Domande orale programmazione APA

- Questa è una raccolta di domande prese dagli orali di APA negli anni: domande prese e riordinate da qui.
- Le domande sono state private della risposta e inserite senza i doppioni (o almeno ci ho provato).
- Data una stringa individuare le sottostringhe tra le virgole e inserirle in una lista ordinata
- Dato un Binary Search Tree nel quale per ogni nodo è presente una lista di stringhe, liberare il tutto (free)
- Dato un vettore di stringhe dividere in due sottovettori le stringhe che iniziano per vocali e consonanti (ti da una funzione per il check); ritornare per riferimento i due vettori (allocati di dimensione corretta, basta leggere una volta la matrice)
- Tutte le sequenze di numeri 0-9 lunghe k per cui vengono rispettati dei vincoli (vuole il pruning)
- Creare una coda a priorità, in cui la priorità è la distanza dall'origine di un punto nel piano cartesiano
- Dato un vettore di interi mettere gli elementi pari in una lista e quelli dispari in un'altra; tenere le liste ordinate.
- Dato un Binary Search Tree in cui ogni nodo ha come Item liste concatenate implementare una funzione freebst() per l'eliminazione dell'intero Binary Search Tree
- Data una matrice con tanti 0, trasformarla in vettore di puntatori a liste in cui indice vettore è riga, nella lista c'è il valore diverso da 0 e colonna.
- Cammino / ciclo lunghezza k in un DI
- Da matrice a liste di adiacenza di un grafo
- Lettura file e caricamento in un vettore di struct, passato per riferimento alla funzione che deve fare tutto
- Stack implementato come ADT di I categoria con vettore
- Componenti connesse di un grafo con lista delle adiacenze e per ogni componente contare quanti vertici ha
- Costruisci un ADT lista. In questa lista ricerca un Item e cancellalo
- Cammini semplici tra sorgente e destinazione in un grafo
- Due matrici: le entrate della seconda matrice sono la media delle caselle adiacenti della prima matrice. La seconda matrice è passata come puntatore (***) quindi deve essere allocata dinamicamente nella funzione
- Dato un vertice di un grafo trovare tutti i cicli di lunghezza k
- Data una lista linkata semplice, rimuovere gli elementi la cui posizione in lista è pari
- Dato un grafo si ritorni il numero di cicli presenti lungo un cammino di lunghezza k
- Data una stringa e noto un carattere di terminazione (ad es "#"), dividere la stringa in tutte le sottostringhe separate tra loro dal terminatore, per poi inserire tali sottostringhe in un vettore di stringhe supposto vuoto e passato per riferimento; ritornare infine il numero di sottostringhe trovate. Prototipo: int* funzione(char **stringa, char ***strvett)
- Dato un albero N-ario, contare tutti i suoi nodi (suggerimento: implementazione left child - right sibling e utilizzo di una funzione ricorsiva in grado di visitare tutto l'albero)
- Implementare un set come ADT di I categoria ed implementare una funzione merge che fonde due set in uno
- Funzione che indichi se è presente un ciclo di lunghezza >= k
- Merge di due liste
- Invertire una lista
- Creare una struttura dati di una lista doppio linkata con FIFO

- Vettore di interi: creare due liste, una con gli elementi pari e una con quelli dispari, mantenendo le liste ordinate
- Dato un vettore con le cifre 0-9, generare tutti gli insiemi di 5 di questi elementi, in modo tale che:
  - A partire dal primo, siano sistemati come pari, dispari, pari, dispari, ecc
  - Ogni elemento della soluzione sia strettamente maggiore del successivo
- Merge di due stringhe ordinate in una sola sempre ordinata
- Inserimento in hash con linear chaning senza duplicati del valore in input
- Insertion sort su un vettore di puntatori a stringhe
- Inserimento in un Binary Search Tree, con funzione ricorsiva di inserzione
- Dato un vettore di stringhe dividerlo e ritornare due vettori. Il primo conterrà le stringhe inizianti per carattere, il secondo quelle inizianti cono numeri.
- Dato un grafo implementare la funzione GraphEdges (che ritorna tutti gli archi)
- Disposizioni semplici (Considerato un vettore di caratteri con tutte le lettere dell'alfabeto generare tutte le parole di 10 caratteri considerando che in ogni parola una stessa lettera può essere ripetuta massimo 2 volte).
- Selection sort su un vettore di puntatori a stringhe
- Data una matrice di caratteri (vettore di stringhe) allocare dinamicamente un vettore di puntatori a stringa in cui ogni riga ha la lunghezza che serve e non una lunghezza generica
- Data una matrice R*C con stringhe terminate dal carattere '*', allocarne una dinamica, con le righe di lunghezza pari alla parola da copiare e copiarne il contenuto.
- Definire la struct di un nodo di un albero n-ario e scrivere una funzione che ritorni il numero di foglie
- Date due liste ordinate di stringhe, fonderle in una terza mantenendo l'ordine
- Determinazione delle componenti fortemente connesse in un grafo
- Vettore di puntatori a struct di tipo punto, definire la struct punto e leggere da file i punti
- Data una struttura FIFO passata per parametro ritornare un contatore che indica il numero di elementi e stampare
- StackPop con stack implementato come ADT di I classe (vettore)
- Find della QuickUnion
- Date 2 stringhe implementare una funzione che copiasse in una terza il contenuto della prima meno le lettere in comune con la seconda
- Trovare la foglia di altezza minima in un Binary Search Tree
- Trovare il grado di un vertice in un grafo implementato con matrice di adiacenze
- Estrazione da buffer circolare
- FIFO come ADT di I classe ed estrazione
- Ricerca in un Binary Search Tree
- Svuotare un vettore dai numeri negativi e ricompattarlo (con complessità lineare)
- Compattare una matrice in un vettore di puntatori a interi eliminando tutti gli zeri
- Problema simile a quello dello zaino (powerset)
- Data una matrice passata come riferimento calcolare la media dei vaolir locali per ogni punto riga colonna e restituire tutto in una nuova matrice anch'essa passata come riferimento
- Implementare funzione Heapify
- Trasformare una lista di stringhe in un vettore di stringhe, poi ordinarlo con un sort a piacere
- Dato un Binary Search Tree e un intero k, scrivere una funzione che elimini tutti i figli sinistri a partire dal livello k-esimo in giù
- Ricerca iterativa in un Binary Search Tree
- Inversione di una lista

- Binary Search Tree: funzione che ritorna il numero di nodi che hanno esattamento un figlio
- Inserimento in una coda a priorità
- Stampare a video tutti i cammini semplici di lunghezza k partendo da un vertice dato A
- Inserzione in una lista ordinata
- Dato un Binary Search Tree, funzione in grado di calcolare il numero di: nodi con un solo figlio, numero di nodi sinistri, numero di nodi destri, numero di nodi totali
- Data una struttura wrapper con un intero contenente il numero di stringhe e un puntatore a un vettore di puntatori a stringa, leggi un file contenente delle stringhe, contale, alloca opportunamente la struttura e il vettore, dopo di che inserisci le stringhe nel vettore (è possibile leggere il file più volte)
- Codice inserimento in foglia in un Binary Search Tree
- Data una lista che contiene interi casuali generare altre due liste: una che contenga solo gli interi pari e l'altra solo quelli dispari
- Generare tutti i numeri binari di n elementi di cui almeno d siano 1
- Dato un vettore di puntatori a struct Point{int x; int y} applicare insertion o selection sort su quel vettore in base alla distanza dall'origine
- Scrivere una funzione che dato un grafo ritorni due puntatori a grafi uno con archi a valore positivo e l'altro con quelli a valori negativi
- Wrapper con puntatori a puntatori a struttura, con relativa funzione di allocazione
- Generare delle stringhe con le lettere alfabetiche, ogni lettera può essere ripetuta al massimo p volte e la sequenza di lettere ripetute può essere lunga al massimo q
- Funzione che dato un grafo calcoli tutti i cammini che si possono effettuare da ogni singolo vertice e che abbiano lunghezza minima k
- Generare anagrammi di una stringa, col vincolo di consonanti e vocali non consecutive
- Componenti connesse tramite una ricerca in profondità
- Liste doppio linkate come ADT
- Inserzione in foglia in un Binary Search Tree
- Dato un grafo e un suo vertice di partenza determinare un cammino di lunghezza 5
- Ricerca del massimo in un Binary Search Tree
- HeapSort
- Trasformazione di una matrice di char NxN in un vettore di stringhe allocate dinamicamente della giusta lunghezza
- Implementare funzioni Init, Aggiungi, Estrai di una coda a priorità implementata come ADT di I classe
- Problema di ottimizzazione simile al problema dello zaino
- Verificare se un Binary Search Tree è contenuto in un altro di lunghezza maggiore oppure no
- Codice visita in ampiezza
- Permutazioni semplici
- File di testo con numero di righe e a seguire N righe con cognome e nome. Va inserito tutto in un vettore di puntatori ad Item, allocando stringhe e Item dinamicamente
- Funzione di verifica del gioco Forza4, cioè data una matrice verificare che ci siano 4 '1' consecutivi sulla riga / colonna / diagonale
- Funzione che ritorna il numero di elementi presenti al livello i-esimo di un heap con i dato
- Problema dello zaino discreto, esaustivo, ricorsivo con pruning
- Data una lista eliminare i nodi pari
- Dato un albero rappresentato come left child – right sibling contare il numero di figli di ogni nodo (solo quelli diretti)

- Dato un file contenente informazioni su persone (nome e età), inserire i dati in una coda implementata come ADT di I classe salvando i dati letti in una struct persona prima dell'inserzione in coda
- Algoritmo QuickUnion e differenza con Weighted QuickUnion
- Inserimento in FIFO con buffer circolare
- Enumerazione di tutti i cammini semplici tra due vertici di un grafo
- Inserimento in un Binary Search Tree in radice
- Inserzione e cancellazione in una lista doppio linkata
- Data una matrice di interi poco densa, trasformarla in una lista di liste in modo che si possa tornare alla matrice iniziale partendo dalla lista
- Dato un grafo, calcolare tutti i cicli non per forza semplici di lunghezza k che partono da un nodo dato
- Ricerca in profondità che rileva cicli, un ciclo può essere costituito da più passaggi sullo stesso vertice (prima versione O(n^2), poi ottimizzare)
- Dato un grafo non orientato non pesato e dato un vertice di partenza, contare tutti i cicli anche non semplici di lunghezza k
- Dato un Binary Search Tree contare i nodi che hanno k figli