0 - Domande orali

2 febbraio 2022

- Che cos'è uno heap
- Come viene implementato (albero binario quasi completo)
- Definizione di maxHeap (e accenni a minHeap)
- Condizione di bilanciamento (definizione di albero binario quasi completo)
- Dato il disegno di un albero, dire se è AVL
- Modificare l'albero al punto precedente per renderlo uno heap
- Cosa garantisce il bilanciamento in uno heap (costi logaritmici e rappresentazione tramite array "senza buchi")
- Come mai un albero che degenera in una lista risulta più inefficiente rispetto ad un albero con bilanciamento (costi lineari VS costi logaritmici)
- In che momento dell'heapSort serve muoversi dalla radice verso la foglia (durante risistema)
- Dato uno heap, scrivere l'array associato e dare il nome di questo array (array posizionale)
- Fare un'esecuzione di heapSort dato un heap (spostare la radice nell'ultima foglia, applicare risistema)
- Come mai si applica risistema al posto di prendere il massimo dei figli e portarli nella radice ad ogni iterazione (si perde la condizione di bilanciamento)
- Numero di confronti (creaHeap + effettivo heapSort)
- Spazio utilizzato (in loco senza spazio aggiuntivo)
- Dare un algoritmo che ha le stesse prestazioni in termini di confronti ma che usa più memoria (mergeSort perchè usa un array ausiliario)
- Memoria aggiuntiva in mergeSort oltre all'array ausiliario (logaritmica per lo stack)
- Prestazioni di quickSort in tempo e memoria
- Differenza tra confronti e tempo (negli ordinamenti i confronti sono le operazioni più dispendiose)
- Quando la stima dei confronti equivale alla stima del tempo (quando sono O(1))
- Quando la stima dei confronti vale di più della stima del tempo (stringhe)
- Confronto sulle stringhe quanto costa (O(m) con m lunghezza della stringa)
- Definizione di algoritmo greedy
- In che tipo di problemi si usa un algoritmo greedy (problemi di ottimizzazione)

`TEMPO 45 MINUTI`

- MergeSort: definizione in modo ricorsivo

- Problema del merge
- Numero di confronti (minimo e massimo)
- Uso dello spazio
- Cos'è un albero in termini di grafi (non orientato, connesso e privo di cicli)
- Problemi relativi agli alberi (albero ricoprente minimo)
- Definizione di albero ricoprente minimo
- Cosa vuol dire che il grafo è connesso (tra ogni coppia di vertici ∃cammino)
- Cos'è una foresta
- Differenza tra alberi in termini di grafi e di alberi con radice
- Definizione di albero binario di ricerca
- Che visita va fatta per avere i nodi in ordine
- Esistono alberi di ricerca non binari
- Utilità B-alberi e alberi 2-3

`TEMPO 15 MINUTI`

- Definizione di albero ricoprente minimo
- Algoritmi per trovare l'albero ricoprente minimo (Kruskal e Prim)
- Spiegare Kruskal (BRAVO CHE HAI SCELTO KRUSKAL)
- Rappresentazione del grafo per Kruskal
- Strutture extra per l'algoritmo
- Costo di Kruskal
- Come cambia la complessità se si usa BubbleSort per ordinare gli archi
- Spiegazione del BubbleSort
- Numero confronti (caso migliore e peggiore)
- Algoritmo che fa sempre n^2 confronti (selectionSort)
- Esempi di algoritmi non basati sui confronti
- Limitazioni di questi algoritmi
- Se si usano i float, RadixSort funziona (no, F)

`TEMPO 25 MINUTI`

3 febbraio 2022

- Discussione esercizi esame
- Spiegazione quickSort in modo ricorsivo
- Tecnica che viene utilizzata (Divide et Impera)
- Altri algoritmi che utilizzano Divide et Impera (mergeSort)

- Differenza tra quickSort e mergeSort
- Dato un vettore, applicare la procedura Partiziona

`TEMPO 25 MINUTI`

- Struttura heap (albero binario quasi completo)
- Cosa garantisce questa struttura (altezza logaritmica)
- Differenza tra maxHeap e minHeap
- Come rappresentare un heap utilizzando poca memoria (vettore posizionale)
- Dato un nodo, come trovare il nodo padre (se indice è dispari --> posizione / 2, se indice è pari --> (posizione - 2) / 2)
- Applicazioni heap (heapSort, code con priorità)
- Cosa sono le code di priorità e per cosa vengono usate (si preleva ogni volta l'elemento di)
- Differenza coda normale - coda di priorità
- Che struttura usa LIFO (pile)
- Operazioni su un coda di priorità con costo associato
- Come mai si applica risistema al posto di prendere il massimo dei figli e portarli nella radice ad ogni iterazione (si perde la condizione di bilanciamento)
- Alberi binari di ricerca
- In cosa può degenerare un albero (lista, costi lineari)
- Condizioni di bilanciamento per garantire un'altezza logaritmica
- Alberi AVL
- Alberi perfettamente bilanciati
- Differenza tra questi ultimi 2 alberi
- Perchè i bilanciati sono shit (i bilanciati hanno un risistema costoso)
- Alberi 2-3
- Come si effettua la ricerca negli alberi 2-3
- Come si effettua l'inserimento negli alberi 2-3 (split)

`TEMPO 25 MINUTI`

- Tecnica greedy
- Applicazione al problema dello zaino
- Dire se trova sempre la soluzione ottima
- Dire se esistono algoritmi greedy che trovano sempre la soluzione ottima (Kruskal, Prim, Dijkstra)
- Formulare Kruskal
- Dato un grafo, applicare Kruskal
- Come viene usata la struttura Union-Find in Kruskal

- Come vengono rappresentate le strutture Union-Find
- Struttura QuickFind (rappresentazione e union normale/bilanciata)
- Struttura QuickUnion (rappresentazione)
- Quando l'altezza dell'albero cresce durante una union in QuickUnion (quando i due alberi hanno la stessa altezza)
- QuickFind con compressione di cammino
- In che algoritmi ci sono problemi con cicli negativi (algoritmi sulla ricerca dei cammini minimi)

`TEMPO 30 MINUTI`

10 febbraio 2022

- Descrizione heapSort
- Definizione struttura heap
- Come mai viene usato un albero binario quasi completo
- Rappresentazione di un heap come array posizionale
- Come mai si applica risistema al posto di prendere il massimo dei figli e portarli nella radice ad ogni iterazione (si perde la condizione di bilanciamento)
- Descrizione quickSort
- Complessità in tempo e spazio di quickSort
- Complessità in tempo e spazio di heapSort
- Algoritmi di ordinamento elementati che hanno buone prestazioni nel caso migliore
- Descrizione bubbleSort
- Caso migliore bubbleSort
- Tecniche greedy
- Algoritmo greedy che trova sempre la soluzione ottima

`TEMPO 20 MINUTI`