C.d.S. Informatica, Università di Bologna
Algoritmi e Strutture Dati

Esercizio 1.[x punti]
Data una sequenza di n numeri interi $x_1, ..., x_n$ diciamo che (x_i, x_{i+1}) sono una coppia di numeri consecutivi se $x_i + 1 = x_{i+1}$. Ad esempio nella sequenza $(12, 13, 24, 25, 26, 35, 67)$ ci sono 3 coppie di numeri consecutivi: $(12, 13)$, $(24, 25)$ e $(25, 26)$. Scrivere un algoritmo **ricorsivo** che utilizzi la **tecnica divide-et-impera** e che calcoli quante coppie di numeri consecutivi sono contenute in una sequenza di n numeri interi $x_1, ..., x_n$ ricevuta in input. Valutare in ordine di grandezza il costo computazionale dell'algoritmo proposto. **Vietato usare il Master Theorem !!!**

Esercizio 1.[x punti]
Scrivere un algoritmo che ordina n numeri compresi tra 0 e $n^2 - 1$ in tempo $O(n)$. **Suggerimento:** *combinare counting-sort e radix-sort.* Motivare le risposte.

Esercizio 1.[x punti]
È possibile ordinare un Heap in tempo $o(n \log n)$?

Esercizio 1.[x punti]
Sia $G = (V, E)$ un grafo non orientato pesato tale che per ogni $e \in E$: $W(e) \in \{1, 2, 3, 4\}$. Sia $s \in V$. Scrivere un algoritmo che calcoli l'albero dei cammini minimi da s a tutti gli altri nodi in tempo $O(|V| + |E|)$. Suggerimento: Applicare BFS su un grafo G' ottenuto da G apportando opportune modifiche.

Esercizio 1.[x punti]
Sia V un vettore di n numeri interi positivi. Definiamo i seguenti due problemi computazionali:
MIN-GAP (DIFFERENZA MINIMA): determinare una coppia di indici $1 \leq i < j \leq n$ tali per cui per ogni $1 \leq k < h \leq n : |V[i] - V[j]| \leq |V[k] - V[h]|$.
MAX-GAP (DIFFERENZA MASSIMA): determinare una coppia di indici $1 \leq i < j \leq n$ tali per cui per ogni $1 \leq k < h \leq n : |V[i] - V[j]| \geq |V[k] - V[h]|$.

Fornire gli algoritmi con il minor costo computazionale possibile per MIN-GAP e per MAX-GAP. Discutere il loro costo computazionale.

Esercizio 1.[x punti]
Un grafo è 2-colorabile se è possibile assegnare ad ogni nodo il colore rosso o nero in modo tale che tutte le coppie di nodi adiacenti non abbiano lo stesso colore. Trovare un algoritmo efficiente per dire se un grafo è 2-colorabile oppure no. Discutere il costo computazionale dell'algoritmo fornito. (suggerimento: usare una BFS modificata)

Esercizio 1.[x punti]
Sia e l'arco di peso minimo in un grafo G. Dimostrare che esiste un MST per G che contiene e. (Suggerimento: procedere per assurdo. Se l'arco di peso minimo non stesse in nessun MST allora ...)

Esercizio 1.[x punti]
Siano $G_1 = (V_1, E_1)$ e $G_2 = (V_2, E_2)$ due grafi.
Siano T_1 e T_2 due MST per G_1 e G_2 rispettivamente.
Sia $G = (V_1 \bigcup V_2, E_1 \bigcup E_2 \bigcup E)$ dove E è un qualsiasi insieme non vuoto di archi che collegano nodi in V_1 con nodi in V_2.
Sia e un arco di peso minimo in E.
Sia $T = T_1 \bigcup T_2 \bigcup \{e\}$.
Dimostrare che T non è un MST per G fornendo un controesempio.

Esercizio 1.[x punti]

Scrivere una procedura RICORSIVA che prenda in input un albero binario A e restituisca in output un albero B ottenuto da A eliminando tutte le foglie che hanno un valore maggiore di quello del padre e di quello del fratello. La procedura deve essere ricorsiva e scritta in pseudo codice. Commentare il codice. Analizzare il costo computazionale della procedura proposta.

Esercizio 1.[x punti]

Sia G un grafo connesso non orientato. Siano $u_1, ..., u_k$ k nodi di G. Scrivere una procedura che restituisca il numero di nodi equidistanti da $u_1, ..., u_k$. Discuterne il costo computazionale al variare di k.

Esercizio 1.[x punti]

Si supponga di disporre di un ipotetico algoritmo di *quasi* ordinamento `QuasiSort` che, preso in ingresso un vettore A di n elementi, ne ordina $n/2$ elementi arbitrari collocandoli quindi nella metÁ sinistra $A[1..n/2]$. Si consideri il seguente algoritmo ricorsivo, dove n Á una potenza del 2:

```
program AlgRic( A[1..n] )
QuasiSort( A[1..n] )
AlgRic( A[n/2+1..n] )
Merge( A[1..n/2], A[n/2+1..n] )
```

La procedura Merge è quella standard impiegata per la fusione di due sequenze ordinate. Utilizzando l'algoritmo `AlgRic`, dimostrare che `QuasiSort` ha una complessità asintotica superiore a $O(n)$ confronti.

Esercizio 1.[x punti]

Scrivere una procedura RICORSIVA che prenda in input un albero A (non necessariamente binario! ogni nodo può avere un numero qualsiasi di figli) e restituisca in output un albero B ottenuto da A guardandolo allo specchio. Per la soluzione dell'esercizio è possibile usare senza implelentare una funzione INVERTI che prende una lista e la inverte.

Esercizio 1.[x punti]

Inserire negli spazi contrassegnati dai puntini Θ, o oppure ω. Giustificare la risposta!

$$\begin{aligned} n \cdot log_2(n) \text{ è} \quad & \ldots\ldots \quad (n^{1.01} + n \cdot log_2(n)) \\ (log(n))^n \text{ è} \quad & \ldots\ldots \quad (n^{1.02}) \\ (log(n))^n \text{ è} \quad & \ldots\ldots \quad (n^{log(n)}) \end{aligned}$$

Esercizio 1.[x punti]

Bip-Dominating-Set: Dato un grafo bipartito (L, R, E) e un intero positivo k dire se esiste un sottoinsieme di nodi di L di cardinalità minore o uguale a k che domini tutti i nodi in R (un insieme di nodi A domina un insieme di nodi B se ogni nodo di B è raggiunto da almeno un arco che parte da un nodo di A).

Trip-Dominating-Set: Dato un grafo tripartito (L, C, R, E) e un intero positivo k dire se esiste un sottoinsieme di nodi di L di cardinalità minore o uguale a k che domini tutti i nodi in R e in C.

Dimostrare che se Trip-Dominating-Set si risolve in tempo polinomiale allora anche Bip-Dominating-Set si risolve in tempo polinomiale.

Esercizio 1.[x punti]

Dare una procedura polinomiale (quella con il più basso costo computazionale possibile) che preso un grafo G in input restituisca **si** se G è aciclico e **no** altrimenti. Dimostrare la sua correttezza e fornire una analisi del suo costo computazionale.

Esercizio 1.[x punti]

Dare una procedura (quella con il più basso costo computazionale possibile) per ordinare un Red-Black Tree. Analizzare il suo costo computazionale. Dare una procedura (quella con il più basso costo computazionale possibile) per costruire un BST. Analizzare il suo costo computazionale. Dimostrare che non esiste nessun algoritmo per costruire un BST con costo computazionale $o(n\,log(n))$.

Esercizio 1.[x punti]

Scrivere una procedura RICORSIVA che prenda in input un albero binario A e restituisca in output un albero B ottenuto da A eliminando tutti i nodi che hanno un solo figlio. La procedura deve essere ricorsiva e scritta in pseudo codice. Analizzare il costo computazionale della procedura proposta.

Esercizio 1.[x punti]

Scrivere una procedura (la piİ semplice possibile) che prenda in input un vettore V di numeri (non ordinato) e restituisca in output un red-black tree T che contiene tutti i numeri di V. Vietato usare la procedura di inserimento in un red-black tree vista a lezione. Discutere il suo costo computazionale. Dare un limite inferiore di complessità (il migliore possibile) per questo problema.

Esercizio 1.[x punti]

Dare la procedura con il costo computazionale minore possibile per ordinare lessicograficamente n stringhe di k caratteri (k costante indipendente da n). Cosa succede se k non è costante ?

Esercizio 1.[x punti]

Scrivere una FUNZIONE RICORSIVA che prende in input il puntatore alla radice di un albero binario e restituisce il numero di nodi dell'albero che si trovano a distanza pari dalla radice. Argomentare la sua correttezza e analizzare il suo costo computazionale. Vietato usare variabili globali !!!

Esercizio 1.[x punti]

Inserire negli spazi contrassegnati dai puntini Θ, o oppure ω. Giustificare la risposta!

$$\begin{aligned} 1 \text{ è } & \ldots\ldots & sen(n)+1 \\ (log(n))^n \text{ è } & \ldots\ldots & n^{1+log(n)} \\ (log(n))^n \text{ è } & \ldots\ldots & n^{log^2(n)} \end{aligned}$$

Esercizio 1.[x punti]

Sia $G=(V,E)$ un grafo pesato e non orientato e T un albero di copertura minimo per G. Supponiamo di diminuire il peso di un arco in E. Scrivere un algoritmo per ricalcolare un albero di copertura minimo per il grafo modificato. Descrivere le strutture dati utilizzate. Calcolare il costo computazionale dell'algoritmo proposto e provarne la correttezza.

Esercizio 1.[x punti]

Scrivere una PROCEDURA RICORSIVA che prende in input il puntatore alla radice di un albero binario (i cui nodi contengono solo 1 o 0) e modifica il contenuto dei nodi dell'albero nel modo seguente: Sia n un qualsiasi nodo dell'albero. Se n contiene 1 allora al termine della computazione deve contenere il numero di nodi dell'albero radicato in n. Se n contiene 0 allora al termine della computazione deve contenere 0. Argomentare la sua correttezza e analizzare il suo costo computazionale. Vietato usare variabili globali !!!

Esercizio 1.[x punti]

Data una sequenza di n numeri interi $x_1, ..., x_n$ diciamo che (x_i, x_{i+1}) sono una coppia di fratelli se sono uguali. Ad esempio nella sequenza $(12, 12, 24, 25, 25, 35, 67)$ ci sono 2 coppie di fratelli: $(12, 12)$ e $(25, 25)$. Scrivere un algoritmo **ricorsivo** che utilizzi la **tecnica divide-et-impera** e che calcoli quante coppie di fratelli sono contenute in una sequenza di n numeri interi $x_1, ..., x_n$ ricevuta in input. Valutare in ordine di grandezza il costo computazionale dell'algoritmo proposto. **Vietato usare il Master Theorem !!!**

Esercizio 1.[x punti]

Scrivere una FUNZIONE RICORSIVA che prende in input il puntatore alla radice di un albero binario e restituisca la lunghezza della sequenza monotona più lunga partendo dalla radice. Argomentare la sua correttezza e analizzare il suo costo computazionale. Vietato usare variabili globali !!!

Esercizio 1.[x punti]

Inserire negli spazi contrassegnati dai puntini Θ, o oppure ω. Giustificare la risposta!

$$\begin{aligned} cos(n) \text{ è} \quad & \ldots\ldots \quad sen(n) + 1 \\ n \text{ è} \quad & \ldots\ldots \quad n^{sen(n)} \\ n! \text{ è} \quad & \ldots\ldots \quad n^n \end{aligned}$$

Esercizio 1.[x punti]

Fornire un algoritmo greedy polinomiale per risolvere max-sat. Dimostrare che non è ottimo. Dimostrare che max-sat è un problema in NP.