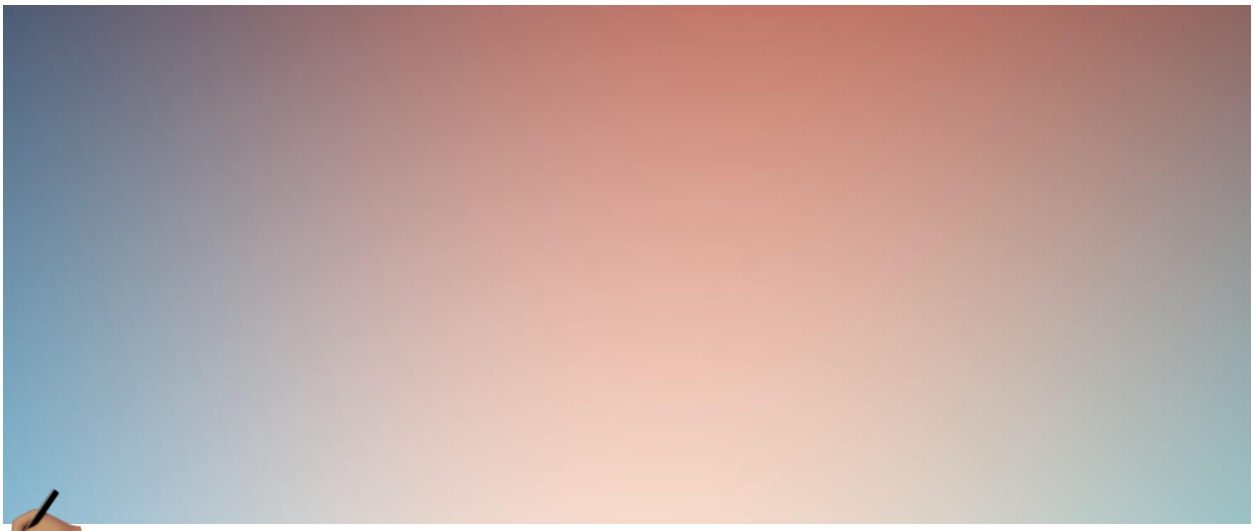


# Esercizi Svolti

## Programmazione Dinamica

### Interval Scheduling Pesato

#### Possibile traccia

Generalmente, se all'esame viene dato un esercizio su Interval Scheduling Pesato, in input vengono forniti i tempi di inizio, i tempi di fine e i relativi pesi dei job. La traccia potrebbe chiedere di calcolare i $p(j)$ relativi ai job e poi di fornire l'array $M$ dei valori computati dall'algoritmo. Inoltre, cosa più importante, è richiesto di fornire il valore della soluzione ottima e di contrassegnare con un cerchio le entrate ispezionate dall'algoritmo.

#### Svolgimento + Esempio

Un'istanza di Interval Scheduling Pesato si presenta generalmente così:

- $s_1 = 7, f_1 = 9, v_1 = 4$
- $s_2 = 1, f_2 = 4, v_2 = 3$
- $s_3 = 5, f_3 = 6, v_3 = 8$
- $s_4 = 3, f_4 = 8, v_4 = 13$

La prima cosa da fare, come dice anche l'algoritmo, è ordinare i job in base ai loro tempi di fine.

- $s_1 = 1, f_1 = 4, v_1 = 3$
- $s_2 = 5, f_2 = 6, v_2 = 8$
- $s_3 = 3, f_3 = 8, v_3 = 13$
- $s_4 = 7, f_4 = 9, v_4 = 4$

Adesso, il primo punto dell'esercizio richiede di calcolare i valori $p(j)$ per ogni job. Sappiamo che il valore $p(j)$ per un job $j$ è definito come l'indice del job più vicino a $j$, tra i job che finiscono prima dell'inizio di $j$. Non ci hai capito nulla? Comunque è l'ultimo job compatibile con $j$ in termini di tempo. Con l'esempio sarà più chiaro.

- $p(1)$ = essendo il primo job, banalmente non esiste un job compatibile che finisce prima di $s_1$, quindi $\mathbf{0}$.

- $p(2)$ = in questo caso, notiamo come $f_1 = 4$, che è $\leq s_2$, quindi $5$, quindi abbiamo trovato un job compatibile. Essendo l'unico job che lo precede $\mathbf{1}$
- $p(3)$ = qui, invece, notiamo come non esista un job precedente tale che la sua fine sia $\leq 3$, quindi automaticamente, non esiste nessun job compatibile $\mathbf{0}$
- $p(4)$ = notiamo come ci siano ben due job compatibili che precedono il job $4$, infatti $6$ e $4$ sono entrambi $\leq 7$, quindi $\mathbf{2}$

Ottenuti i $p(j)$, adesso, possiamo procedere col calcolo dei valori della tabella $M$, ricordandoci sempre la $OPT$, che è fondamentale. In questo caso, non ci interessa del caso base, dato che i job ci sono.

- $OPT(1) = max\{v_1 + OPT(p(1)), OPT(1-1)\} = max\{3+0, 0\} = \mathbf{3}$
- $OPT(2) = max\{v_2 + OPT(p(2)), OPT(2-1)\} = max\{8+3, 3\} = \mathbf{11}$
- $OPT(3) = max\{v_3 + OPT(p(3)), OPT(3-1)\} = max\{13+0, 11\} = \mathbf{13}$
- $OPT(4) = max\{v_4 + OPT(p(4)), OPT(4-1)\} = max\{4+11, 11\} = \mathbf{15}$

Sicuramente, possiamo già scrivere che $OPT(4) = 15$ rappresenta il valore massimo ottenibile considerando tutti i job fino al job $4$. Adesso, però, viene chiesto di fornire la soluzione ottima, cerchiando le entrate dell'algoritmo sulle celle dell'array. Esiste un modo piuttosto intuitivo, senza dover per forza ricorrere all'algoritmo della stampa della soluzione ottima. Partiamo dall'ultimo job, cioè $OPT(4) = 15$ e verifichiamo se è $>$ di $OPT(3) = 13$. Banalmente sì, quindi il job $\mathbf{4}$ è incluso nella soluzione. Passiamo a $p(4) = 2$, quindi analizziamo il job $2$ e vediamo se includerlo nella soluzione. Vediamo come $OPT(2) = 11 > OPT(1) = 3$, quindi includiamo anche il job $\mathbf{2}$ nella soluzione. Passiamo a $p(2) = 1$, quindi analizziamo il job $1$, notando come $OPT(1) = 3 > OPT(0) = 0$, quindi anche il job $\mathbf{1}$ è incluso nella soluzione. Andando verso $p(1) = 0$ , raggiungiamo l'ultimo job, quindi non serve andare oltre. Quindi:

- $OPT(j) = [1, 2, 4]$

Che è il nostro sottoinsieme compatibile con massimo peso totale di $15$. Dato che viene chiesto di cerchiare le entrate nell'array, andremo, ovviamente, a cerchiare $15$, $11$ e $3$.

## Subset Sum/Zaino

### Possibile traccia

Generalmente, se all'esame viene dato un esercizio su Subset Sum, in input vengono forniti i pesi dei relativi job con il limite sul processore $W$. La traccia potrebbe richiedere sicuramente di disegnare la tabella $M$ dei valori computati dall'algoritmo e viene chiesto di fornire la soluzione ottima e il suo valore, indicando con un cerchio le celle su cui viene invocato l'algoritmo che stampa la soluzione ottima. In realtà, vale lo stesso svolgimento anche per il Problema dello Zaino, soltanto che cambierà l'istanza in input, poiché avremo i valori degli oggetti, ma per il calcolo del valore e della soluzione ottima è pressoché identico

### Svolgimento + Esempio

Un'istanza di Subset Sum si presenta generalmente così:

- $w_1 = 2, w_2 = 1, w_3 = 5, w_4 = 2$
- $W = 6$

La prima cosa da fare è sicuramente disegnare la tabella, in cui le righe saranno il numero di job, mentre le colonne è il limite del processore. È fondamentale partire sempre da $0$. E, per prassi, riempiamo sempre la prima riga e la prima colonna di $0$.

| $i/w$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 0 | **0** | **0** | **0** | **0** | **0** | **0** | **0** |
| 1 | **0** | **0** | **2** | **2** | **2** | **2** | **2** |
| 2 | **0** | **2** | **3** | **3** | **3** | **3** | **3** |
| 3 | **0** | **2** | **3** | **3** | **3** | **5** | **5** |

| $i/w$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 4 | **0** | **2** | **3** | **4** | **5** | **5** | **5** |

Adesso, bisogna ricordarsi la OPT, mostreremo tutto il processo passo-passo.

- $OPT(1,1) = w_1 = 2 > 1 = OPT(0,1) = \mathbf{0}$
- $OPT(1,2) = max\{OPT(0,2), 2 + OPT(0,0)\} = max\{0,2\} = \mathbf{2}$
- $OPT(1,3) = max\{OPT(0,3), 2 + OPT(0,1)\} = max\{0,2\} = \mathbf{2}$
- $OPT(1,4) = max\{OPT(0,4), 2 + OPT(0,2)\} = max\{0,2\} = \mathbf{2}$
- $OPT(1,5) = max\{OPT(0,5), 2 + OPT(0,3)\} = max\{0,2\} = \mathbf{2}$
- $OPT(1,6) = max\{OPT(0,6), 2 + OPT(0,4)\} = max\{0,2\} = \mathbf{2}$
- $OPT(2,1) = max\{OPT(1,1), 1 + OPT(1,0)\} = max\{2,1\} = \mathbf{2}$
- $OPT(2,2) = max\{OPT(1,2), 1 + OPT(1,1)\} = max\{2,3\} = \mathbf{3}$
- $OPT(2,3) = max\{OPT(1,3), 1 + OPT(1,2)\} = max\{2,3\} = \mathbf{3}$
- $OPT(2,4) = max\{OPT(1,4), 1 + OPT(1,3)\} = max\{2,3\} = \mathbf{3}$
- $OPT(2,5) = max\{OPT(1,5), 1 + OPT(1,4)\} = max\{2,3\} = \mathbf{3}$
- $OPT(2,6) = max\{OPT(1,6), 1 + OPT(1,5)\} = max\{2,3\} = \mathbf{3}$
- $OPT(3,1) = w_3 = 5 > 1 = OPT(2,1) = \mathbf{2}$
- $OPT(3,2) = w_3 = 5 > 2 = OPT(2,2) = \mathbf{3}$
- $OPT(3,3) = w_1 = 5 > 3 = OPT(2,3) = \mathbf{3}$
- $OPT(3,4) = w_1 = 5 > 4 = OPT(2,4) = \mathbf{3}$
- $OPT(3,5) = max\{OPT(2,5), 5 + OPT(2,0)\} = max\{3,5\} = \mathbf{5}$
- $OPT(3,6) = max\{OPT(2,6), 5 + OPT(2,1)\} = max\{3,5\} = \mathbf{5}$
- $OPT(4,1) = w_2 = 2 > 1 = OPT(3,1) = \mathbf{2}$
- $OPT(4,2) = max\{OPT(3,2), 2 + OPT(3,0)\} = max\{3,2\} = \mathbf{3}$
- $OPT(4,3) = max\{OPT(3,3), 2 + OPT(3,1)\} = max\{3,4\} = \mathbf{4}$
- $OPT(4,4) = max\{OPT(3,4), 2 + OPT(3,2)\} = max\{3,5\} = \mathbf{5}$
- $OPT(4,5) = max\{OPT(3,5), 2 + OPT(3,3)\} = max\{5,5\} = \mathbf{5}$
- $OPT(4,6) = max\{OPT(3,6), 2 + OPT(3,4)\} = max\{5,5\} = \mathbf{5}$

Adesso, l'ultimo elemento in basso a destra, cioè $OPT(4,6) = 5$ è il valore della nostra soluzione ottima. Viene chiesto, però, di fornire la soluzione ottima e di evidenziare le celle della tabella in cui viene eseguito l'algoritmo di stampa. Ancora una volta, non serve ricordare lo pseudocodice, poiché esiste un metodo piuttosto intuitivo. Partiamo dal valore della nostra soluzione ottima e confrontiamo con quello immediatamente sopra. Notiamo come $OPT(4,6) = 5 = OPT(3,6)$, quindi possiamo concludere che il job $4$ non è incluso nella soluzione ottima. Confrontiamo $OPT(3,6) = 5 \neq OPT(2,6) = 3$, quindi il job $\mathbf{3}$ è incluso nella soluzione ottima. Riduciamo $W$ del peso di $w_3 = 5$, quindi $W = 6 - 5 = 1$. Ci spostiamo, quindi, a sinistra, di cinque unità, andando a finire in $OPT(2,1) = 2$ che notiamo subito essere diverso da $OPT(1,1)$. Vuol dire che il job $\mathbf{2}$ è incluso nella soluzione ottima. Andiamo nuovamente a ridurre il peso di $W = 1 - 1 = 0$. Possiamo fermarci, dato che abbiamo azzerato il limite del processore. Concludiamo che i job da includere nella soluzione ottima sono $w_2$ e $w_3$, con un peso totale di $5 + 1 = 6$, che rispetta il limite del processore. Nella tabella, ho anche evidenziato le celle in cui finisce l'algoritmo.

## Minimum Coin Change

## Possibile traccia

Generalmente, se all'esame viene dato un esercizio sul Minimum Coin Change, in input vengono forniti i valori delle monete e il valore della banconota da cambiare. La traccia potrebbe richiedere sicuramente di disegnare la tabella $M$ dei valori computati dall'algoritmo e viene chiesto di fornire la soluzione ottima e il suo valore, indicando con un cerchio le celle su cui viene invocato l'algoritmo che stampa la soluzione ottima.

## Svolgimento + Esempio

Un'istanza di *Minimum Coin Change* si presenta così:

- $v_1 = 1, v_2 = 3, v_3 = 4$
- $V = 6$

Esistono due modi per poter riempire la tabella. Il primo è quello più formale, basandoci sulla $OPT$, mentre il secondo basta andarsene per una semplice logica. Li vedremo entrambi.

| $i/v$ | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $\mathbf{v_1}$ | **0** | **1** | **2** | **3** | **4** | **5** | **6** |
| $\mathbf{v_2}$ | **0** | **1** | **2** | **1** | **2** | **3** | **2** |
| $\mathbf{v_3}$ | **0** | **1** | **2** | **1** | **1** | **2** | **2** |

- La prima riga è generalmente sempre uguale. Un'istanza di *Minimum Coin Change* prevede sempre a prescindere una moneta dal valore $1$. Avendo a disposizione una moneta dal valore $1$, basta riempire le celle con l'esatto valore della banconota in quel momento. È prassi, si va sul sicuro.
- Essendo $v_2 = 3 > v$ nelle prime due colonne, quindi riempiremo semplicemente con $v$.
- $OPT(2,3) = min\{OPT(1,3), 1 + OPT(2,0)\} = min\{3,1\} = \mathbf{1}$
- $OPT(2,4) = min\{OPT(1,4), 1 + OPT(2,1)\} = min\{4,2\} = \mathbf{2}$
- $OPT(2,5) = min\{OPT(1,5), 1 + OPT(2,2)\} = min\{5,3\} = \mathbf{3}$
- $OPT(2,6) = min\{OPT(1,6), 1 + OPT(2,3)\} = min\{6,2\} = \mathbf{2}$
- Anche in questo caso, $v_4 > v$ per la prima, la seconda e la terza colonna, quindi $OPT(1,v) = 1$, $OPT(2,v) = 2$, $OPT(3,v) = 1$.
- $OPT(3,4) = min\{OPT(2,4), 1 + OPT(3,0)\} = min\{2,1\} = \mathbf{1}$
- $OPT(3,5) = min\{OPT(2,5), 1 + OPT(3,1)\} = min\{3,2\} = \mathbf{2}$
- $OPT(3,6) = min\{OPT(2,6), 1 + OPT(3,2)\} = min\{2,3\} = \mathbf{2}$

Il valore $OPT(3,6) = 2$ rappresenta il minimo numero di monete per ottenere $V = 6$.
In realtà, esiste un metodo molto più intuitivo, ma meno formale, per riempire la tabella.

- Per la prima riga, vale la stessa regola di prima.
- Idem se $v_2 = 3 > v$, quindi $OPT(1,v) = 1$ e $OPT(2,v) = 2$.
- $OPT(2,3)$ = per ottenere $3$, quante monete mi servono avendo a disposizione $v_1$ e $v_2$? $\mathbf{1}$
- $OPT(2,4)$ = per ottenere $4$, quante monete mi servono avendo a disposizione $v_1$ e $v_2$? $\mathbf{2}$

Ovviamente, il discorso è sempre quello, quindi non serve continuare. Andiamo, invece, a fornire la soluzione ottima, cerchiando le celle in cui entra l'algoritmo che la stampa. Partiamo, sempre da $OPT(3,6) = 2$

- $OPT(3,6) = 2 = OPT(2,6)$, quindi la moneta $v_3 = 4$ non fa parte della soluzione ottima.
- Confrontiamo $OPT(2,6)$ con $OPT(2,6 - v_2) + 1$, quindi $OPT(2,3) + 1 = 2$, quindi la moneta $v_2 = 3$ è inclusa nella soluzione ottima.
- Confrontiamo $OPT(2,3) = 1$ e $OPT(1,3) = 3$. I valori non coincidono, quindi la moneta $v_2$ è usata una seconda volta.

Dato che abbiamo usato la moneta $v_2 = 3$ due volte, abbiamo raggiunto il valore della nostra banconota $V = 6$, quindi non serve proseguire oltre. In arancione sono evidenziate le celle in cui si ferma il nostro algoritmo.

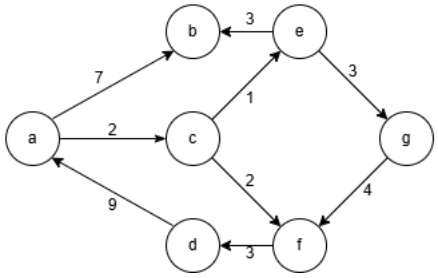
# Bellman-Ford

## Possibile traccia

Generalmente, se all'esame viene dato un esercizio di Bellman-Ford, in input viene fornito un grafo $G$, con un insieme di nodi e di archi e i loro relativi pesi. La traccia potrebbe richiedere sicuramente di fornire i valori delle tabelle $M$ ed $S$, con la tabella $M$ che rappresenta i costi minimi e $S$ con i nodi predecessori. Infine, viene chiesto di fornire il percorso minimo da un nodo qualsiasi $a$ fino al nodo destinazione $g$.

## Svolgimento + Esempio

Un'istanza di Bellman-Ford si presenta come un grafo, di cui conosciamo il numero di nodi, archi e i rispettivi pesi. Ecco come appare:




Iniziamo vedendo come si impostano le varie tabelle. Partiamo dalla tabella $M$. Ad ogni colonna, corrisponde un nodo del grafo, quindi avrà in tutto $V$ colonne. Generalmente, anziché andare in ordine lessicografico, si sceglie sempre di partire dal nodo destinazione, in questo caso $g$. Le righe corrispondono alle iterazioni dell'algoritmo. Con $V$ nodi del grafo, quindi ci sono al massimo $V - 1$ iterazioni, quindi se il grafo ha $7$ nodi, ci saranno $6$ iterazioni.

| $i$ | g | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|
| **0** | **0** | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ |
| **1** | **0** | $\infty$ | $\infty$ | $\infty$ | $\infty$ | **3** | $\infty$ |
| **2** | **0** | $\infty$ | $\infty$ | **4** | $\infty$ | **3** | $\infty$ |
| **3** | **0** | **6** | $\infty$ | **4** | $\infty$ | **3** | $\infty$ |
| **4** | **0** | **6** | $\infty$ | **4** | **15** | **3** | $\infty$ |
| **5** | **0** | **6** | $\infty$ | **4** | **15** | **3** | **18** |
| **6** | **0** | **6** | $\infty$ | **4** | **15** | **3** | **18** |

- $M[0][g] = 0$, banalmente, perché con $0$ passi, se mi trovo già in $g$, automaticamente ho già raggiunto la destinazione e quindi ho costo $0$. Automaticamente, avendo a disposizione $0$ passi, $M[0][a] = M[0][b] = M[0][c] = M[0][d] = M[0][e] = M[0][f] = \infty$, poiché avendo $0$ passi, non posso andare da nessuna parte, quindi costo infinito.
- $M[1][g] = 0$, poiché non serve effettuare nessun aggiornamento, mentre $M[1][a] = M[1][b] = M[1][c] = M[1][d] = \infty$, poiché con un passo non riusciamo a raggiungere il nodo $g$. Discorso diverso per il nodo $e$, poiché con un passo riusciamo a raggiungere $g$, quindi $M[1][e] = min\{\infty, M[0][g] + 3\} = \mathbf{3}$, che aggiorniamo nella tabella.
- $M[2][g] = 0$ e $M[2][e] = 3$ perché non c'è stato alcun aggiornamento. Ovviamente $M[2][a] = M[2][b] = M[2][d] = \infty$, poiché in due passi, da questi nodi, non giungiamo a $g$. Invece, ci toccherà aggiornare $c$, poiché in due passi possiamo raggiungere $g$, quindi $M[2][c] = min\{\infty, M[1][e] + 1\} = \mathbf{4}$, che aggiorniamo nella tabella.
- $M[3][g] = 0$, $M[3][e] = 3$ e $M[3][c] = 4$ poiché non c'è stato nessun aggiornamento. Ovviamente, anche $M[3][b] = M[3][d] = M[3][f] = \infty$, poiché, in tre passi, non giungiamo a $g$. Ci tocca, ora, aggiornare $a$, poiché in tre passi

possiamo raggiungere $g$, quindi $M[3][a] = min\{\infty, M[2][c] + 2\} = \mathbf{6}$, che aggiorniamo nella tabella.

- $M[4][g] = 0$, $M[4][e] = 3$, $M[4][c] = 4$ e $M[4][a] = 6$, poiché non c'è stato alcun aggiornamento. Ovviamente, $M[4][b] = M[4][f] = \mathbf{\infty}$, poiché, in quattro passi, non giungiamo a $g$. Ci tocca, quindi, aggiornare $d$, poiché in quattro passi possiamo raggiungere $g$, quindi $M[4][d] = min\{\infty, M[3][a] + 9\} = \mathbf{15}$, che aggiorniamo nella tabella.
- $M[5][g] = 0$, $M[5][e] = 3$, $M[5][c] = 4$, $M[5][a] = 6$ e $M[5][d] = 15$ poiché non c'è stato alcun aggiornamento. Ovviamente, $M[5][b] = \mathbf{\infty}$, poiché, in cinque passi, non giungiamo a $g$. Ci tocca, quindi, aggiornare $f$, poiché in cinque passi possiamo raggiungere $g$, quindi $M[5][f] = min\{\infty, M[4][d] + 3\} = \mathbf{18}$, che aggiorniamo nella tabella.
- Notiamo, adesso, come nel nostro grafo, il nodo $b$ non abbia nodi uscenti. Automaticamente, anche con sei iterazioni dell'algoritmo, comunque non giungeremo mai a $g$, quindi andremo semplicemente a riscrivere i dati della riga precedente. Piccolo trucchetto, se un nodo non ha archi uscenti, automaticamente, la sua colonna può essere riempita di $\infty$.

Abbiamo, quindi, calcolato i valori della tabella $M$, cioè i costi minimi. Adesso, andremo a calcolare la tabella $S$, dei nodi predecessori, che è quella che ci serve poi, per trovare il cammino minimo da $a$ a $g$.

| $i$ | g | a | b | c | d | e | f |
|---|---|---|---|---|---|---|---|
| **0** | **g** | **∅** | **∅** | **∅** | **∅** | **∅** | **∅** |
| **1** | **g** | **∅** | **∅** | **∅** | **∅** | **g** | **∅** |
| **2** | **g** | **∅** | **∅** | **e** | **∅** | **g** | **∅** |
| **3** | **g** | **c** | **∅** | **e** | **∅** | **g** | **∅** |
| **4** | **g** | **c** | **∅** | **e** | **a** | **g** | **∅** |
| **5** | **g** | **c** | **∅** | **e** | **a** | **g** | **d** |
| **6** | **g** | **c** | **∅** | **e** | **a** | **g** | **d** |

- $S[0][g]$, banalmente, trovandoci già in $g$, abbiamo $0$ passi, quindi raggiungiamo già $g$. Automaticamente, con $0$ passi, non abbiamo predecessori, quindi $S[0][a] = S[0][b] = S[0][c] = S[0][d] = S[0][e] = S[0][f] = \mathbf{\emptyset}$.
- $S[1][g] = g$, poiché non c'è stato alcun aggiornamento. Avendo a disposizione un passo, per calcolare la tabella $M$ abbiamo visto che da $e$ possiamo giungere a $g$, quindi andremo ad aggiornare il predecessore di $e$. Ovviamente, sempre in base ai calcoli precedenti, con un passo non giungiamo a nessun'altro degli altri nodi, quindi il predecessore è sempre nullo.
- $S[2][g] = g$ e $S[2][e] = g$, poiché non c'è stato nessun aggiornamento. Avendo a disposizione due passi, abbiamo visto che il nodo $c$ può raggiungere $g$, quindi aggiorniamo il predecessore di $c$ come $e$, lasciando nulli tutti gli altri.
- $S[3][g] = g$, $S[3][e] = g$ e $S[3][c] = e$, poiché non c'è stato nessun aggiornamento. Avendo a disposizione tre passi, abbiamo visto che il nodo $a$ può raggiungere $g$, quindi aggiorniamo il predecessore di $a$ come $c$, lasciando nulli tutti gli altri.
- $S[4][g] = g$, $S[4][e] = g$, $S[4][c] = e$ e $S[4][a] = c$, poiché non c'è stato alcun aggiornamento. Avendo a disposizione quattro passi, abbiamo visto che il nodo $d$ può raggiungere $g$, quindi aggiorniamo il predecessore di $d$ ad $a$, lasciando nulli tutti gli altri.
- $S[5][g] = g$, $S[5][e] = g$, $S[5][c] = e$, $S[5][a] = c$ e $S[5][d] = a$, poiché non c'è stato alcun aggiornamento. Avendo a disposizione cinque passi, abbiamo visto che il nodo $f$ può raggiungere $g$, quindi aggiorniamo il predecessore di $f$ a $d$, lasciando nulli tutti gli altri.
- Ovviamente, vale lo stesso discorso per la tabella M. Il nodo b non ha archi uscenti, a prescindere non possiamo andare da nessuna parte, quindi ricopiamo semplicemente l'ultima riga.

Adesso, l'esercizio chiede comunque di calcolare il cammino minimo da $a \rightarrow g$. Avendo calcolato entrambe le tabelle, risulta piuttosto banale ottenerlo, infatti basterà guardare la tabella $S$ e risalire a ritroso partendo da $g$. Notiamo che il predecessore di $g$ è $e$, il predecessore di $e$ è $c$ e il predecessore di $c$ è $a$. Siamo giunti al nodo sorgente, quindi il cammino minimo è:

- $a \rightarrow c \rightarrow e \rightarrow g$

## Sottosequenza Comune Più Lunga

### Possibile traccia

Generalmente, se all'esame viene dato un esercizio sulla Sottosequenza Comune Più lunga, in input vengono fornite due stringhe $x$ e $y$, composte da un insieme di caratteri. La traccia potrebbe richiedere sicuramente di fornire i valori della tabella $M[i,j]$. Ovviamente, viene sempre chiesto di fornire il valore della soluzione ottima, che sarebbe la sottosequenza della stringa in comune più la soluzione ottima, evidenziando le celle della tabella su cui viene invocato l'algoritmo che stampa la soluzione ottima.

### Svolgimento + Esempio

Un'istanza di Sottosequenza Comune Più Lunga, si presenta così:

- $x = BACBDAB$
- $y = BDCABA$

| $i/j$ | ε | B | D | C | A | B | A |
|---|---|---|---|---|---|---|---|
| ε | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | ↖ 1 | ← 1 | ← 1 | ← 1 | ↖ 1 | ← 1 |
| A | 0 | ↑ 1 | ↑ 1 | ↑ 1 | ↖ 2 | ← 2 | ↖ 2 |
| C | 0 | ↑ 1 | ↑ 1 | ↖ 2 | ↑ 2 | ↑ 2 | ↑ 2 |
| B | 0 | ↖ 1 | ↑ 1 | ↑ 2 | ↑ 2 | ↖ 3 | ← 3 |
| D | 0 | ↑ 1 | ↖ 2 | ↑ 2 | ↑ 2 | ↑ 3 | ↑ 3 |
| A | 0 | ↑ 1 | ↑ 2 | ↑ 2 | ↖ 3 | ↑ 3 | ↖ 4 |
| B | 0 | ↖ 1 | ↑ 2 | ↑ 2 | ↑ 3 | ↖ 4 | ↑ 4 |

- La prima riga e la prima colonna corrispondono sempre alla stringa vuota $\epsilon$. Automaticamente, essendo stringa vuota, non ci sarà nessun valore, quindi sarà semplicemente uguale a $0$.
- $OPT(B,B)$ = notiamo subito come c'è un match tra caratteri, quindi riempiremo la cella con $OPT(\epsilon,\epsilon)+1 = \mathbf{1}$, andando ad apporre una freccia in diagonale per poi ricostruirci la soluzione ottima.
- $OPT(B,D)$= $B \neq D$, quindi $max\{OPT(B,B), OPT(\epsilon,D)\} = \mathbf{1}$, andando a mettere una freccia orizzontale.
- $OPT(B,C)$ = $B \neq C$, quindi $max\{OPT(B,D), OPT(\epsilon,D)\} = \mathbf{1}$.
- $OPT(B,A)$ = $B \neq A$, quindi $max\{OPT(B,C), OPT(\epsilon,A)\} = \mathbf{1}$.
- $OPT(B,B)$ = $B = B$, quindi $OPT(\epsilon,A)+1 = \mathbf{1}$.
- $OPT(B,A)$ = $B \neq A$, quindi $max\{OPT(B,B), OPT(\epsilon,A)\} = \mathbf{1}$.
- $OPT(A,B)$ = $A \neq B$, quindi $max\{OPT(A,\epsilon), OPT(B,B)\} = \mathbf{1}$.
- $OPT(A,D)$ = $A \neq D$, quindi $max\{OPT(A,\epsilon), OPT(B,B)\} = \mathbf{1}$. Essendo uguali entrambi i valori nelle celle, per prassi, la freccia si mette sempre alla cella superiore, tanto comunque non ci servirà per la costruzione della soluzione ottima.
- $OPT(A,C)$ = $A \neq C$, quindi $max\{OPT(A,D), OPT(B,C)\} = \mathbf{1}$.
- $OPT(A,A)$ = $A = A$, quindi $OPT(B,C)+1 = \mathbf{2}$.
- $OPT(A,B)$ = $A \neq B$, quindi $max\{OPT(A,A), OPT(B,B)\} = \mathbf{2}$.
- $OPT(A,A)$ = $A = A$, quindi $OPT(B,B)+1 = \mathbf{2}$.
- $OPT(C,B)$ = $C \neq B$, quindi $max\{OPT(C,\epsilon), OPT(A,B)\} = \mathbf{1}$.

- $OPT(C, D)$ = $C \neq D$, quindi $max\{OPT(C, B), OPT(A, D)\} = \mathbf{1}$.
- $OPT(C, C)$ = $C = C$, quindi $OPT(A, D) + 1 = \mathbf{2}$.
- $OPT(C, A)$ = $C \neq A$, quindi $max\{OPT(C, C), OPT(A, A)\} = \mathbf{2}$.
- $OPT(C, B)$ = $C \neq B$, quindi $max\{OPT(C, A), OPT(A, B)\} = \mathbf{2}$.
- $OPT(C, A)$ = $C \neq A$, quindi $max\{OPT(C, B), OPT(A, A)\} = \mathbf{2}$.
- $OPT(B, B)$ = $B = B$, quindi $OPT(C, \epsilon) + 1 = \mathbf{1}$.
- $OPT(B, D)$ = $B \neq D$, quindi $max\{OPT(B, B), OPT(B, D)\} = \mathbf{1}$.
- $OPT(B, C)$ = $B \neq C$, quindi $max\{OPT(B, D), OPT(C, C)\} = \mathbf{2}$.
- $OPT(B, A)$ = $B \neq A$, quindi $max\{OPT(B, D), OPT(C, A)\} = \mathbf{2}$.
- $OPT(B, B)$ = $B = B$, quindi $OPT(C, A) + 1 = \mathbf{3}$.
- $OPT(B, A)$ = $B \neq A$, quindi $max\{OPT(B, B), OPT(C, A)\} = \mathbf{3}$.
- $OPT(D, B)$ = $D \neq B$, quindi $max\{OPT(D, \epsilon), OPT(B, B)\} = \mathbf{1}$.
- $OPT(D, D)$ = $D = D$, quindi $OPT(B, B) + 1 = \mathbf{2}$.
- $OPT(D, C)$ = $D \neq C$, quindi $max\{OPT(D, D), OPT(B, C)\} = \mathbf{2}$.
- $OPT(D, A)$ = $D \neq A$, quindi $max\{OPT(D, D), OPT(B, A)\} = \mathbf{2}$.
- $OPT(D, B)$ = $D \neq B$, quindi $max\{OPT(D, A), OPT(B, B)\} = \mathbf{3}$.
- $OPT(D, A)$ = $D \neq A$, quindi $max\{OPT(D, B), OPT(B, A)\} = \mathbf{3}$.
- $OPT(A, B)$ = $A \neq B$, quindi $max\{OPT(A, \epsilon), OPT(D, B)\} = \mathbf{1}$.
- $OPT(A, D)$ = $A \neq D$, quindi $max\{OPT(A, B), OPT(D, D)\} = \mathbf{2}$.
- $OPT(A, C)$ = $A \neq C$, quindi $max\{OPT(A, D), OPT(D, C)\} = \mathbf{2}$.
- $OPT(A, A)$ = $A = A$, quindi $OPT(D, C) + 1 = \mathbf{3}$.
- $OPT(A, B)$ = $A \neq B$, quindi $max\{OPT(A, A), OPT(D, B)\} = \mathbf{3}$.
- $OPT(A, A)$ = $A = A$, quindi $OPT(D, B) + 1 = \mathbf{4}$.
- $OPT(B, B)$ = $B = B$, quindi $OPT(A, \epsilon) + 1 = \mathbf{1}$.
- $OPT(B, D)$ = $B \neq D$, quindi $max\{OPT(B, B), OPT(A, D)\} = \mathbf{2}$.
- $OPT(B, C)$ = $B \neq C$, quindi $max\{OPT(B, D), OPT(A, C)\} = \mathbf{2}$.
- $OPT(B, A)$ = $B \neq A$, quindi $max\{OPT(B, C), OPT(A, A)\} = \mathbf{3}$.
- $OPT(B, B)$ = $B = B$, quindi $OPT(A, A) + 1 = \mathbf{4}$.
- $OPT(B, A)$ = $B \neq A$, quindi $max\{OPT(B, B), OPT(A, A)\} = \mathbf{4}$.

In realtà, le frecce non sono esplicitamente richieste, ma servono per poterci costruire, poi, la soluzione ottima, ossia la nostra sottosequenza in comune. Infatti basterà semplicemente prendere quei caratteri in cui è presente la freccia diagonale, cioè in cui c'è un match. Partiamo dall'ultima cella in basso a destra, muovendoci nel senso della freccia, fino a quando non troviamo una freccia in diagonale. Non appena trovata la freccia, ci spostiamo nel suo senso e aggiungiamo quel carattere alla fine della sottosequenza comune, al punto che risulterà essere:

- $BABA$

# Algoritmi Greedy

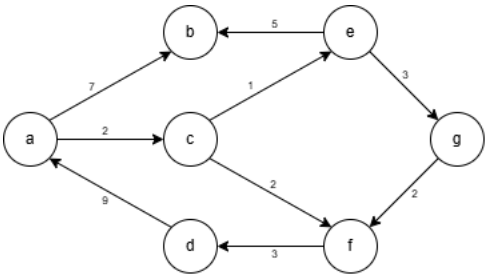
## Algoritmo di Dijkstra per i Cammini Minimi

### Possibile traccia

Generalmente, se all'esame viene dato un esercizio su Dijkstra, in input viene fornito un grafo orientato. La traccia potrebbe chiedere di dimostrare l'esecuzione dell'algoritmo di Dijkstra su questo grafo, a partire da un nodo sorgente arbitrario, di solito $a$, mostrando, ad ogni passo, il contenuto dell'insieme $S$ dei nodi esplorati, della coda $Q$ e anche dell'albero $T$ dei cammini minimi.

### Svolgimento + Esempio

Un'istanza di Dijkstra si presenta generalmente così:




Quindi come un grafo direzionato, con degli archi pesati. Come già anticipato, viene richiesto di mostrare il contenuto dell'insieme $S$, dell'albero $T$ e della coda $Q$. Inizializziamo la nostra istanza:

- $S = \{\emptyset\}$
  $T = \{\emptyset\}$
  $Q = \{(0, a), (\infty, b), (\infty, c), (\infty, d), (\infty, e), (\infty, f), (\infty, g)\}$
  Ovviamente, all'inizio non avremo esplorato nessun nodo, quindi sia
  $S$ che $T$ saranno vuoti. Nella coda, inseriamo tutti i nodi inizialmente con distanza infinita e inizializziamo la sorgente come distanza $0$.

Vediamo, ora, il secondo passo ed estraiamo il minimo dalla coda.

- $S = \{a\}$
  $T = \{\emptyset\}$
  $Q = \{(7, b), (2, c), (\infty, d), (\infty, e), (\infty, f), (\infty, g)\}$
  Estraiamo il minimo dalla coda, in questo caso
  $a$ ed inseriamolo tra i nodi esplorati. Ovviamente, dato che in $T$ vanno gli archi, per ora non abbiamo esplorato nessun arco, quindi resta vuoto. Effettuiamo, allora, l'operazione di `ChangeKey()`, aggiornando le distanze. In questo caso, eliminando $a$, scopriamo sia $b$ che $c$ e modifichiamo la loro distanza.
- $S = \{a, c\}$
  $T = \{(a, c)\}$
  $Q = \{(7, b), (\infty, d), (3, e), (4, f), (\infty, g)\}$
  Estraiamo il minimo, in questo caso
  $c$, aggiungendolo ad $S$. Inoltre, aggiungiamo l'arco $(a, c)$ all'albero dei percorsi minimi e andiamo ad effettuare la solita operazione di `ChangeKey()`. In questo caso, possiamo arrivare ad $e$, facendo $2 + 1 = 3$ e anche ad $f$, facendo $2 + 2 = 4$
  .
- $S = \{a, c, e\}$
  $T = \{(a, c), (c, e)\}$
  $Q = \{(7, b), (\infty, d), (4, f), (6, g)\}$
  Estraiamo il minimo, in questo caso
  $e$, aggiungendolo ad $S$. Inoltre, aggiungiamo anche l'arco $(c, e)$ ai percorsi minimi. Con la `ChangeKey()`, aggiorniamo $g$, poiché possiamo arrivarci facendo $2 + 1 + 3 = 6$.
- $S = \{a, c, e, f\}$

$T = \{(a,c),(c,e),(c,f)\}$
$Q = \{(7,b),(7,d),(6,g)\}$
Estraiamo il minimo, in questo caso
$f$, aggiungendolo ad $S$. Inoltre, aggiungiamo l'arco $(c,f)$ a $T$. Con la `ChangeKey()`, aggiorniamo $d$, poiché possiamo arrivarci facendo $2+2+3=7$.

- $S = \{a,c,e,f,g\}$
  $T = \{(a,c),(c,e),(c,f),(e,g)\}$
  $Q = \{(7,b),(7,d)\}$
  Estraiamo il minimo, in questo caso
  $g$, aggiungendolo ad $S$. Inoltre, aggiungiamo l'arco $(e,g)$ a $T$. Con la `ChangeKey()`, non serve aggiornare nient'altro.
- $S = \{a,c,e,f,g,b\}$
  $T = \{(a,c),(c,e),(c,f),(e,g),(a,b)\}$
  $Q = \{(7,d)\}$
  In questo caso, avendo due pesi uguali, andiamo a prendere quello in ordine lessicografico, ossia
  $b$, inserendolo in $S$. Inoltre, aggiungiamo l'arco $(a,b)$ a $T$. Anche qui, con la `ChangeKey()` non serve modificare nient'altro.
- $S = \{a,c,e,f,g,b,d\}$
  $T = \{(a,c),(c,e),(c,f),(e,g),(a,b),(f,d)\}$
  $Q = \{\emptyset\}$
  Avendo un solo altro nodo nella coda, lo estraiamo, completando sia
  $S$ che $T$, uscendo dal `while` poiché la coda è vuota e restituendo, appunto $T$.

L'insieme $T$ richiesto è l'albero dei percorsi minimi, cioè contiene gli archi scelti per costruire i percorsi più brevi dalla sorgente $a$ ad ogni altro nodo.

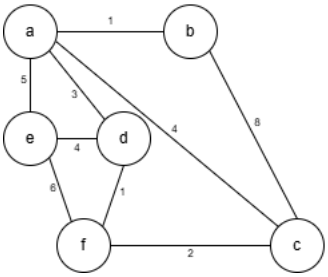
## Algoritmo di Prim per l'MST

### Possibile traccia

Generalmente, se all'esame viene dato un esercizio su Prim, in input viene fornito un grafo non orientato. La traccia potrebbe chiedere di dimostrare l'esecuzione dell'algoritmo di Prim su questo grafo, a partire da un nodo sorgente arbitrario, di solito $a$, mostrando, ad ogni passo, il contenuto dell'insieme $S$ dei nodi esplorati, della coda $Q$ e anche dell'albero $T$, che rappresenta, appunto, il nostro $MST$.

### Svolgimento + Esempio

Un'istanza di Prim si presenta generalmente così:




Quindi come un grafo non direzionato, con degli archi pesati. Come già anticipato, viene richiesto di mostrare il contenuto dell'insieme $S$, dell'albero $MST$ $T$ e della coda $Q$. Inizializziamo la nostra istanza:

- $S = \{\emptyset\}$
  $T = \{\emptyset\}$
  $Q = \{(\infty,a),(\infty,b),(\infty,c),(\infty,d),(\infty,e),(\infty,f)\}$

Ovviamente, all'inizio non avremo esplorato nessun nodo, quindi sia
$S$ che $T$ saranno vuoti. Nella coda, inseriamo tutti i nodi inizialmente con distanza infinita, senza, però, inizializzare la sorgente.

- $S = \{a\}$
  $T = \{\emptyset\}$
  $Q = \{(1, b), (4, c), (3, d), (5, e), (\infty, f)\}$
  La sorgente viene specificata dall'esercizio, in questo caso, quindi, estraiamo
  $a$ dalla coda, andando poi ad aggiornare le distanze dei nodi scoperti. Come possiamo ben vedere, $a$ è collegato a quasi tutti i nodi, quindi aggiorniamo.
- $S = \{a, b\}$
  $T = \{(a, b)\}$
  $Q = \{(4, c), (3, d), (5, e), (\infty, f)\}$
  Estraiamo il nodo di costo minimo, in questo caso
  $b$ e aggiungiamo all'insieme $S$ dei nodi esplorati, aggiungendo poi l'arco $(a, b)$ all'insieme $T$. Dobbiamo, ora, aggiornare i vicini di $b$. In questo caso, c'è solo $c$, ma da $b$ ci arrivo con costo $8$, quindi essendo $8 > 4$, non serve aggiornare ulteriormente.
- $S = \{a, b, d\}$
  $T = \{(a, b), (a, d)\}$
  $Q = \{(4, c), (4, e), (1, f)\}$
  Estraiamo il nodo di costo minimo, in questo caso
  $d$ e aggiungiamo all'insieme $S$ dei nodi esplorati, aggiungendo poi l'arco $(a, d)$ all'insieme $T$. Dobbiamo, ora, aggiornare i vicini di $d$. Innanzitutto, scopriamo $f$, andando ad aggiornarlo, mentre ora, arriviamo ad $e$ con $4$, $4 < 5$, quindi aggiorniamo anche $e$.
- $S = \{a, b, d, f\}$
  $T = \{(a, b), (a, d), (d, f)\}$
  $Q = \{(2, c), (4, e)\}$
  Estraiamo il nodo di costo minimo, in questo caso
  $f$ e aggiungiamo all'insieme $S$ dei nodi esplorati, aggiungendo poi l'arco $(d, f)$ all'insieme $T$. Dobbiamo, ora, aggiornare i vicini di $f$. In questo caso, non serve aggiornare $e$, poiché da $f$ ci arriviamo con costo $6$ e $6 > 4$, mentre invece a $c$ ci arriviamo con costo $2$ e $2 < 4$, quindi aggiorniamo.
- $S = \{a, b, d, f, c\}$
  $T = \{(a, b), (a, d), (d, f), (f, c)\}$
  $Q = \{(4, e)\}$
  Estraiamo il nodo di costo minimo, in questo caso
  $c$ e aggiungiamo all'insieme $S$ dei nodi esplorati, aggiungendo poi l'arco $(f, c)$ all'insieme $T$. Dobbiamo, ora, aggiornare i vicini di $c$. In questo caso, c non arriva ad e, quindi non serve aggiornare nulla
- $S = \{a, b, d, f, c, e\}$
  $T = \{(a, b), (a, d), (d, f), (f, c), (d, e)\}$
  $Q = \{\emptyset\}$
  Estraiamo l'ultimo nodo rimasto, ossia
  $e$ e aggiungiamo all'insieme $S$ dei nodi esplorati, aggiungendo poi l'arco $(d, e)$ all'insieme $T$. La coda è vuota, l'algoritmo termina.

L'insieme $T$ richiesto è l'albero $MST$, cioè un sottoinsieme di archi di un grafo che collega tutti i nodi senza formare cicli e con il minimo peso totale. In questo caso, il minimo peso totale risulta essere:

- $1 + 3 + 1 + 2 + 4 = 11$

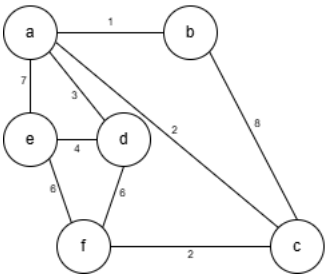
## Algoritmo di Kruskal

### Possibile traccia

Generalmente, se all'esame viene dato un esercizio su Kruskal, in input viene fornito un grafo non orientato. La traccia potrebbe chiedere di dimostrare l'esecuzione dell'algoritmo di Kruskal su questo grafo, mostrando ad ogni passo la foresta di alberi ottenuta.

### Svolgimento + Esempio

Un'istanza di Kruskal si presenta generalmente così:




La prima cosa da fare è ordinare gli archi in base ai costi.

- $(a, b) = 1$
  $(a, c) = 2$
  $(c, d) = 2$
  $(c, f) = 2$
  $(a, d) = 3$
  $(d, e) = 4$
  $(d, f) = 6$
  $(e, f) = 6$
  $(a, e) = 7$
  $(b, c) = 8$

Adesso, creiamo un singleton per ogni nodo del nostro grafo e indichiamo con $F$ la nostra foresta:

- $F = \{a\}, \{b\}, \{c\}, \{d\}, \{e\}, \{f\}$

Ora, in base all'ordine degli archi, aggiungiamo un arco all'$MST$, stando attenti a non creare cicli:

- $(a, b) = 1 \rightarrow$ non crea un ciclo, quindi lo aggiungiamo alla foresta e all'$MST$.

  $F = \{a, b\}, \{c\}, \{d\}, \{e\}, \{f\}$
  $T = \{(a, b)\}$
- $(a, c) = 2 \rightarrow$ non crea un ciclo, quindi aggiungiamolo alla foresta e all'albero.

  $F = \{a, b, c\}, \{d\}, \{e\}, \{f\}$
  $T = \{(a, b), (a, c)\}$
- $(c, d) = 2 \rightarrow$ non crea un ciclo, quindi aggiungiamolo alla foresta e all'albero.

  $F = \{a, b, c, d\}, \{e\}$
  $T = \{(a, b), (a, c), (c, d)\}$
- $(c, f) = 2 \rightarrow$ non crea un ciclo, quindi aggiungiamolo alla foresta e all'albero.

$F = \{a, b, c, d, f\}, \{e\}$
$T = \{(a, b), (a, c), (c, d), (c, f)\}$

- $(a, d) = 3 \rightarrow$ crea un ciclo, quindi lo scartiamo a rimaniamo tutto invariato.

  $F = \{a, b, c, d, f\}, \{e\}$
  $T = \{(a, b), (a, c), (c, d), (c, f)\}$

- $(d, e) = 4 \rightarrow$ non crea un ciclo, quindi aggiungiamolo alla foresta e all'albero.

  $F = \{a, b, c, d, e, f\}$
  $T = \{(a, b), (a, c), (c, d), (c, f), (d, e)\}$

Ora, possiamo fermarci, poiché tutti i nodi sono connessi e nell'$MST$ abbiamo esattamente $n - 1 = 5$ archi, per $n = 6$ nodi. Quindi restituiamo semplicemente il nostro $MST$ ottenuto:

- $T = \{(a, b), (a, c), (c, d), (c, f), (d, e)\}$