ALGORITMI E STRUTTURE DATI

Dr. Manuela Montangero

A.A. 2022/23

Cammini minimi su grafi: singola sorgente e pesi non negativi

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Cammini minimi da sorgente singola

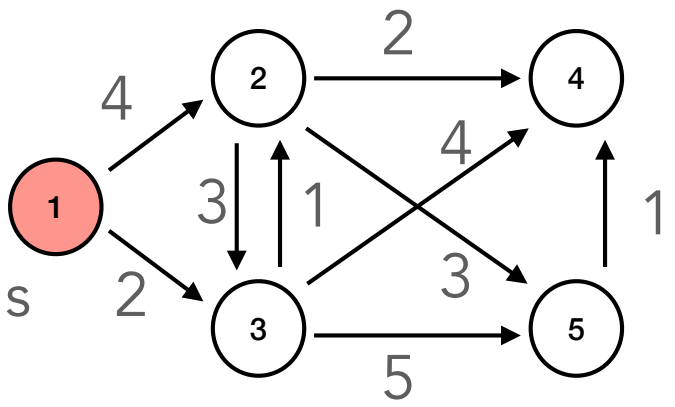
PROBLEMA:

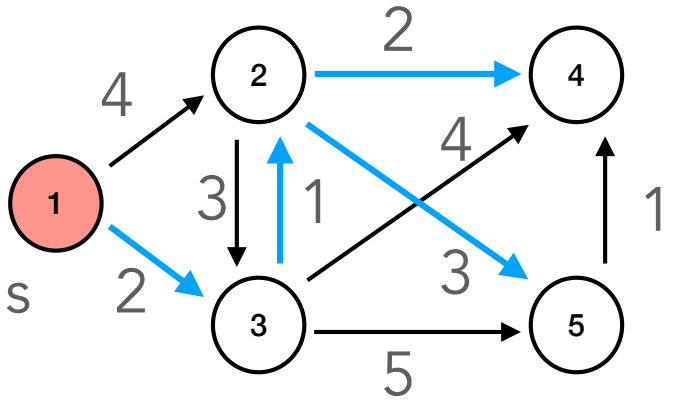
INPUT: Grafo $G = (V, E)$ (diretto/indiretto)

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



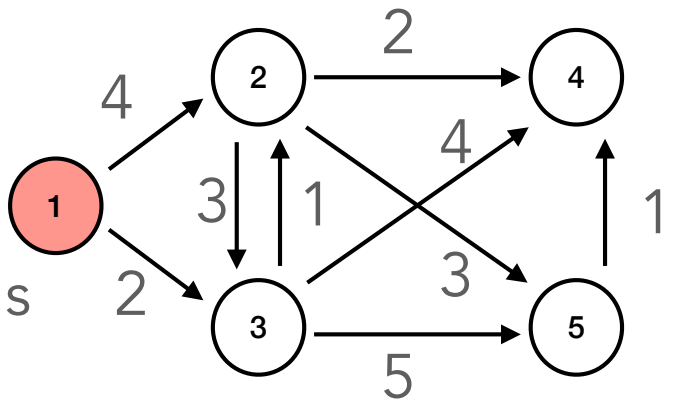


Albero dei cammini minimi

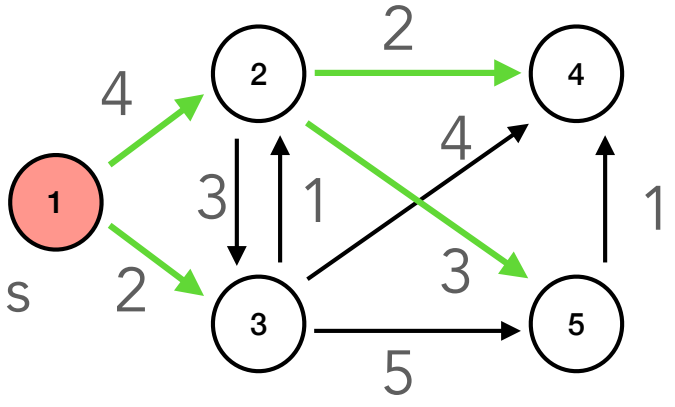
Cammini minimi da sorgente singola

DOMANDA:

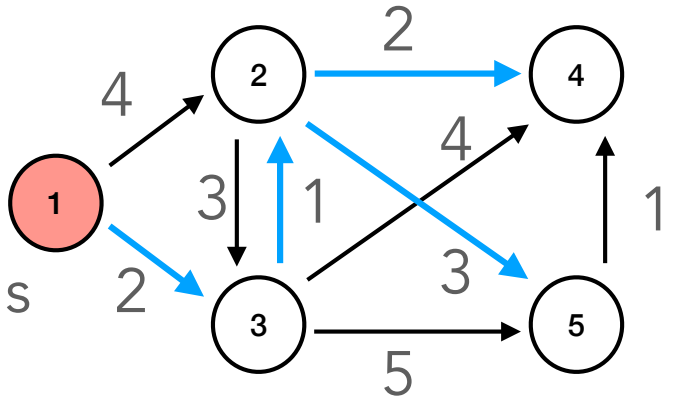
Se gli archi hanno un peso ≥ 0, la BFS risolve ancora il problema dei cammini minimi da singola sorgente?



BFS



Cammini minimi

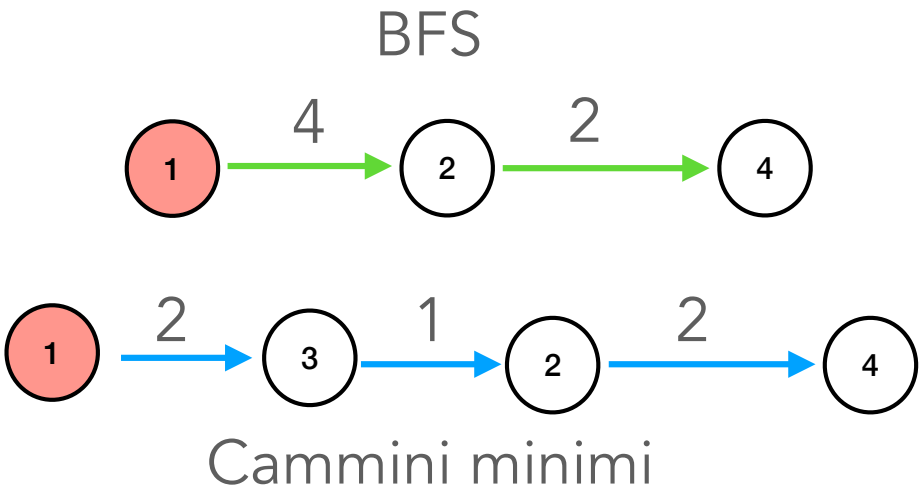


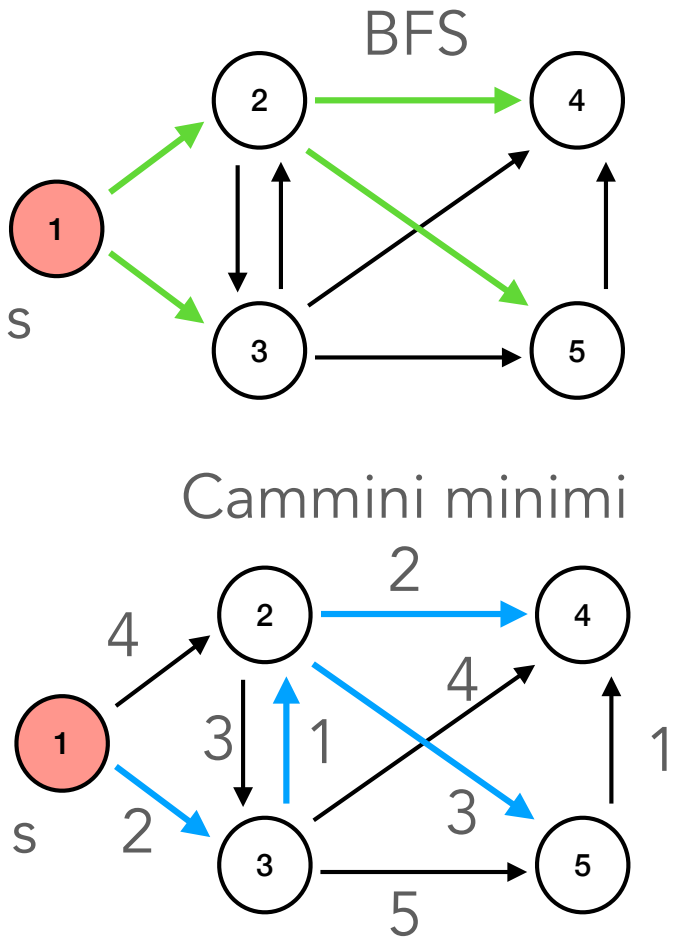
RISPOSTA: cosi' com'è NO

Cammini minimi da sorgente singola

OSSERVAZIONE

Un cammino con più archi può essere più "corto" di uno con meno archi

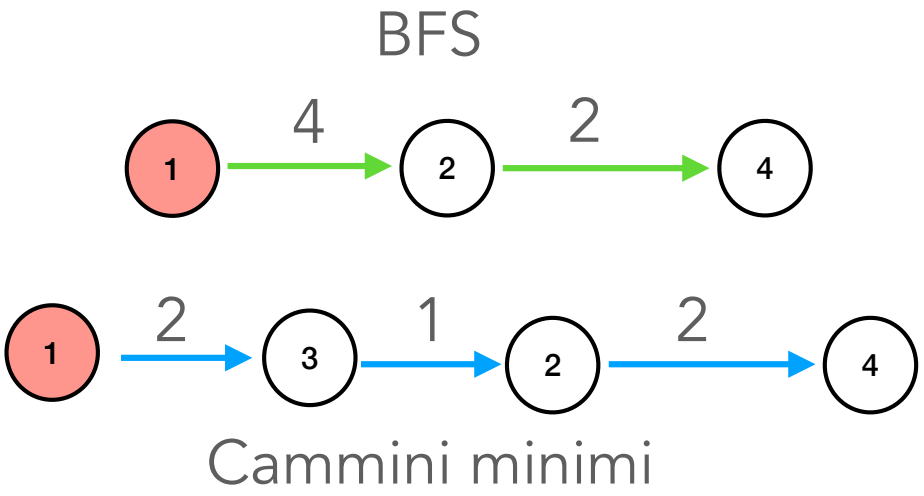


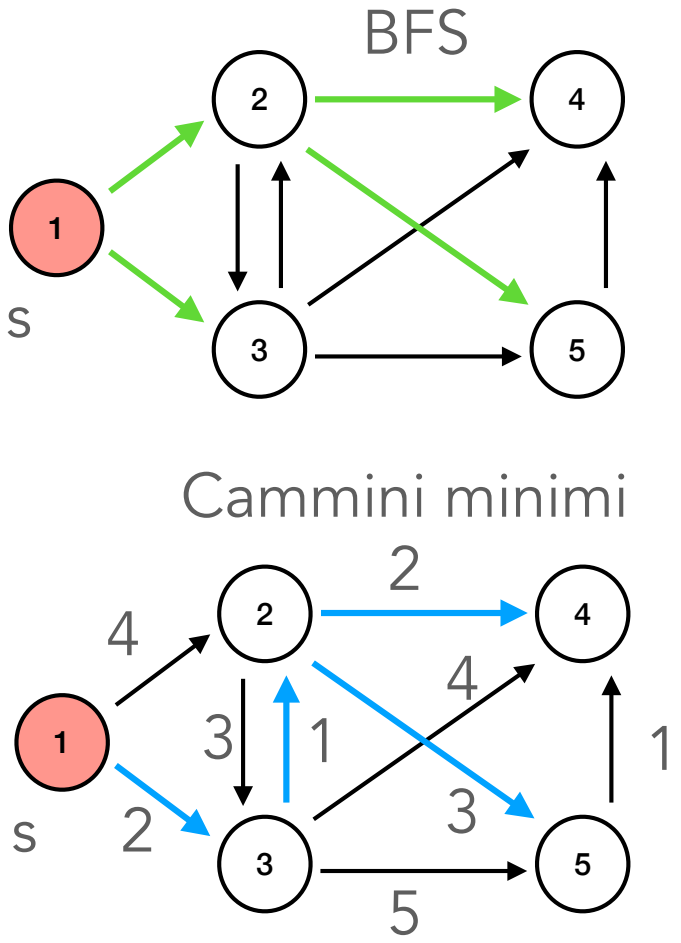


Cammini minimi da sorgente singola

OSSERVAZIONE

Durante la BFS,
l'analisi del nodo 3
riporta al nodo 2
con un cammino più corto!





Cammini minimi da sorgente singola

COME nella BFS:

- Usiamo dist[1..n] per ricordare le distanze dei nodi dalla sorgente
- Ci sono nodi scoperti (dist[] $\neq +\infty$) e nodi non ancora scoperti (dist[] = $+\infty$)
- Ad ogni iterazione: scegliamo un nodo tra quelli scoperti ed esploriamo i vicini

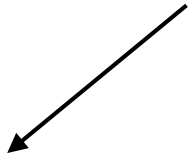
DIVERSAMENTE nella BFS:

- Ci sono nodi scoperti con distanza calcolata **non** ancora **definitiva**, e nodi scoperti con distanza calcolata **definitiva**
- la distanza di un nodo u dalla sorgente viene aggiornata solo se viene trovato un cammino più breve (anche se dist[u] $\neq +\infty$)
- Ad ogni iterazione: come scegliamo il prossimo nodo da esplorare?

Cammini minimi da sorgente singola

DIVERSAMENTE nella BFS:

- Ci sono nodi scoperti con distanza calcolata **non** ancora **definitiva**, e nodi scoperti con distanza calcolata **definitiva**
- la distanza di un nodo u dalla sorgente viene aggiornata solo se viene trovato un cammino più breve (anche se dist[u] $\neq +\infty$)
- Ad ogni iterazione: come scegliamo il prossimo nodo da esplorare?



Scegliamo tra i nodi scoperti ma non visitati quello con **dist[.] minima**

ALGORITMO di DIJKSTRA

Un nodo è visitato quando abbiamo esplorato i suoi vicini

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

`G = (V,E)` indiretto/diretto

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

 while "ci sono nodi scoperti" do
  u:= nodo scoperto con dist[.] minima

  for all (u,v) ∈ E do
   if dist[v] = +∞
    then
     marca v come nodo scoperto
   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)

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

`G = (V,E)` indiretto/diretto

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

 while "ci sono nodi scoperti" do
  u:= nodo scoperto con dist[.] minima

  for all (u,v) ∈ E do
   if dist[v] = +∞
    then
     marca v come nodo scoperto
   if dist[v] > dist[u] + c(u,v)
    then
     dist[v] := dist[u] + c(u,v)
     prev[v] := u
 return prev[]
```

INIZIALIZZAZIONE

Ci serve una struttura dati efficace per gestire i nodi scoperti

⇩

CODA CON PRIORITA'

RILASSAMENTO

arco (u,v)

Cammini minimi da sorgente singola

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

 Q := Make_priority_queue({(v,dist[v])|v = 1,…,n})
                          // coda con priorità: gli elementi sono i
                          //  nodi di V con priorità le relative dist[]

 while NOT is_empty_queue(Q) do
   u:= DeQueue(Q) // restituisce l'elemento con priorità
                  // massima (valore minimo) e lo toglie
                  // dalla coda
   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
         Decrease_Priority(Q,v,dist[v])
 return prev[]
```

$O(|V|)$

$|V|$ volte

RILASSAMENTO arco (u,v)

$O(|E|)$ per tutte le iterazioni

$O(|E|)$ volte

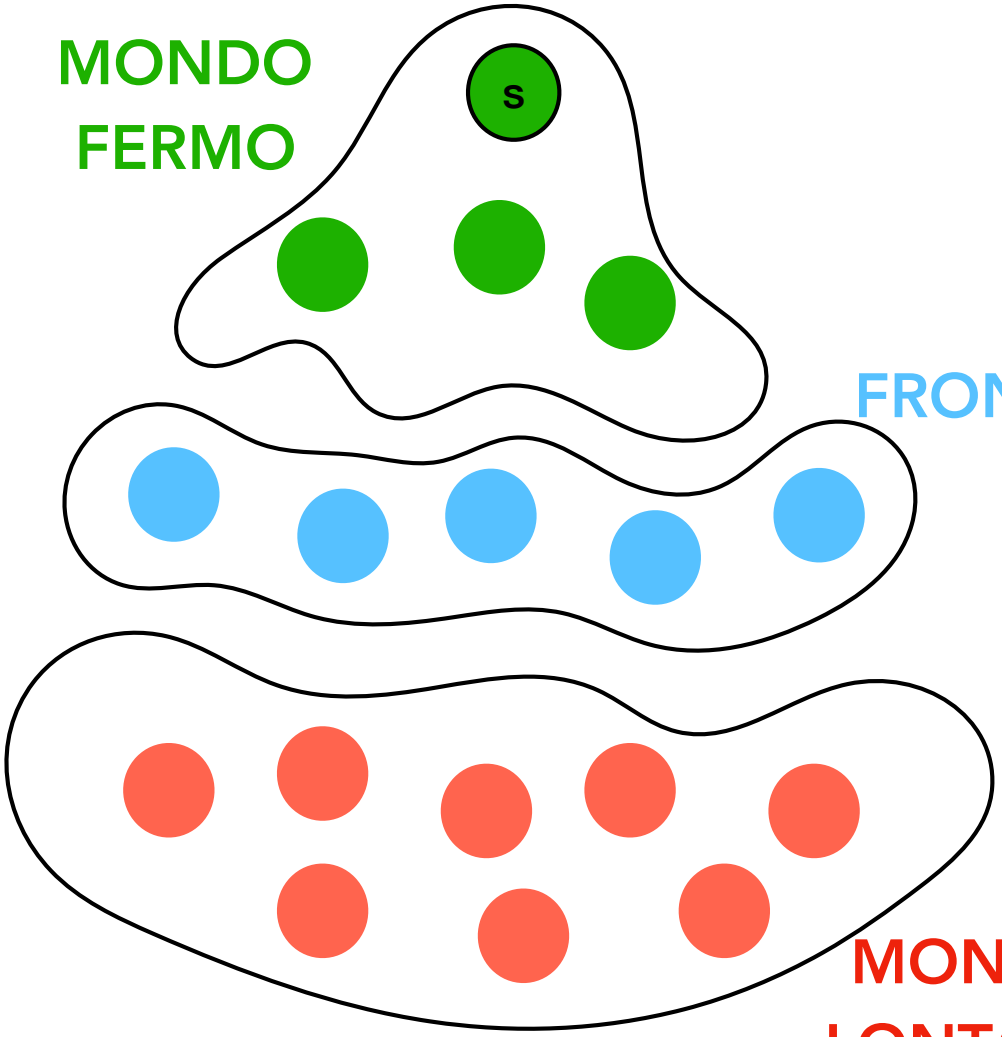
$O(|V|+|E|)$

+ costo inizializzazione coda con priorità + costo $|V|$ DeQueue

+ costo $O(|E|)$ Decrease_Priority

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



VISITATI

u dist[u] ≠ +∞
NON cambierà più

SCOPERTI, NON VISITATI

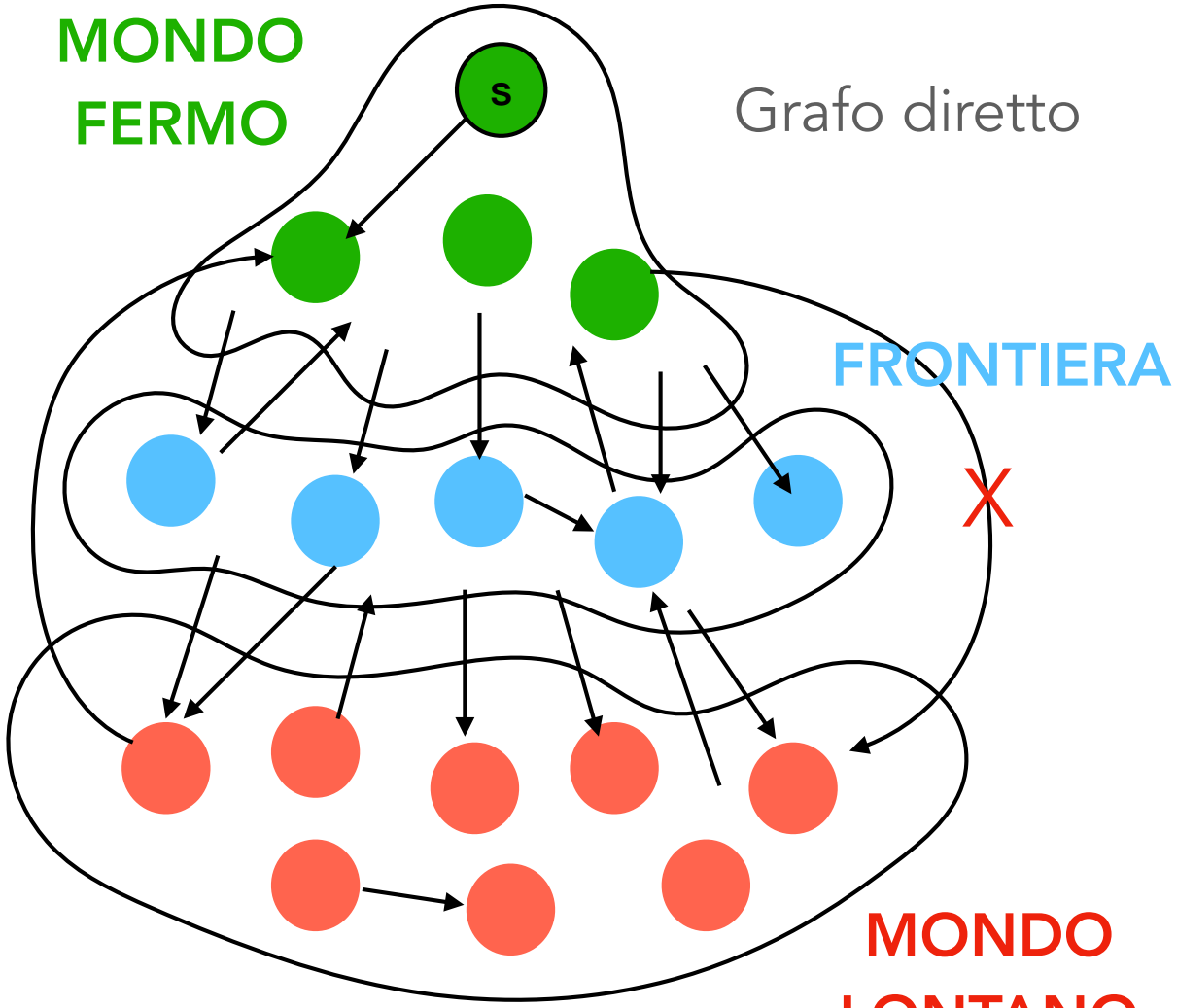
u dist[u] ≠ +∞
POTREBBE cambiare

NON SCOPERTI

u dist[u] = +∞

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



VISITATI

u dist[u] ≠ +∞
NON cambierà più

SCOPERTI, NON VISITATI

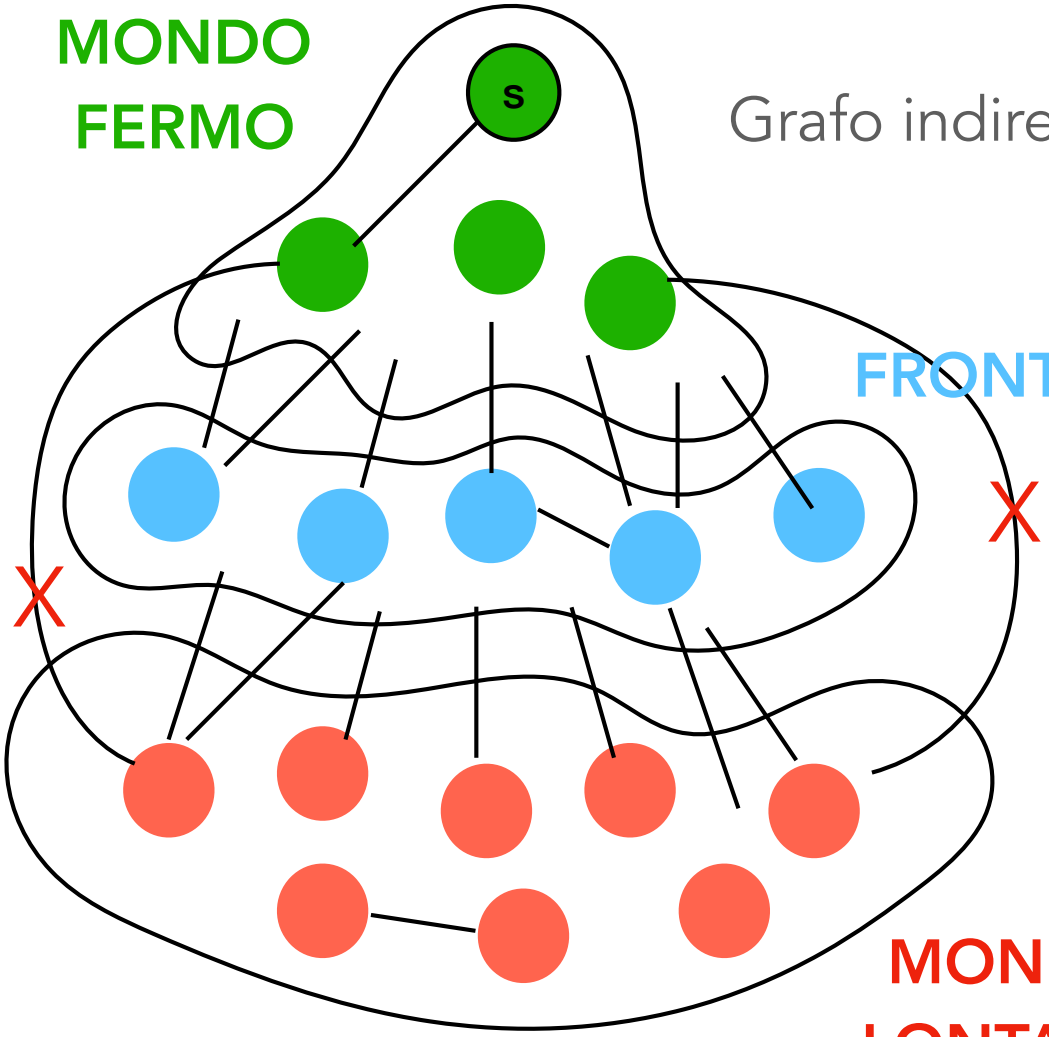
u dist[u] ≠ +∞
POTREBBE cambiare

NON SCOPERTI

u dist[u] = +∞

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



VISITATI

u dist[u] ≠ +∞
NON cambierà più

SCOPERTI, NON VISITATI

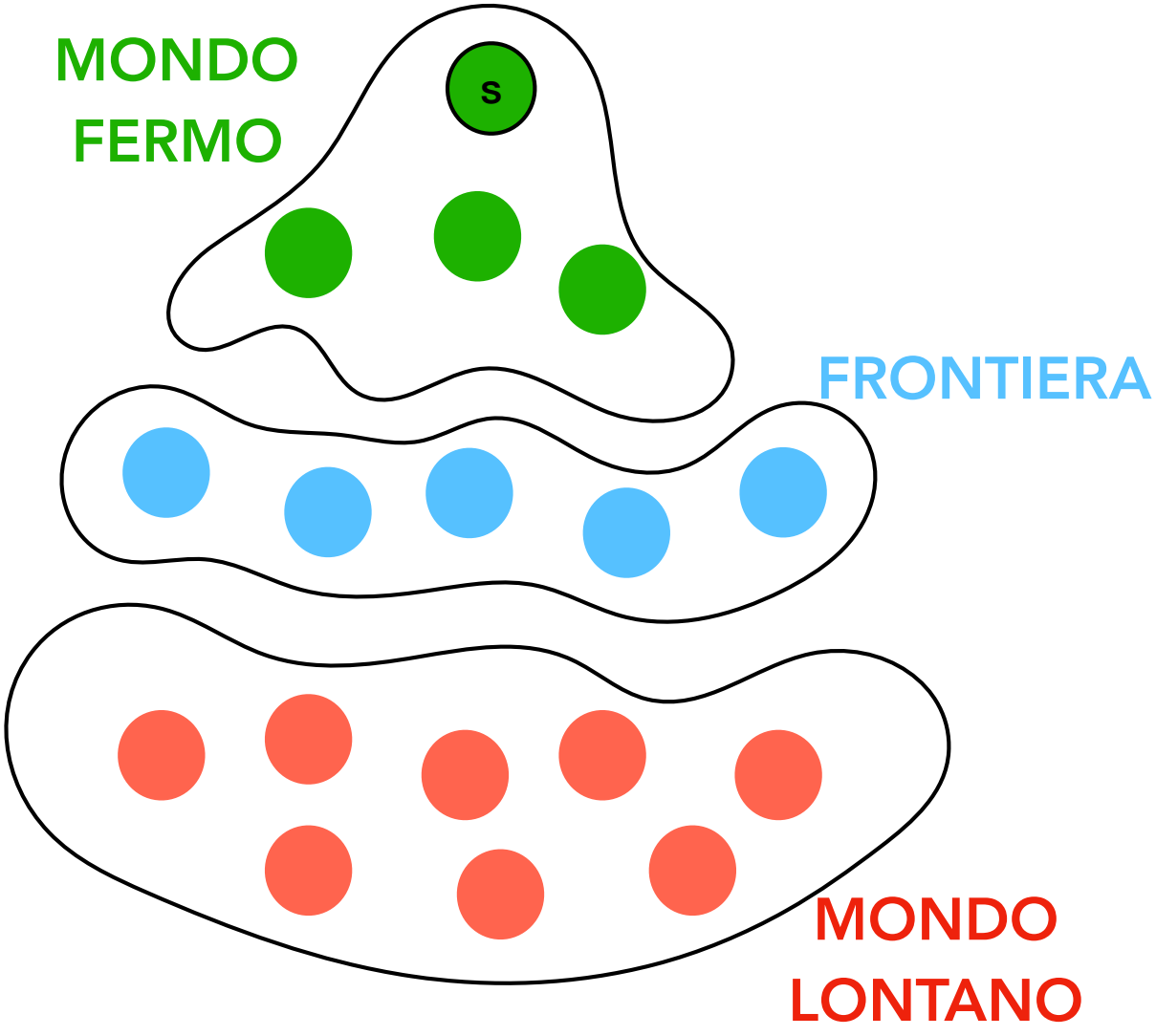
u dist[u] ≠ +∞
POTREBBE cambiare

NON SCOPERTI

u dist[u] = +∞

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

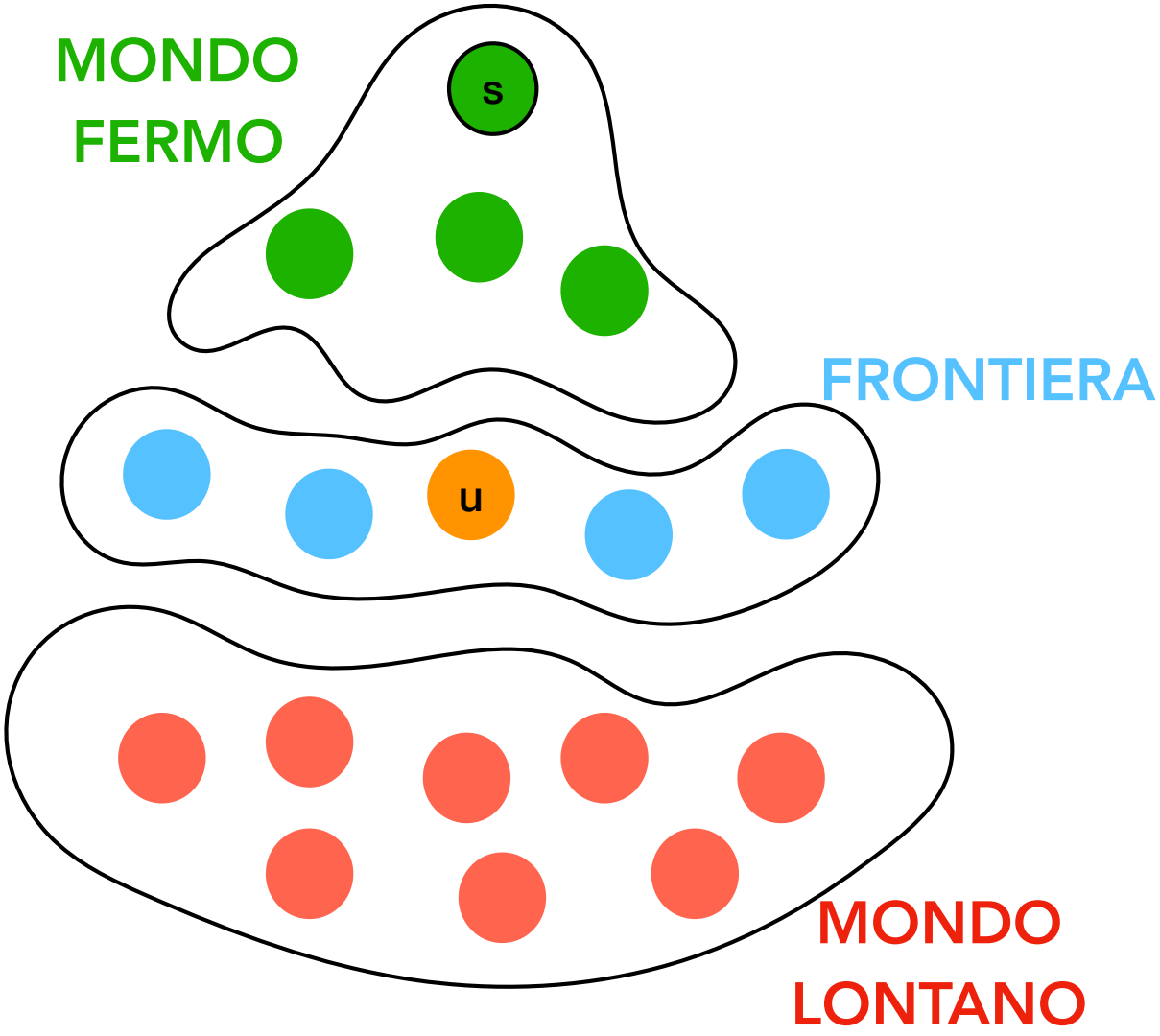


ITERAZIONE

- Scegliamo il nodo della FRONTIERA con minimo dist[.]
- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA
- Il nodo scelto si sposta nel MONDO FERMO

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

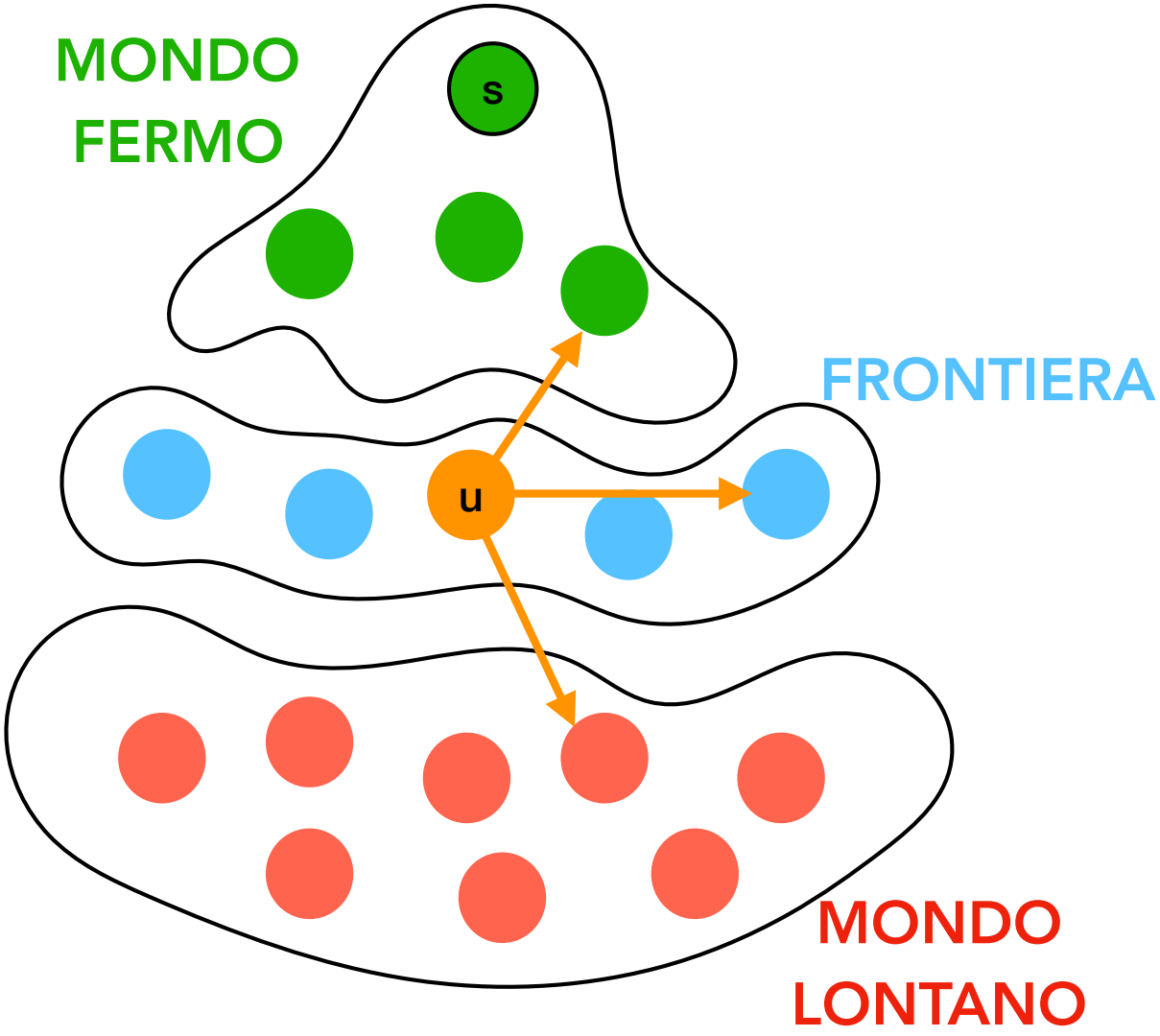


ITERAZIONE

- Scegliamo il nodo della FRONTIERA con minimo dist[.]

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

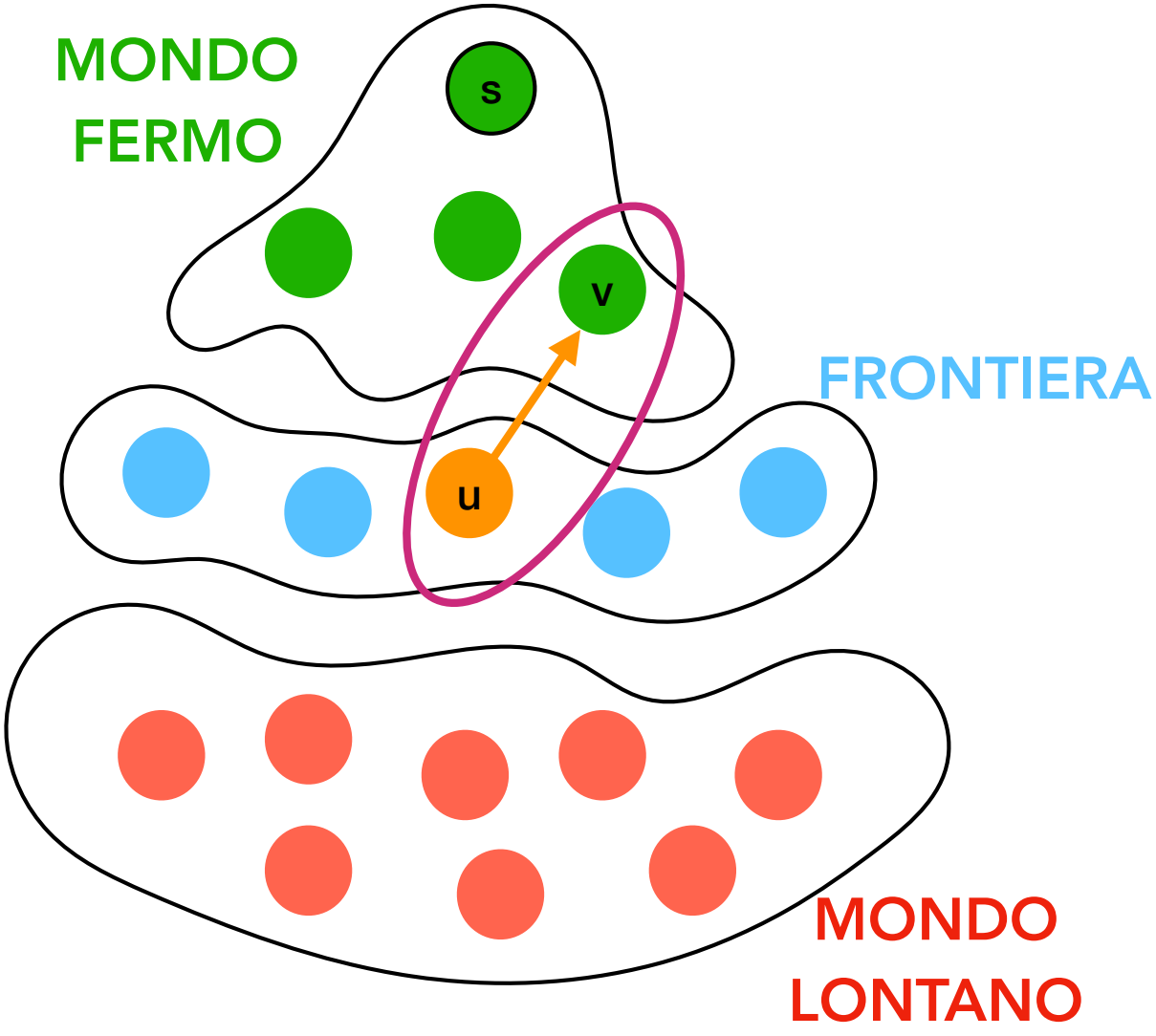


ITERAZIONE

- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



ITERAZIONE

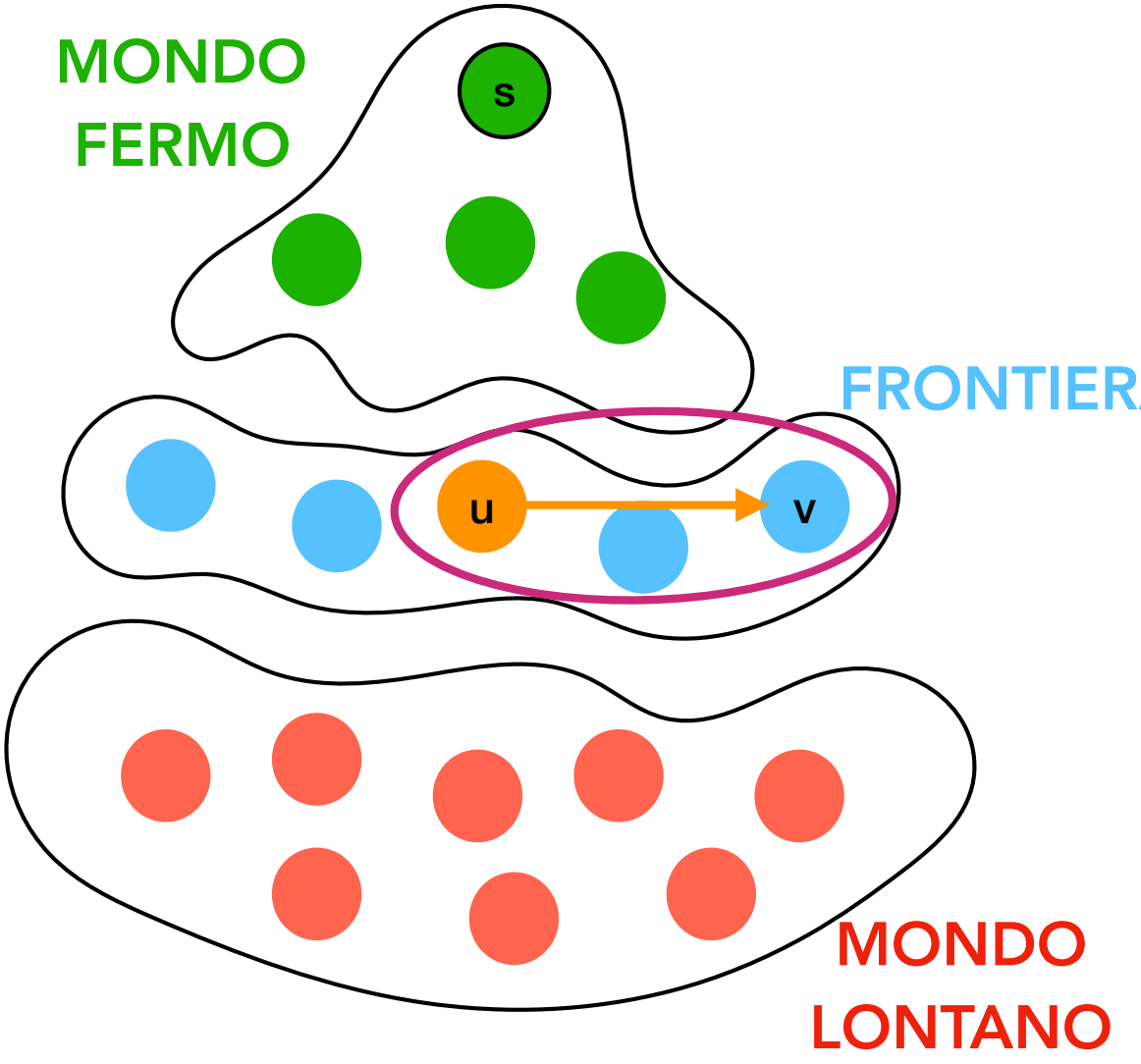
- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA

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

MAI

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



ITERAZIONE

- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA

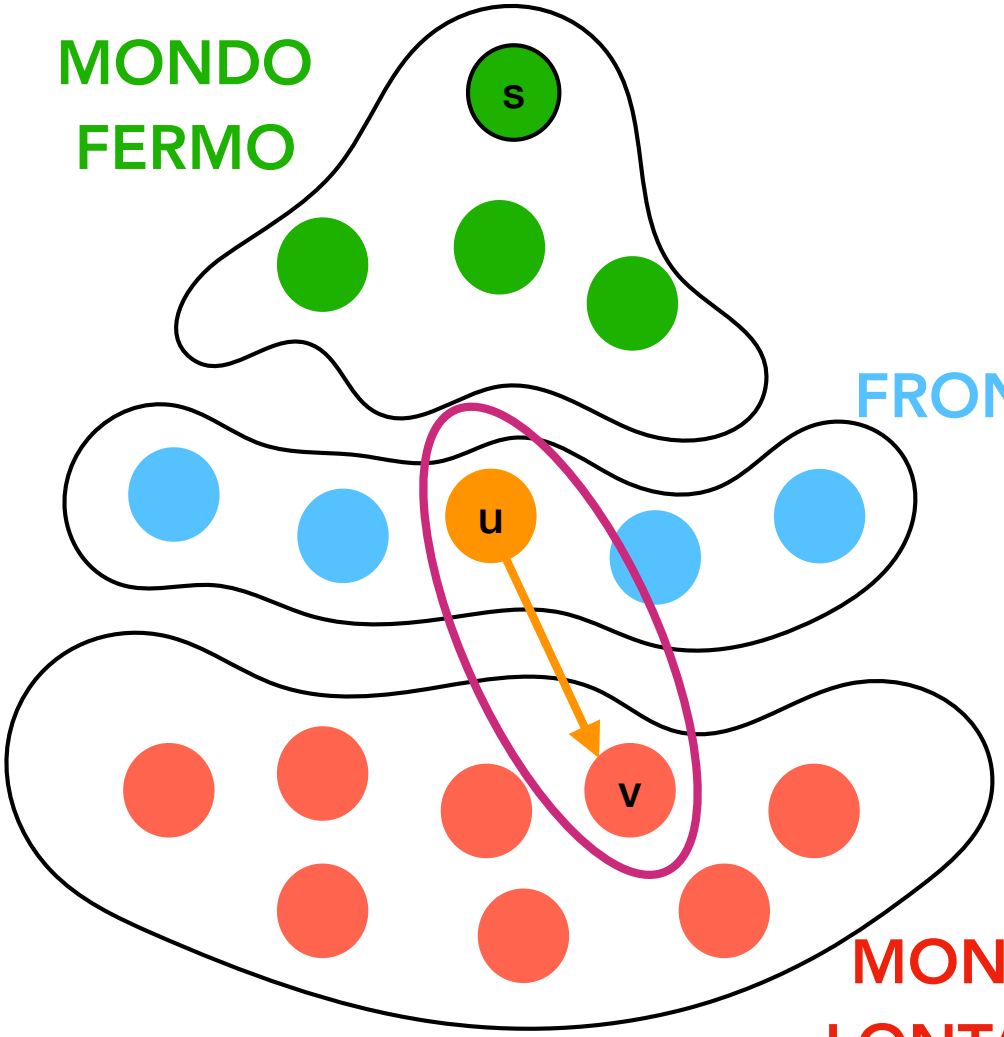
dist[**u**] + c(**u**,**v**) < dist[**v**] ?

SI —> dist[**v**] := dist[**u**] + c(**u**,**v**)

NO —> niente

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



ITERAZIONE

- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA

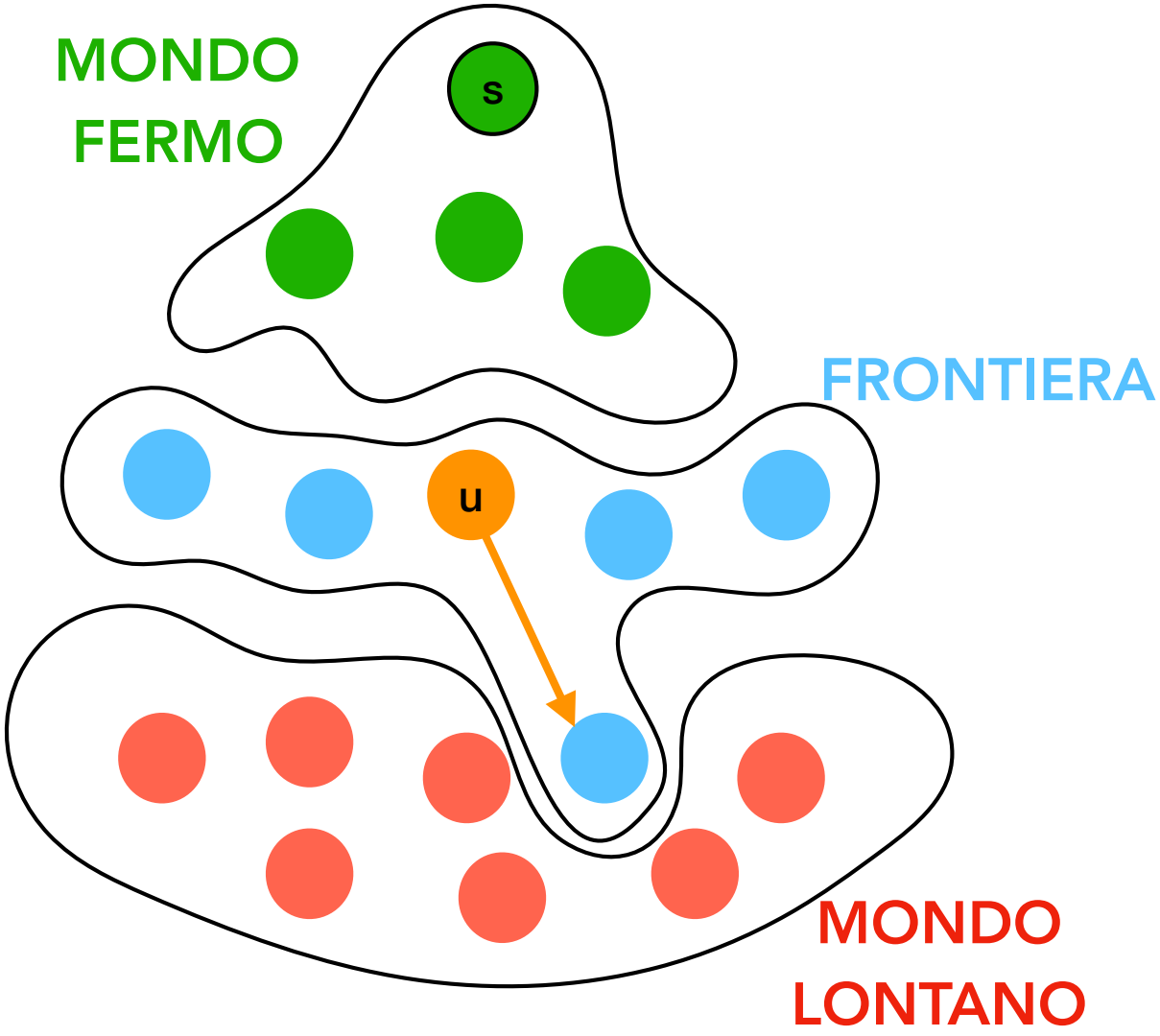
dist[**u**] + c(**u**,**v**) < dist[**v**] ?

SEMPRE

—> dist[**v**] := dist[**u**] + c(**u**,**v**)

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

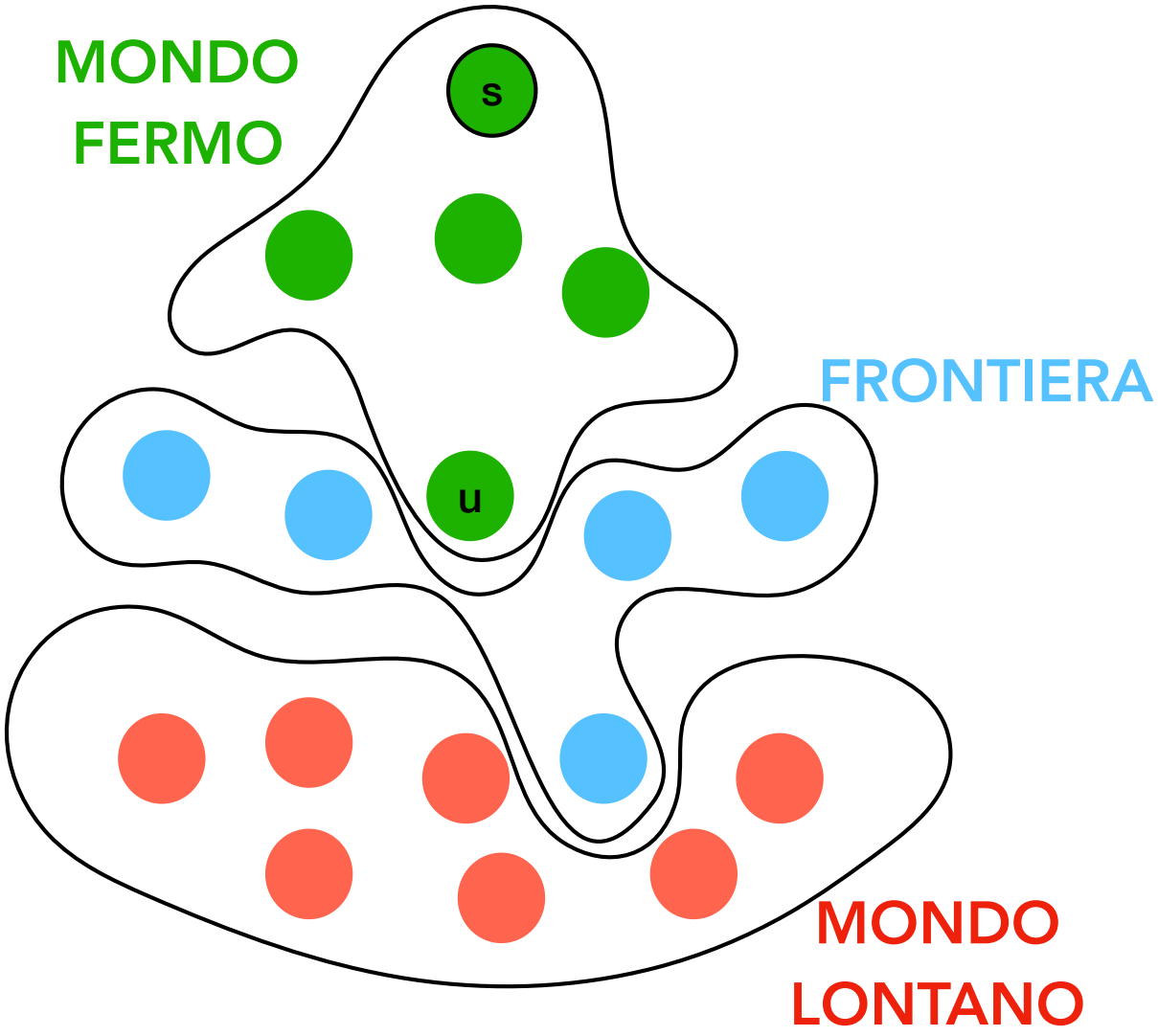


ITERAZIONE

- Rilassiamo gli archi verso i vicini ed eventualmente spostiamo i vicini dal MONDO LONTANO alla FRONTIERA

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



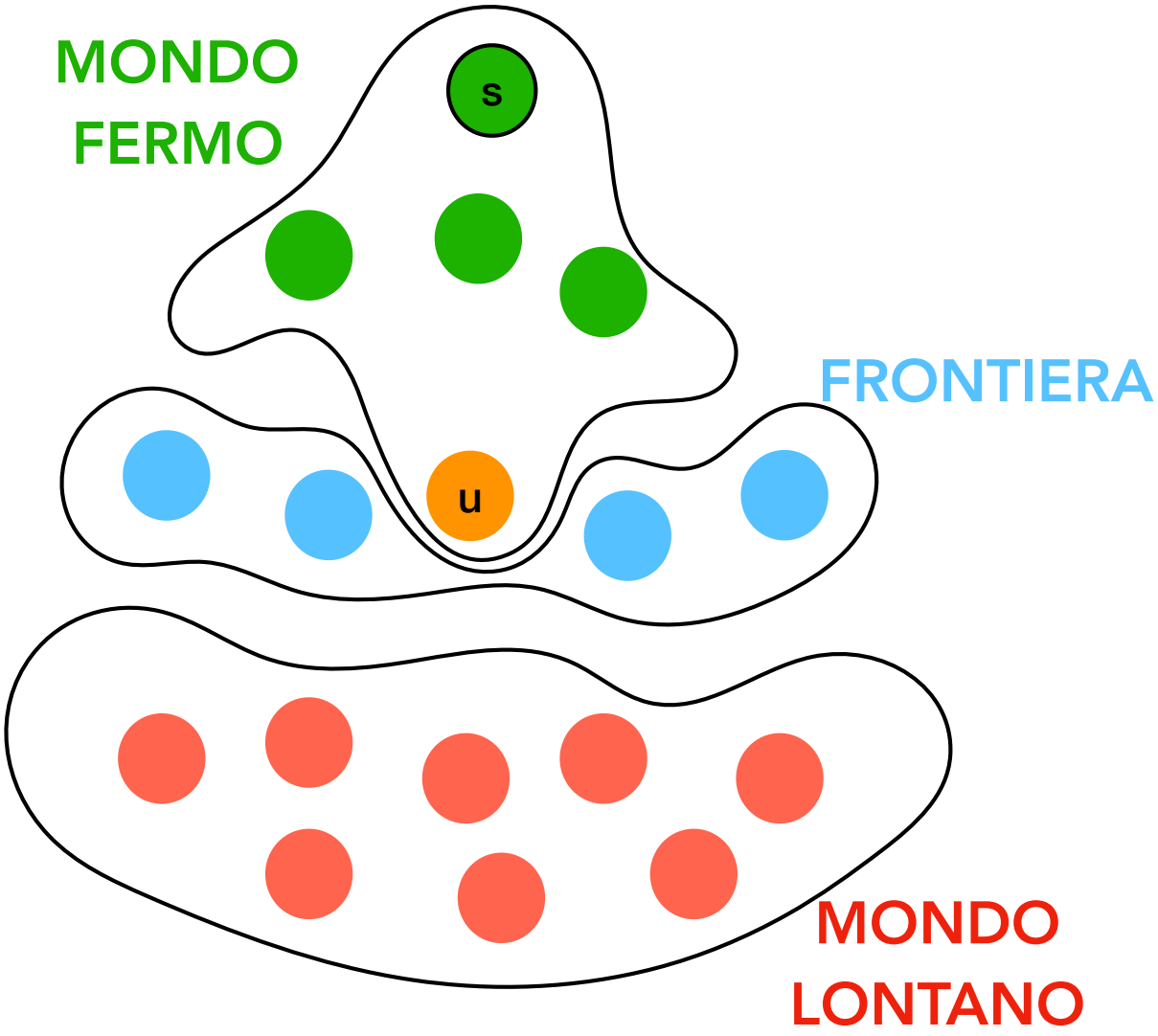
ITERAZIONE

- Il nodo scelto si sposta nel MONDO FERMO

E' corretto?

Cammini minimi da sorgente singola

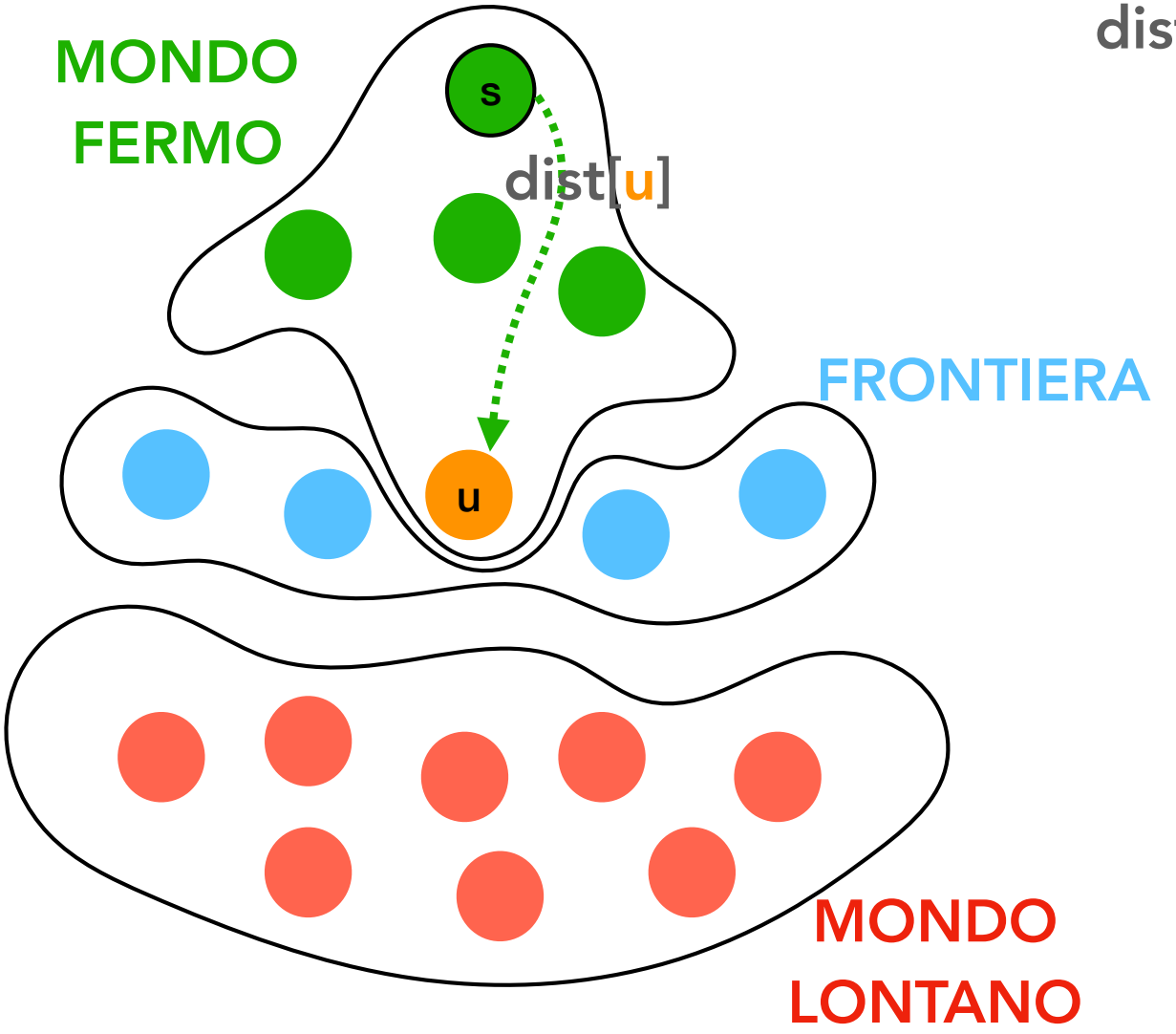
ALGORITMO di DIJKSTRA



Quando un
nodo **u** esce dalla
FRONTIERA e
entra nel MONDO FERMO
dist[u] registra correttamente
la sua distanza dalla sorgente?

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



dist[u] è minima tra i nodi della FRONTIERA

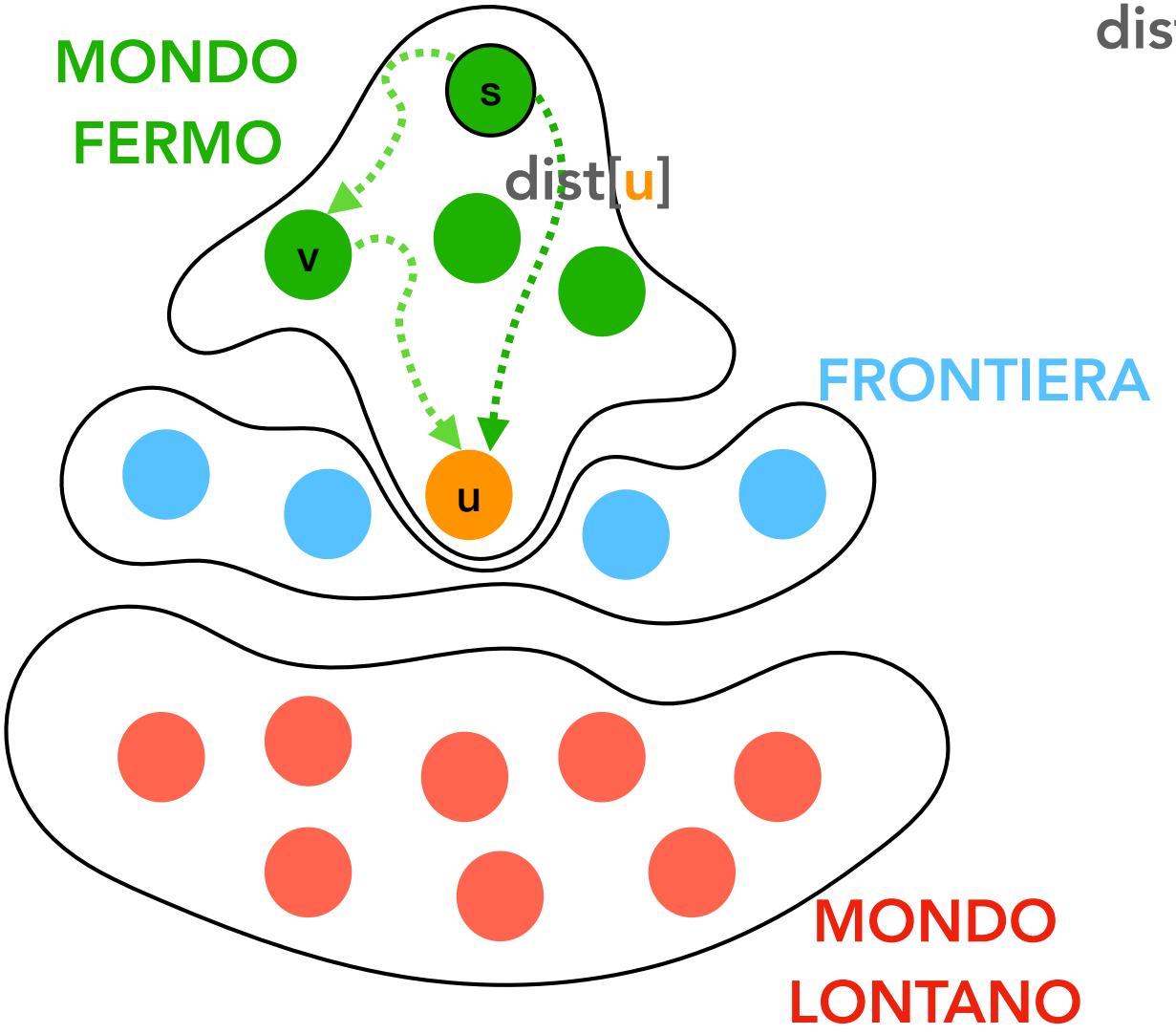
È possibile trovare
un altro cammino da **s** a **u**
più corto di **dist[u]**?

NO,
algoritmo
giusto

SI,
algoritmo
sbagliato

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



dist[u] è minima tra i nodi della FRONTIERA

È possibile trovare
un altro cammino da **s** a **u**
più corto di **dist[u]**…

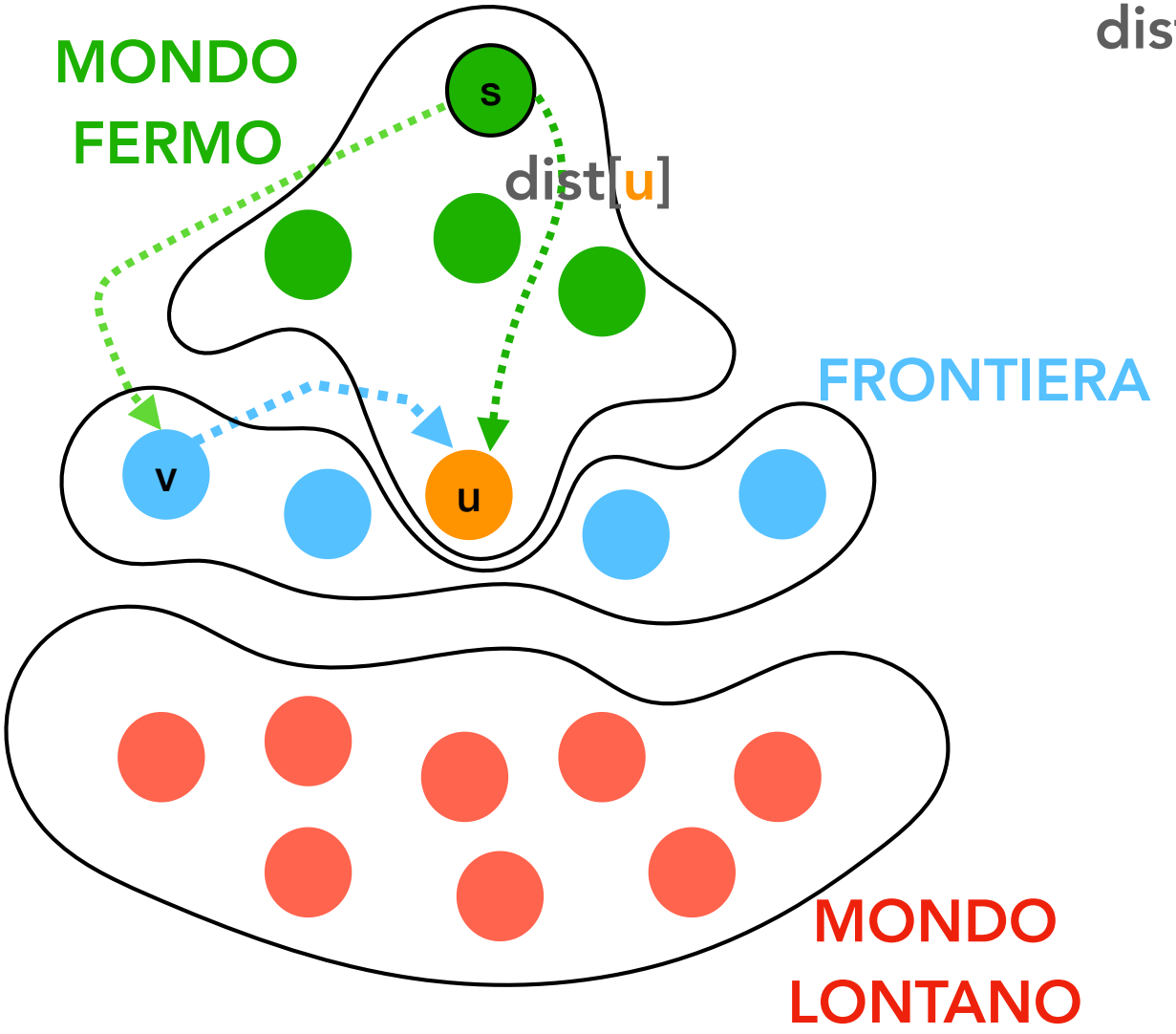
..che arriva dal
MONDO FERMO?

NO

Lo avremmo già trovato

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



dist[u] è minima tra i nodi della FRONTIERA

È possibile trovare un altro cammino da **s** a **u** più corto di **dist[u]**…

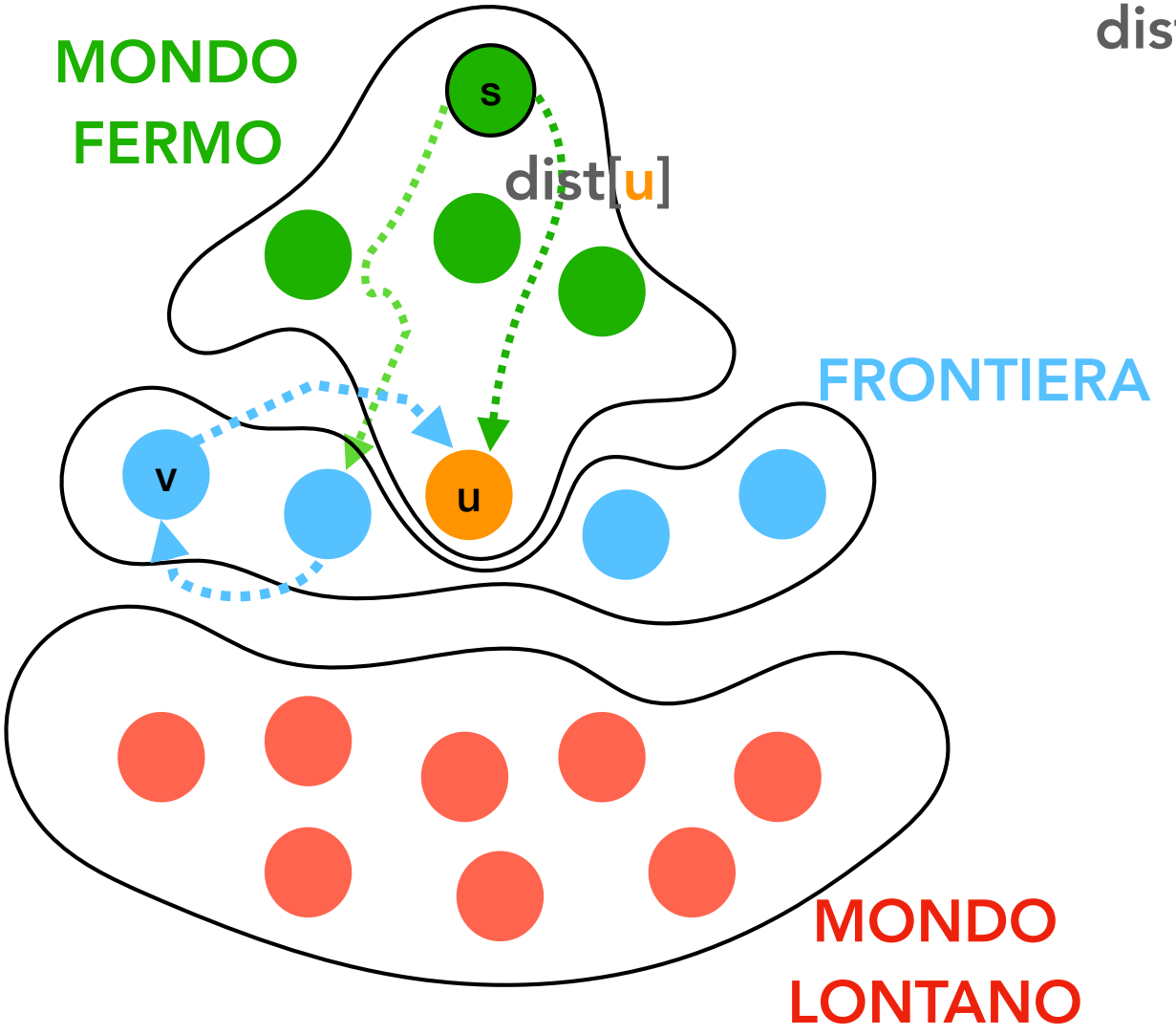
…che arriva dalla FRONTIERA?

NO

dist[**v**] + c(**v**,**u**) ≥ dist[**u**]

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



dist[u] è minima tra i nodi della FRONTIERA

È possibile trovare
un altro cammino da **s** a **u**
più corto di **dist[u]**…

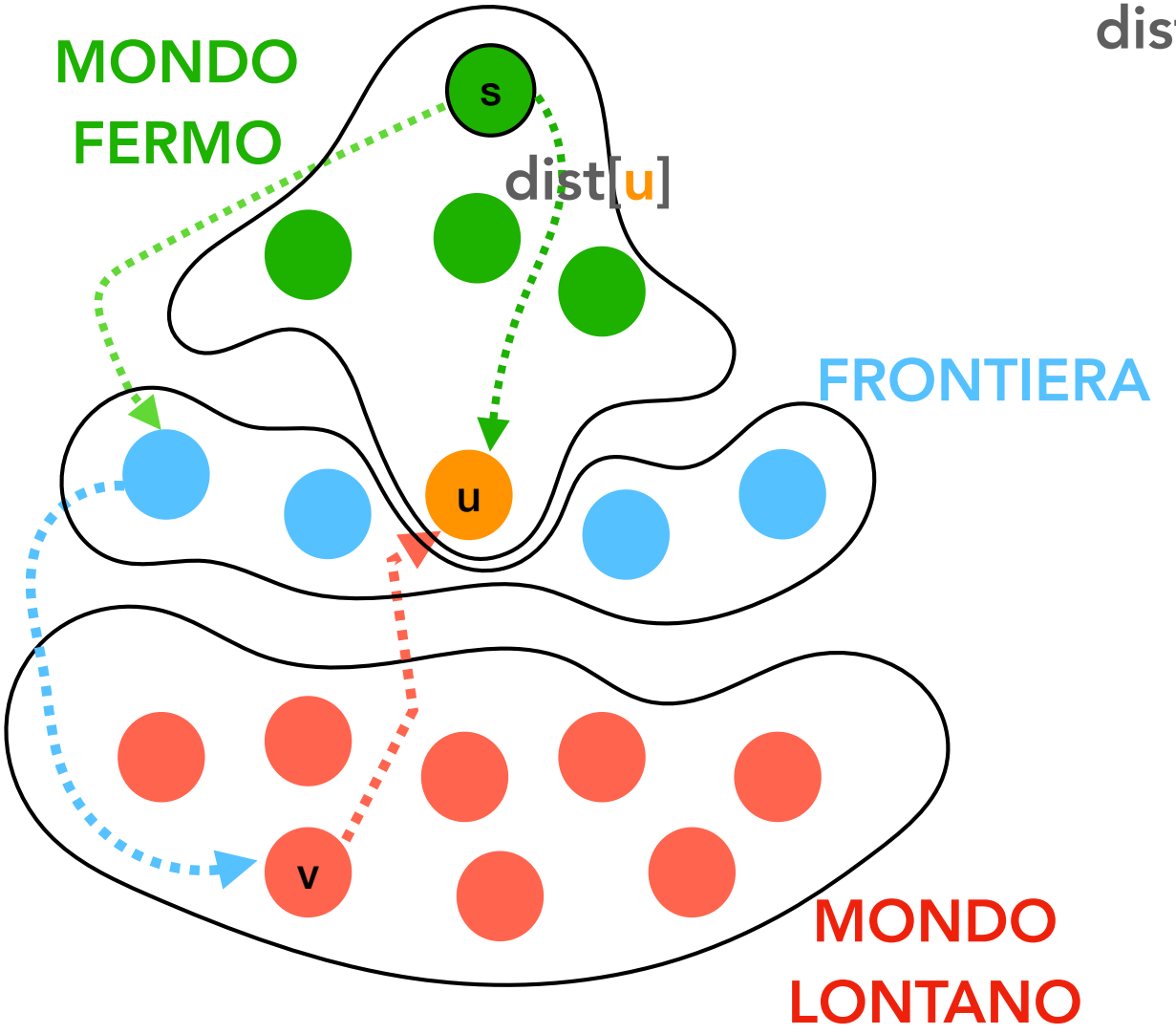
…che arriva dalla FRONTIERA?

NO

dist[**v**] + c(**v**,**u**) ≥ dist[**u**]

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



dist[u] è minima tra i nodi della FRONTIERA

È possibile trovare
un altro cammino da **s** a **u**
più corto di **dist[u]**…

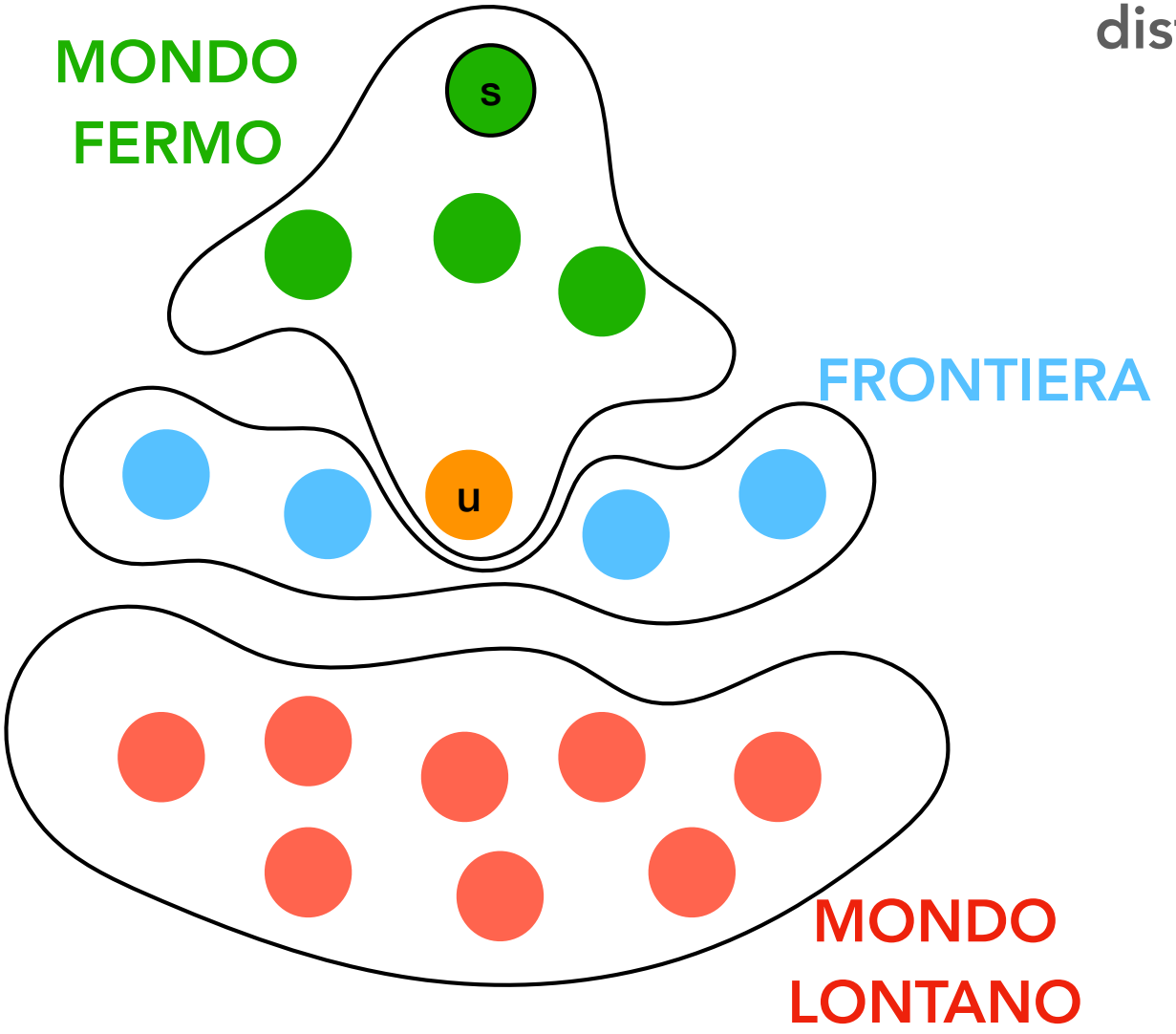
…che arriva dal
MONDO LONTANO?

NO

dist[**v**] + c(**v**,**u**) ≥ dist[**u**]

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA

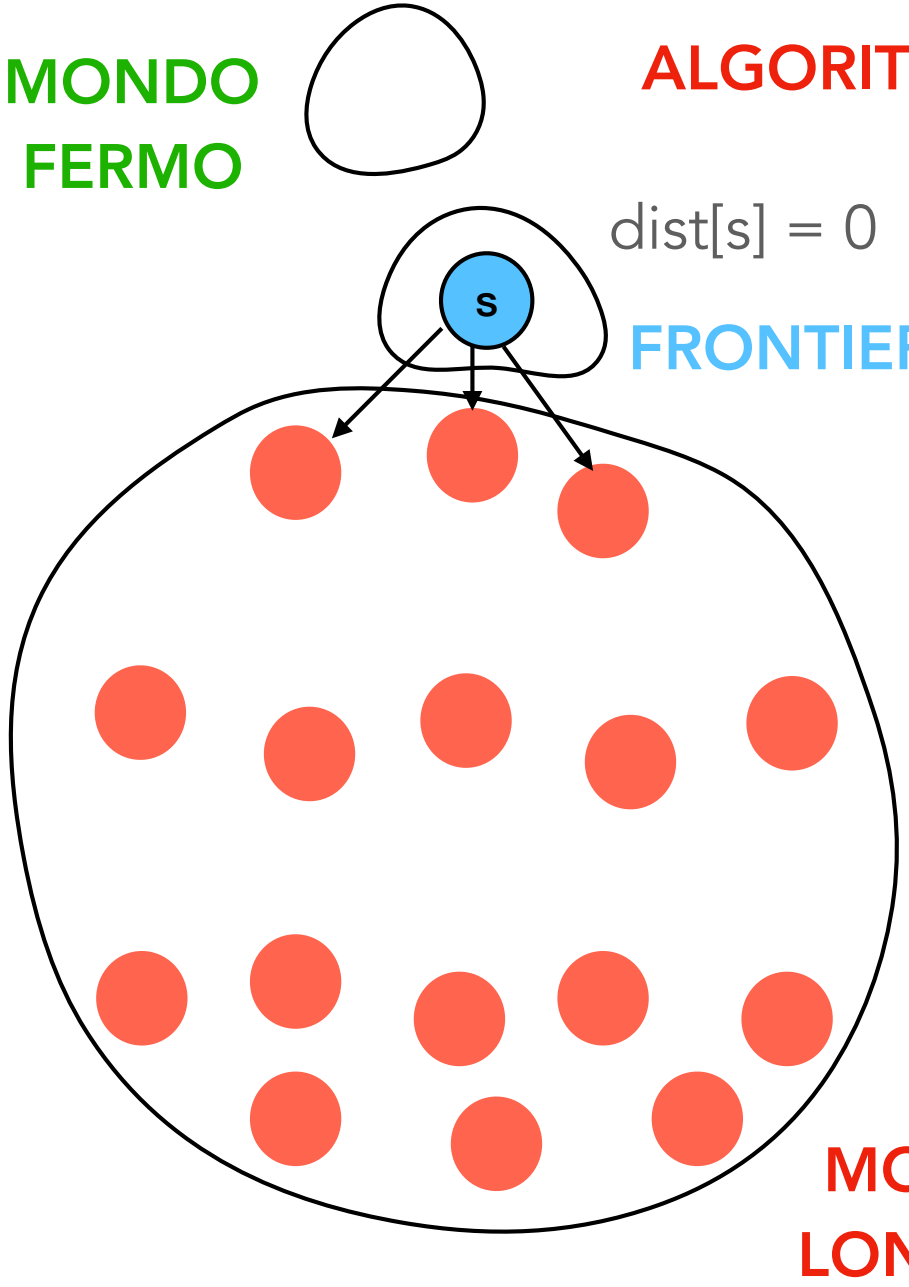


dist[u] è minima tra i nodi della FRONTIERA

È possibile trovare un altro cammino da **s** a **u** più corto di **dist[u]**…

NO, algoritmo giusto

Cammini minimi da sorgente singola

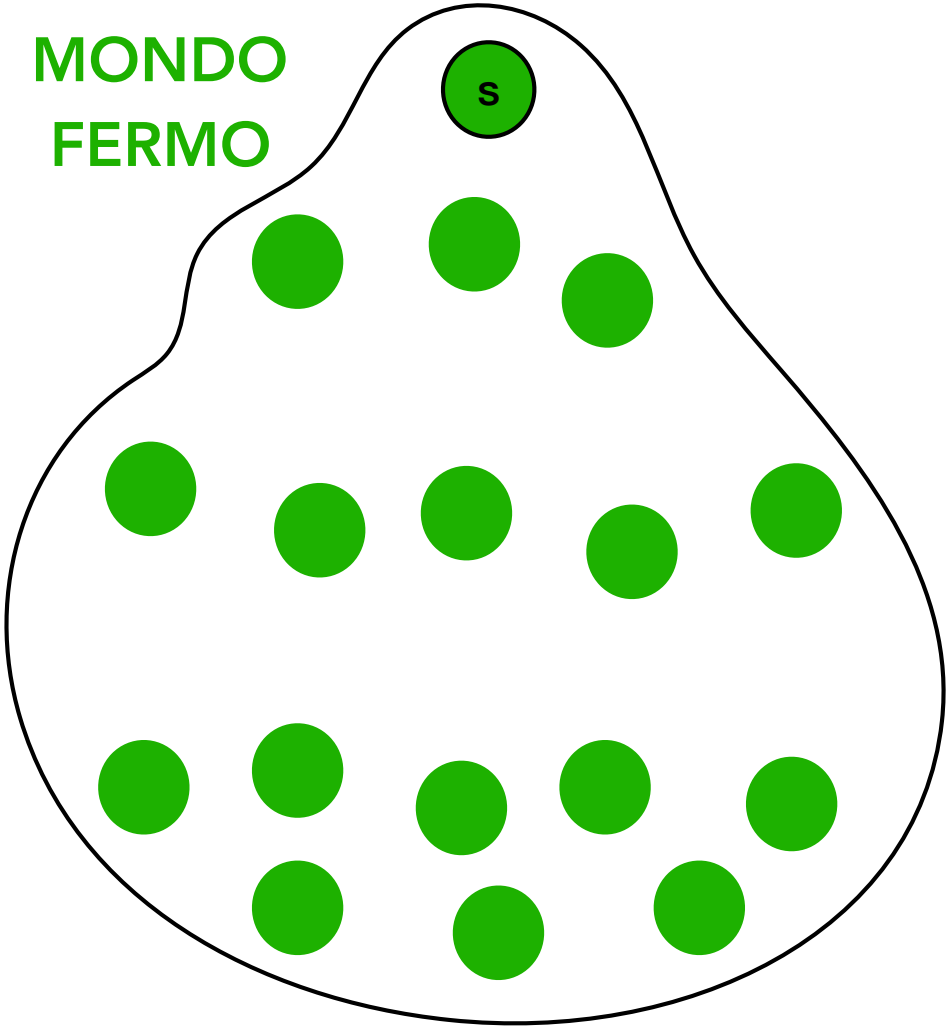


COME INIZIAMO?

MONDO FERMO: vuoto
FRONTIERA: contiene solo s
MONDO LONTANO:
contiene tutti gli altri nodi

Cammini minimi da sorgente singola

ALGORITMO di DIJKSTRA



QUANDO ci FERMIAMO?

Quando tutti i nodi
raggiungibili da s
sono nel MONDO FERMO