ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

BFS su grafi

UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Visita di grafi

Strategia per **analizzare i nodi del grafo** una volta sola

VISITA IN PROFONDITÀ
(**Depth First Search - DFS**)

- Estensione della pre/post-order DFS sugli alberi:
  dopo aver visitato un nodo si continua la visita cercando "di allontanarsi" sempre più.
- Strategia utilizzata per visitare tutti i nodi del grafo, anche se ci sono più componenti

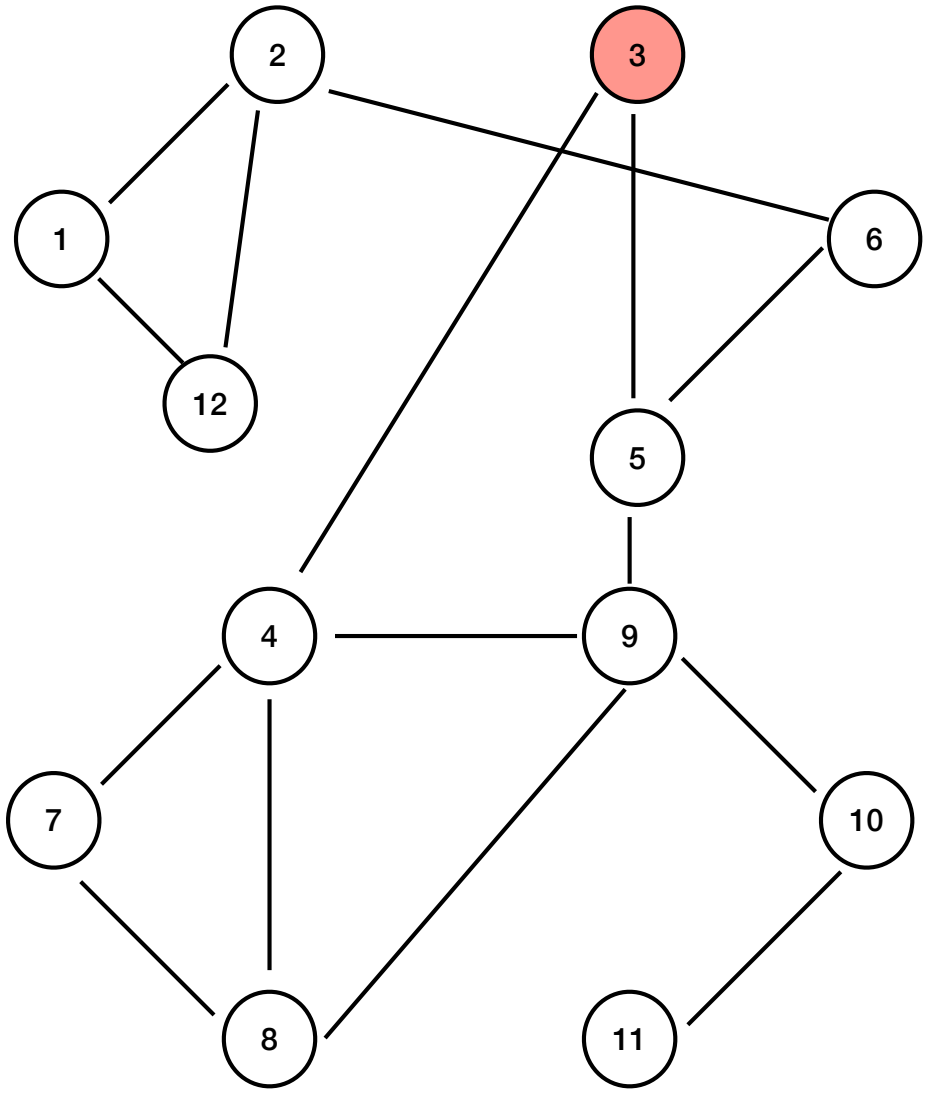
VISITA IN AMPIEZZA
(**Breadth First Search - BFS**)

- Estensione della BFS sugli alberi:
  partendo da un nodo, si visitano tutti i suoi vicini, poi i vicini dei vicini, e cosi' via.
- **Strategia utilizzata per visitare tutti i nodi raggiungibili da un nodo sorgente**

BFS su grafi

3 sorgente della BFS

- partendo da un nodo, si visitano tutti i suoi vicini, poi i vicini dei vicini, e cosi' via.



INDIRETTO

BFS su grafi

3 sorgente della BFS

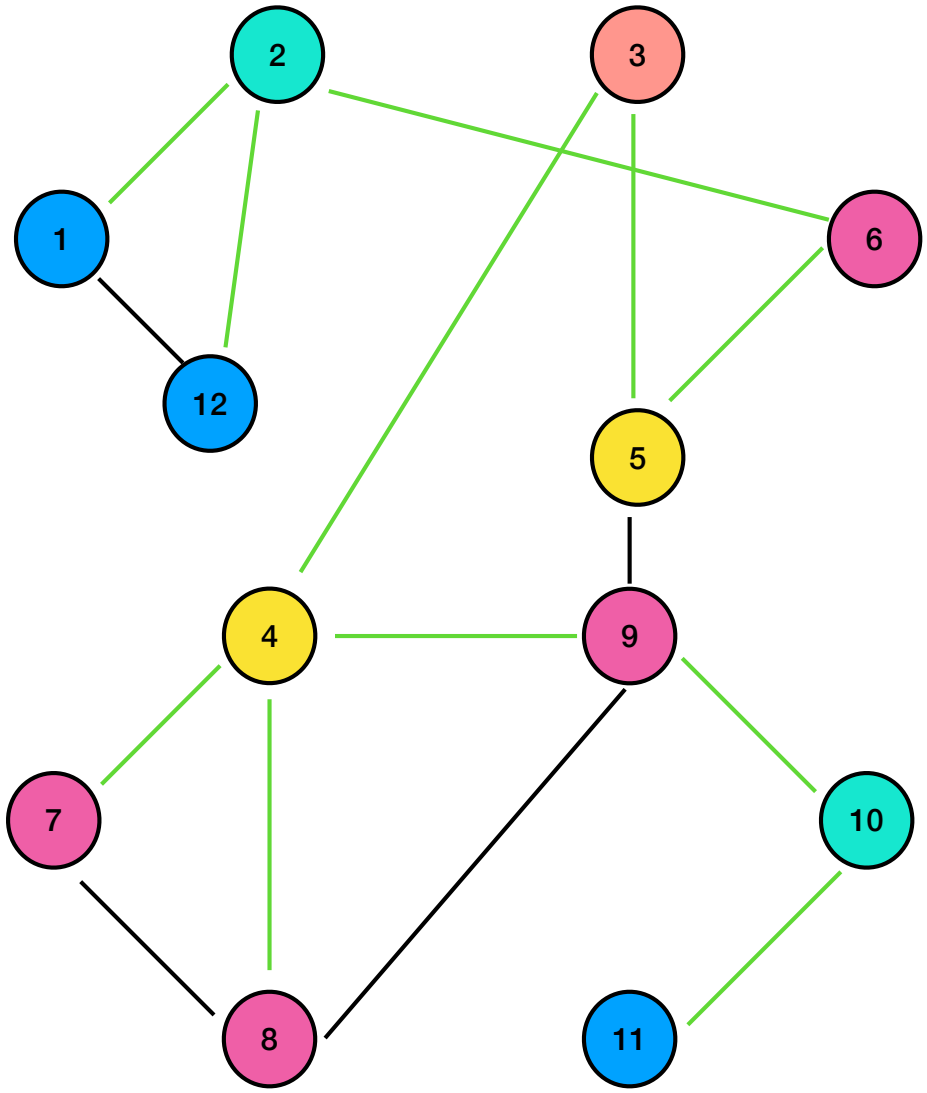
vicini della sorgente

vicini dei vicini della sorgente
(non ancora visitati)

vicini dei vicini dei vicini della sorgente
(non ancora visitati)

vicini dei vicini dei vicini dei vicini della sorgente
(non ancora visitati)

archi albero di copertura



INDIRETTO

BFS su grafi

3 sorgente della BFS

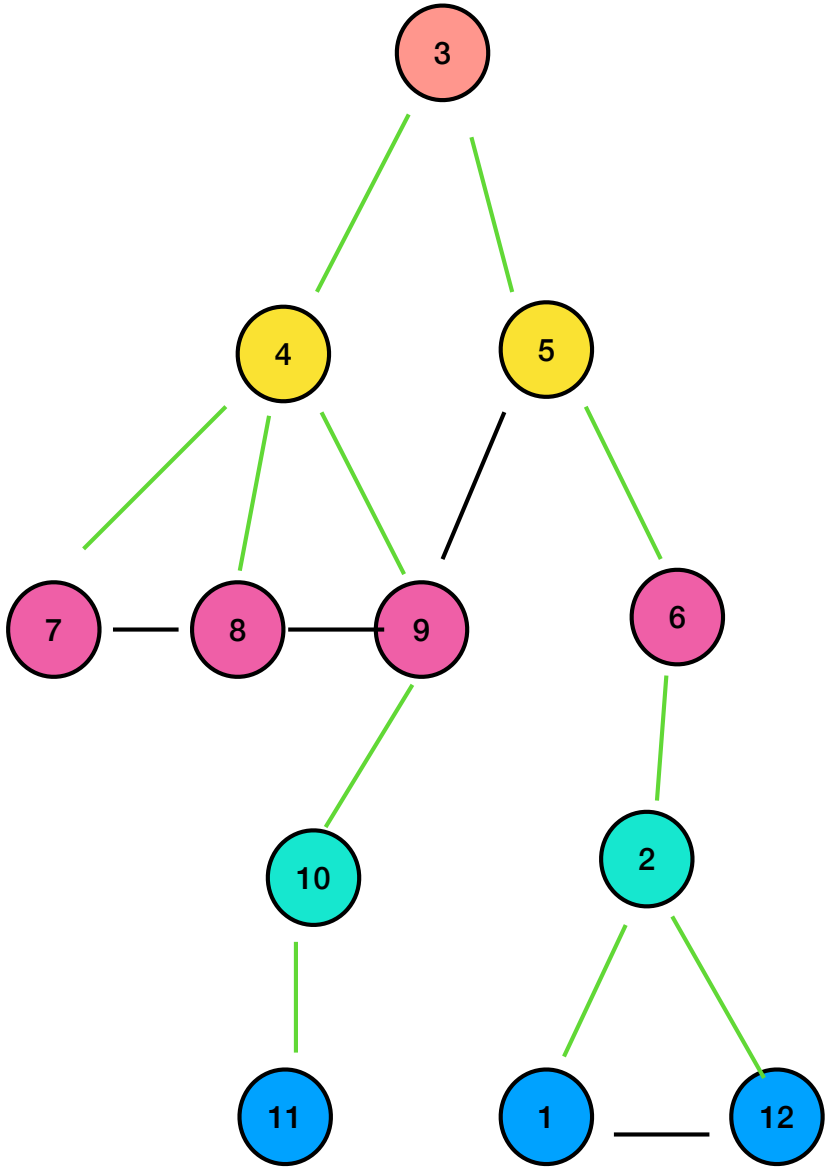
vicini della sorgente

vicini dei vicini della sorgente
(non ancora visitati)

vicini dei vicini dei vicini della sorgente
(non ancora visitati)

vicini dei vicini dei vicini dei vicini della sorgente
(non ancora visitati)

archi albero di copertura



INDIRETTO

BFS su grafi

3 sorgente della BFS

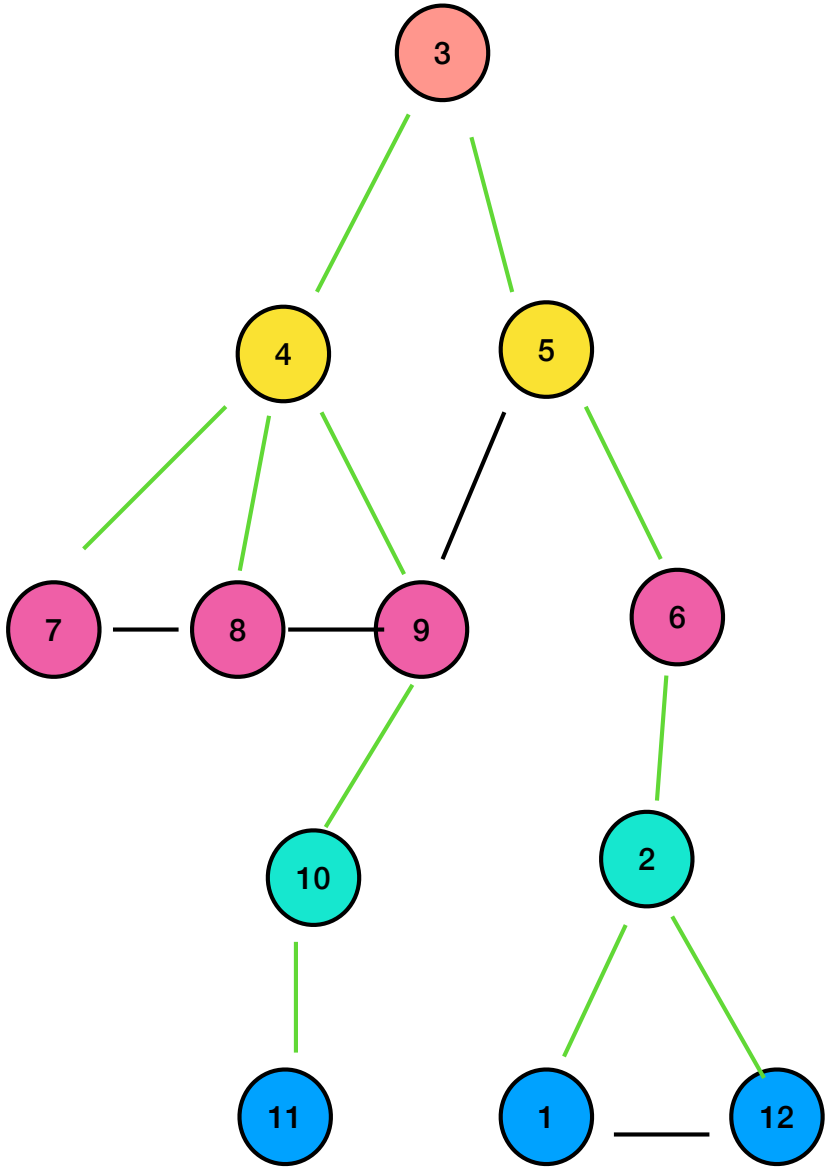
distanza 1 della sorgente

distanza 2 della sorgente

distanza 3 della sorgente

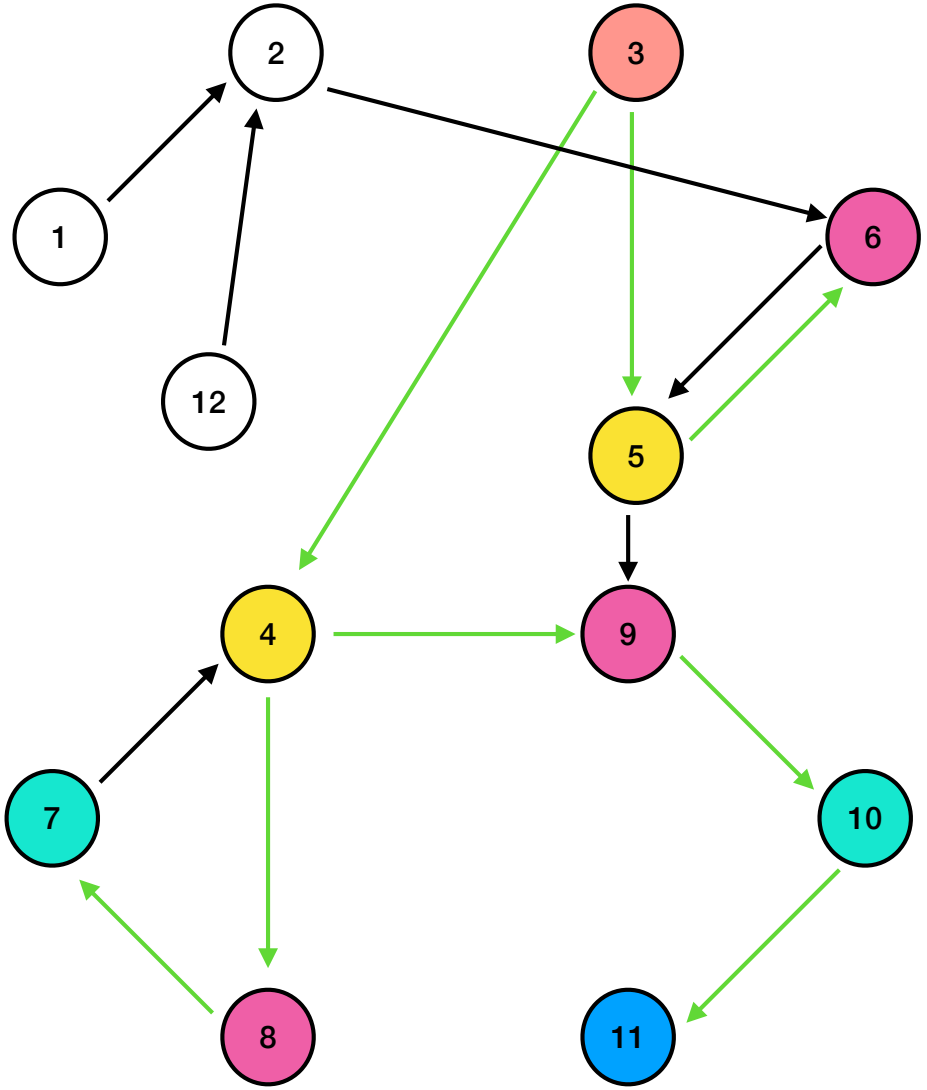
distanza 4 della sorgente

archi albero di copertura



INDIRETTO

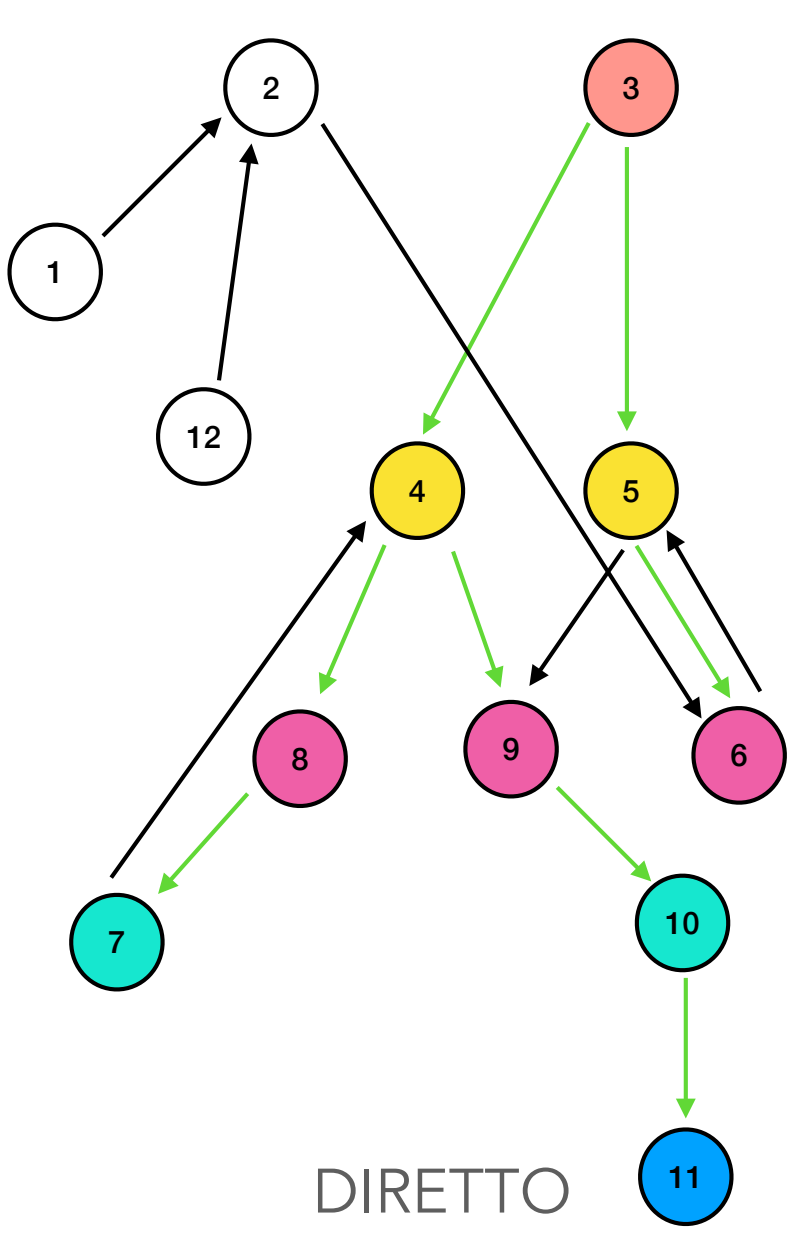
BFS su grafi



3 sorgente della BFS

DIRETTO

BFS su grafi



DIRETTO

3 sorgente della BFS

distanza 1 della sorgente

distanza 2 della sorgente

distanza 3 della sorgente

distanza 4 della sorgente

NON RAGGIUNGIBILE della sorgente

BFS su grafi

G = (V,E) indiretto/diretto

```
BFS(G,s)
 for all u ∈ V
  visited[u] := FALSE
 visited[s] := TRUE
 Q := new_queue() // coda FIFO
 enqueue(Q,s)
 while NOT is_empty_queue(Q)do
  u:= dequeue(Q)
  // esame di u
  for all (u,v)∈E do
   if visited[v] = FALSE
    then
     enqueue(Q,v)
     visited[v] := TRUE
```

Possiamo modificare il codice per calcolare le distanze dalla radice e l'albero di copertura?

`dist[u]` memorizza la distanza da `s` a `u`

Distanza tra `u` e `v`: numero di archi sul cammino più breve in `G` tra `u` e `v`

`prev[u]` memorizza il predecessore (padre) di `u` nell'albero di copertura dalla visita BFS con sorgente `s`

BFS su grafi

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

```
BFS(G,s)
 for all u ∈ V
  dist[u] := +∞
  prev[u] := 0
 dist[s] := 0
 Q := new_queue() // coda FIFO
 enqueue(Q,s)
 while NOT is_empty_queue(Q)do
  u:= dequeue(Q)
  eventuale esame di u
  for all (u,v)∈E do
   if dist[v] = +∞
    then
     enqueue(Q,v)
     dist[v] := dist[u] + 1
     prev[v] := u
```

`dist[u]` memorizza la distanza da `s` a `u`

Distanza tra `u` e `v`:
numero di archi sul cammino più breve in `G` tra `u` e `v`

`prev[u]` memorizza il predecessore (padre) di `u` nell'albero di copertura dalla visita BFS con sorgente `s`

`dist[]` viene usato anche per controllare se un nodo e' già stato visitato

N.B. nella pratica $+\infty$ può essere sostituito con un qualunque valore $> n$

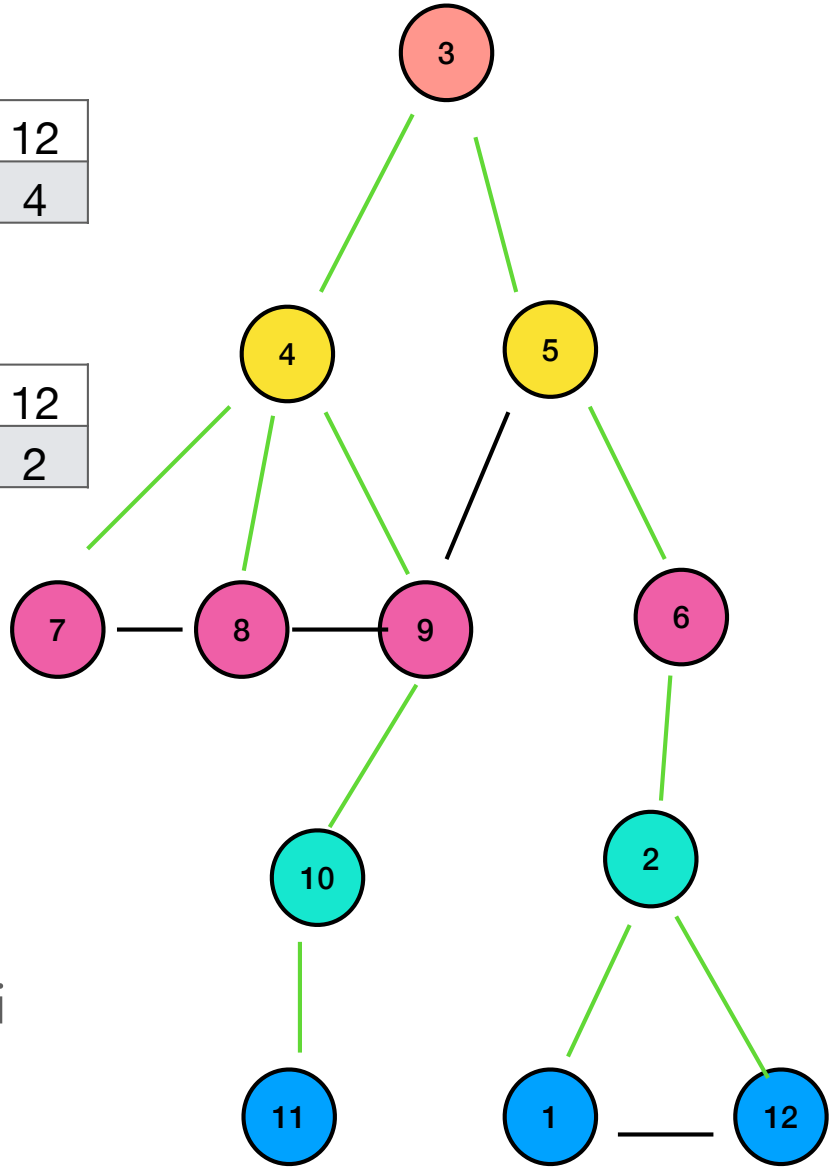
BFS su grafi

`dist`

1	2	3	4	5	6	7	8	9	10	11	12
4	3	0	1	1	2	2	2	2	3	4	4

`prev`

1	2	3	4	5	6	7	8	9	10	11	12
2	6	0	3	3	5	4	4	4	9	10	2

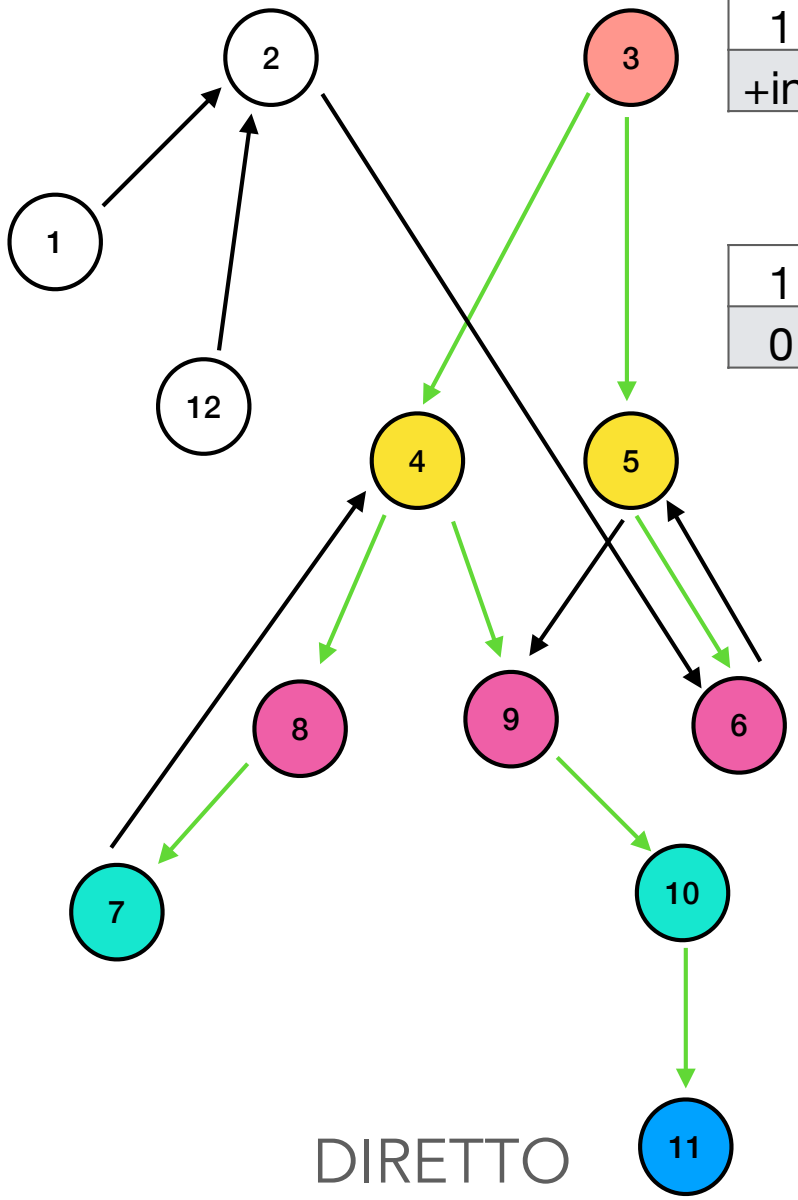


Gli archi che non fanno parte dell'albero di copertura possono essere tra:

- nodi dell'albero allo stesso livello (es. (7,8))
- nodi dell'albero che si trovano su due livelli consecutivi (es. (9,5))

INDIRETTO

BFS su grafi



DIRETTO

dist

1	2	3	4	5	6	7	8	9	10	11	12
+inf	+inf	0	1	1	2	3	2	2	3	4	+inf

prev

1	2	3	4	5	6	7	8	9	10	11	12
0	0	0	3	3	5	8	4	5	9	10	NIL

Gli archi che non fanno parte dell'albero di copertura possono essere:

- da un nodo ad un altro che si trova allo stesso livello dell'albero
- da un nodo ad un altro che si trova al livello successivo dell'albero (es. (5,9))
- da un nodo ad un suo antenato nell'albero (es. (7,4))
- da un nodo non nell'albero ad un nodo nell'albero (es. (2,6))

BFS su grafi

G = (V,E) indiretto/diretto

```
BFS(G,s)
 for all u ∈ V
  dist[u] := +∞
  prev[u] := 0
 dist[s] := 0
 Q := new_queue() // coda FIFO
 enqueue(Q,s)
 while NOT is_empty_queue(Q)do
  u:= dequeue(Q)
  eventuale esame di u
  for all (u,v)∈E do
   if dist[v] = +∞
    then
     enqueue(Q,v)
     dist[v] := dist[u] + 1
     prev[v] := u
```

dist[ ] viene usato anche per controllare se un nodo e' già stato visitato

Costo computazionale?

- Ogni nodo (raggiungibile da s) viene inserito in coda esattamente una volta

  ⇒ $O(|V|)$ enqueue

  +

- Ogni nodo in coda viene estratto dalla coda esattamente una volta

  ⇒ $O(|V|)$ dequeue

  +

- Ogni arco e' considerato esattamente:
  - 1 volta se il grafo e' diretto
  - 2 volte se il grafo e' indiretto

  per $\Theta(1)$ operazioni

  ⇒ $O(|E|)$

  =

enqueue/dequeue $\in \Theta(1)$ ⇒ $O(|V|+|E|)$

BFS su grafi

`dist[u]` memorizza
la distanza da `s` a `u`

Distanza tra `u` e `v`:
numero di archi sul cammino
più breve in `G` tra `u` e `v`

L'albero BFS dipende dall'ordine in cui
sono visitati gli archi incidenti,
LE DISTANZE NO

BFS su grafi

`dist[u]` memorizza
la distanza da `s` a `u`

Distanza tra `u` e `v`:
numero di archi sul cammino
più breve in `G` tra `u` e `v`

Perché la BFS calcola correttamente le distanze
tra il nodo sorgente s e i nodi raggiungibili da s?

Dimostrazione formale in
video pillola + slide che seguono
(facoltativa)

BFS su grafi

All'inizio di OGNI ITERAZIONE esiste un **d** < n tale che:

I nodi del grafo possono essere divisi in tre insiemi:

- **MONDO FERMO**: nodi gia' visitati, vicini esplorati, **dist[.] ≤ d corretto**
- **FRONTIERA**: nodi già scoperti, vicini NON esplorati, **d ≤ dist[.] ≤ d+1 corretto**
- **MONDO LONTANO**: nodi non visitati, **dist[.] = +∞**

La coda contiene i nodi della **FRONTIERA:**
nodi con distanza d dalla sorgente, seguiti, eventualmente,
da nodi a distanza d+1 dalla sorgente

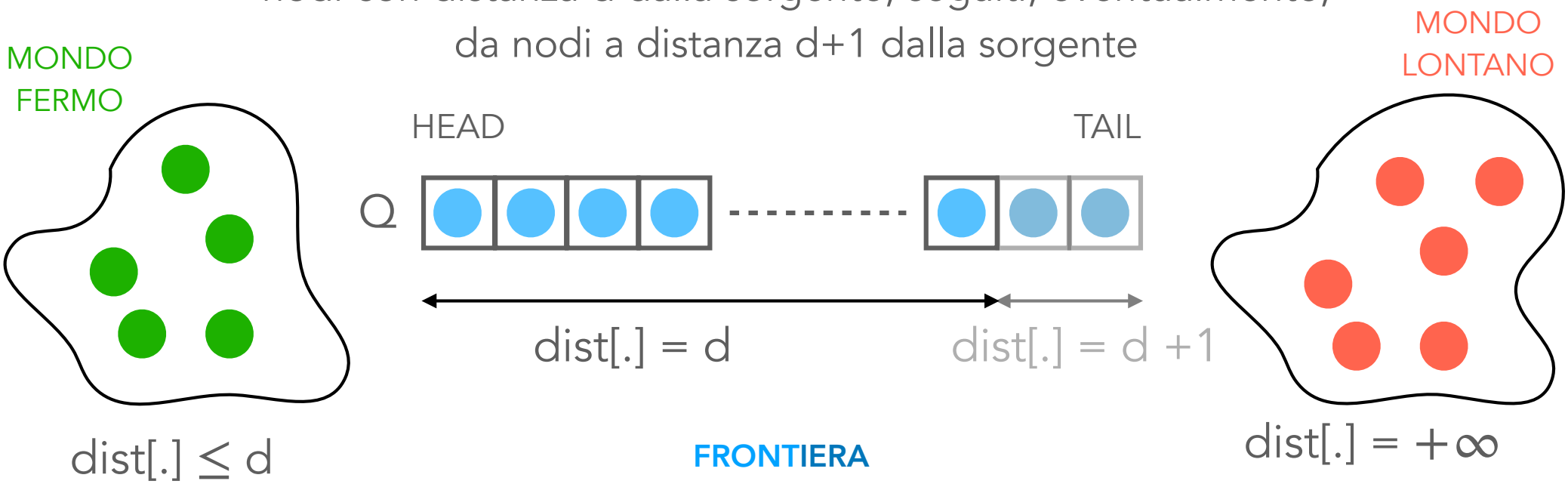
BFS su grafi

All'inizio di OGNI ITERAZIONE esiste un **d** < n tale che:

I nodi del grafo possono essere divisi in tre insiemi:

- **MONDO FERMO**: nodi gia' visitati, vicini esplorati, **dist[.] ≤ d corretto**
- **FRONTIERA**: nodi già scoperti, vicini NON esplorati, **d ≤ dist[.] ≤ d+1 corretto**
- **MONDO LONTANO**: nodi non visitati, **dist[.] = +∞**

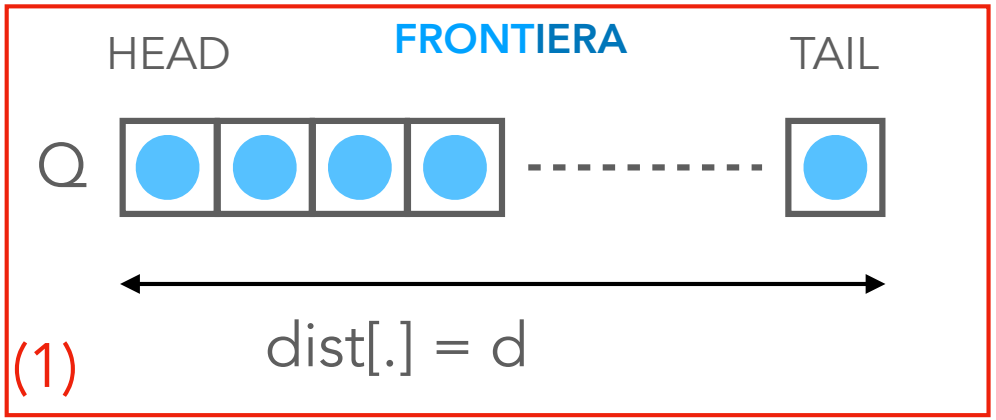
La coda contiene i nodi della **FRONTIERA:**
nodi con distanza d dalla sorgente, seguiti, eventualmente,
da nodi a distanza d+1 dalla sorgente



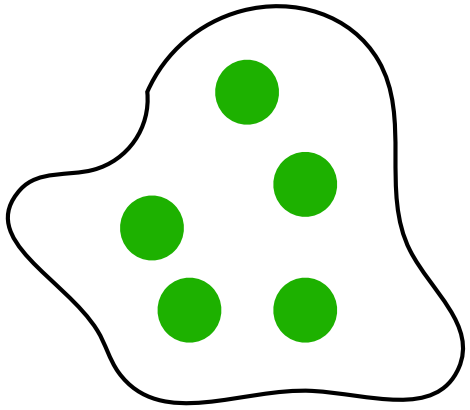
BFS su grafi

All'inizio di un'iterazione generica, per d < n:

caso (1)

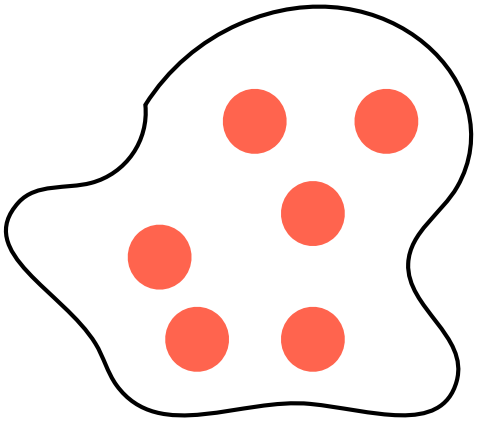


MONDO FERMO



dist[.] ≤ d

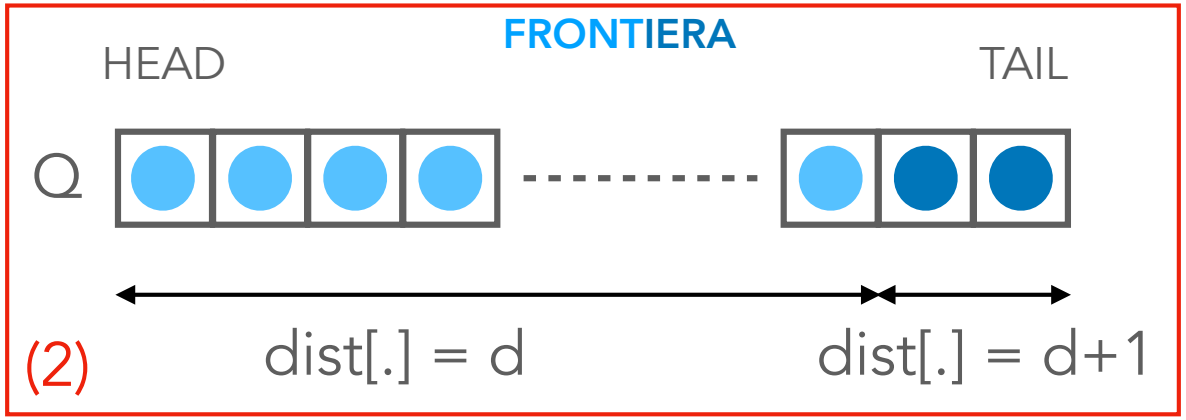
MONDO LONTANO



dist[.] = +∞

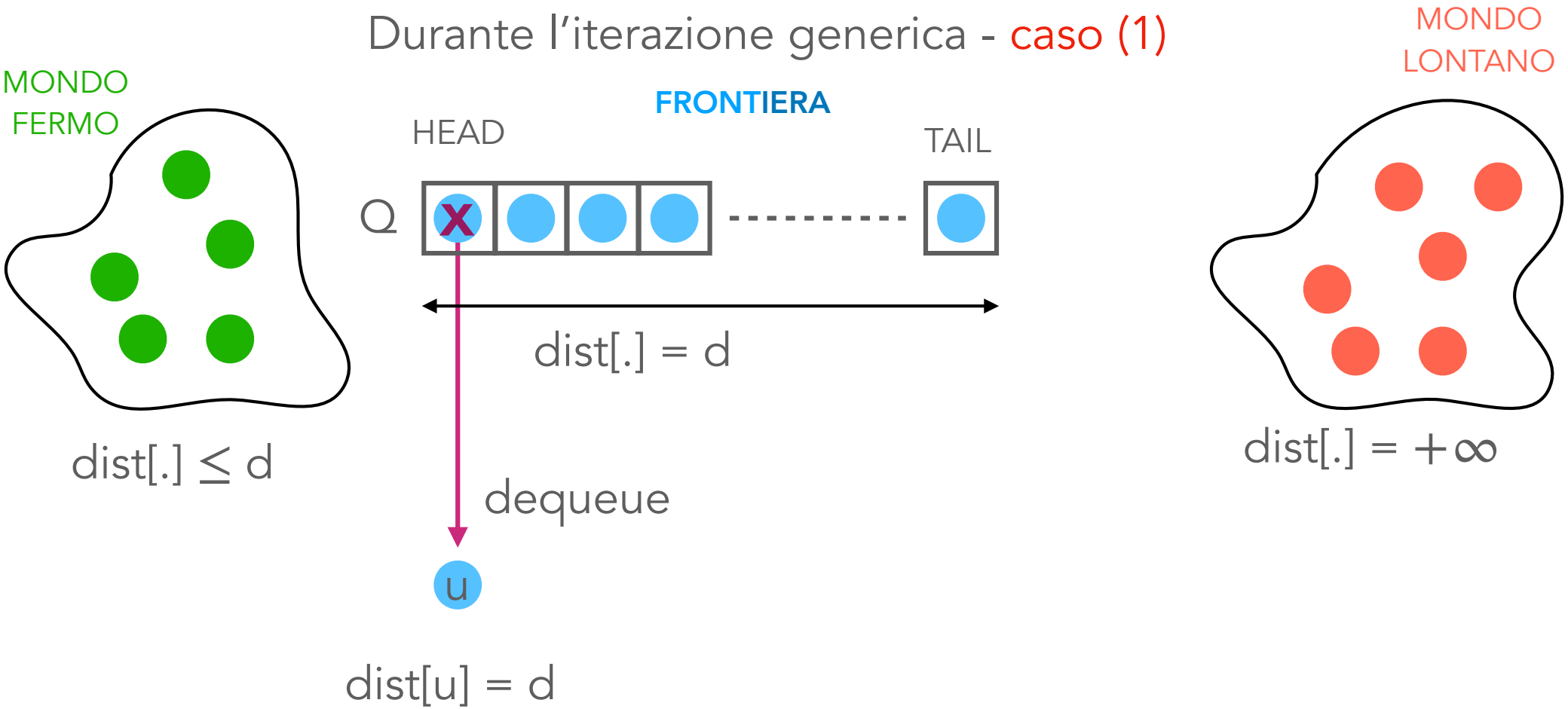
OPPURE

caso (2)



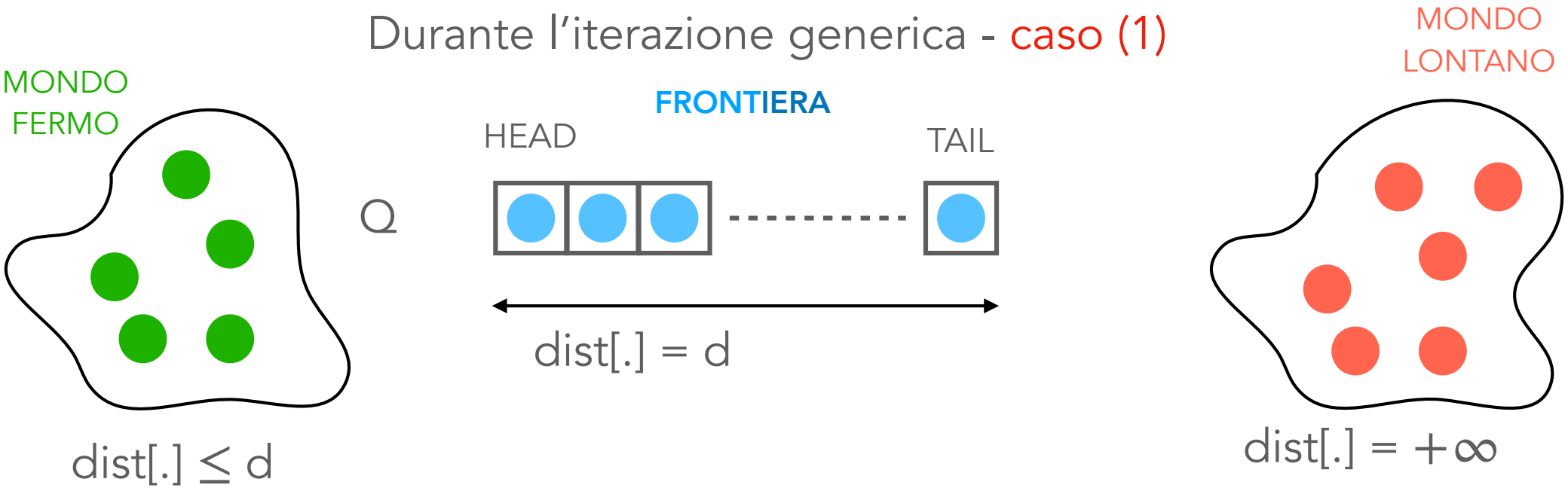
BFS su grafi

Durante l'iterazione generica - caso (1)



BFS su grafi

Durante l'iterazione generica - caso (1)

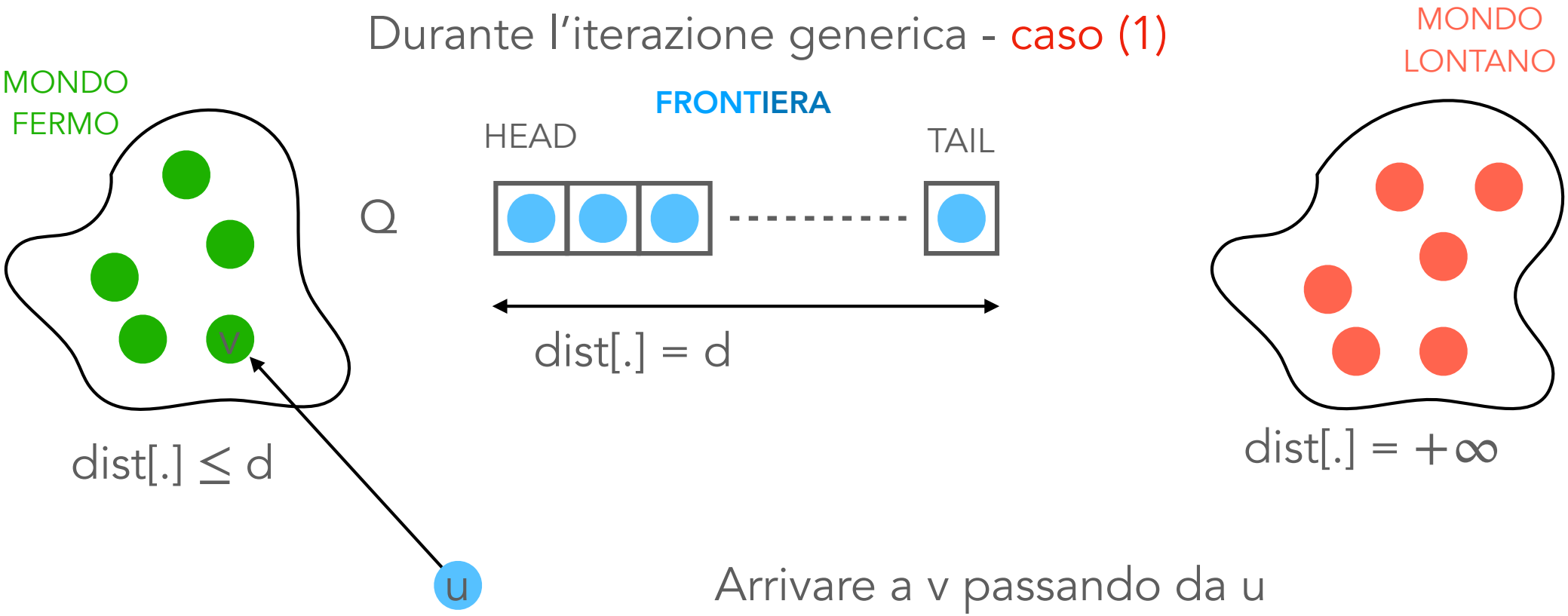


u esplorazione degli archi incidenti

dist[u] = d

BFS su grafi

Durante l'iterazione generica - caso (1)

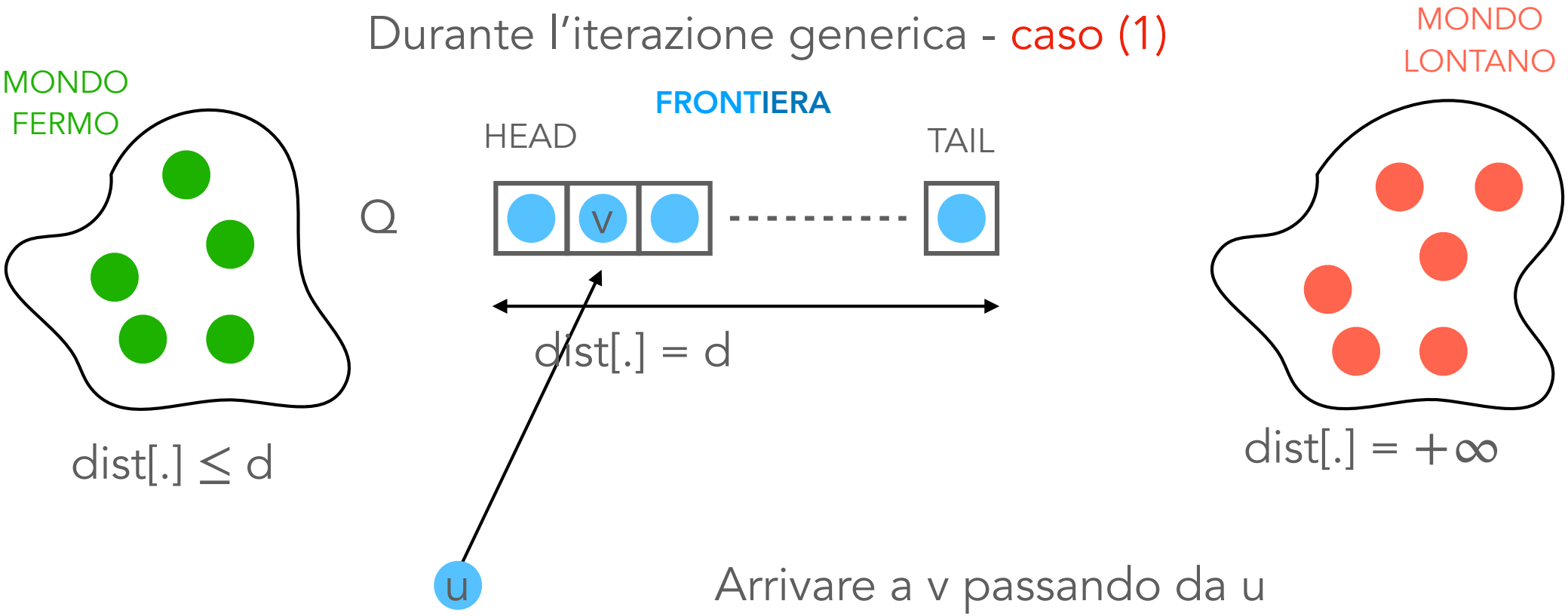


Arrivare a v passando da u
ALLUNGA il cammino già trovato
da s a v

I vicini di v sono già stati esplorati,
v NON entra nella frontiera

BFS su grafi

Durante l'iterazione generica - caso (1)

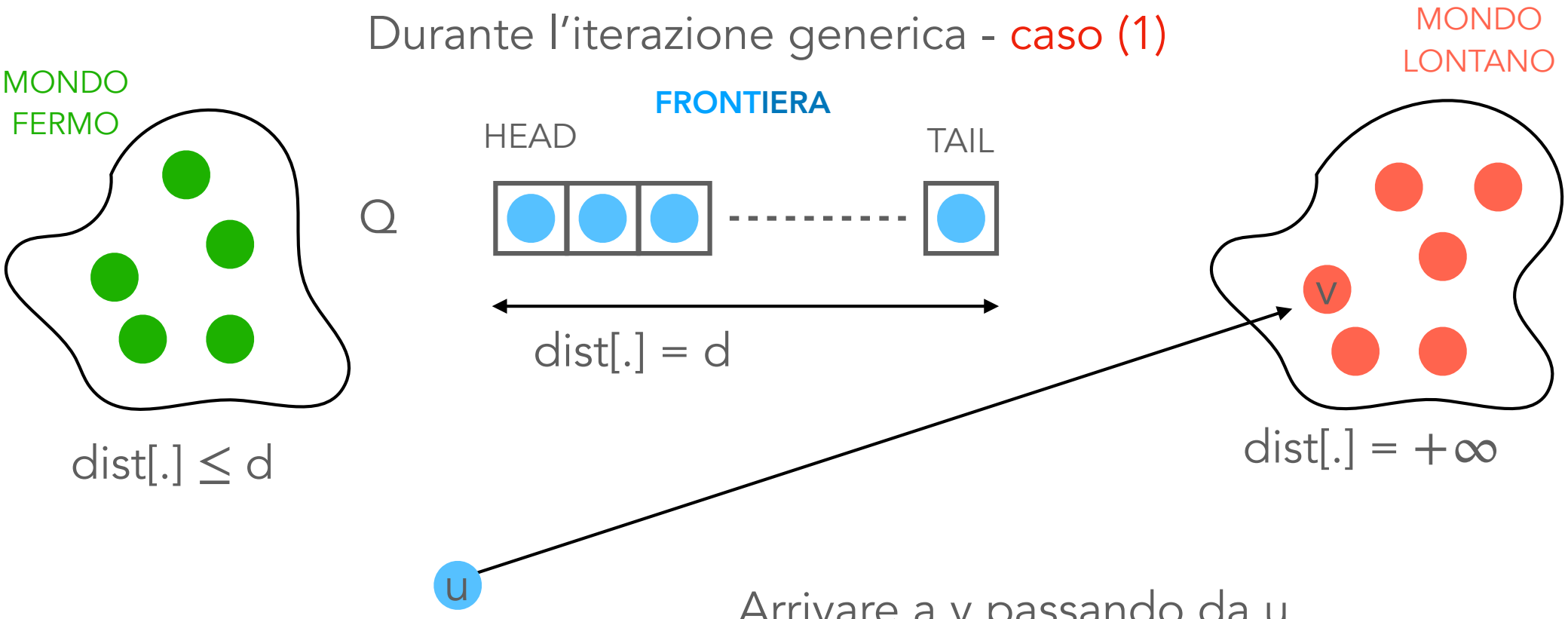


Arrivare a v passando da u
ALLUNGA il cammino già trovato
da s a v

I vicini di v NON sono già stati esplorati,
ma v e' già nella frontiera

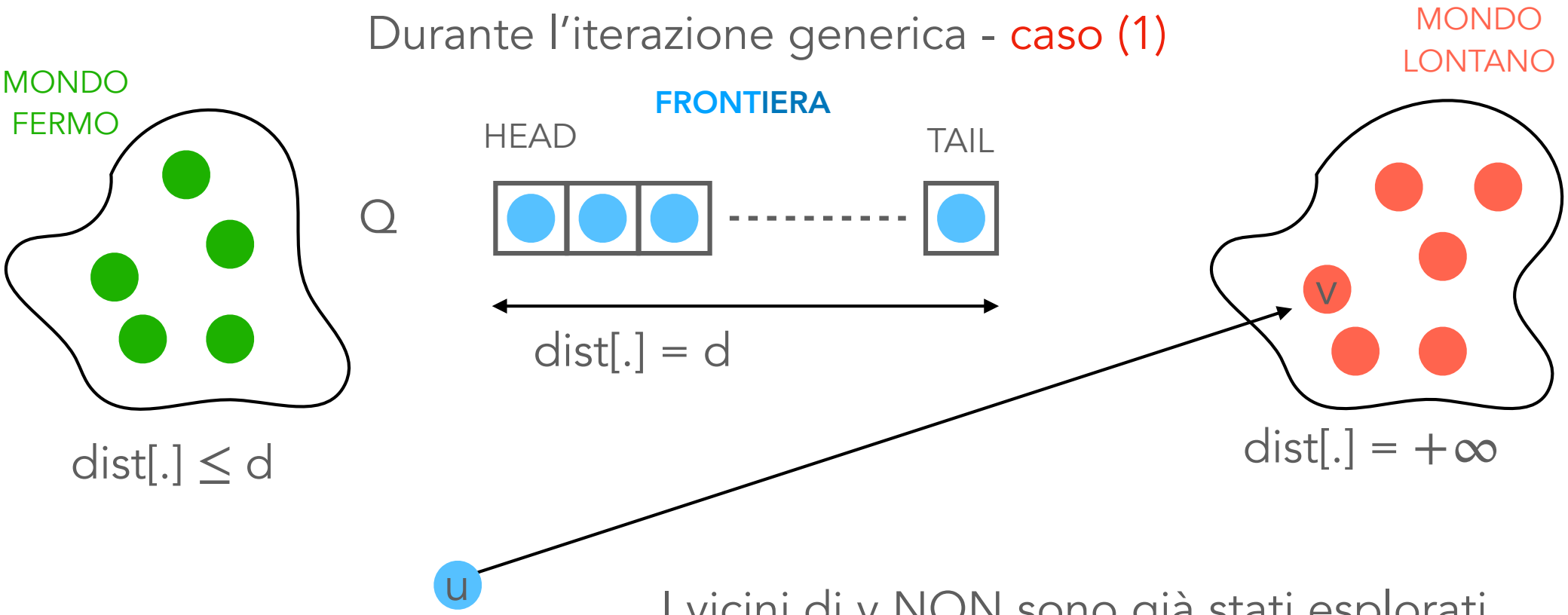
BFS su grafi

Durante l'iterazione generica - caso (1)



BFS su grafi

Durante l'iterazione generica - caso (1)

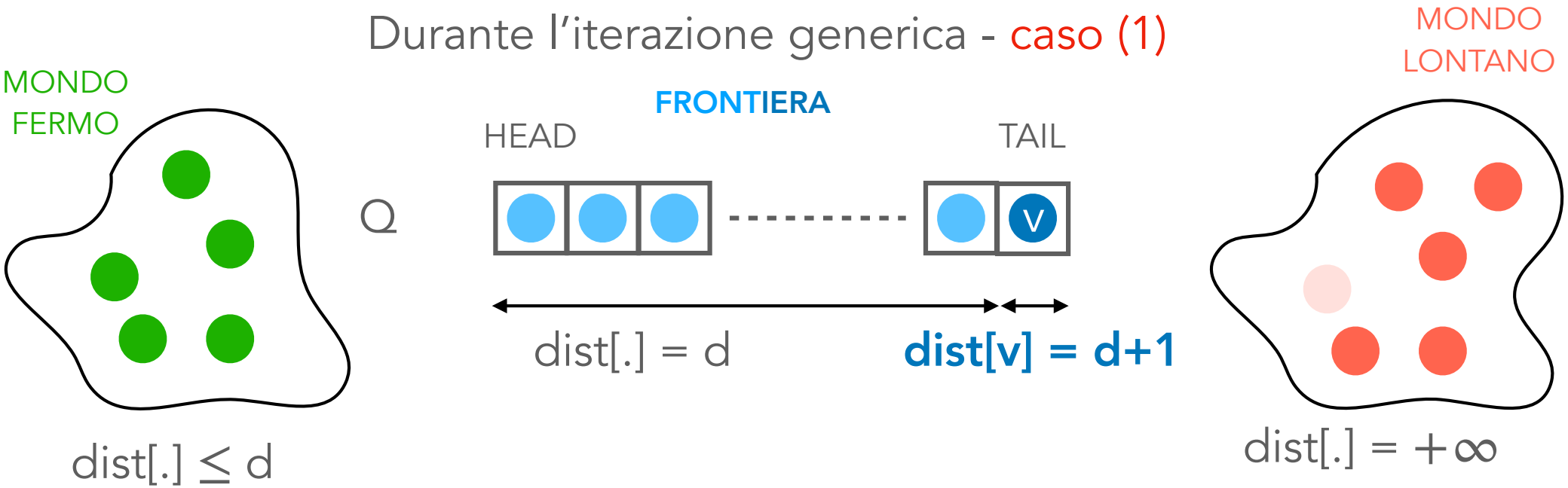


I vicini di v NON sono già stati esplorati,
v NON è ancora nella frontiera,
v viene inserito nella frontiera con **dist[v] = d+1**

NON sarà possibile trovare un
cammino più breve da s a v

BFS su grafi

Durante l'iterazione generica - caso (1)



u

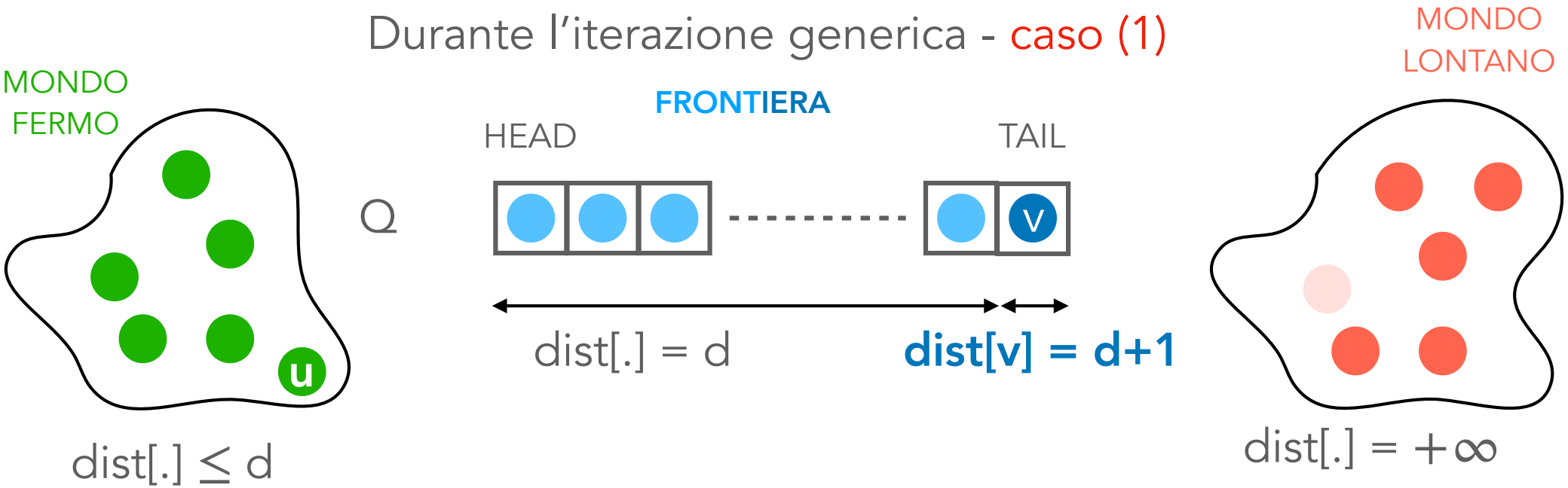
dist[u] = d

I vicini di v NON sono già stati esplorati,
v NON e' ancora nella frontiera,
v viene inserito nella frontiera con **dist[v] = d+1**

NON sarà possibile trovare un
cammino più breve da s a v

BFS su grafi

Durante l'iterazione generica - caso (1)



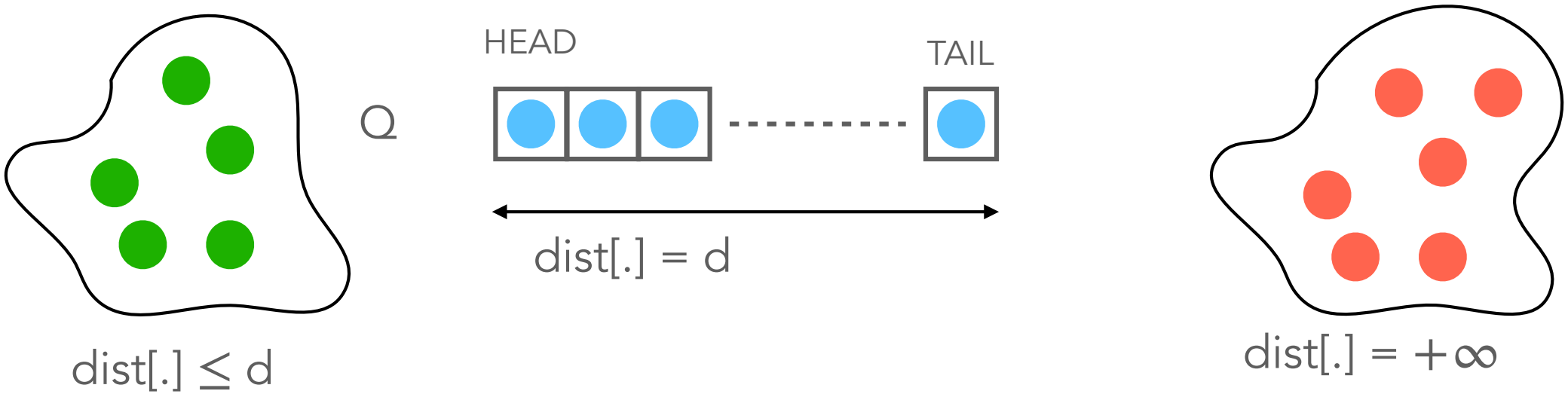
dist[u] = d

u è stato scoperto,
i vicini di u sono stati esplorati

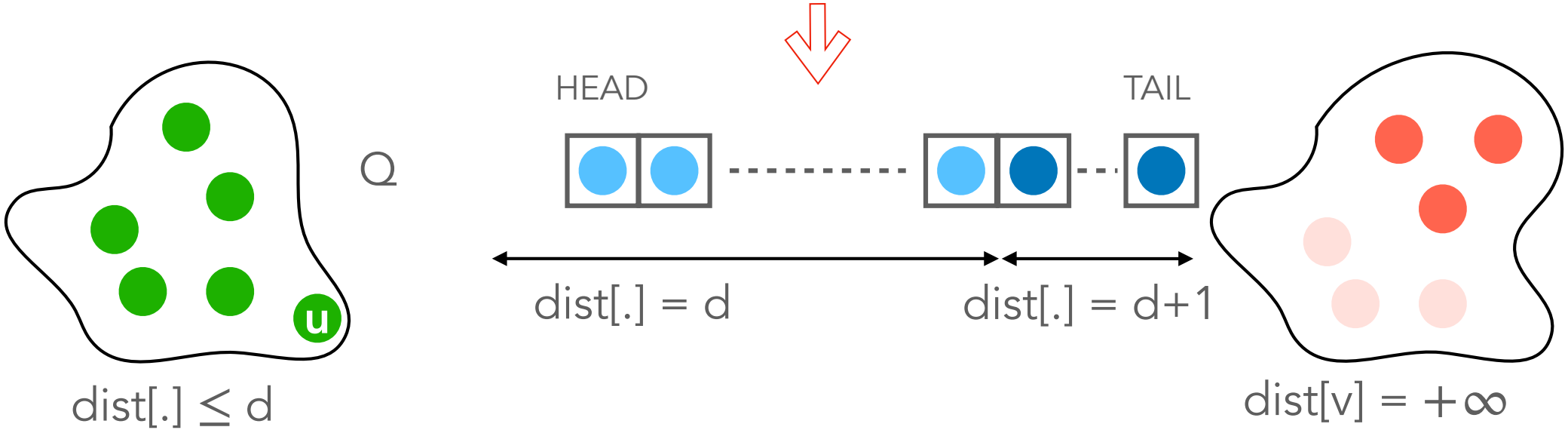
u entra nel MONDO FERMO

BFS su grafi

All'inizio dell'iterazione generica - caso (1)



Alla fine dell'iterazione generica - caso (1)



BFS su grafi

All'inizio dell'iterazione generica successiva

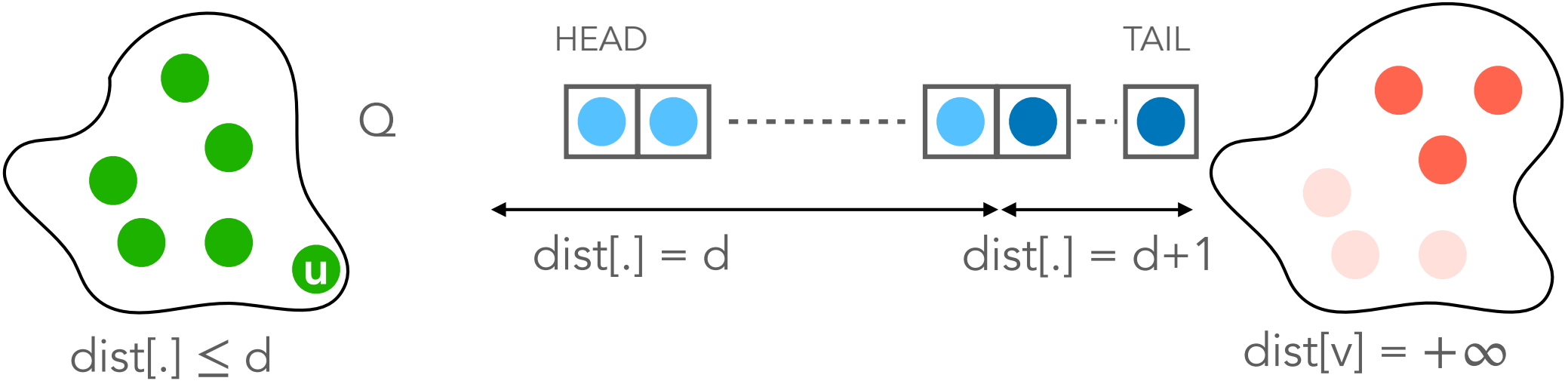
I nodi del grafo possono essere divisi in tre insiemi:

- **MONDO FERMO**: nodi gia' visitati, vicini esplorati, **dist[.] ≤ d corretto**
- **FRONTIERA**: nodi già scoperti, vicini NON esplorati, **d ≤ dist[.] ≤ d+1 corretto**
- **MONDO LONTANO**: nodi non visitati, **dist[.] = +∞**

Per lo stesso **d** dell'iterazione precedente

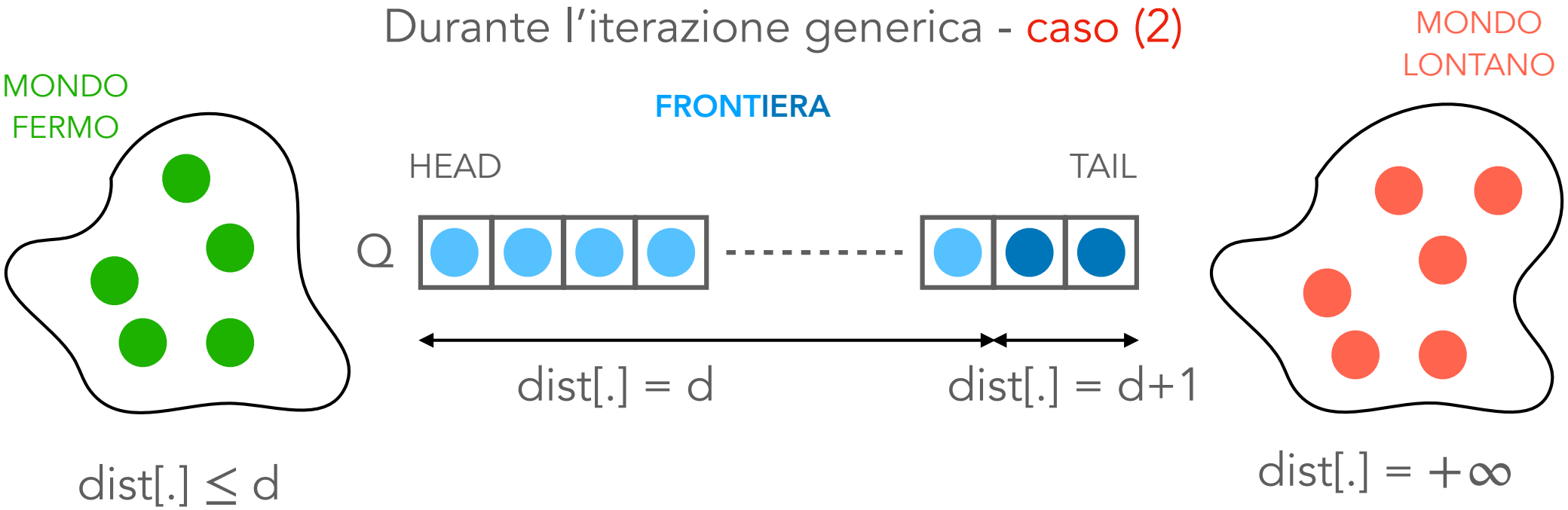
⇑

Alla fine dell'iterazione generica - caso (1)



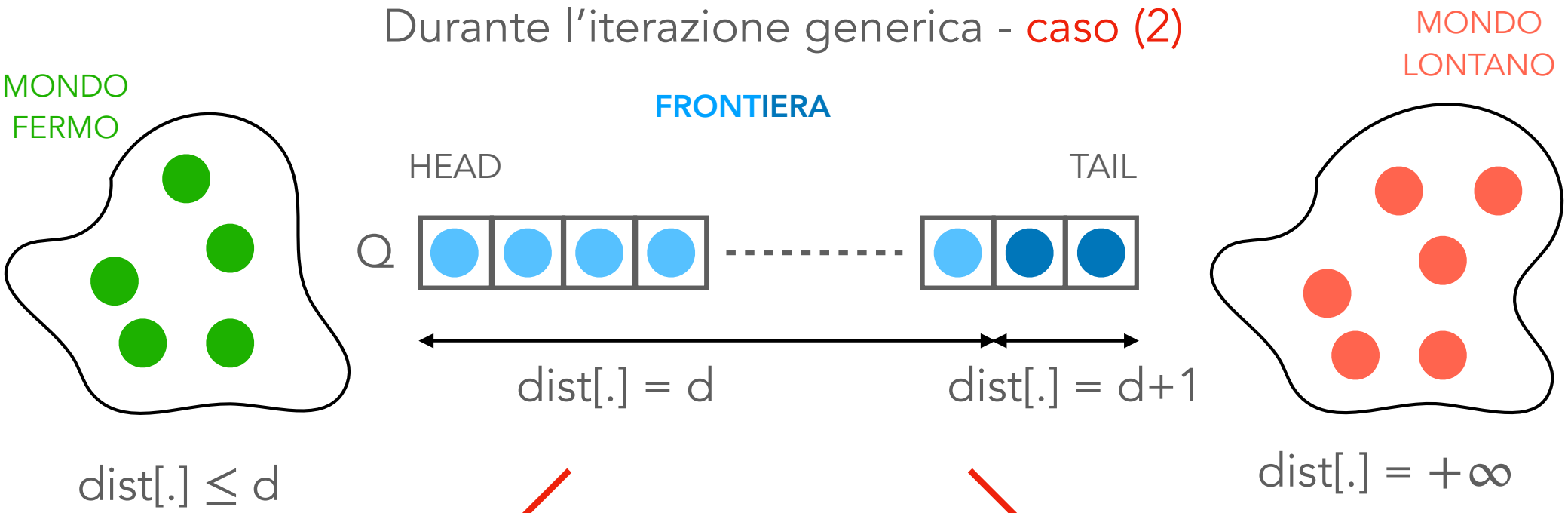
BFS su grafi

Durante l'iterazione generica - caso (2)



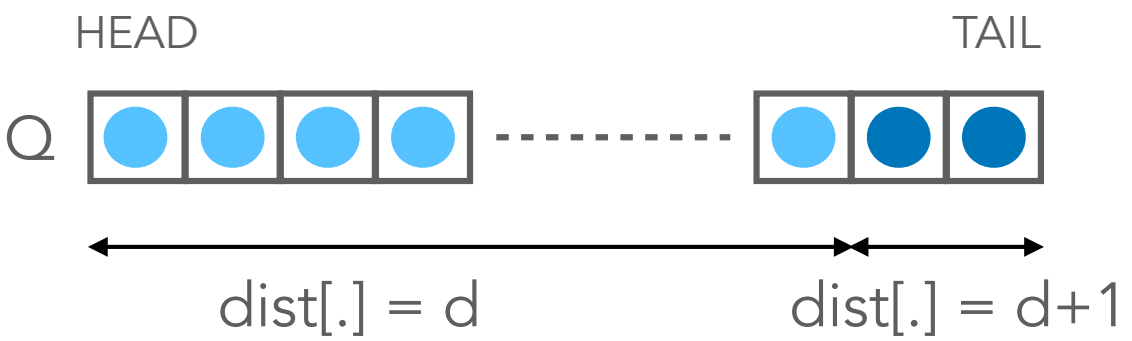
BFS su grafi

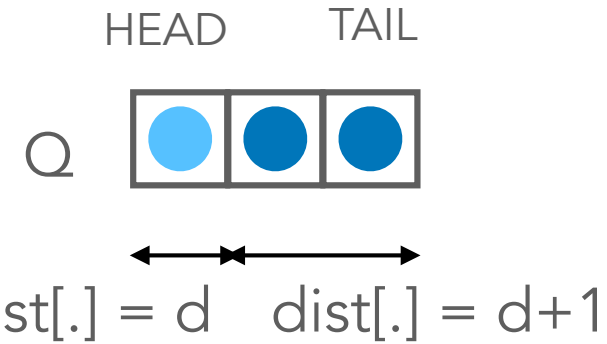
Durante l'iterazione generica - caso (2)



caso (2.1)

caso (2.2)



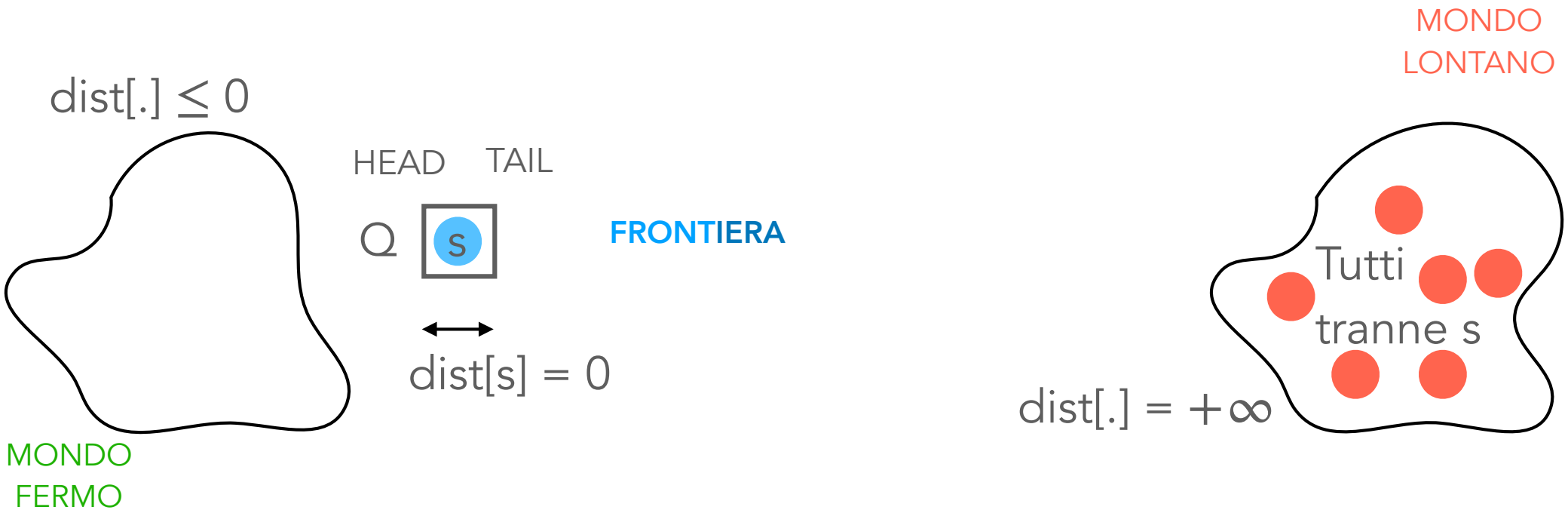


BFS su grafi

COME SI INIZIA?

All'inizio della prima iterazione **d=0**:

- **MONDO FERMO**: vuoto (nessun nodo già visitato)
- **FRONTIERA**: contiene solo s e dist[s] = 0 corretto - unico nodo già scoperto
- **MONDO LONTANO**: tutti i nodi tranne s (altri nodi non scoperti)



BFS su grafi

COME TERMINIAMO?

Alla fine dell'ultima iterazione, per qualche **d<n**:

- **MONDO FERMO**: tutti i nodi sono stati visitati
- **FRONTIERA**: vuota (nessun nodo scoperto ma non visitato)
- **MONDO LONTANO**: vuoto (nessun nodo non ancora scoperto)

$dist[.] \leq d$

FRONTIERA

$dist[.] = +\infty$

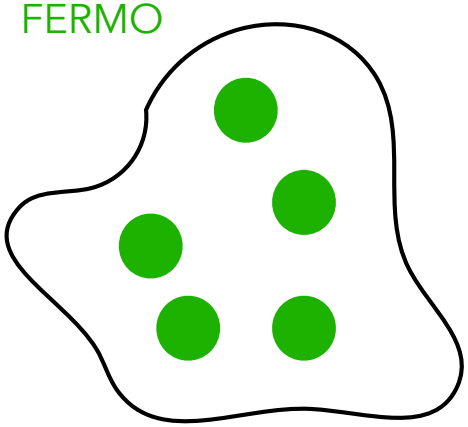
BFS su grafi

COME TERMINIAMO?

Alla fine dell'ultima iterazione, per qualche **d<n**:

- **MONDO FERMO**: tutti i nodi sono stati visitati
- **FRONTIERA**: vuota (nessun nodo scoperto ma non visitato)
- **MONDO LONTANO**: vuoto (nessun nodo non ancora scoperto)

MONDO FERMO



dist[.] ≤ d

DOMANDA:

quanto vale **d** alla fine?

FRONTIERA

MONDO LONTANO

dist[.] = +∞