Algoritmi e Strutture Dati

Lezione 23

14 novembre 2025

ALBERO RICOPRENTE MINIMO:

una strategia greedy alternativa

Problema: Dato $G=(V,E)$ non orientato connesso con una funzione peso $w: E \to \mathbb{R}$ trovare un albero ricoprente $T=(V,E_T)$ di peso minimo

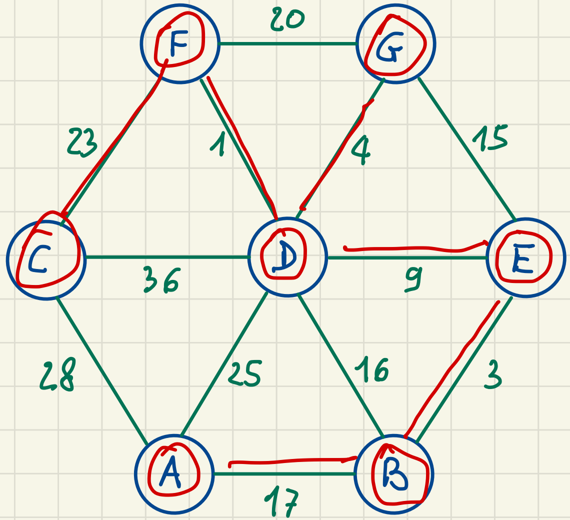
Inizialmente:

T: albero costituito da un unico vertice

Algoritmo di Prim

Ad ogni passo:

si espande T aggiungendo l'arco (x,y) di peso minimo con un vertice in T e l'altro non in T



ALGORITMO Prim (Grafo $G=(V,E,\omega)$) $\to$ albero

$T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

WHILE T ha meno di n nodi DO

  sia (x,y) l'arco di peso minimo con x in T e y non in T

  aggiungi a T il vertice y e l'arco (x,y)

RETURN T

ALGORITMO di PRIM : correttezza

Teo L'algoritmo di Prim trova un albero ricoprente minimo

PRIM vs KRUSKAL

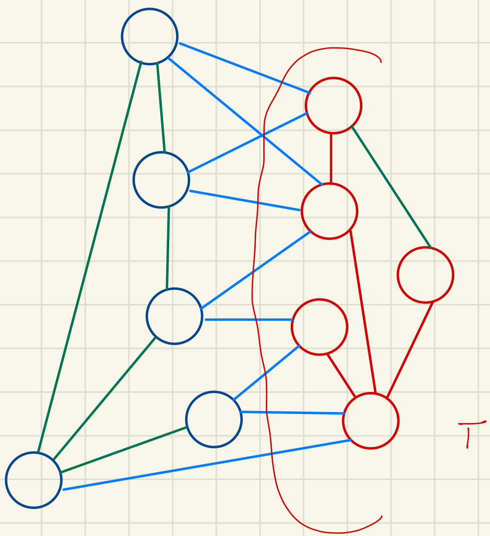
Strategie greedy differenti

- KRUSKAL
  - soluzione parziale: foresta di alberi con insieme di vertici V
  - inizialmente: tutti i vertici, nessun arco

- PRIM
  - soluzione parziale: albero $T = (V_T, E_T)$ con $V_T \subseteq V$

    $E_T \subseteq E \cap (V_T \times V_T)$
  - inizialmente: albero formato da un unico vertice

ALGORITMO di PRIM: implementazione



ALGORITMO Prim (Grafo $G=(V,E,w)$) $\to$ albero

$T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

WHILE T ha meno di n nodi DO

sia (x,y) l'arco di peso minimo con x in T e y non in T

aggiungi a T il vertice y e l'arco (x,y)

RETURN T

ALGORITMO di PRIM: implementazione

ALGORITMO Prim (Grafo $G=(V,E,w)$) $\rightarrow$ albero

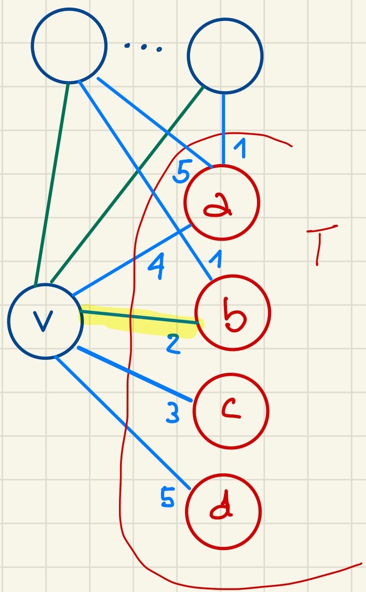
T $\leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

WHILE T ha meno di n nodi DO

sia (x,y) l'arco di peso minimo con x in T e y non in T

aggiungi a T il vertice y e l'arco (x,y)

RETURN T



$d[v] = 2$

$vicino[v] = b$

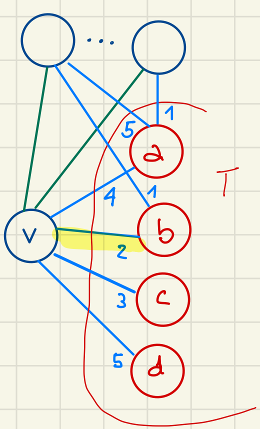
ALGORITMO di PRIM: implementazione

ALGORITMO Prim (Grafo $G = (V, E, \omega)$) $\rightarrow$ albero
  $T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi
  WHILE T ha meno di n nodi DO
    sia (x,y) l'arco di peso minimo con x in T e y non in T
    aggiungi a T il vertice y e l'arco (x,y)
  RETURN T

AD OGNI PASSO PER OGNI VERTICE v NON ANCORA IN T CONSIDERIAMO:

$d[v]$ = minimo peso degli archi tra v e vertici in T (∞ se non ce ne sono)

$$d[v] = \min \{ \omega(u,v) \mid u \in V_T \} \cup \{ \infty \} \qquad T = (V_T, E_T)$$



$vicino[v]$ = vertice u in T t.c. $d[v] = \omega(u,v)$

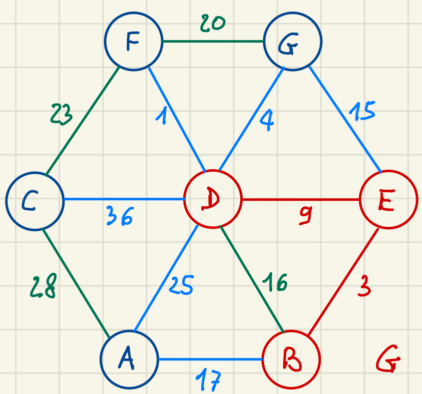
ALGORITMO di PRIM: implementazione

$d[v]$ = minimo peso degli archi
tra v e vertici in T
(∞ se non ce ne sono)

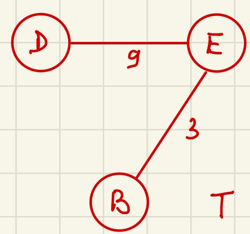
$d[v] = \min\{\omega(u,v) \mid u \in V_T\} \cup \{\infty\}$ $\quad T = (V_T, E_T)$

$vicino[v]$ = vertice u t.c. $d[v] = \omega(u,v)$

ALGORITMO Prim (Grafo $G = (V, E, \omega)$) → albero
  $T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi
  WHILE T ha meno di n nodi DO
    sia (x,y) l'arco di peso minimo
    con x in T e y non in T
    aggiungi a T il vertice y e l'arco (x,y)
  RETURN T



G



T

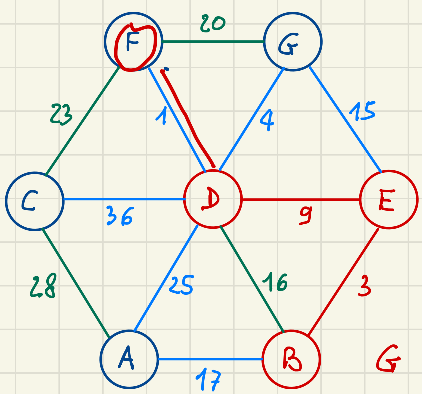
	d	vicino
A	17	B
C	36	D
F	1	D
G	4	D

ALGORITMO di PRIM: implementazione

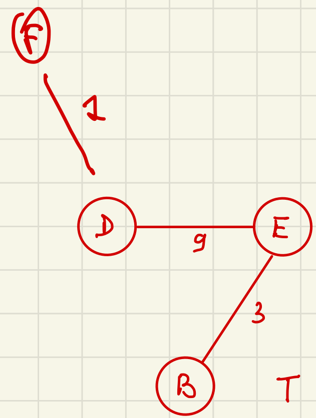
$d[v]$ = minimo peso degli archi tra v e vertici in T (∞ se non ce ne sono)

$$d[v] = \min\{\omega(u,v) \mid u \in V_T\} \cup \{\infty\} \qquad T=(V_T, E_T)$$

vicino$[v]$ = vertice u t.c. $d[v] = \omega(u,v)$



G



T

	d	vicino
A	17	B
C	~~36~~ 23	~~D~~ F
~~F~~	~~1~~	~~D~~
G	4	D

ALGORITMO Prim (Grafo $G=(V,E,\omega)$) $\to$ albero

Siano d e vicino due array con indici in V

FOR EACH $v \in V$ DO
    $d[v] \leftarrow \infty$
$T \leftarrow (V_T \leftarrow \emptyset,\ E_T \leftarrow \emptyset)$

DO
    $y \leftarrow$ elemento di $V \setminus V_T$ con $d[y]$ minima
    $V_T \leftarrow V_T \cup \{y\}$
    IF $d[y] \neq \infty$ THEN   // falsa solo alla prima iterazione
        $x \leftarrow vicino[y]$
        $E_T \leftarrow E_T \cup \{(x,y)\}$
    FOR EACH $(y,z) \in E$ DO
        IF $z \notin V_T$ AND $\omega(y,z) < d[z]$ THEN
            $d[z] \leftarrow \omega(y,z)$
            $vicino[z] \leftarrow y$
WHILE $V \setminus V_T \neq \emptyset$
RETURN T

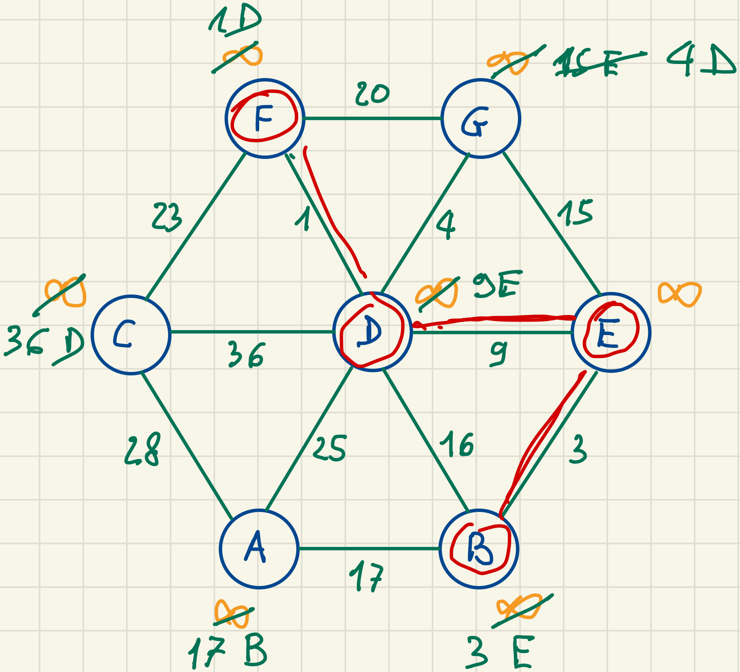
ALGORITMO Prim (Grafo $G=(V,E,\omega)$) $\to$ albero
    $T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi
    WHILE T ha meno di n nodi DO
        sia (x,y) l'arco di peso minimo con x in T e y non in T
        aggiungi a T il vertice y e l'arco (x,y)
    RETURN T

Inizialmente
Struttura vuota:
ogni vertice v ha $d[v] = \infty$

Ad ogni passo:
Si preleva vertice y con $d[y]$ minima

Si aggiunge a T il vertice y e l'arco (x,y) con $x = vicino[y]$
(caso particolare: prima iterazione)

Per ogni z non in T adiacente a y:
se la distanza da T è diminuita
si aggiornano $d[z]$ e $vicino[z]$




```
ALGORITMO Prim (Grafo G=(V,E,ω)) → albero
  Siano d e vicino due array con indici in V
  FOR EACH v∈V DO
  |  d[v] ← ∞
  T ← (V_T ← ∅, E_T ← ∅)
  DO
  |  y ← elemento di V∖V_T con d[y] minima
  |  V_T ← V_T ∪ {y}
  |  IF d[y] ≠ ∞ THEN          //Falsa solo alla prima iterazione
  |  |  x ← vicino[y]
  |  |  E_T ← E_T ∪ {(x,y)}
  |  FOR EACH (y,z) ∈ E DO
  |  |  IF z ∉ V_T AND ω(y,z) < d[z] THEN
  |  |  |  d[z] ← ω(y,z)
  |  |  |  vicino[z] ← y
  WHILE V∖V_T ≠ ∅
  RETURN T
```

ALGORITMO Prim $(\text{Grafo } G=(V,E,\omega)) \to$ albero

Sia C una coda con priorità vuota

Siano d e vicino due array con indici in V

FOR EACH $v \in V$ DO

  $d[v] \leftarrow \infty$ $C.\text{insert}(v, \infty)$

$T \leftarrow (V_T \leftarrow \emptyset,\ E_T \leftarrow \emptyset)$

DO

  ~~$y \leftarrow$ elemento di $V \setminus V_T$ con $d[y]$ minima~~ $y \leftarrow C.\text{deleteMin}()$

  $V_T \leftarrow V_T \cup \{y\}$

  IF $d[y] \neq \infty$ THEN //falsa solo alla prima iterazione

    $x \leftarrow \text{vicino}[y]$

    $E_T \leftarrow E_T \cup \{(x,y)\}$

  FOR EACH $(y,z) \in E$ DO

    IF $z \notin V_T$ AND $\omega(y,z) < d[z]$ THEN

      $d[z] \leftarrow \omega(y,z)$ $C.\text{changeKey}(z, \omega(y,z))$

      $\text{vicino}[z] \leftarrow y$

WHILE ~~$V \setminus V_T \neq \emptyset$~~ $C \neq \emptyset$

RETURN T

ALGORITMO Prim $(\text{Grafo } G=(V,E,\omega)) \to$ albero

$T \leftarrow$ albero costituito da un unico vertice $s \in V$ qualsiasi

WHILE T ha meno di n nodi DO

  sia (x,y) l'arco di peso minimo con x in T e y non in T

  aggiungi a T il vertice y e l'arco (x,y)

RETURN T

Inizialmente

Struttura vuota: ogni vertice v ha $d[v]=\infty$

Ad ogni passo:

Si preleva vertice y con $d[y]$ minima

Si aggiunge a T il vertice y e l'arco (x,y) con $x=\text{vicino}[y]$ (caso particolare: prima iterazione)

Per ogni z non in T adiacente a y: se la distanza da T è diminuita si aggiornano $d[z]$ e $\text{vicino}[z]$

ALGORITMO di PRIM: implementazione con code con priorità

Rappresentazione di G: liste di adiacenza

Coda con priorità C: heap di n elementi (vettore posizionale)

		Tempo
$\boxed{1}$	riempi coda ≡ riempi array	$O(n)$
$\boxed{2}$	ciclo DO-WHILE: n iterazioni	
$\boxed{2a}$	deleteMin costo singolo $O(\log n)$, n volte	$O(n \log n)$
$\boxed{2b}$		$O(n)$
$\boxed{2c}$	intestazione del ciclo lista di adiacenza: $\delta(y)$ iterazioni → totale $2m$	$O(m)$
$\boxed{2d}$	changeKey $O(\log n)$ costo singolo al max m volte	$O(m \log n)$

$$O(n \log n) + O(m \log n) = O(m \log n)$$

$n-1 \leq m \leq 2^2$

ALGORITMO Prim (Grafo $G = (V, E, \omega)$) → albero

Sia C una coda con priorità vuota

Siano d e vicino due array con indici in V

```
[1] FOR EACH v ∈ V DO
        d[v] ← ∞
        C.insert(v, ∞)
    T ← (V_T ← ∅, E_T ← ∅)
[2] DO
        y ← C.deleteMin()                  [a]
        V_T ← V_T ∪ {y}
        IF d[y] ≠ ∞ THEN                   [b]  // falsa solo alla prima iterazione
            x ← vicino[y]
            E_T ← E_T ∪ {(x,y)}
        FOR EACH (y,z) ∈ E DO              [c]
            IF z ∉ V_T AND ω(y,z) < d[z] THEN    [d]
                d[z] ← ω(y,z)
                C.changeKey(z, ω(y,z))
                vicino[z] ← y
    WHILE C ≠ ∅
    RETURN T
```

Grafo: lista adiacenza / incidenza

$$T(n, m) = O(m \lg n)$$

code con priorità implementare con min-heap

Heap di Fibonacci → changeKey $O(1)$ quando la chiave si riduce

$$T(n, m) = O(m + n \lg n)$$