Progettazione di Algoritmi	**Anno Accademico 2021–2022**

Esercizi

Ugo Vaccaro

Esercizi su Grafi: Parte Prima

Importante: Si ricorda che ogni algoritmo và accompagnato da una argomentazione sul perchè calcola correttamente l'output e da un'analisi della sua complessità di tempo. Inoltre, si possono usare algoritmi visti a lezione (come BFS, DFS, etc.) senza necessariamente riportare il relativo pseudocodice, purchè lo si menzioni esplicitamente. In generale, di ogni algoritmo è preferibile presentare il relativo pseudocodice. Tuttavia, anche una sola descrizione a parole dell'idea dell'algoritmo (purchè precisa e corretta) verrà accettata all'esame.

1. *Esercizio*: Progettare ed analizzare un algoritmo che, prendendo in input un grafo non orientato $G = (V, E)$, determina se G contiene o meno cicli. La complessità dell'algoritmo deve essere $O(|V|)$, *indipendentemente* da $|E|$.

$\diamond$

2. *Esercizio*: Dato un grafo non diretto $G = (V, E)$, e tre vertici $u, v, w \in V$, descrivere ed analizzare un algoritmo che ritorna `True` se esiste un cammino in G che parte da u ed arriva a v, passando per w, ritorna `False` altrimenti. Il cammino può anche contenere vertici ripetuti.

$\diamond$

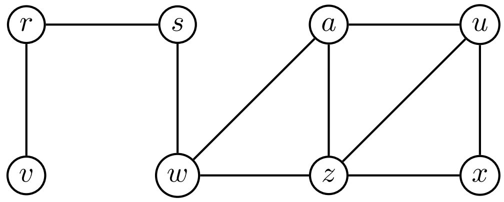
3. *Esercizio*: Dato un grafo diretto $G = (V, E)$, e due vertici $u, v \in V$, descrivere ed analizzare un algoritmo che ritorna `True` se esiste un cammino in G che parte da u ed arriva a v, e contemporaneamente esiste un cammino in G che parte da v ed arriva a u, ritorna `False` altrimenti.

$\diamond$

4. *Esercizio*: Sia $G = (V, E)$ un grafo diretto, $V = \{1, 2, \ldots, n\}$, rappresentato mediante la sua matrice delle adiacenze $A = [a_{ij}]$, in cui $a_{ij} = 1$ se e solo se esiste un arco *diretto* dal vertice i al vertice j. Diremo che un vertice $u \in V$ è un *pozzo* se u non ha archi uscenti da esso ed ha $n - 1$ archi entranti. Determinare in tempo $O(n)$ se G ha un pozzo o meno.

$\diamond$

5. *Esercizio*: Dato il grafo G rappresentato in figura



eseguire su di esso l'algoritmo `BFS`(G, s), indicando ad ogni passo dell'algoritmo sia l'arco considerato dall'algoritmo che la composizione della coda Q. Si calcoli inoltre sia l'albero BFS risultante che i valori $d[c]$ per ogni nodo c del grafo.

◇

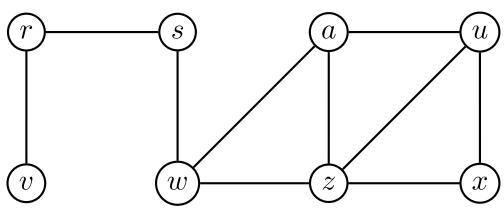
6. *Esercizio*: Progettare ed analizzare un algoritmo per il seguente problema:

   Input: Un grafo non diretto $G = (V, E)$.

   Output: "Sì" se esiste un arco $e \in E$ che si può rimuovere da G, in modo tale che $G' = (V, E - \{e\})$ sia connesso, "No", altrimenti.

◇

7. *Esercizio*: Dato il grafo G rappresentato in figura

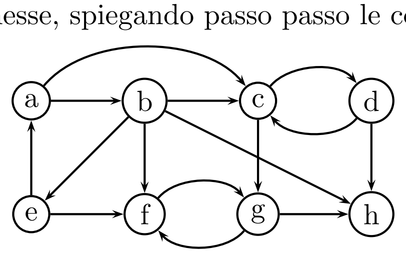


   eseguire su di esso l'algoritmo `DFS`(G, s), indicando ad ogni passo dell'algoritmo sia l'arco considerato dall'algoritmo che la composizione dello stack Q. Si calcoli inoltre sia l'albero DFS risultante.

◇

8. *Esercizio*:

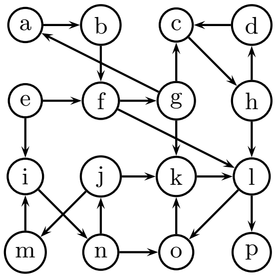
   Dato il grafo rappresentato in figura, eseguire su di esso l'algoritmo per il calcolo delle componenti fortementi connesse, spiegando passo passo le computazioni fatte.



◇

9. *Esercizio*:

   Dato il grafo rappresentato in figura, eseguire su di esso l'algoritmo per il calcolo delle componenti fortementi connesse, spiegando passo passo le computazioni fatte.

◇

10. *Esercizio*: Dato un grafo $G = (V, E)$ non diretto, il suo diametro $D(G)$ è definito come la distanza tra i due vertici di G massimalmente distanti, se G è connesso, è definito come ∞ altrimenti. Formalmente:

$$D(G) = \begin{cases} \infty & \text{se } G \text{ non è connesso} \\ \max_{u,v \in V} d(u,v) & \text{altrimenti} \end{cases}$$

dove $d(u, v)$ è la distanza tra i due vertici u e v. Analogamente, l'eccentricità di un vertice u è definita come

$$e(u) = \begin{cases} \infty & \text{se } G \text{ non è connesso} \\ \max_{v \in V} d(u,v) & \text{altrimenti} \end{cases}$$

mentre il raggio $R(G)$ di G è definito come $R(G) = \min_{u \in V} e(u)$.

Si dia un algoritmo che, dato in input un grafo $G = (V, E)$ non diretto, calcoli $D(G)$ e $R(G)$.

◇

11. *Esercizio*: Data la rappresentazione mediante liste di adiacenza di un grafo *diretto* $G = (V, E)$, scrivere un algoritmo che determini il numero di archi uscenti da ogni vertice ed un separato algoritmo per determinare il numero di archi entranti in ogni vertice.

◇

12. *Esercizio*: Il quadrato di un grafo orientato $G = (V, E)$ è il grafo $G^2 = (V, E^2)$ tale che $(u, w) \in E^2$ se e solo se $\exists v : (u, v) \in E \wedge (v, w) \in E$. In altre parole, se esiste un percorso di due archi fra i nodi u e w. Scrivere un algoritmo che, dato un grafo G rappresentato con matrice di adiacenza, restituisce il grafo G^2.

◇

13. *Esercizio*: Descrivere un algoritmo per il seguente problema. Dato in input un grafo orientato $G = (V, E)$ rappresentato tramite liste di adiacenza e un nodo $s \in V$, restituire il numero dei nodi in V raggiungibili da s che si trovano alla massima distanza da s.

◇

14. *Esercizio*: Descrivere un algoritmo per il seguente problema:

    Input: un grafo diretto $G = (V, E)$ rappresentato mediante liste di adiacenza.

    Output: per ogni nodo $v \in V$ il numero dei nodi w raggiungibili da v (ovvero il numero dei nodi $w \in V$ per i quali esiste un cammino orientato da v a w).

$\diamond$

15. *Esercizio*: Descrivere un algoritmo per il seguente problema:

    Input: un grafo diretto $G = (V, E)$ rappresentato mediante liste di adiacenza, un nodo $v \in V$ ed un intero k

    Output: il numero di nodi del grafo a distanza $\leq k$ da v.

$\diamond$

16. *Esercizio*: Descrivere un algoritmo per il seguente problema:

    Input: un grafo non diretto $G = (V, E)$ rappresentato mediante liste di adiacenza

    Output: il numero di componenti connesse di G.

$\diamond$

17. *Esercizio*: Dato un grafo non diretto $G = (V, E)$, un triangolo in esso è una tripla di vertici $a, b, c \in V$ tali che $(a, b), (b, c), (c, a) \in E$. Descrivere un algoritmo per il seguente problema:

    Input: un grafo non diretto $G = (V, E)$ rappresentato mediante matrice di adiacenza;
    Output: il numero di triangoli in G.

$\diamond$

18. *Esercizio*: Dato un grafo non diretto $G = (V, E)$, con $V = \{1, \ldots, n\}$ ed un array di interi $A[1 \ldots n]$, G è detto ben colorato se per ogni $\{i, j\} \in E$ vale che $A[i] \neq A[j]$. Si descriva un algoritmo per il seguente problema:

    Input: $G = (V, E)$ rappresentato mediante matrice di adiacenza, vettore A
    Output: "SI", se G è ben colorato, "NO", altrimenti.

    Si risolva l'esercizio anche sotto l'ipotesi che di G sia rappresentato mediante liste di adiacenze.

$\diamond$

19. *Esercizio*: Dato un grafo non diretto $G = (V, E)$, con $V = \{1, \ldots, n\}$ ed un sottoinsieme $S \subseteq V$, S è detto un VERTEX COVER per G se per ogni $\{i, j\} \in E$ vale $\{i, j\} \cap S \neq \emptyset$. Si descriva un algoritmo per il seguente problema:

    Input: $G = (V, E)$ rappresentato mediante matrice di adiacenza, sottoinsieme $S \subseteq V$
    Output: "SI", se S è un VERTEX COVER "NO", altrimenti.

    Si risolva l'esercizio anche sotto l'ipotesi che di G sia rappresentato mediante liste di adiacenze.

$\diamond$

20. *Esercizio*: Descrivere un algoritmo per il seguente problema:

    Input: un grafo diretto $G = (V, E)$ rappresentato mediante liste di adiacenza, due nodi $v, w \in V$.

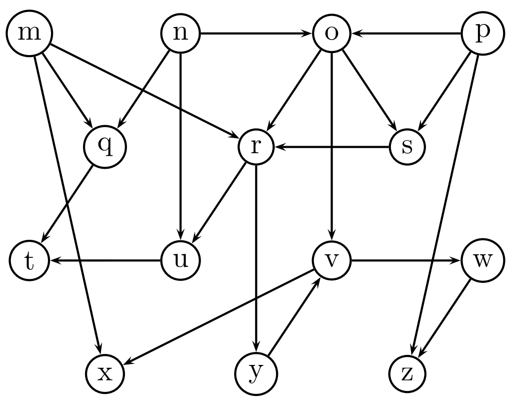
    Output: Il numero di nodi raggiungibili da v che si trovano alla stessa distanza sia da v che da w.

◇

21. *Esercizio*: Si descriva l'algoritmo per il calcolo dell'ordinamento topologico di un DAG, argomentando la correttezza e valutandone la complessità di tempo Giustificare con *precisione* le relative affermazioni effettuate.

◇

22. *Esercizio*: Dato il grafo rappresentato in figura



determinarne un ordinamento topologico, applicando passo passo l'algoritmo visto a lezione.

◇

23. *Esercizio*: Si provino le seguenti affermazioni:

    (a) Se G ammette un ordinamento topologico allora G è necessariamente senza cicli

    (b) Se G è un DAG allora ha un vertice senza archi entranti.

◇

24. *Esercizio*: Siano $C_1, \ldots, C_k$ le componenti fortemente connesse di un grafo diretto $G = (V, E)$. Dato il grafo $G' = (V', E')$, dove $V' = \{C_1, \ldots, C_k\}$ e c'è un arco diretto (C_i, C_j) dal "vertice" C_i al "vertice" C_j se e solo se, *nel grafo* $G = (V, E)$ esiste un qualche arco della forma (u, v), dove $u \in C_i$ e $v \in C_j$.

    Si provi che $G' = (V', E')$ è un DAG, ovvero non ha cicli.

Esercizi su Code a Prioritá

25. *Esercizio*: Sia dato un Max-Heap rappresentato da un array $A[1 \dots n]$. Si scriva un procedura (*in pseudocodice, per favore*) che, in tempo $O(\log n)$, avendo in input A ed un indice i, $1 \le i \le n$, cancelli l'elemento $A[i]$ dallo heap. L'array in output dovrá ancora essere un Max-Heap. Si illustri la esecuzione dell'algoritmo sul seguente heap, con $i = 2$ (*L'algoritmo puó usare, al suo interno, chiamata a procedure note viste a lezione, purché se ne spieghi chiaramente l'utilizzo e la funzione*. Come suggerimento, si proceda in modo analogo alla cancellazione dell'elemento nella radice dello Heap, così come visto a lezione).

1	2	3	4	5	6	7	8	9	10
23	18	19	16	15	10	9	7	6	2

◇

26. *Esercizio*: Si dia la rappresentazione mediante albero del Min-Heap:

    $A = [1, 4, 5, 9, 8, 7, 6, 10, 11, 15, 12, 13]$.

    Si ridisegni sia l'albero che il vettore corrispondente dopo l'esecuzione delle seguenti operazioni: (a) Heap-Extract-Min(A), seguita da (b) Heap-Insert($A, 0$)

◇

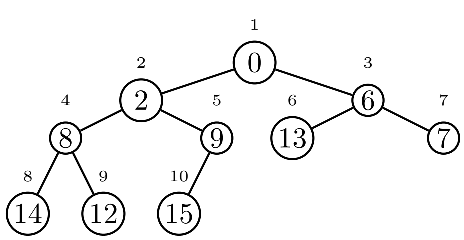
27. *Esercizio*: Si assuma che n numeri diversi tra di loro siano memorizzati in un Min-Heap. Si dia un algoritmo che in tempo $O(1)$ trovi il terzo elemento piú piccolo di A. *Sotto l'ipotesi che i sia costante*, si dica come l'i-esimo elemento piú piccolo di A possa essere trovato in tempo $O(1)$.

◇

28. *Esercizio*: Dato un array di n numeri $A[1 \dots n]$, si presenti un algoritmo che, avendo in input l'array A, ritorni True se A rappresenta un Min-Heap, ritorni il piú piccolo indice i su cui la proprietá dello heap é violata, altrimenti. Si valuti la complessitá dell'algoritmo proposto.

◇

29. *Esercizio*: Si esegua l'algoritmo HeapSort(A) sullo heap



descrivendo esplicitamente tutti i passi dell'algoritmo.

◇

30. *Esercizio*: Dato un array A di n elementi, tutti distinti tra di loro, rappresentante un Max-Heap.

   (a) In che posizione di A compare l'elemento massimo?

   (b) In quali posizioni di A può comparire il secondo massimo?

   (c) Per $k = 3, 4$, in che posizioni di A può comparire il k-esmo elemento piú grande di A?

   (d) In che posizioni di A può comparire l'elemento minimo di A?

   Giustificare le risposte.