# Algoritmi e Programmazione
# Esercizi di teoria

1. **On-line connectivity: quickfind**
   Sia data la seguente sequenza di coppie, dove la relazione i - j indica che il nodo i è adiacente al nodo j:

   4-8, 7-3, 5-9, 2-9, 5-6, 8-0

   si applichi un algoritmo di on-line connectivity con quickfind, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale.

   **Soluzione:**

   configurazione iniziale

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   4-8

| 0 | 1 | 2 | 3 | 8 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   7-3

| 0 | 1 | 2 | 3 | 8 | 5 | 6 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   5-9

| 0 | 1 | 2 | 3 | 8 | 9 | 6 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   2-9

| 0 | 1 | 9 | 3 | 8 | 9 | 6 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

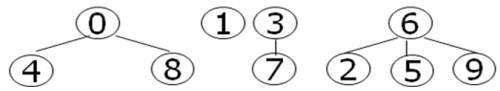
   5-6

| 0 | 1 | 6 | 3 | 8 | 6 | 6 | 3 | 8 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   8-0

| 0 | 1 | 6 | 3 | 0 | 6 | 6 | 3 | 0 | 6 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   rappresentazione ad albero della configurazione finale




2. **On-line connectivity: quickunion**
   Sia data la seguente sequenza di coppie, dove la relazione i - j indica che il nodo i è adiacente al nodo j:

   4-8, 7-3, 5-9, 9-4, 5-6

   si applichi un algoritmo di on-line connectivity con quickunion, riportando a ogni passo il contenuto del vettore e la foresta di alberi al passo finale.

   **Soluzione:**

   configurazione iniziale

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   4-8

| 0 | 1 | 2 | 3 | 8 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

   7-3

| 0 | 1 | 2 | 3 | 8 | 5 | 6 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

5-9

| 0 | 1 | 2 | 3 | 8 | 9 | 6 | 3 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

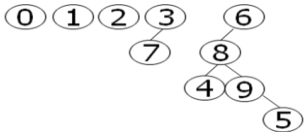
9-4

| 0 | 1 | 2 | 3 | 8 | 9 | 6 | 3 | 8 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

5-6

| 0 | 1 | 2 | 3 | 8 | 9 | 6 | 3 | 6 | 8 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

rappresentazione ad albero della configurazione finale




### 3. Equazioni alle ricorrenze con unfolding

Si risolva la seguente equazione alle ricorrenze mediante sviluppo (unfolding):

$$T(n) = 3T(n/2) + n^2 \qquad n \geq 2$$
$$T(1) = 1$$

**Soluzione:**

$T(n) = O(n^2)$

### 4. Insertion sort

- Si ordini in maniera discendente la seguente sequenza di interi mediante insertion sort, indicando i passaggi:

12 1 34 4 5 7 0 9 17 25 6 8 10

**Soluzione:**

| 12 | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | | | | | | | | | | | |
| 34 | 12 | 1 | | | | | | | | | | |
| 34 | 12 | 4 | 1 | | | | | | | | | |
| 34 | 12 | 5 | 4 | 1 | | | | | | | | |
| 34 | 12 | 7 | 5 | 4 | 1 | | | | | | | |
| 34 | 12 | 7 | 5 | 4 | 1 | 0 | | | | | | |
| 34 | 12 | 9 | 7 | 5 | 4 | 1 | 0 | | | | | |
| 34 | 17 | 12 | 9 | 7 | 5 | 4 | 1 | 0 | | | | |
| 34 | 25 | 17 | 12 | 9 | 7 | 5 | 4 | 1 | 0 | | | |
| 34 | 25 | 17 | 12 | 9 | 7 | 6 | 5 | 4 | 1 | 0 | | |
| 34 | 25 | 17 | 12 | 9 | 8 | 7 | 6 | 5 | 4 | 1 | 0 | |
| 34 | 25 | 17 | 12 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 1 | 0 |

- Si ordini in maniera discendente la seguente sequenza di interi mediante insertion sort, indicando i passaggi: 17 25 6 8 4 10 15 67

**Soluzione:**

| 17 | | | | | | | |
|---|---|---|---|---|---|---|---|
| 25 | 17 | | | | | | |
| 25 | 17 | 6 | | | | | |
| 25 | 17 | 8 | 6 | | | | |
| 25 | 17 | 8 | 6 | 4 | | | |
| 25 | 17 | 10 | 8 | 6 | 4 | | |
| 25 | 17 | 15 | 10 | 8 | 6 | 4 | |
| 67 | 25 | 17 | 15 | 10 | 8 | 6 | 4 |

## 5. Selection sort

Si ordini in maniera ascendente la seguente sequenza di interi mediante selection sort, indicando i passaggi:

12 1 34 4 5 7 0 9 17 25 6 8 10

**Soluzione**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 12 | 1 | 34 | 4 | 5 | 7 | 0 | 9 | 17 | 25 | 6 | 8 | 10 |
| 0 | 1 | 34 | 4 | 5 | 7 | 12 | 9 | 17 | 25 | 6 | 8 | 10 |
| 0 | 1 | 34 | 4 | 5 | 7 | 12 | 9 | 17 | 25 | 6 | 8 | 10 |
| 0 | 1 | 4 | 34 | 5 | 7 | 12 | 9 | 17 | 25 | 6 | 8 | 10 |

## 6. Counting sort

Si ordini in maniera ascendente mediante counting-sort il seguente vettore di interi, indicando i passaggi rilevanti: 7 2 1 2 0 4 2 5 2 3 1 7 7 0 1

**Soluzione:**

occorrenze semplici

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 3 | 4 | 1 | 1 | 1 | 0 | 3 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

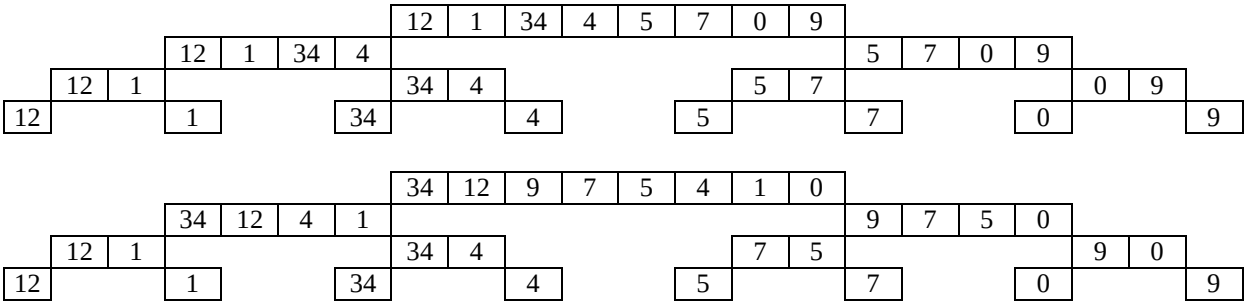
occorrenze multiple

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 2 | 5 | 9 | 10 | 11 | 12 | 12 | 15 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |

## 7. Merge sort

Si ordini in maniera discendente la seguente sequenza di interi mediante merge sort, indicando i passaggi rilevanti: 12 1 34 4 5 7 0 9

**Soluzione:**



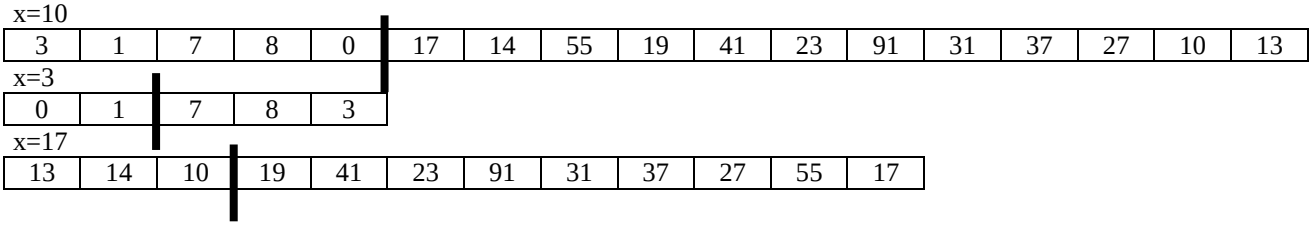

## 8. Quicksort

- Sia data la sequenza di interi, supposta memorizzata in un vettore:

  10 1 27 14 17 0 8 55 19 41 23 91 31 37 7 3 13

  Si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento ascendente**.** NB: I passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente. Si scelga come pivot l'elemento all'estremità sinistra del vettore considerato.

  **Soluzione:**




- Sia data la sequenza di interi, supposta memorizzata in un vettore:

44 24 29 43 17 35 74 28 61 41 114 48 53 58 40 32

Si eseguano i primi 2 passi dell'algoritmo di quicksort per ottenere un ordinamento ascendente**.** NB: I passi sono da intendersi, impropriamente, come in ampiezza sull'albero della ricorsione, non in profondità. Si chiede, pertanto, che siano ritornate le 2 partizioni del vettore originale e le due partizioni delle partizioni trovate al punto precedente. Si scelga come pivot l'elemento all'estremità sinistra del vettore considerato.

**Soluzione:**

x=44

| 32 | 24 | 29 | 43 | 17 | 35 | 40 | 28 | 41 | 61 | 114 | 48 | 53 | 58 | 75 | 44 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

x=32

| 28 | 24 | 29 | 17 | 43 | 35 | 40 | 32 | 41 |
|---|---|---|---|---|---|---|---|---|

x=61

| 44 | 58 | 48 | 53 | 114 | 75 | 61 |
|---|---|---|---|---|---|---|

## 9. Heap, heapsort

Sia data la sequenza di interi, supposta memorizzata in un vettore:

10 1 27 14 17 0 8 55 19 41 23 91 31 37 7 3 13

Si costruisca lo heap ad essa corrispondente. Si utilizzi un vettore come struttura dati. Si riportino graficamente i passi significativi della costruzione dello heap ed il risultato finale. Si ipotizzi che, alla fine, nella radice dello heap sia memorizzato il valore massimo.

**Soluzione:**

| 91 | 41 | 55 | 19 | 27 | 31 | 37 | 13 | 17 | 14 | 23 | 0 | 10 | 8 | 7 | 1 | 3 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

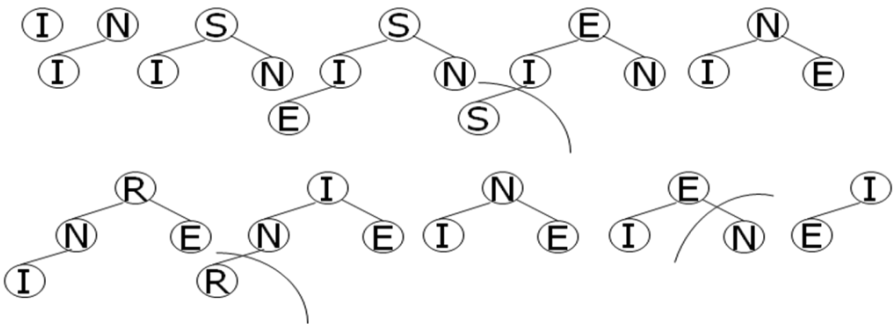
## 10. Code a priorità

- Sia data una coda a priorità inizialmente vuota implementata mediante uno heap. Sia data la sequenza di caratteri:

  I N S E * R * * T

  dove ad ogni lettera corrisponde un inserimento nella coda a priorità e al carattere * un'estrazione con cancellazione del massimo. Si ipotizzi che la priorità massima sia associata alla lettera Z e quella minima alla lettera A. Si riporti la configurazione della coda a priorità dopo ogni operazione e la sequenza dei valori restituiti dalle estrazioni con cancellazione del massimo. Nel caso di chiavi ripetute, le si distingua mediante pedice crescente (ad esempio E, E $\Rightarrow$ E1, E2; E1 < E2).

  **Soluzione**



- Si inserisca la seguente sequenza di interi in una coda a priorità:

  44 24 29 43 17 35 74 28 61 41 114 48 53 58 40 32

  Si ipotizzi di usare un heap come struttura dati. Si disegni la struttura ai diversi passi dell'inserzione. Si ipotizzi che la priorità massima sia associata alla chiave a valore minimo.

  **Soluzione:**

| 17 | 24 | 29 | 28 | 41 | 35 | 40 | 32 | 61 | 43 | 114 | 48 | 53 | 74 | 58 | 44 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

**11. Le pile (stack)**

Sia dato uno stack inizialmente vuoto implementato mediante un vettore. Sia data la sequenza di caratteri: Q U E S T O * E * * S T * A * * * C K * dove ad ogni lettera corrisponde un push nello stack e al carattere * un pop. Si riporti la configurazione dello stack dopo ogni operazione e la sequenza dei valori restituiti dalle operazioni pop.

**Soluzione:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | Q | | QUESTO | | QUESS | S | QUES |
| | QU | O | QUEST | | QUESST | S | QUE |
| | QUE | | QUESTE | T | QUESS | | QUEC |
| | QUES | E | QUEST | | QUESSA | | QUECK |
| | QUEST | T | QUES | A | QUESS | K | QUEC |

**12. Le code (queue)**

Sia data una coda FIFO inizialmente vuota implementata mediante un vettore circolare. Sia data la sequenza di caratteri:

C O D A T I * P * * O F I * F * * * O *

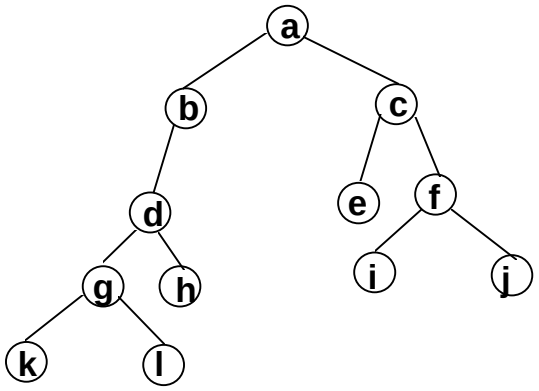
dove ad ogni lettera corrisponde un inserimento nella coda FIFO e al carattere * un'estrazione. Si riporti la configurazione della coda FIFO dopo ogni operazione e la sequenza dei valori restituiti dalle estrazioni.

**Soluzione:**

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | C | | CODATI | | ATIPO | T | IPOFIF |
| | CO | C | ODATI | | ATIPOF | I | POFIF |
| | COD | | ODATIP | | ATIPOFI | P | OFIF |
| | CODA | O | DATIP | A | TIPOFI | | OFIFO |
| | CODAT | D | ATIP | | TIPOFIF | O | FIFO |

**13. Attraversamenti di alberi binari in-ordine, pre-ordine, post-ordine, conversioni**

- Si visiti il seguente albero in preorder e postorder. Come output è richiesto un elenco di nodi.




**Soluzione:**

Preorder: a, b, d, g, k, l, h, c, e, f, i, j, postorder: k, l, g, h, d, b, e, i, j, f, c, a

- Si converta la seguente espressione in forma postfissa

  A * ( ( B * C ) + ( D * ( E + F ) ) )

  **Soluzione:** A B C * D E F + * + *

- Si trasformi la seguente espressione in forma infissa

  (((A+B) + C*(D+E)) + F) * (G+H)

  in forma:
  - prefissa
  - postfissa

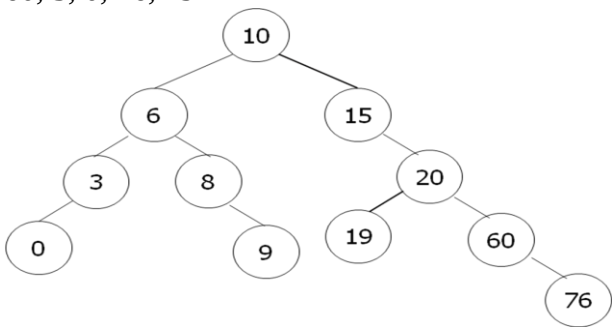
  visitando l'albero corrispondente. Sia l'output una stringa di caratteri.

**Soluzione:**

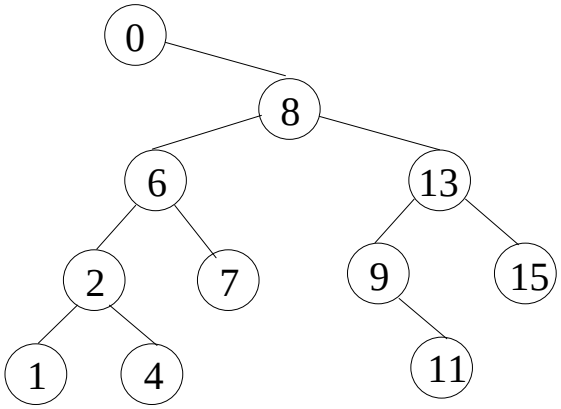
- prefissa: * + + + A B * C + D E F + G H
- postfissa: A B + C D E + * + F + G H + *

**14. Gli alberi binari di ricerca (BST)**

- Si supponga di aver memorizzato tutti i numeri compresi tra 1 e 1000 in un albero di ricerca binario e che si stia cercando il numero 363. Quali tra queste **non** possono essere le sequenze esaminate durante la ricerca?
  2, 252, 401, 398, 330, 344, 397, 363
  924, 220, 911, 244, 898, 258, 362, 363
  925, 202, 911, 240, 912, 245, 363
  2, 399, 387, 219, 266, 382, 381, 278, 363
  935, 278, 347, 621, 299, 392, 358, 363
  **Soluzione:** III e V sequenza
- Si inseriscano in sequenza le seguenti chiavi in un BST inizialmente vuoto: 10, 15, 6, 8, 9, 20, 60, 3, 0, 76, 19

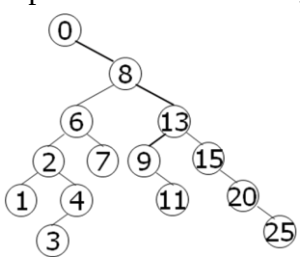



- Sia dato il seguente BST:

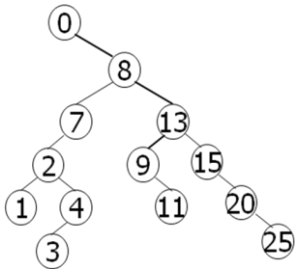



  - si eseguano sul BST di figura le operazioni in sequenza, dove + indica un'inserzione e – una cancellazione: +20, +25, +3, -6, -13, +19
  - si partizioni il BST di partenza attorno alla chiave mediana
  - si determini sul BST di partenza la nona chiave più piccola (k=8)
  - si determino sul BST di partenza il successore di 4 e il predecessore di 9
  - si inserisca in radice sul BST di partenza la chiave 3.

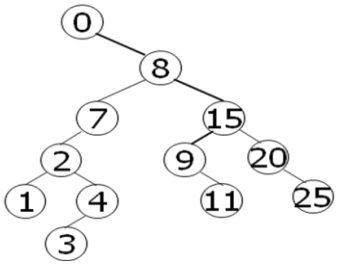
  **Soluzione**:
  dopo le inserzioni di 20, 25, 3



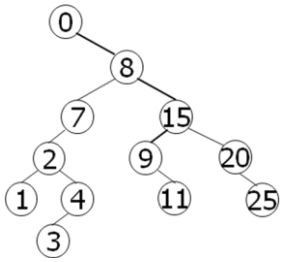
dopo la cancellazione di 6



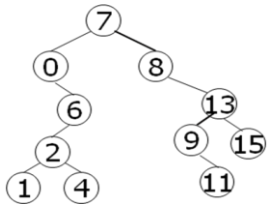

dopo la cancellazione di 13



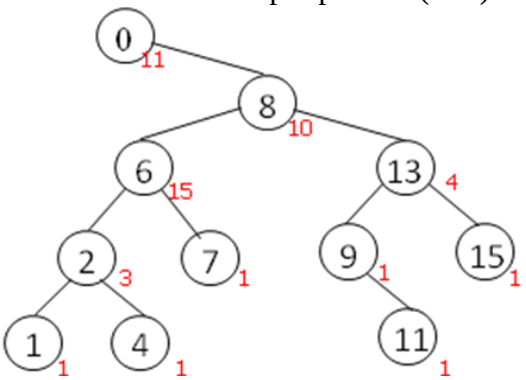

dopo l'inserzione di 19




partizionamento attorno alla chiave mediana (7)

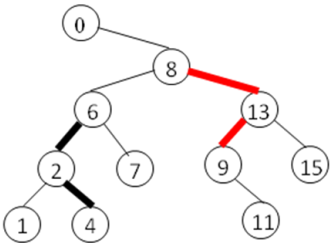



cerco la nona chiave più piccola (k=8)

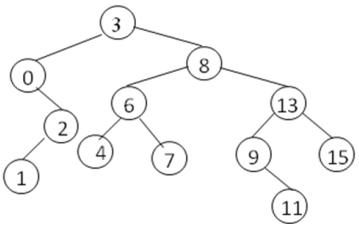



Dalla radice guardo il figlio SX, t=0<8, scendo a DX, cerco la k -t-1-ma chiave più piccola (7), guardo il figlio SX, t=5, 5<7, scendo a DX, cerco la k -t-1-ma chiave più piccola (1), guardo il figlio SX, t=2>1, scendo a SX con k=1, gurado il figlio SX, t=0>1, scendo a DX e trovo t=K, la chiave è 11

succ di 4 e pred di 9




inserimento di 3 in radice



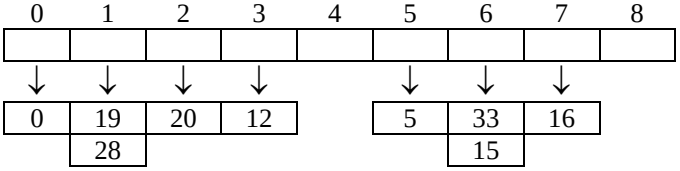
# Algoritmi e Programmazione
# Esercizi di teoria

## 1. Le tabelle di hash

### Linear chaining

- (**1 punto**) Sia data la sequenza di chiavi intere 5, 28, 19, 15, 20, 33, 12, 16, 10. Si riporti la struttura di una tabella di hash di dimensione 9, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza di cui sopra. Si supponga di utilizzare il linear chaining. Si definisca un'opportuna funzione di hash.

  **Soluzione:** h(k) = k mod 9




### Open Addressing con linear probing

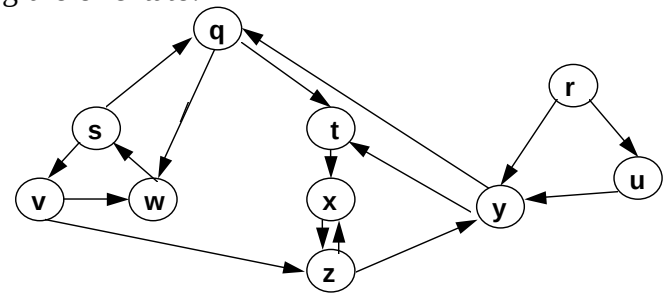
- (**2 punti**) Sia data la sequenza di chiavi EASYQUESTION, dove ciascun carattere è individuato dal suo ordine progressivo nell'alfabeto (A=1, ...., Z=26). Si riporti la struttura di una tabella di hash di dimensione 13, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza di cui sopra. Si supponga di utilizzare l'open addressing con linear probing e che la funzione di hash sia $h(k) = k \bmod 13$.

### Open Addressing con double hashing

- (**2 punti**) Sia data la sequenza di chiavi ADFKKADGCADH, dove ciascun carattere è individuato dal suo ordine progressivo nell'alfabeto (A=1, ...., Z=26). Si riporti la struttura di una tabella di hash di dimensione 19, inizialmente supposta vuota, in cui avvenga l'inserimento della sequenza di cui sopra. Si supponga di utilizzare l'open addressing con il double hashing scegliendo opportunamente la funzione di hash.

## 2. Gli algoritmi di visita dei grafi in profondità, ampiezza, etichettatura archi, componenti fortemente connesse

- Sia dato il seguente grafo orientato:



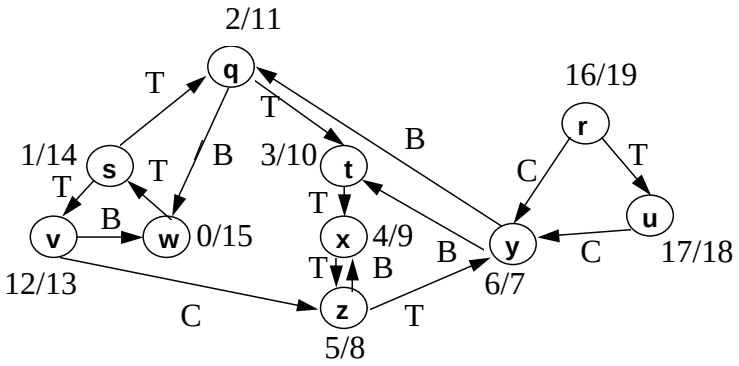

Considerando **w** come vertice di partenza:

- se ne effettui una visita in profondità,. Si elenchino i nodi nell'ordine risultante dalla visita e si indichino per ognuno di essi i tempi di scoperta e di fine elaborazione nel formato tempo1/tempo2 (**2 punti**);
- lo si ridisegni, etichettando ogni suo arco come **T** (tree), **B** (back), **F** (forward), **C** (cross) (**1.5 punti**);
- se ne determinino le componenti fortemente connesse (**2 punti**)
- se ne effettui una visita in ampiezza (**1 punto**)

Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.

**Soluzione:**

BFS: w | s | v q | z t | x y
SCC1 = {s, v, w, q, t, x, z, y}
SCC2 = {r} SCC3 = {u}



Sia dato il seguente grafo non orientato rappresentato come lista delle adiacenze:

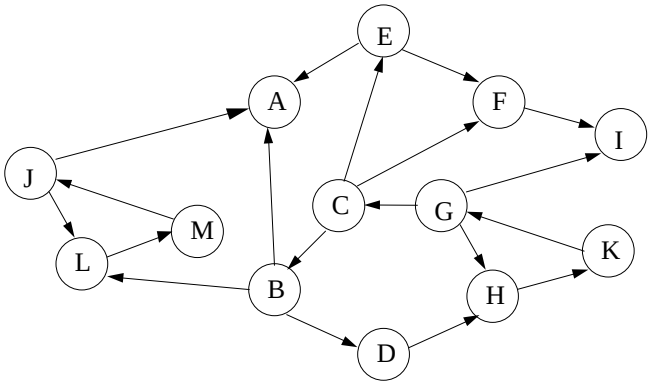
**A B C F G H**
**B A I**
**C A I**
**D E F**
**E D F G**
**F A D E**
**G A E**
**H A I**
**I B C H**
**J K**
**K J**

- se ne effettui una visita in profondità, considerando **A** come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico. Si etichettino indicando per ognuno di essi i tempi di scoperta e di processamento nel formato tempo1/tempo2.
- lo si ridisegni, etichettando ogni suo arco come **T** (tree) o **B** (back), considerando **A** come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico.
- se ne determinino i punti di articolazione. Si consideri **A** come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico.
- se ne effettui una visita in ampiezza, considerando A come vertice di partenza. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico. Si elenchino i nodi nell'ordine risultante dalla visita.

**Soluzione:**

- A 0/19, B 1/8, C 3/4, D 10/15, E 11/14, F 9/16, G 12/13, H 5/6, I 2/7, J 18/21, K 19/20
- tutti gli archi sono T, tranne AH, AG, AC, FE che sono B
- punto di articolazione: A
- BFS: A | B C F G H | I D E

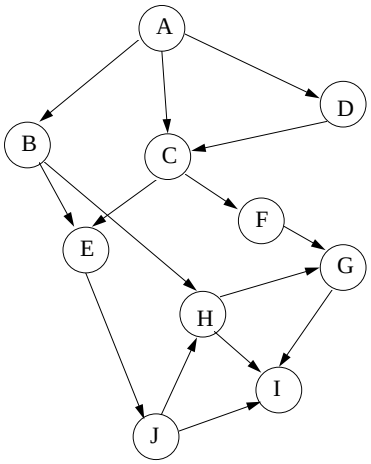
- Sia dato il seguente grafo orientato: si determinino le componenti fortemente connesse del seguente grafo orientato. Si consideri **A** come vertice di partenza **(2 punti)**. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.




**Soluzione :**

SCC1: {I} SCC2: {F} SCC3: {A} SCC4: {J, L, M} SCC5: {E} SCC6: {B, C, D, G, H, K}

- Sia dato il seguente grafo orientato:

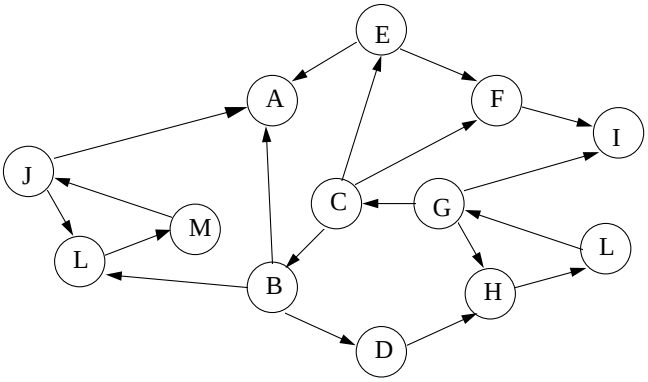



Considerando **A** come vertice di partenza:

- se ne effettui una visita in profondità,. Si elenchino i nodi nell'ordine risultante dalla visita e si indichino per ognuno di essi i tempi di scoperta e di fine elaborazione nel formato tempo1/tempo2 (2 punti);
- lo si ridisegni, etichettando ogni suo arco come **T** (tree), **B** (back), **F** (forward), **C** (cross) (1.5 punti);

Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.

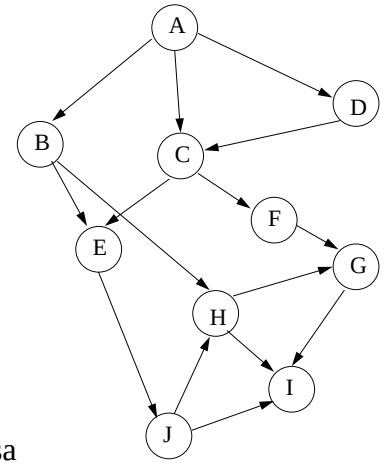
- **(2 punti)** Sia dato il seguente grafo orientato. Lo si visiti in profondità considerando **A** come vertice di partenza. Si elenchino i nodi nell'ordine risultante dalla visita e si indichino per ognuno di essi i tempi di scoperta e di fine elaborazione nel formato tempo1/tempo2. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.




**Soluzione:**
A 1/2; B 3/26; C 8/15; D 4/19; E 9/14; F 10/13; G 7/16; H 5/18; I 11/12; L 6/17, J 22/23; L 20/25; M 21/24

**3. DAG: ordinamento topologico**

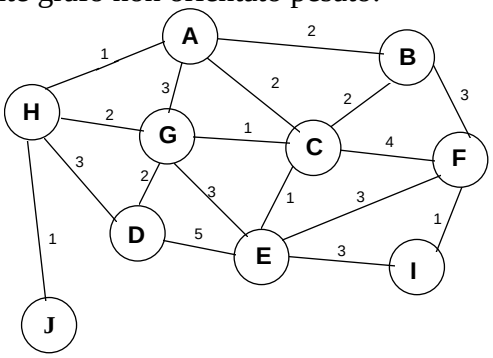
(**1.5 punti**) Si ordini topologicamente il seguente DAG. Qualora necessario, si trattino i vertici secondo l'ordine alfabetico e si assuma che la lista delle adiacenze sia anch'essa ordinata alfabeticamente.




## 4. Gli alberi ricoprenti minimi

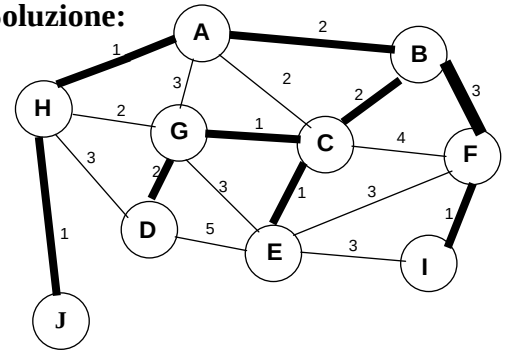
### L'algoritmo di Kruskal

- (**2 punti**) Sia dato il seguente grafo non orientato pesato:



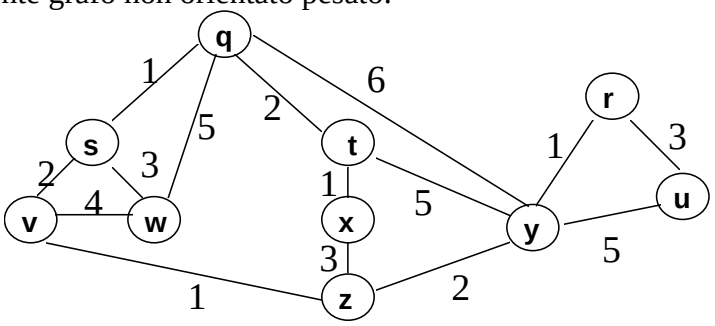

se ne determini un minimum spanning tree, ritornando come risultato l'albero ed il valore del peso minimo applicando l'algoritmo di Kruskal.

**Soluzione:**



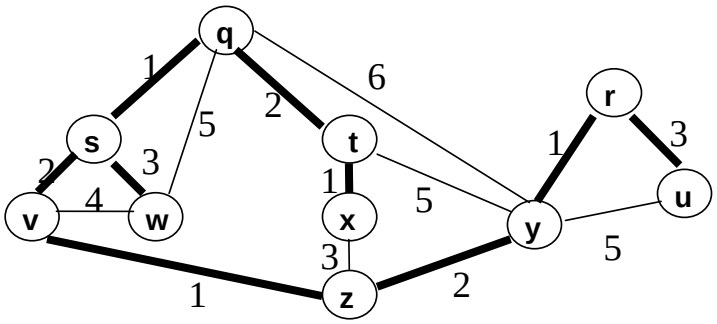

### L'algoritmo di Prim

- **(2 punti)** Sia dato il seguente grafo non orientato pesato:



se ne determini un minimum spanning tree, ritornando come risultato l'albero ed il valore del peso minimo applicando l'algoritmo di Prim, a partire dal vertice **q**.

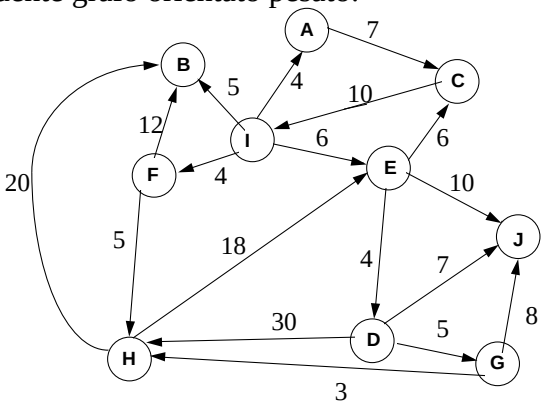
**Soluzione:**




## 5. I cammini minimi

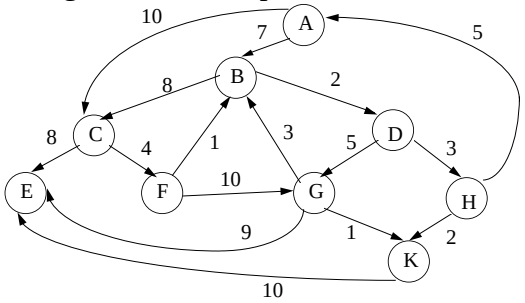
### L'algoritmo di Dijkstra

- **(2.5 punti)** Sia dato il seguente grafo orientato pesato:




si determinino i valori di tutti i cammini minimi che collegano il vertice **A** con ogni altro vertice mediante l'algoritmo di Dijkstra . Si assuma, qualora necessario, un ordine alfabetico per i vertici e gli archi

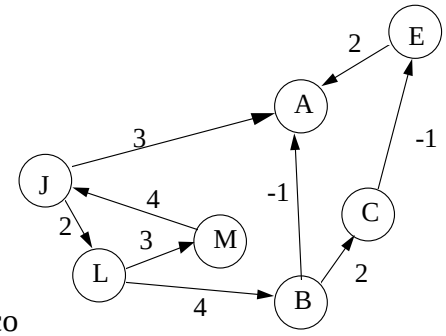
- **(2.5 punti)** Sia dato il seguente grafo orientato pesato:




si determinino i valori di tutti i cammini minimi che collegano il vertice **A** con ogni altro vertice mediante l'algoritmo di Dijkstra. Si assuma, qualora necessario, un ordine alfabetico per i vertici e gli archi.

### L'algoritmo di Bellman-Ford

- **(2.5 punti)** Si determinino per il seguente grafo orientato pesato mediante l'algoritmo di Bellman-Ford i valori di tutti i cammini minimi che collegano il vertice **J** con ogni altro vertice. Si assuma, qualora necessario, un ordine

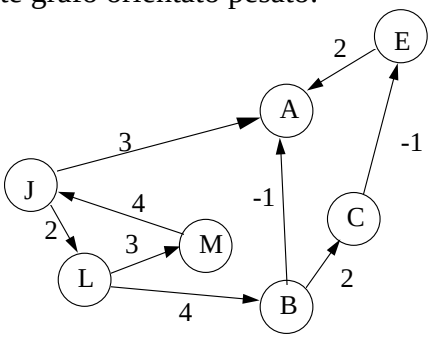



alfabetico per i vertici e gli archi.

**Soluzione:**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | ∞ | 3 | 3 | E | ∞ | ∞ | 6 | M | ∞ | 5 | 5 |
| B | ∞ | 6 | 6 | J | 0 | 0 | 0 | | | | |
| C | ∞ | ∞ | 7 | L | ∞ | 2 | 2 | | | | |

- **(2.5 punti)** Sia dato il seguente grafo orientato pesato:




si determinino i valori di tutti i cammini minimi che collegano il vertice **J** con ogni altro vertice mediante l'algoritmo di Bellman-Ford. Si assuma, qualora necessario, un ordine alfabetico per i vertici e gli archi.

**Soluzione:**

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| A | ∞ | 3 | 3 | E | ∞ | ∞ | 6 | M | ∞ | 5 | 5 |
| B | ∞ | 6 | 6 | J | 0 | 0 | 0 | | | | |
| C | ∞ | ∞ | 7 | L | ∞ | 2 | 2 | | | | |