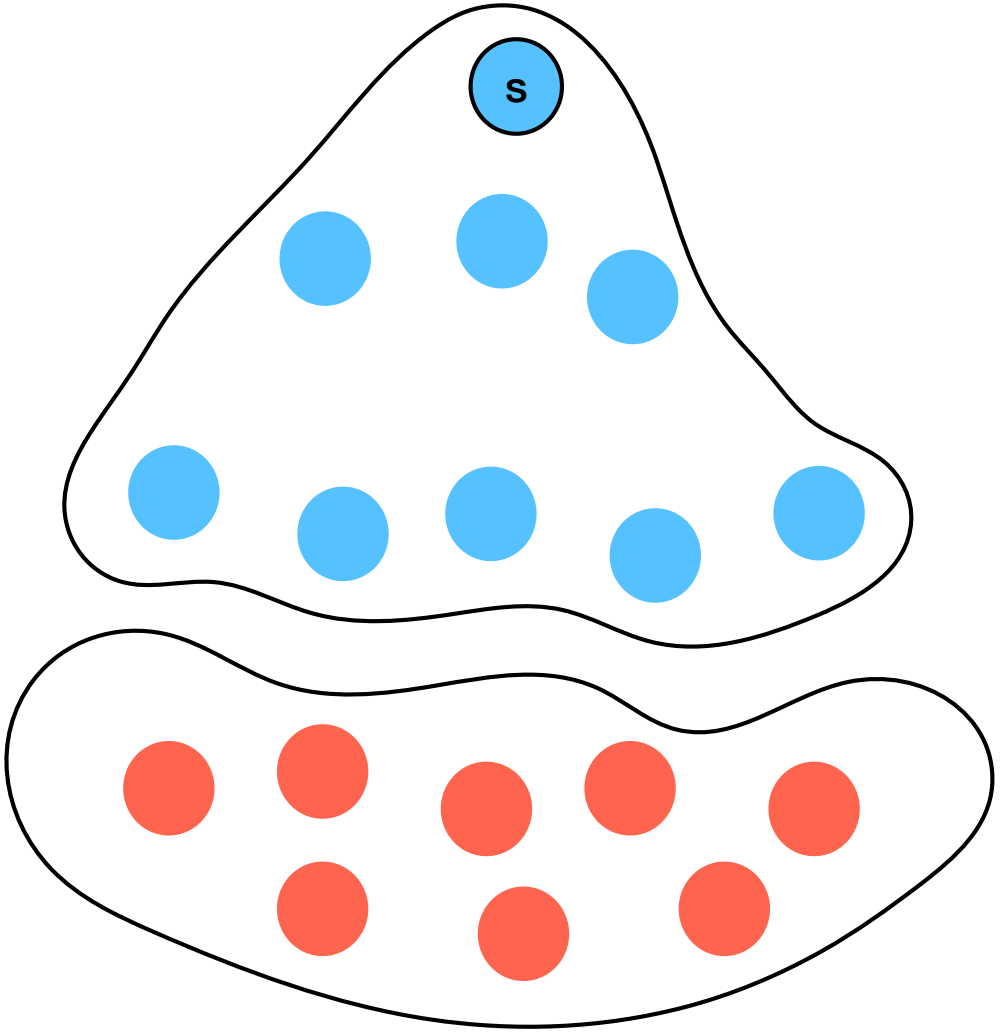
SI

se c(**u**,**v**) < 0 e sufficientemente piccolo!

NON ESISTE IL MONDO FERMO!!

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD



SCOPERTI

u dist[u] ≠ +∞
POTREBBE cambiare

NON SCOPERTI

u dist[u] = +∞

- Rilassiamo TUTTI gli archi ripetutamente!

Quante volte?

|V| - 1 volte è sufficiente

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

```
Bellman-Ford(G,c,s)
 for all u ∈ V
  dist[u] := +∞
  prev[u] := 0
 dist[s] := 0

 for i=1 to |V|-1 do
  for all (u,v) ∈ E do
   if dist[v] > dist[u] + c(u,v)
    then
     dist[v] := dist[u] + c(u,v)
     prev[v] := u

 return prev[]
```

INIZIALIZZAZIONE

RILASSAMENTO

arco (u,v)

Costo computazionale

$O(|E||V|)$

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v

`prev[u1] = s`

Prima iterazione
di rilassamenti

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v

`prev[u1] = s`

`prev[u2] = u1`

Seconda iterazione
di rilassamenti

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v

`prev[u1] = s`

`prev[u3] = u2`

`prev[u2] = u1`

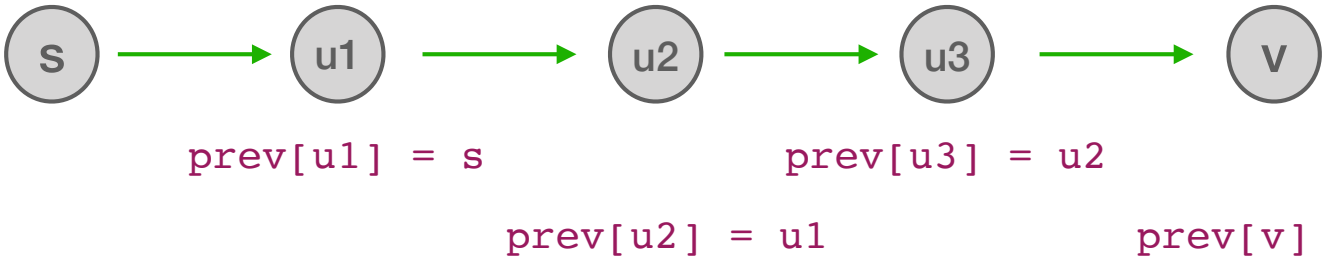
Terza iterazione
di rilassamenti

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v



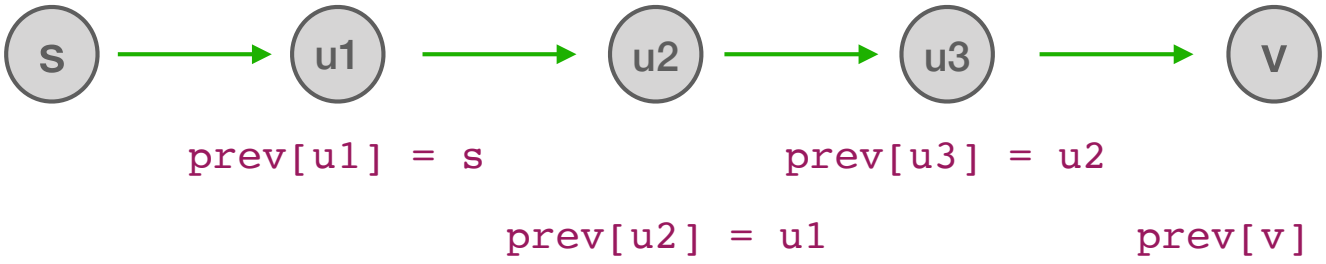
Quarta iterazione
di rilassamenti

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

Perché funziona??

cammino minimo da s a v



Nelle successive iterazioni
di rilassamenti NON cambieranno più

⇒ Iteriamo un numero sufficiente di volte per
coprire tutti i cammini minimi di ogni possibile lunghezza!

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

E se non avessimo la garanzia
che nel grafo non ci sono cicli negativi?

AGGIUNGIAMO un'iterazione di RILASSAMENTO

Se troviamo un cammino più breve per un solo nodo
⇒ nel grafo c'è un ciclo negativo!

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

```
Bellman-Ford(G,c,s)
 for all u ∈ V
  dist[u] := +∞
  prev[u] := 0
 dist[s] := 0

 for i=1 to |V|-1 do
  for all (u,v) ∈ E do
   if dist[v] > dist[u] + c(u,v)
    then
     dist[v] := dist[u] + c(u,v)
     prev[v] := u

 for all (u,v) ∈ E do
  if dist[v] > dist[v] + c(u,v)
   then return "esiste un ciclo negativo"

 return prev[]
```

INIZIALIZZAZIONE

RILASSAMENTO
arco (u,v)

DETERMINAZIONE
ciclo negativo

Costo computazionale $O(|E||V|)$

Cammini minimi da sorgente singola

ALGORITMO di BELLMAN-FORD

ESERCIZIO

Modificare l'algoritmo di Bellman Ford per terminare anticipatamente nel caso in cui si converga prima delle |V|-1 iterazioni

SUGGERIMENTO

Usare un **flag**
(una variabile che memorizza un valore booleano vero o falso)
che segnala l'avvenuta o non avvenuta
modifica della lunghezza di un cammino

```
Bellman-Ford(G,c,s)
 for all u ∈ V
  dist[u] := +∞
  prev[u] := 0

 dist[s] := 0
 continua := True

 while i<|V| AND continua do
  continua := False
  for all (u,v) ∈ E do
   if dist[v] > dist[u] + c(u,v)
    then
     dist[v] := dist[u] + c(u,v)
     prev[v] := u
     continua := True

 if continua
  then
   for all (u,v) ∈ E do
    if dist[v] > dist[v] + c(u,v)
     then return "ciclo negativo"
 return prev[]
```

ALGORITMO di BELLMAN-FORD
con "uscita anticipata"

`continua = True`

è necessaria almeno un'altra iterazione del ciclo `while`

`continua = False`

non sono necessarie altre iterazioni del ciclo `while`

Costo computazionale

$O(|E||V|)$

Cammini minimi da sorgente singola

ESERCIZIO

Scrivere lo pseudocodice dell'algoritmo
visto al lezione per calcolare i cammini minimi
da una sorgente nel caso in cui il grafo sia un DAG

INPUT: DAG G = (V,E), c : E->R, s in V
OUTPUT: albero dei cammini minimi con sorgente s

OSSERVAZIONI

1. Idea algoritmo: linearizzare il DAG e rilassare gli archi dei nodi presi in ordine topologico
2. Il nodo s può essere un nodo qualunque, non necessariamente il primo dell'ordinamento topologico
3. È possibile chiamare la funzione che abbiamo visto a lezione per l'ordinamento topologico di un DAG (ricordatevi che restituisce una pila)