Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 22 giugno 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `22 24 23 12 17 25 16`

(a) Disegnate l'albero AVL

(b) Disegnate l'albero 2-3

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato* dell'albero AVL

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL

(f) L'albero AVL ottenuto al punto (a) è perfettamente bilanciato?

(g) Se avete risposto NO al punto (f), disegnate un albero di ricerca *perfettamente bilanciato* contenente gli stessi numeri

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`22 24 23 12 17 25 16`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Sia $G = (V, E)$ un *grafo orientato* con una funzione peso $\omega : E \to \mathbb{N}$ che associa ad ogni arco un peso non negativo. La *distanza* di un vertice v da un vertice u è il minimo tra i pesi di tutti i cammini che iniziano nel vertice u e terminano nel vertice v, dove il peso di un cammino è dato dalla somma dei pesi degli archi che lo costituiscono.

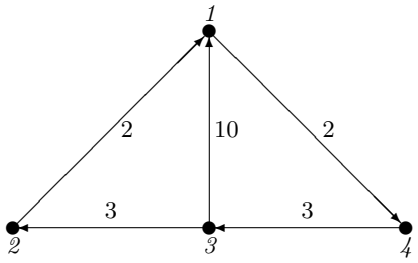
Si vuole progettare un algoritmo che ricevendo in ingresso un grafo orientato G con una funzione peso ω determini:

- il valore massimo della distanza tra due vertici del grafo,
- per ogni vertice u del grafo, il vertice v più distante (o uno dei vertici più distanti) da u.

Si supponga che i vertici siano indicati con gli interi *1,2,...,n*.

Esempio 1. Per il grafo rappresentato nella figura a destra si ha:

- massima distanza tra due vertici: 8
- vertice più distante da *1*: *2*
- vertice più distante da *2*: *3*
- vertice più distante da *3*: *4*
- vertice più distante da *4*: *1*



Svolgete i seguenti punti ***nell'ordine indicato.***

(a) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice un algoritmo che risolva il problema, *adattando uno degli algoritmi presentati nel corso* (indicate esplicitamente quale algoritmo utilizzate).

(b) Fornite una stima del tempo di calcolo dell'algoritmo ottenuto, in funzione del numero n di vertici del grafo in ingresso, giustificando la risposta.

(c) Nel caso il grafo in ingresso non sia fortemente connesso, il valore massimo della distanza tra due vertici sarà $+\infty$. Indicate come si possa modificare l'algoritmo descritto al punto (a) in modo che fornisca in output anche:

- l'elenco di tutti i vertici che appartengono alla stessa componente fortemente connessa del vertice *1*,
- il valore massimo della distanza tra due vertici della componente fortemente connessa del vertice *1*.

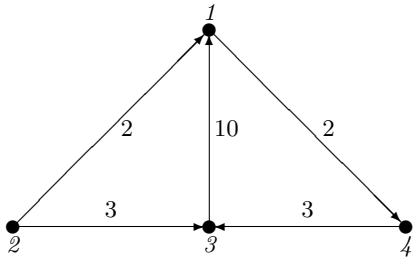
Indicate inoltre se, asintoticamente, il tempo di calcolo cambia nell'algoritmo così modificato rispetto al tempo stimato al punto (b).

Note

- Nella descrizione e nello pseudocodice dell'algoritmo potete utilizzare direttamente il valore $+\infty$.

Esempio 2. Per il grafo rappresentato nella figura a destra si ha:

- vertici della componente fortemente connessa che contiene *1*: *1, 3, 4*
- massima distanza tra due vertici della componente fortemente connessa che contiene *1*: 13



Cognome.....

Nome.....

Matricola.....

Algoritmi e Strutture Dati

Prova scritta del 22 giugno 2018

TEMPO DISPONIBILE: 2 ore

Le risposte agli esercizi 1 e 2 devono essere scritte negli appositi riquadri su questo foglio (risposte scritte su altri fogli non saranno considerate). La soluzione dell'esercizio 3 va scritta su uno dei fogli di protocollo forniti. Le brutte copie NON devono essere consegnate. *Ricordatevi di scrivere cognome e nome su TUTTI i fogli.*

1. Considerate un albero AVL e un albero 2-3 ottenuti inserendo uno dopo l'altro, nell'ordine indicato, i seguenti numeri a partire da alberi inizialmente vuoti: `20 22 21 10 15 23 14`

(a) Disegnate l'albero AVL

(b) Disegnate l'albero 2-3

(c) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ampiezza* dell'albero AVL

(d) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine anticipato* dell'albero AVL

(e) Scrivete l'elenco dei valori dei nodi ottenuto mediante la visita in *ordine posticipato* dell'albero AVL

(f) L'albero AVL ottenuto al punto (a) è perfettamente bilanciato?

(g) Se avete risposto NO al punto (f), disegnate un albero di ricerca *perfettamente bilanciato* contenente gli stessi numeri

2. La seguente sequenza di numeri, memorizzata in un array, deve essere ordinata in modo crescente:
`20 22 21 10 15 23 14`

(a) Supponete di ordinare la sequenza mediante l'algoritmo `quickSort`, scegliendo come perno il primo elemento. Indicate il contenuto dell'array dopo avere effettuato la partizione, prima delle chiamate ricorsive di `quickSort`.

(b) Supponete di ordinare la sequenza mediante l'algoritmo `heapSort`. Indicate il contenuto dell'array dopo averlo trasformato in uno heap.

(c) Supponete di ordinare la sequenza mediante l'algoritmo `bubbleSort`. Indicate il contenuto dell'array dopo la prima iterazione del ciclo principale dell'algoritmo.

3. Sia $G = (V, E)$ un *grafo orientato* con una funzione peso $\omega : E \to \mathbb{N}$ che associa ad ogni arco un peso non negativo. La *distanza* di un vertice v da un vertice u è il minimo tra i pesi di tutti i cammini che iniziano nel vertice u e terminano nel vertice v, dove il peso di un cammino è dato dalla somma dei pesi degli archi che lo costituiscono.

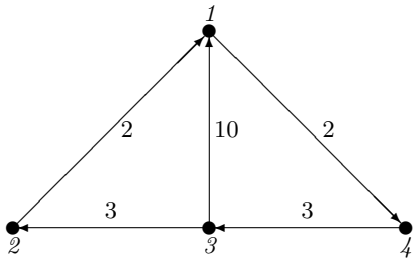
Si vuole progettare un algoritmo che ricevendo in ingresso un grafo orientato G con una funzione peso ω determini:

- il valore massimo della distanza tra due vertici del grafo,
- per ogni vertice u del grafo, il vertice v più distante (o uno dei vertici più distanti) da u.

Si supponga che i vertici siano indicati con gli interi *1,2,...,n*.

Esempio 1. Per il grafo rappresentato nella figura a destra si ha:

- massima distanza tra due vertici: 8
- vertice più distante da *1*: *2*
- vertice più distante da *2*: *3*
- vertice più distante da *3*: *4*
- vertice più distante da *4*: *1*



Svolgete i seguenti punti ***nell'ordine indicato.***

(a) Descrivete *sinteticamente a parole* e poi *ad alto livello* in pseudocodice un algoritmo che risolva il problema, *adattando uno degli algoritmi presentati nel corso* (indicate esplicitamente quale algoritmo utilizzate).

(b) Fornite una stima del tempo di calcolo dell'algoritmo ottenuto, in funzione del numero n di vertici del grafo in ingresso, giustificando la risposta.

(c) Nel caso il grafo in ingresso non sia fortemente connesso, il valore massimo della distanza tra due vertici sarà $+\infty$. Indicate come si possa modificare l'algoritmo descritto al punto (a) in modo che fornisca in output anche:

- l'elenco di tutti i vertici che appartengono alla stessa componente fortemente connessa del vertice *1*,
- il valore massimo della distanza tra due vertici della componente fortemente connessa del vertice *1*.

Indicate inoltre se, asintoticamente, il tempo di calcolo cambia nell'algoritmo così modificato rispetto al tempo stimato al punto (b).

Note

- Nella descrizione e nello pseudocodice dell'algoritmo potete utilizzare direttamente il valore $+\infty$.

Esempio 2. Per il grafo rappresentato nella figura a destra si ha:

- vertici della componente fortemente connessa che contiene *1*: *1, 3, 4*
- massima distanza tra due vertici della componente fortemente connessa che contiene *1*: 13