ESEMPI DI DOMANDE A CROCETTE

❖ Primo semestre

- È possibile ordinare un array di n numeri compresi tra 1 ed n^2 in: **$O(n)$, $O(n \log n)$, $O(n^2)$, $O(n^2/\log n)$**
- Il problema dell'ordinamento appartiene a: **$\Omega(n)$, $\Theta(n \log n)$, $O(n^2)$**
- Bucket Sort è applicabile solo quando i dati sono distribuiti uniformemente: **F**
- Se Radix Sort è applicabile, allora è applicabile anche Counting Sort: **V**
- In un RB-albero è possibile mantenere in tempo logaritmico un campo che indica il numero di nodi dell'intero albero con chiave minore alla chiave del nodo corrente: **F**
- Il problema della selezione del mediano di un array appartiene a: **$\Omega(n)$, $O(n^2)$**
- In un RB-albero è possibile mantenere in tempo logaritmico un campo Q[x] che indica il quadrato del numero di nodi nel sottoalbero radicato in x: **V**
- È possibile ordinare in tempo asintoticamente lineare: **array di numeri complessi**, **array di voti di laurea**
- Non è possibile ordinare in tempo asintoticamente lineare: **array di reali**, **array di razionali**
- Esiste una implementazione degli insiemi disgiunti tale che la complessità di n operazioni è $O(n \log n)$: **V**
- È possibile rimuovere tutte le radici di uno heap binomiale in tempo $O(\log^2 n)$: **V**
- Non esistono algoritmi deterministici lineari per il problema della selezione: **F**
- Se Bucket Sort è applicabile, allora anche Radix Sort è applicabile: **F**
- Il problema della moltiplicazione di due matrici appartiene ad $O(n^5)$: **V** **perché appartiene a $\Theta(n^3)$**
- Quick Sort funziona in tempo pessimo quadratico: **V**
- È possibile unire due heap binomiali in tempo logaritmico: **V**
- È possibile unire due heap binomiali in tempo lineare: **V** **perché $O(n) > \Theta(\log n)$**
- Non esistono algoritmi probabilistici lineari per il problema della selezione: **F**
- Il problema della moltiplicazione di due matrici appartiene ad $O(n^2)$: **F** **perché appartiene a $\Theta(n^3)$**
- Quick Sort funziona in tempo pessimo n log n: **F**
- Non è possibile unire due heap binomiali in tempo logaritmico: **F**
- È possibile ordinare un array di numeri razionali in tempo cubico: **V** **perché appartiene a $\Theta(n \log n)$**
- Non è possibile ordinare in tempo lineare un array di razionali in [0, 10] con numeratore limitato: **V** **perché $O(n) < \Theta(n \log n)$**
- È possibile trovare il mediano di un array non ordinato in tempo logaritmico: **F** **perché Select appartiene a $\Theta(n)$**
- Bucket Sort è applicabile se i dati sono distribuiti in: **rettangolo**, **cerchio goniometrico**, **gaussiana**, **razionali dell'intervallo [0,1]**
- Bucket Sort non è applicabile se i dati sono distribuiti in: **naturali dell'intervallo [1,100]**

<table>
<tr><th></th><th>COMPLESSITÀ</th><th>STABILE</th><th>IN LOCO</th></tr>
<tr><td>Insertion Sort</td><td>n^2</td><td>SI</td><td>SI</td></tr>
<tr><td>Merge Sort</td><td>n log n</td><td>SI</td><td>NO</td></tr>
<tr><td>Heap Sort</td><td>n log n</td><td>NO</td><td>SI</td></tr>
<tr><td>Quick Sort</td><td>n^2</td><td rowspan="2">NO</td><td rowspan="2">SI</td></tr>
<tr><td>Quick Sort probabilistico</td><td>n log n</td></tr>
<tr><td>Counting Sort</td><td>n+k</td><td rowspan="3">SI</td><td rowspan="3">NO</td></tr>
<tr><td>Radix Sort</td><td>l(n+k)</td></tr>
<tr><td>Bucket Sort</td><td>n</td></tr>
<tr><td>Select, Min, Max</td><td>n</td><td>/</td><td>/</td></tr>
</table>

❖ Secondo semestre

- Il problema dei cammini minimi ammette soluzione solo se non esistono cicli con un arco di costo negativo: F
- L'algoritmo di Ford-Fulkerson funziona solo se non vi sono archi con capacità 0: F
- Un albero di cammini minimi è un albero di copertura: V
- La complessità dell'algoritmo Floyd-Warshall è in $\Omega(V^2)$: V **perché appartiene a $\Theta(V^3)$**
- La complessità della visita in ampiezza in un grafo completo è $\Theta(V^2)$: V
- Il problema dei cammini minimi con sorgente singola appartiene a: **O((V+E) log V)**
- L'algoritmo di Prim si basa su una tecnica Greedy: V
- Il calcolo delle componenti fortemente connesse di un grafo appartiene a $\Omega(V \log V)$: F **perché appartiene a $\Theta(V+E)$**
- Il diametro di un grafo è finito se e solo se il grafo è connesso: F
- Un grafo è rappresentabile con liste di adiacenza solo se è connesso: F **perché si può fare sempre**
- È possibile verificare se un grafo orientato è aciclico in $\Theta(E)$: F **perché appartiene a $\Theta(V+E)$**
- L'algoritmo del simplesso viene usato per risolvere problemi di programmazione lineare: V
- L'algoritmo di Dijkstra per i cammini minimi è applicabile solo se non esistono cicli negativi: V
- L'algoritmo di Johnson per i cammini minimi tra tutte le coppie produce risposte corrette solo se applicato a grafi sparsi: F
- Le matrici di adiacenza non sono indicate per rappresentare grafi sparsi: V
- È possibile verificare se un grafo orientato è aciclico in $\Theta(V+E)$: V
- Un grafo orientato è bipartito se e solo se la sua chiusura transitiva è bipartita: V
- Nella programmazione lineare il costo della soluzione del problema duale è sempre uguale al costo della soluzione del problema primale: V
- L'algoritmo di Johnson per i cammini minimi tra tutte le coppie produce risposte corrette anche quando applicato a grafi non sparsi: V
- Le matrici di adiacenza sono particolarmente indicate per rappresentare grafi sparsi: F
- È possibile verificare se un grafo orientato è aciclico in $\Theta(V^2)$: V **perché appartiene a $\Theta(V+E)$**
- È possibile verificare se un grafo non orientato è bipartito in $\Theta(V+E)$: V
- È possibile verificare se un grafo orientato è aciclico in $\Theta(V^2+E)$: V **perché appartiene a $\Theta(V+E)$**
- L'algoritmo di Dijkstra per i cammini minimi è applicabile quando non esistono cicli negativi: V
- L'algoritmo di Johnson per i cammini minimi tra tutte le coppie produce risposte corrette anche se applicato a grafi sparsi: V
- Il problema di verificare se un grafo è bipartito appartiene a: **$\Theta(V+E)$, $O(V^2)$**
- Le matrici di adiacenza non possono essere utilizzate per rappresentare grafi sparsi: F
- Se in un grafo orientato tutti gli archi hanno lo stesso peso, non negativo, allora i cammini minimi possono essere calcolati in tempo $O(V+E)$: V
- Un grafo con un solo arco è sempre bipartito: F
- Il problema del flusso massimo non può essere espresso come problema di programmazione lineare: F
- L'algoritmo di Johnson per i cammini minimi tra tutte le coppie non funziona quando esistono archi negativi: F
- In un grafo aciclico esiste al più un nodo che può raggiungere tutti gli altri nodi: V
- La chiusura transitiva di un grafo aciclico è un grafo aciclico: V
- Se un grafo è orientato, allora la sua matrice di adiacenza è simmetrica: F
- In una rete di flusso, un flusso è massimo se e solo se non esistono tagli non saturi: F
- Se un algoritmo funziona sui grafi orientati, allora funziona anche sui grafi non orientati: V