Alcuni esercizi assegnati

Qui si raccolgono alcuni esercizi significativi assegnati durante il corso:

1. Realizzare `InsertionSort` in modo ricorsivo
2. Realizzare una funzione `Dup(A,p,r)` che verifica, in modo divide et impera, la presenza di duplicati nell'array A.
3. Risolvere le seguenti ricorrenze:
    - $T(n) = 2 \cdot T(n/2) + \log(n)$
    - $T(n) = 2 \cdot T(n/2) + n^2$
    - $T(n) = 2 \cdot T(n/2) + n \cdot \log(n)$
    - $T(n) = a \cdot T(n-1) + b$
4. Realizzare una funzione `Select(A, k)` che restituisce l'elemento che occuperebbe la k-ma posizione nell'array `A` ordinato. Detta `n` la dimensione dell'array, trovare soluzioni:
    - `O(n log(n))`
    - `O(n + k log(n))`
    - `O(n log(k))`

   Per gli ultimi due casi il suggerimento è quello di usare un MinHeap e un MaxHeap rispettivamente.
5. Realizzare con approccio divide et impera una funzione `Inv(A,p,r)` che ritorna il numero di inversioni in `A[p,r]`, ovvero il numero di coppie di indici`i, j` tali che `i < j` e `A[i] > A[j]`. [Suggerimento: Modificare il MergeSort]
6. Realizzare una procedure per fondere due alberi binari, il primo completo, il secondo quasi completo, della stessa altezza, che soddisfano la proprietà di Max-Heap, mantenendo questa proprietà. Cercare una soluzione di costo `O(log(n))` dove `n` è il numero di elementi titali dei due alberi.
7. Dire se esiste un algoritmo di tempo lineare per elencare gli elementi di un max-heap in ordine descrescente. Descrivere l'algoritmo oppure motivare l'impossibilità di realizzarlo.
8. Realizzare una implementazione degli alberi binari di ricerca nella quale il campo `p` (padre) è sostituito dal campo `succ` (successore).
9. Dimostrare che un albero binario è un Albero Binario di Ricerca se e solo se una visita simmetrica visita i nodi in ordine di chiave crescente.
10. Scrivere un algoritmo per determinare un insieme minimo di monete, di tagli`5`, `2` e `1`, che totalizzi un dato resto `r`.
11. Date `n` lezioni A_1, ..., A_n, ciascuna con il suo tempo di inizio s_i e fine $f_i$scrivere un algoritmo per allocare tutte le lezioni in un numero minimo di aule.
12. Date `n` lezioni A_1, ..., A_n, ciascuna con il suo tempo di inizio s_i e fine $f_i$scrivere un algoritmo per allocare il massimo numero di lezioni in un numero prefissato `m` di aule.
13. Implementare una coda FIFO con due pile, garantendo per le operazioni di inserimento ed estrazione costo ammortizzato `O(1)`.
14. Realizzare un contatore come array di bit `A[0,k]`, con operazioni di incremento e reset, garantendo per le operazioni costo ammortizzato `O(1)`.
15. Risolvere le seguenti ricorrenze:

- $T(n) = 4 \cdot T(n/16) + \sqrt{n}$
- $T(n) = 4 \cdot T(n/16) + n$
- $T(n) = 4 \cdot T(n/16) + \log(n)$
- $T(n) = T(n-1) + n^2$

nei primi tre casi utilizzando il master theorem, nell'ultimo il metodo di sostituzione.

16. Realizzare un algoritmo ricorsivo che verifica se un albero binario è completo e valutarne la complessità.
17. Dato un insieme di file di dimensione `f1, ..., fn` e un disco di capacità `d`, realizzare un algoritmo greedy che massimizza il numero di file nel disco, dimostrandone la correttezza.
18. Realizzare una generalizzazione del Mergesort nel quale l'array viene diviso in `k` parti invece che in 2. Valutarne la complessità.
19. Tradurre nel modo più diretto possibile la definizione dei numeri di Fibonacci (ovvero `fib(0) = fib(1) =1`, `fib(n+2) = fib(n+1) + fib(n)`in un algoritmo ricorsivo. Mostrare che la complessità è limitata inferiormente da α^n per una opportuna costante $\alpha > 1$ e fornire un algoritmo iterativo (stile programmazione dinamica, bottom up) di complessità lineare.
20. Si consideri una variante degli alberi binari di ricerca nei quali ogni nodo `x`ha un attributo addizionale `f` che rappresenta il numero di foglie nel sottoalbero radicato in `x`. Definire una funzione `Insert(T,z)` che inserisce un nodo.
21. Data una tabella hash con indirizzamento aperto, di dimensione `m=7`operante con doppio hash basato sulle funzioni `h1(k)= k mod m` e `h2(k) = 1 + k mod (m-1)`, descrivere cosa si ottiene inserendo i valori `10, 15, 22, 31, 43`.

Soluzioni non discusse in classe:

- Verificare che le rotazioni per gli Interval Trees hanno complessità `O(1)`.
- Data una sequenza di numeri $a_1, a_2, \ldots, a_n$, possibilmente negativi, trovare `i, j`, con $i < j$ tale che la somma $a_i + a_{i+1} + \ldots + a_j$ sia massima.