‘Si prega gentilmente di mettere anche la data dell’appello. Mettete o in testa o in coda o in ordine, preferibilmente in ordine, ma fondamentale mettete la data dell’appello per favore. Possibilmente mettere il voto come “Titolo”, in alto a SX c’è scritto “Testo normale” accanto ad “Arial”, seleziona “Titolo” per il voto.

Grazie e complimenti di averlo superato.

20 -> 23 (28/09/2023)

Teoria (Camurati):

- Equazione alle ricorrenze nel decrease and conquer più focus sul numero di sottoproblemi che possono essere generati e sul come compattare l’equazione usando un’unica formula matematica per tutti i sottoproblemi (sommatoria per k di T(n-k)) + Es Fibonacci: T(n)=T(Fn-1)+T(n-2)+1
- Bellman-Ford: che approccio applica, a che scopo e limite superiore del numero di passi con motivazione (ogni passo->arco in più nella soluzione, v-1 archi messi in MASSIMO v-1 passi)
- Etichettatura degli archi nella DFS con spiegazione di come riconoscerli durante l’esecuzione del programm

Programmazione (Palena):

- Definisci la tabella di HASH come quasi ADT e implementa l’inserimento con linear probing
- Definisci le strutture dati di nodo e lista in un modo a scelta e implementa una funzione che, dato un intero k, elimini tutti i nodi con x->Val<k.

Commento: Camurati apprezza molto le risposte dirette, dopo 10 minuti di spiegazione mi ha fermato e mi ha detto “si, quindi come li riconosco durante l'algoritmo…"; se comunque la risposta è corretta è soddisfatto.

Palena: amore, ha dato tutto il tempo per fare schemi o correggere in automatico gli errori senza arrabbiarsi o mettere fretta.

16 -> 19 (28/09/2023) (appello 20/09/2023)

Teoria (Camurati)

- SCC, grafo fortemente connesso e Kosaraju, kernel dag
- Etichettatura archi DFS e come si trovano (vettori pre e post)
- Complessità BSTrotate (O(1)) e quando si usa (per esempio BSTPartition)

Programmazione (Palena)

Definire grafo tramite madj, poi effettuarne la trasposizione

- Implementare BST e searchR

Commento: Come già specificato nel file, Camurati apprezza molto le risposte precise e dritte al punto, è meglio pensarci su piuttosto che fare “giri strani” per arrivare alla risposta corretta. Palena bravissimo, lascia tutto il tempo per riflettere e, in caso di errore, te lo fa notare in modo tale da poter farti formulare una soluzione corretta.

19 -> 21 (25/09/2023)

Teoria(Camurati)

- Zaino discreto e continuo
- Programmazione greedy
- Algoritmo di Kosaraju
- Ordinamento topologico
- Ordinamento topologico inverso

Programmazione(Cabodi)

- Data una lista eliminare un nodo ogni due
- Dato un DAG contare i vertici sink e source

20+1.75 lab -> 22 (30/06/2023)

Teoria (Camurati)

- Algoritmo di Kruskal (che paradigma usa e quali sono le strutture dati che utilizza)
- Come determinare la funzione di hash per una stringa

Programmazione (Cabodi)

- Dato un BST, una chiave k e un intero m stampare se esiste la chiave k e le m chiavi successive (ovvero se la chiave è Milano e m=2 stampare Milano, Modena, Napoli)
- Dato un grafo orientato e un vertice v, determinare quanti vertici sono raggiungibili da v con al massimo 2 passi

19 + 1 lab → 21 (30/06/2023)

Teoria(Camurati)

- Quando si verifica il caso peggiore nel QuickSort, equazione alle ricorrenze, a cosa è dovuto il termine n nell’equazione delle ricorrenze
- Codici di Huffman
- Che struttura dati utilizza Dijkstra, che approccio utilizza e dove sta nell’algoritmo l’approccio greedy

Programmazione (Pasini)

- Data una lista del tipo 1->3->6 riempire i buchi e ottenere 1->2->3->4->5->6
- Trovare i nodi non completi in un BST nell’intervallo (i1, i2)

21(0.5 punti lab) -> 23 (30/06/2023)

Teoria(Camurati)

- struttura generale equazione alle ricorrenze
- struttura equazione ricorrenze per decrease and conquer
- definizione stack frame e utilizzo per funzioni ricorsive
- cos'è lo stack overflow e se una funzione tail recursive può provocare stack overflow

Programmazione(Pasini)

- Data una lista orientata non pesata L di interi riempire i "buchi" tramite una funzione fill dove i "buchi" sono gli interi che mancano tra due nodi per essere consecutivi.

  (attraversamento lista con puntatore al precedente se la differenza nel valore dei nodi eccede 1 aggiungo nodo t con t -> val = p->val+1, bisogna gestire l'incremento dentro il for perchè se facciamo aggiunta il precedente diventa nuovo nodo e successivo diventa t->next)

  Es. 1 -> 2 -> 4-> 7

  deve diventare 1->2->3->4->5->6->7

- Determinare tutti i cammini semplici in un grafo G partendo da un vertice v con lunghezza massima L

  (funzione allPath che è una dfs modificata dove la condizione terminazione è cnt >= L e backtrack smarcando vertici come visitati sul vettore pre[])

27 (2 punti lab) -> 30 (03/03/2023)

Programmazione (Cabodi)

- Su un grafo pesato e orientato, trovare triplette di vertici connessi a->b->c e aggiungere un arco che unisca a->c con matrice delle adiacenze (e lista delle adiacenze, a voce). Risolto con 3 for annidati per avere i tre nodi a,b,c. A voce mi ha chiesto a quali particolari dovessi fare attenzione (se l'arco a->c come lo gestisco? Possibilità di creare multigrafo…)
- Su BST, trovare successore del successore di un nodo data la sua chiave (tralasciando il caso in cui il successore si trova più in alto). Risolto impostando una funzione ricorsiva che, una volta trovato il nodo, chiamava select(rango=1) sul sottoalbero dx.

Commento: Tranquillo. Spesso ti ferma senza che tu debba scrivere il codice per intero. Fa domande di ragionamento sull'approccio proposto, ti chiede come gestiresti certi problemi, ecc…

Teoria (Camurati)

- Complessità DFS con lista e matrice delle adiacenze
- Gestione delle collisioni con hash table

- Fattore di carico. Tra che estremi può variare e che significato ha per linear chaining e open addressing?
- Come eliminare elementi da hash table con linear chaining e open addressing?

Commento: Apprezza risposte sintetiche e dritte al punto. Se dici qualcosa di impreciso ti fa una domanda per farti notare l'errore affinché tu possa correggerti.

Voto 20->22 03/03/2023

Teoria(Camurati)

-Che cos'è lo stack frame, complessità spaziale delle funzioni ricorsive tail-recursive

-Complessità di Fibonacci

-Equazione alle ricorrenze del divide and conquer voleva la spiegazione di cosa sono C(n) e D(n)

Programmazione (Palena)

- implementare uno stack con un vettore dinamico, poi fare la push

-Inserimento in una tabella di Hash con double hashing, voleva anche i prototipi delle funzione di Hash

Commento: Palena ti fa sentire tranquillo e ti lascia tutto il tempo necessario, Camurati preferisce le risposte precise e insiste finché non ottiene una risposta corretta e precisa.

26+2 -> 29 (03/03/2023)

Teoria (Camurati)

- Modi in cui abbiamo visto il calcolo del powerset (in particolare voleva che gli spiegassi quello con l'approccio Divide Et Impera).
- Cos'è un bridge e come verificare per un arco di un grafo se questo è un bridge.
- Rapporto tra visita in ampiezza e calcolo dei cammini minimi.

Programmazione (Pasini)

- Implementare una tabella di hash in cui le collisioni sono gestite tramite linear chaining e scrivere una funzione di ricerca di una data stringa all'interno della tabella.
- Dato un vertice v di un grafo, trovare tutto il "vicinato" di quel vertice, ovvero tutti i vertici raggiungibili da v che distano al più di un certo valore k da v. (Ho implementato una bfs in cui nel momento in cui si supera la distanza k da v la funzione smette di inserire i vertici adiacenti nella coda).

Commento: Camurati è tranquillo anche se molto freddo, io dopo la prima domanda a cui non ho saputo rispondere bene sono andato nel pallone e ho fatto schifo a quelle dopo (ho percepito il suo tentativo di non incazzarsi mentre dicevo cagate .-.). Per quanto riguarda Pasini è un patatone, mette subito a proprio agio e lascia tutto il tempo di riflettere al problema e, nell'eventuale difficoltà, cerca di darti degli input per la

risoluzione dell'esercizio, mantenendo sempre un tono molto tranquillo. L'unico consiglio che posso darvi è di non farvi prendere dal panico, per quanto mi riguarda l'orale è stato decisamente meno peggio di quanto mi aspettassi, nonostante abbia quasi fatto scena muta nella parte di teoria. Unica cosa, con Camurati vi consiglio di non dire cose se non le sapete, sparare cose a caso potrebbe essere controproducente (un banale "non lo so" penso sia meglio). Per il resto vedrete che l'orale in sè è molto meglio di quello che sembra. Buona fortuna a tutti!

29+2->30L (03/03/2023)

Teoria (Camurati)

- È applicabile la programmazione dinamica ai cammini minimi? Se si, dimostra
- Condizioni di applicazione della programmazione dinamica oltre alla presenza della sottostruttura ottima della soluzione -> complessità polinomiale dei sottoproblemi indipendenti e complessità polinomiale della loro ricombinazione
- I vantaggi della programmazione dinamica rispetto ad un approccio top-down ricorsivo -> la ricorsione assume tutti i sottoproblemi indipendenti e di ugual natura, risolvendo già problemi che aveva risolto prima, vantaggio per la memoria occupata -> si rischia stack overflow nella ricorsione

Camurati molto tranquillo, anche ascoltando l'orale di altri miei colleghi, aiutava e dava suggerimenti. Consiglio di pensarci un pò prima di formulare una risposta, senza buttarsi a capofitto, ovviamente apprezza risposte veloci e coincise.

Programmazione (Palena)

- Definisci ADT I classe lista singolo linkata e con campo intero val. Devi scrivere sia il .h che il .c, solo per la struttura dati, non è necessario implementare l'architettura per evitare l'inclusione multipla del file. Fa domande sugli ADT applicati al problema, nel caso ad esempio se la struttura wrapper o la struct node della lista fosse implementata nel .h, sarebbe ancora ADT I classe o meno e perchè. Successivamente ha dato un prototipo di funzione void intersection(LIST l1, LIST l2) bisognava trasformare la prima nell'intersezione delle due. Liste non ordinate e senza doppioni.
- Vengono lanciati k dadi a 6 facce, stampa tutte le soluzioni che hanno somma delle facce > T e almeno una faccia doppia. Chiede di individuare il modello del calcolo combinatorio e se è possibile fare pruning.

Palena ti lascia prendere tempo per pensarci sia nel trovare gli errori sia a trovare la soluzione. Ti aiuta a trovare gli errori dandoti suggerimenti. Dopo ti dice cosa hai sbagliato, è preciso su questo aspetto, ma abbastanza tranquillo.

20 -> 25 (03/03/2023) (2 punti bonus lab)

Teoria (Camurati)

- Heapify caratteristiche e complessità
- Spiegare in cosa consiste la memoization

Programmazione (Cabodi)

- Dato un albero binario, calcolare il numero medio di nodi per livello (basta contare i nodi e trovare l'altezza dell'albero)
- Dato un grafo non orientato, trovare una quaterna di vertici tali che ogni vertice sia adiacente almeno ad altri due della quaterna(essenzialmente per ogni vertice fare 4 cicli annidati per poi verificare che nella lista delle adiacenze del 4 vertice ci sia il vertice di partenza) (il problema consiste nel trovare un ciclo di lunghezza 4)

22 -> 24 (0.75 punti lab) (03/03/2023)

Teoria (Camurati)

- Paradigma Greedy con esempio di funzione di appetibilità STATICA (selezione attività) e DINAMICA (alg. Dijkstra).
- HeapBuild con complessità.
- Come calcolare la chiave di una stringa per le tabelle di Hash.

Programmazione (Palena)

- Definire stack come ADT di 1 classe in cui il vettore è dinamico, implementare funzione push con realloc del vettore.
- Problema dei 6 dadi (già scritta nelle domande precedenti, si applicano le combinaz. rip. con k=numero dadi) con pruning.

25+ (2 laboratorio) ->25 (02/03/2023)

Teoria (Camurati)

- Componenti fortemente connesse e Kosaraju
- Algoritmo di Er (cosa fa e come funziona la sua ricorsione)

Programmazione (Pasini)

- Date 2 liste ordinate con ripetizioni, ritornare una terza con solo gli elementi comuni e senza ripetizioni
- Controllare se un grafo è bipartito

(Commento: sapendo un po di teoria e, anche senza saperlo fare, dimostrando di aver capito il problema di programmazione si dovrebbe passare, good luck)

28 -> 30

Teoria (Camurati)

- Euclide-Lamé, come funziona (parametri x,y; se y==0 mi fermo altrimenti ricorro con y, x%y) e complessità (numeri di fibonacci consecutivi per caso peggiore, O(n) con n indice del numero di fibonacci oppure O(log y))
- Heapsort, come funziona e complessità
- inserimenti nella priority queue (io ho detto heap e lista ordinata)

Programmazione (Pasini)

- lista che non termina ma l'ultimo elemento ha come next uno dei precedenti (è già spiegato nel file)
- trovare tutti i vertici a distanza esattamente k da uno dato (bfs)

20+1.25 -> 25 (02/03/2023)

Teoria (Camurati)

- kernel DAG
- Bellman Ford e al passo i-esimo quanti archi formano il cammino minimo
- Definizione di funzione tail-recursive

Programmazione (Pasini)

- Implementare una funzione list f(list l1, list l2) che restituisca una lista composta dall'intersezione delle due liste ordinate e di gestire eventualmente i duplicati.
- Grafo non orientato: contare tutti i cammini a partire da un vertice v di lunghezza k

17 ->20 (1.75 lab) (02/03/2023)

Teoria (Cabodi)

- Inserzione e estrazione in un BST e relative complessità
- algoritmi greedy, selezione di attività

Programmazione (Pasini)

- dati due vettori di interi ordinati e la loro dimensione, scrivere una funzione che allochi un terzo vettore di interi in cui inserire gli interi che sono presenti in tutti e due i vettori (intersezione). Il terzo vettore e la sua dimensione devono essere poi disponibili anche al chiamante della funzione.
- dato un albero ternario, scrivere una funzione che calcoli quanti cammini di lunghezza dispari sono presenti tra radice e foglie, ogni volta che si attraversa un arco la lunghezza cresce di 1.

18 ->20 (1.75 lab) (02/03/2023)

Teoria (Camurati)

- Complessità QuickSort, in che caso si ricade nel caso peggiore, in che caso non siamo nel caso peggiore ma siamo in un caso particolarmente sfortunato
- Come generare casualmente un grafo e come evitare multigrafi in questo caso -> (rappresentando il grafo con una matrice di adiacenza, perché in questo caso se i vertici i,j sono già connessi lo vedo con la matrice con costo O(1))

Programmazione (Palena)

- Definire stack come ADT di 1 classe, implementare funzione push
- Definire albero binario a scelta se come quasi ADT o ADT di 1 classe, implementare funzione che conta il numero di nodi al livello k

27 -> ? + 1 lab (02/03/2023)

Teoria (camurati)

- Etichette archi (t,b,f,c)
- equazione delle ricorrenze divide and conquer

Programmazione (palena)

- Problema dei 6 dadi (dati k dadi verificare con un modello del calcolo combinatorio e con del pruning che la somma sia maggiore di T e la soluzione è valida se almeno due facce sono uguali)
- Implementazione di ADT di prima classe di un albero binario e verifica che due alberi binari siano isomorfi

28 -> 30 + 1.25 lab (02/03/2023)

Teoria (Camurati):

- Cos'è un codice libero da prefisso, c'è perdita di dati con la codifica libera da prefisso?
- Fammi un esempio di codifica NON libera da prefisso
- Archi non possibili nei DAG e perchè
- Perchè un cammino minimo è un cammino semplice

Commento: ti aiuta ad auto-correggerti, silenzio = la risposta è giusta, meglio prendersi qualche secondo prima di rispondere piuttosto che vomitare parole (si potrebbe innervosire). Consiglio: risposte brevi e coincise, niente fronzoli inutili (possono andare a tuo svantaggio lol).

Programmazione (Pasini):

- date due liste singole linkate ordinate, con possibili elementi duplicati entro una stessa lista, fai una funzione che ritorna una terza lista contenente i nodi comuni alle due liste, evitando i duplicati. ES: l1: 2>3>3>4, l2: 2>3>3>5 risultato l3: 2>3
- dato un grafo, calcola tutte le bipartizioni possibili (vedi pacco slide 00 per definizione grafo bipartito). Richiesta esplicita di usare un modello combinatorio per generare le partizioni e una funzione di verifica per verificare se la parzione del grafo da origine ad un grafo bipartito.

Commento: nel chill, a parte le bestemmie dovute al portale che lagga.

26->? (02/03/2023)

Programmazione (Cabodi):

- Determinare un cammino (quasi) semplice in un grafo: ossia un cammino con 2 ripetizioni
- trasformare un heap in un albero binario: ossia farlo passare da vettore ad una implementazione tramite nodi con puntatori

Teoria (Camurati):

- Grafo completo, numero di archi, quale modello del calcolo combinatorio si utilizza per determinare questo numero?
- Fattore di carico
- Cammino di Eulero e lemmi corrispondenti

29->30 (02/03/2023)

Programmazione (Cabodi):

- A partire da una stringa s, realizzare il codice di una funzione che fa le permutazioni considerando eventuali sequenze di vocali o di consonanti come un'unica lettera. Es. torno considerata come t o rn o . Voleva prevalentemente sapere come salvare la stringa per poter fare le permutazioni, basta un vettore dove l'indice corrisponde all'inizio di una sequenza e il valore a quell'indice corrisponde alla fine
- Scrivere una funzione di verifica per un grafo quasi completo-k, ovvero un grafo completo a cui mancano al massimo k archi e tale che ad ogni vertice manchi al più un arco.

Teoria (Camurati):

- Problema della colorabilità di un grafo
- Che cos'è lo stack frame
- Teorema e corollario per gli archi sicuri in un MST

25->25 con 1 punto lab (02/03/2023)

Teoria (Camurati):

- Grafo: tipi di rappresentazioni e vantaggi/svantaggi per ognuna (vettore di liste, lista di liste e matrice di adiacenza), le varie complessità
- Algoritmo di Kosaraju
- Massimo comun divisore: algoritmo di euclide

Programmazione (Pasini):

- Date 2 liste ordinate singolo linkate l1, l2: creare una funzione che ritorni una nuova lista formata dall'intersezione delle due liste l1 l2, ignorando i duplicati

- Grafo non orientato: trovare tutti i nodi raggiungibili a partire da un vertice v con al massimo k passi (voleva l'algoritmo di visita in ampiezza)

17->19 con 1.25 lab (01/03/2023)

Teoria (Cabodi):

- BST Partition
- Algoritmo di Bellman-Ford e la sua complessità nel caso migliore e peggiore

Programmazione (Pasini):

- Dati due vettori v1 e v2 (ordinati) con le relative dimensioni d1 e d2, scrivere una funzione void f(int *v1, int *v2, int d1, int d2,...) che allochi un terzo vettore che contenga la differenza simmetrica fra v1 e v2. (gli elementi ottenuti dall'unione di v1 e v2 - la loro intersezione) escludendo i duplicati
- Dato un albero Left-child, Right-sibling (LCRS), andare a riempire per ogni nodo il campo che indica il numero di figli "diretti" (ovvero si considera solo il figlio left + i suoi fratelli).

Note: Cabodi è tranquillo sulla parte di teoria, ho sbagliato la complessità di B.F. ma avendogli spiegato bene l'algoritmo era soddisfatto.

Pasini ti lascia il tempo di scrivere e ragionare, consiglio di soffermarsi a pensare alla strategia giusta da adottare prima di buttarsi sul codice. Se la programmazione dovesse andare male (nel mio caso non sono riuscito a imboccare in nessuno dei problemi la strada corretta) non demoralizzatevi, se fate bene la parte di teoria dovreste passarlo.

26->30 con 2 punti lab (01/03/2023)

Teoria (Camurati, circa 15 minuti per entrambi):

- Rango in un BST, BST select, complessità, informazioni da aggiungere ad albero
- Grafi bipartiti: definizione e esempio di applicazione
- Codici senza prefisso e Huffman

Programmazione (Palena, non sono riuscito a quantificare il tempo, eravamo in tre ad alternarci):

- Definire ADT di I classe lista, ed effettuare n rotazioni di lista senza creare nodi aggiuntivi (spiega lui cosa si intende per rotazione di lista, ovvero l'ultimo elemento va in testa alla lista, e la lista termina al penultimo)
- Lancio di 6 dadi, bisogna ottenere almeno un doppio e superare una certa soglia come somma. Definire modello combinatorio adatto a risoluzione (combinazioni con ripetizioni), impostarlo, supponendo come date le funzioni di verifica per le due condizioni, e impostare del pruning (in questo caso, solo sulla condizione della soglia di somma, perché sul doppio il pruning non si può attuare, essendo una condizione che si può avverare anche sull'ultimo lancio di dadi)

18->19 con 1.25 punti lab (01/03/2023)

Teoria: (Cabodi)

1- Tabelle di hash: Cos'è il linear chaining? Che complessità ha? Qual è il caso peggiore?

2- Algoritmo di Kosaraju: Che cos'è e cosa fa? I DAG possono essere preda dell'algoritmo di Kosaraju (la risposta è "no", perché servono dei grafi orientati con dei cicli, altrimenti le componenti fortemente connesse sarebbero tutti i singoli vertici)

Programmazione: (Pasini)

1- Data una lista del tipo

3>5>8>9

con almeno due elementi, scrivere un programma tale da generare i "nodi mancanti" per riempire la lista

3>**4**>5>**6**>**7**>8>9

2- Dato un grafo, generare le sue bipartizioni. Il grafo va rappresentato con una matrice di adiacenza o con la lista di adiacenza (risposta: con la matrice di adiacenze)? (Credo sia un problema di calcolo combinatorio da risolvere con le partizioni)

/*Ho fatto particolarmente schifo all'esame, ma evidentemente non abbastanza da essere bocciato. La seconda domanda di teoria l'ho sbagliata perché non mi sono spiegato bene (curate l'esposizione, fa veramente tanto la differenza) e nella seconda domanda di programmazione mi sono fatto spiegare cosa fosse un grafo bipartito perché non ne avevo mai sentito parlare. Un professore severo mi avrebbe dato in pasto ai cani in campagna, ma nè Cabodi nè Pasini lo sono e non vi umilieranno in alcun modo, quindi state tranquilli sotto questo punto di vista. Vi assicuro che è stato uno tra i peggiori esami che abbia mai dato a livello di performance. Pasini mi guardava schifato ogni volta che aprivo bocca, probabilmente stavo spiegando i miei ragionamenti nel modo più contorto e meno intuitivo del mondo, ma è stato comunque clemente (proprio per questo vi ripeto di curare l'esposizione, dovete spiegare bene quello che sapete). Nel primo esercizio (nonostante credo fosse giusto a livello funzionale) mi ha detto "Non c'è un solo motivo per farlo così, c'era una maniera più semplice". Insomma, probabilmente mi hanno odiato dall'inizio alla fine, ma alla fine l'ho passato. E se l'ho passato io, potete riuscirci anche voi, ve lo posso assicurare.*/

19->19 con 1,75 punti lab (01/03/2023)

Teoria (cabodi)

- Definizione di programmazione dinamica
- Classificazione degli archi in un grafo

Programmazione (Palena)

- Definizione di Stack e Implementazione di uno stack-push
- Trovare i 10 valori minimi in un BST

27->29 con 1.75 punti lab (01/03/2023)

Teoria (Cabodi) (è buono, innegabile, solo che ti spara domande a profusione ed è molto botta e risposta, nel mio caso ⅔ le ho trovate rognose):

- Interval BST, definizione della struttura, come si esplorano e tutto il contenuto della struct nodo degli interval bst
- Complessità per ottenere un grafo trasposto, sia con MADJ che con LADJ.
- Cosa sono i grafi bipartiti, complessità di un algoritmo per verificare se un grafo è bipartito o meno, date le due partizioni ipotetiche

Programmazione (Palena):

- ADT di I classe lista, ed effettuare n rotazioni di lista (sgancio l'elemento in coda e lo inserisco in testa)
- Lancio di 6 dadi, bisogna ottenere almeno due valori uguali e superare una certa soglia T come somma. Definire modello combinatorio adatto a risoluzione (combinazioni con ripetizioni), impostarlo, supponendo come date le funzioni di verifica per le due condizioni, e impostare del pruning (in questo caso, solo sulla condizione della soglia di somma, perché sul doppio il pruning non si può attuare, essendo una condizione che si può avverare anche sull'ultimo lancio di dadi)

voto?

Teoria (sempre Camurati):

- Condizioni per effettuare heapify e complessità
- Definizione di kernel DAG, e in seguito a risposta, definizione di componente fortemente connessa
- Complessità della ricerca dicotomica, e equazione alle ricorrenze

Programmazione (sempre Palena):

- Implementazione di heapify
- Definire struttura per hash con open addressing, e effettuare inserimento con double hashing

22->24 (24/02/23)

Teoria (Camurati):

- Rango in un BST, BST select, complessità, informazioni da aggiungere all'albero.
- Tabelle ad accesso diretto, complessità operazioni, perchè in pratica non vengono utilizzate.

Programmazione (Pasini):

- Funzione che riceve due liste singolo linkate, ordinate e con possibili elementi ripetuti e che restituisce una nuova lista ordinata che contiene gli elementi di entrambe le liste senza duplicati.
- Contare i cammini semplici di lunghezza k.

23->26(con 2 punti di Lab), 18/02/2023

Teoria (Camurati):

- Funzioni ricorsive (stack frame)

- Strutture dati per la ricerca di alberi ricoprenti minimi
- Definizione di componente connessa

programmazione (Palena) :

- Verificare se due alberi binari sono isomorfi
- Stampare tutti i possibili risultati lanciando 6 dadi (la loro somma doveva essere >= di un certo valore e ci deve essere almeno un valore doppio)

24 -> 29 (con 2 punti di LAB, 18/02/2023)

Teoria (Camurati):

- Come terminare una ricorsione una volta trovata una soluzione? (voleva sapere le flag), se in un algoritmo iterativo ho una flag la complessità è o grande o theta?
- Primary clustering, cosa sono e cosa riguardano (sono i cluster delle tabelle di hashing con open addressing e linear probing). Come fare ad evitare che si formino primary clusters? (faccio quadratic probing o double hasing)
- Cos'è un kernel DAG

Programmazione (Pasini):

- Date due liste ordinate che possono avere duplicati (sia nella stessa lista sia tra le due liste) scrivere una funzione di merge che crea una nuova lista senza duplicati
- Dato un grafo orientato e non pesato, contare tutti i cammini di lunghezza k che partono dal nodo (van bene cammini qualsiasi, non devono essere Hamiltoniani o Euleriani)

Note: Camurati è serio ma comunque ti da tempo per pensare. Se sbagli qualcosa non te lo dice esplicitamente ma cerca di farti ragionare in modo che tu possa accorgerti da solo dell'errore. Pasini è più tranquillo

, ti lascia scrivere il codice e poi te lo fa spiegare (volendo puoi anche spiegarlo mentre scrivi). Poi ti dice se ci sono errori o se c'era un modo più semplice per rispondere alla domanda.

23 → ?? (17/02/2023)

Teoria (Camurati):

- Definizione di prefisso / codici liberi da prefissi / Huffman
- Cammino di Eulero
- Complessità della DFS; (theta(V+E) con lista adiacenze, V per l'inizializzazione ed E per la visita ricorsiva; theta(V^2) con matrice adiacenze)

Programmazione (Cabodi):

- Creare una funzione che, data una lista, restituisca una seconda lista composta di tutti e soli gli elementi facenti parte della progressione geometrica (primo, secondo, quarto, ottavo ecc.)
- Dato un grafo, verificare se esistono cicli di lunghezza 2 (a→b, b→a) e se esistono eliminare quello a peso minore / estensione del problema su un ciclo triangolare (a→b, b→c, c→a) cosa fare nel caso in

cui ci siano 2 triangoli con un lato in comune (questa era, a suo dire, una domanda “trabocchetto”, voleva gli rispondessi chiedendogli cosa voleva che facessi tra eliminare comunque il lato trovato, fare prima un controllo, o altre soluzioni possibili).

18 → 20 (+1 lab) (18/02/2023)

Teoria (Camurati):

- Clustering: definizione, dove si ha, come si forma
- Heap: definizione e proprietà
- Se hai un albero binario completo di altezza data h, qual è il numero di nodi in un generico livello?

Programmazione (Pasini):

- dato un vettore di interi e una lista di interi, per ogni valore appartenente al vettore, elimina il nodo avente lo stesso valore
- dato un albero ternario inizializzato, crea una funzione che calcoli la lunghezza di tutti i cammini dispari.

Pareri miei personali: Pasini ti da l’esercizio e ti lascia il tuo tempo per scrivere il codice. Quando hai finito ti chiede di spiegare le tue scelte e nel caso in cui ci fossero degli errori te li sottolinea e basta.
Camurati: se dai una definizione poco precisa, ti ferma e tramite un esempio ti fa capire che c’è qualcosa che non va. Dopo aver capito il tuo “errore” gli puoi ridare la definizione corretta.

26 → 30 (+2 lab) (17/02/2023)

Teoria (Camurati):

- stack frame
- equazione alle ricorrenze della sequenza di Fibonacci
- cancellazione ricorsiva da una lista

Programmazione (Pasini):

- invertire una lista in loco in maniera ricorsiva (poi ha precisato che andava bene anche fatto iterativamente).
- contare tutti cammini semplici di lunghezza k a partire da un vertice dato (l’ho implementato tramite una variante dell’algoritmo per trovare i cammini hamiltoniani).

27→ 30 (+2 lab) (17/02/2023)

pasini:

prima di scrivere qualsiasi linea di codice rifletti e dialoga con lui spiegando la tua idea

NOTA: fa usare le funzione di libreria tipo q init ecc..

1. data una lista singola linkata, la classica solo con puntatore a next, che potrebbe terminare a null, così come richiudersi su se stessa, ritornare null se termina a null o ritornare il nodo su cui si richiude es 1->2->3->4->2->3

   va risolto con un doppio for, il primo che itera sulla lista nodo per nodo, il secondo calcola la distanza tra la testa di lista e la prima istanza del nodo, questa distanza va poi controllata con quella precedente, se risulta minore allora la lista si richiude e ritorni il puntatore al nodo nel for principale, altrimenti continui fino a fine lista e fuori dai for fai un return null

2. dato un grafo (il tipo a scelta tua), un nodo di partenza start e un raggio k, determinare tutto il vicinato di start, ovvero tutti i nodi raggiungibili a partire da start e con raggio k

   ho utilizzato una bfs, devi solo stare attento a mettere nel while il contatore che sia minore o uguale a k per poter terminare prima puoi anche non usare il vettore dei tempi, non ti interessa

camurati:

1. cos'è un dag
2. topological sort

3. b

20->21 (02/03/2023)

•Teoria (Camurati):

Complessità Quick Sort ed equazione alle ricorrenze nel caso peggiore. (O(N^2), T(N)=N + T(N-1) +1). Quando avviene il caso peggiore. (Quando si sceglie un pivot che ti fa scorrere tutto l'array). Dopo aver scelto il peggior pivot la complessità rimane la peggiore per tutte le operazioni? (No, solo per un iterazione, bisognerebbe scegliere ogni volta il peggior pivot).

Cosa è un grafo bipartito. (Un grafo suddiviso in due insiemi i cui vertici di un insieme puntano solo ed esclusivamente ai vertici dell'altro insieme tramite gli archi). Cosa può rappresentare? (La relazione insieme di macchine-insieme di compiti da svolgere).

•Programmazione (palena)

1)Scrivere le strutture dati per implementare una lista e una funzione (il prototipo c'è già e ha un puntatore a wrapper lista) che elimina i nodi con un intero dispari, non si possono usare funzioni di libreria devi scrivere tutto (tipo la delete);

2)Scrivere la struct dell'hash table e la stInsert per il double hashing (solo la Stinsert, le funzioni di hashing sono già fornite ma bisogna scriverne i prototipi, occhio a conoscerle perchè a me le ha chieste);

21->26 (+1.75 Lab) (15/02/23)

•Teoria (Camurati):

BST (definizione), come trovare l'altezza di un BST ricorsivamente, metodi per rappresentare un grafo

•Programmazione (Pasini):

intersezione insiemistica di due vettori ordinati (si poteva fare in O(n) scandendoli parallelamente), contare i cammini di lunghezza k in un grafo.

22+2 -> 26 (15/02/2023)

- Programmazione (palena)
  - Scrivere la struct dell'hash table e la stInsert per il double hashing (solo la Stinsert, le funzioni di hashing sono già fornite)
  - Scrivere ADT di I CLASSE lista, dato il prototipo void list_intersezione(list l1,list l2){...} trovare le intersezioni tra le due liste, modificando la lista1
    esempio:
    lista1 3->4->1->2

    wta2 2->4\\1+
    risultato (dopo modifica della lista1) 4->2
- Teoria (Camurati)
  - cosa è un heap? cosa è un albero quasi completo? quando posso applicare una heapify?

- Tabella ad accesso diretto, costi operazioni di ricerca, inserzione. Perché è sconveniente? Sulle stringhe piccole va ancora bene?
- Cosa sono gli archi back e come posso individuarli

  Palena ti fa ragionare e mette a proprio agio, non ti interrompe subito se sbagli e ti lascia un po per vedere se ti accorgi dell'errore. Ti lasciano ragionare, quindi secondo me meglio pensare a quello che si dice prima di parlare, la maggior parte delle domande ci si arriva se non ti preticipi immediatamente a parlare giusto per dare fiato alla bocca. In bocca a lupo a tutti

15->20 (15/09/2022)

Teoria (Camurati):

- Che cos'è la Relaxation
- Che cos'è la Memoization
- Che cos'è un IBST

Programmazione (Pasini):

- Implementare una funzione merge tra due liste linkate in avanti ordinate per chiave del nodo crescente. Gestire i valori duplicati effettuando l'inserimento di uno solo dei due elementi duplicati.

  Il prototipo fornito è: LISTA f(LISTA l1, LISTA l2, …);

- Implementare una funzione la quale, dato un grafo connesso e non orientato, conta quanti cammini semplici esistono da un vertice sorgente (s) a tutti i vertici del grafo con lunghezza al più uguale a len.

  Il prototipo fornito è: int f(G g, int s, int len);

20->24 (12/07/2022)

Programmazione (Cabodi):

- Data una matrice sparsa implementata come ore in r,c sia -v (che quindi sommato a v dà zero) eliminare il nodo della lista.
  - se non esiste alcun valore in r,c aggiungerlo alla lista
    - si tratta semplicemente di scorrere delle liste. Mi ha detto che potevo utilizzare semplicemente delle funzioni di libreria. Mi ha fatto scrivere pochissimo tipo l'intestazione di un for. Mi ha poi chiesto nei vari casi cosa avrei fatto senza scrivere codice (sopratutto per cancellare un nodo)
- Dato un grafo, cercarne i triangoli (ovvero cicli di lunghezza 3)
    - la mia idea di getto è stata di implementarlo utilizzando le liste di adiacenze e di ciclare su ogni nodo della lista, per ogni nodo ciclare su ogni arco, e andare ancora a ciclare sull'ar co. mentre stavo scrivendo il secondo for ho capito che la complessità sarebbe stata altissima, probabilmente cubica. L'ho detto, ma lui mi ha risposto che andava bene lo stesso. Non mi ha fatto scrivere oltre l'intestazione del terzo for. Mentre scrivevo mi è

poteva utilizzare una dfs e ad ogni arco back verificare se la distanza fosse == 3 e che ritornasse al vertice di partenza, gli ho detto semplicemente anche questa idea e bom.

Commento: Cabodi mette a proprio agio e lascia tutto il tempo necessario per ragionare, anche facendosi disegni su un foglio di carta. Quello che gli interessa davvero è capire come ragioni, a me ha fatto scrivere pochissimo codice. Se sai spiegare bene e giustificare quello che vuoi fare, se sai programmare un minimo sei apposto. Mi sento di consigliare di esercitarsi anche su questo aspetto, il saper spiegare con un discorso abbastanza coeso.

Teoria (Camurati):

- Delete da una tabella di Hash nel caso di Open Addressing
- Codici liberi da prefissi. Un codice a lunghezza fissa deve necessariamente essere libero da prefisso? (Ho risposto di sì, in quanto se i codici sono a lunghezza fissa non esistono altri codici a lunghezza minore che possano essere prefisso, non mi ha corretto quindi presumo sia giusto il mio ragionamento)
- Rango di una chiave in un BST e come si trova. Come faresti a trovarla in un BST in cui non hai a disposizione il numero di nodi presenti nel sotto albero sinistro? (Visita INORDER per poter ottenere in ordine crescente le chiavi del BST, così si può selezionare la piu' piccola) Perchè è' piu' conveniente usare la prima tecnica? (Sostanzialmente se hai un albero bilanciato allora si paga un costo O(logn) mentre con la seconda si deve mettere il BST in un vettore e scandire il vettore con costo O(n), anche qui non mi ha corretto quindi suppongo sia giusto)

21->23(12/07/2022)

Teoria(Camurati):

- Bellman-Ford, perchè si fanno V-1 iterazioni
- Programmazione dinamica, cosa si intende per sottostruttura ottima
- Come determinare i punti di articolazione di un grafo con un algoritmo(si usa dfs)

Programmazione(Palena):

- Data una lista, un min ed un max, rimuovere gli elementi inferiori del min o maggiori del max
- Un esercizio con una sequenza da implementare con le disposizioni ripetute e un eventuale implementazione con Pruning

25->26 (12/07/2022)dinamica

programmazione:

Ricerca in un trinary search tree, un albero dove ogni nodo ha 2 chiavi k1 e k2 con k1 < k2, nei rami a ciclisinistra stanno i valori minori di k1 , a destra quelli maggiori di k2 e nel ramo centrale i valori compresi.

In un grafo con lista delle adiacenze, eliminare un vertice,

teoria:

LIS come risolverla senza programmazione dinamica

DAG cosa sono, tipi di nodi, ordinamento topologico, topologico inverso, e come generare un ordine topologico inverso.

Code a priorità, operazioni, costo delle operazioni nel caso di vettore non ordinato

15->18 (11/07/2022)

Teoria:

- Programmazione dinamica, a quali problemi si applica e quando è possibile utilizzarla
- Dijkstra
- Algoritmi greedy
- Huffman e codici senza prefisso

Programmazione

- A partire dalla struttura dati di una lista definita come ADT di 1° classe definire la struttura di uno stack e implementare le funzioni di push e pop
- Definire la struttura dati di un BST come quasi ADT e poi implementare la funzione di estrazione del minimo iterativamente e la funzione di estrazione del massimo ricorsivamente

16 -> 18 (11/07/2022)insertion sort

Teoria (Camurati):

- SCC e DAG: spiegare SCC e correlazione con DAG oltre a definizione. Cosa succede se unisco i nodi di una SCC?
- Algoritmi di ordinamento: concetto di stabilità. Insertion sort stabile?
- Differenza tra matrice e Lista adiacenze

Programmazione (Pasini):

- Dato un grafo (indifferente madj o lista adj), contare il numero di cicli semplici a partire/terminare solo ed esclusivamente dal/al nodo di interesse

18->19 (11/07/2022)

Teoria (Camurati)

- Funzione Heapify e condizioni necessarie per il suo utilizzo

  Perché è conveniente implementare un heap attraverso un vettore?
- Limite inferiore lasco degli algoritmi di ordinamento e sua dimostrazione
- La programmazione dinamica è applicabile per la ricerca di un cammino minimo?

- sottostruttura ottima dei cammini minimi e massimi (se esiste, dimostrazione)

Programmazione (Pasini)

- intersezione insiemistica di due vettori prototipo: int*f(int*v1,int*v2,int d1, int d2)
- calcolo dei nodi completi compresi tra due livelli di un albero (BST b, int l1, int l2)

19 -> 21 (11/02)7/202

Teoria (Camurati):

- Rango di un BST
- Come fare una select (con rango) se non ho un Order-statistic BST (semplice visita in-order per ordinare i nodi)
- Visita in profondità e componenti connesse

Programmazione (Pasini):

- Data lista di interi crescenti, aggiungere tutti gli interi consecutivi mancanti (es. 1-4-10 -> 1-2-3-4-5-6-7-8-9-10).
- Dato un albero rappresentato come Left-child Right-sibling gia riempito, per ogni nodo riempire il campo che indica quanti figli (diretti) ha

18 -> 20 (06/07/2022)

Teoria (Camurati):

- Algoritmo di Dijkstra, che tipo di strutture dati usa, e che differenza c'è con le strutture dati utilizzate nella bfs (Dijkstra usa code a priorità, bfs usa code semplici)
- Cosa vuol dire relaxation
- Cos'è un DAG, che tipo di nodi particolari ci sono, che tipo di ordinamento possiamo avere
- Complessità della rotazione in un BST

Programmazione (Cabodi):

- Dato un BST, i cui nodi hanno memorizzato anche il puntatore al padre, partendo da un dato nodo, contare quanti sono i nodi allo stesso livello che hanno in comune il padre del padre
- In un grafo non orientato, dati due nodi, trovare, se esiste, un cammino che li connette di lunghezza pari a 3

??->?? 06/07/2022

Teoria (Camurati):

- Open addressing, vantaggi e svantaggi delle tre opzioni
- Cos'è un MST e quanti archi ha
- Qual è il caso peggiore nel quicksort ed equazione alla ricorrenze di questo caso

Programmazione (Pasini):

- Data una lista linkata scrivere una funzione che verifichi se una lista si ripiega su se stessa alla terminazione in un qualsiasi nodo precedente e ritorni il nodo in questione, altrimenti NULL
- Grafo orientato e non pesato, scrivere una funzione che ritorni un vettore con l'elenco dei vertici distanti al massimo k dal vertice v (bfs)

19 -> ?? 06/07/2022

Programmazione:

- Trovare il figlio con chiave k e portarlo in radice del BST e rendere tutto il BST come figlio destro della nuova radice
- Dato un grafo orientato verificare che fosse anche un grafo non orientato con matrice e lista delle adiacenze

15-> ?? (06/07/2022)

Programmazione (Cabodi) :

- crea una funzione che determina lo sbilanciamento di un BST (sbilanciamento = profondità max - profondità min)
- data una stringa di caratteri 1)elimina le vocali 2) trova le coppie adiacenti per ASCII crescente

Teoria (Camurati) :

- cosa è un grafo completo e numero di archi (il numero è uguale se il grafo è orientato o meno? spoiler = no)
- caso peggiore del quicksort (spiega cosa succede + eq. ricorrenze)
- double hashing

15 -> 18 (13/04/2022 - appello straordinario)

Programmazione (Cabodi) :

- Impostare brevemente ADT BST, e funzione oneChildCount che conta tutti i nodi che hanno uno ed un solo figlio
- Dato un grafo, verificare l'esistenza del ciclo a->b->c->a

Teoria (Camurati) :

- Complessità inferiore algoritmi di ordinamento
- Bellman-Ford e perché fornisce soluzione ottima in V-1 passi
- Tutti i casi di gestione di collisione in hashing
- Vantaggi del quadratic probing e del double hashing rispetto a linear probing (clustering)

17 -> 18 (13/04/22 - appello straordinario)

Programmazione (Cabodi):

- Numero di foglie ad un determinato livello in un BST
- Cammino semplice hamiltoniano in un grafo

Teoria (Camurati):

- Programmazione dinamica (parlare in generale)
- Complessità DFS e vantaggi/svantaggi tra matrice e lista di adiacenza
- Rango di un BST ed a cosa serve

18->?? (13/04/2022)

Programmazione(Pasini)

- date due liste di interi ordinate, fonderle in una lista mantenendo l'ordine
- generare sequenze di lettere minuscole senza ordine e senza ripetizione dove il massimo di vocali consecutive è 3

Teoria(Camurati)

- Altezza e profondità in BST cosa significa
- Cos'è una coda a priorità con relative funzioni
- Codici senza prefissi

16->21 (23/02/22)

Programmazione (Pasini)

- Data lista di interi crescenti, aggiungere tutti gli interi consecutivi mancanti (es. 1-4-10 -> 1-2-3-4-5-6-7-8-9-10).
- lancio K dadi a 12 facce, stampare tutti i lanci tali per cui la somma è al massimo S (combinazioni ripetute).

Teoria (Camurati)

- cos'è un BST, valori minimo e massimo dell'altezza (log n, n)
- Bellman-Ford, perché applicabile la programmazione dinamica, sottostruttura ottima di cammino minimo (dimostrazione)
- Open Addressing , Quadratic Probing, vantaggio sul Linear Probing (evita clustering).

27->30L (23/02/2022)

Teoria (Camurati)

- Equazioni alle ricorrenze per le disposizioni semplici, formula generale

- Teorema degli alberi ricoprenti minimi

Programmazione (Patti)

- Implementare le funzioni PQinsert() e PQextr_max() per coda a priorità. Specificare la struttura utilizzata. (io ho scelto di implementarla con un heap di Item)
- Dato un grafo non orientato né pesato, un suo vertice v e la distanza k, stampare tutti i cammini semplici lunghi esattamente k

15->18 (senza lab) (23/02/2022)

Programmazione (Pasini)

- Data una lista di interi con il formato "1 1 1 5 6 6 6 6 6 7 8 8 9 9 9" (ordinata, con possibilità di chiavi uguali consecutive) dopo aver definito il tipo lista, implementare la funzione (prototipo: "void f(L l)") che data in input la lista elimini le chiavi uguali lasciandone solo una
- Dato un albero binario, 2 interi che rappresentano un intervallo di livelli (estremi compresi) e un intero c, implementare una funzione (prototipo: “int conta(BT bt, int l1, int l2, int c”) che conta tutti i valori maggiori di c contenuti nell'intervallo di livelli dato

Teoria (Camurati)

- Complessità minima degli algoritmi di ordinamento basati sul confronto (con dimostrazione).

  Questo limite si rappresenta con O grande, Theta grande o Omega grande?
- Modi di rappresentazione dei grafi

  Quando è vantaggioso usare la lista e quando la matrice

  Complessità di verifica di connessione tra i due vertici nel caso della matrice e nel caso della lista
- Weighted quick union

  Differenza con la quick union, qual è il vantaggio

  Complessità

18->24 (+2 laboratorio) 22/02/2022

Teoria (Camurati):

- Come evitare nel Quick sort il caso peggiore (devi permutare il vettore così ottieni una probabilità di 1/n! di ottenere un vettore ordinato nell'ordine opposto a quello che vuoi ordinare)
- Etichettatura Archi nei grafi orientati
- Linear Probing e Quadratic Probing

Programmazione (Palena)

- Implementare inserzione con Linear Probing
- Dovevo generare una sequenza di numeri decimali tali che se un numero fosse disparo e quello dopo fosse divisibile per 3 e che non abbia un numero di valori pari maggiore di un parametro T, l'ho fatto con disposizioni ripetute e pruning

17->19 senza Lab (08/02/2022)

Teoria (Camurati)

- Applicabilità programmazione dinamica
- Dijkstra adotta questo approccio?
- DAG e topological sort
- Etichettatura archi DAG

Programmazione (Cabodi)

- Contare i nodi di un BST con chiave che rispettava dei parametri. La funziopriorine di validazione era già data e ritornava successo o insuccesso
- La seconda non la ricordo precisamente, riguardava concatenazioni di stringhe con di lunghezza massima che rispettavano una condizione, il modello preciso da usare erano le permutazioni semplici

18->20 (23/02/2022)

Teoria (Camurati) (monofacciale ma comprensivo, non infierisce per il gusto di farlo)

- Come funziona Heapbuild e complessita', proprieta' degli heap e funzione di heapify
- Priority queue, le varie implementazioni e in base a quelle i costi delle diverse operazioni
- Dijkstra, che approccio usa e di quali strutture dati fa uso (voleva sapere principalmente della coda a priorita')

Programmazione (Patti) (anche lui caruccio, ti lascia il tuo tempo e non imbastisce)

- Data una matrice e noti numero di righe e colonne, creare un vettore di liste che copi il contenuto della matrice eliminando gli zeri e che salvi in ogni nodo sia il valore sia il suo indice di colonna
- Dati i numeri da uno a nove, trovare tutte le terne per cui n0 = pari, n1 = dispari e n2 = pari e tali per cui n0>n1>n2

20->20 (08/02/2022)

Teoria (Cabodi)

- Definizioni riguardo dei BST (altezza, profondità, albero completo, albero bilanciato)
- Cammini minimi, complessità Bellman-Ford e Dijkstra (a seconda di come è rappresentato il graintersezionefo), relaxation

Programmazione (Palena)

- Implementare tabella di simboli come ADT di prima classe, funzione di ricerca in tabella con linear probing
- Implementare BST, inserimento in radice

16->21 (+2lab 8/02/2022)

Teoria(Camurati)

- Heapsort come funziona e costo
- Componenti fortemente connesse
- Bst definizioni e rango di un bst
-

Programmazione (Patti)

- Inserimento lista pari e dispari dato un vettore di dimensione n
- BST funzione che conta quanti hanno un figlio destro e figlio sinistro

18->22 con 2 punti bonus (08/02/2022)

Teoria (Camurati)

- HeapBuild (complessità, a cosa serve, cos'è un heap)
- Heapify (in quali condizioni si può usare)
- Rotazioni nei BST (cosa sono, a cosa servono e la loro complessità [spoiler: unitaria])

Programmazione (Pasini)

- Dati due vettori v1 e v2 e le rispettive dimensioni, scrivere una funzione che inserisca in un terzo vettore la differenza insiemistica (unione meno intersezione) e ritorni la sua dimensione
- Dato un albero, verificare che sia un Sum Tree (la somma di tutti i discendenti di un nodo è uguale al valore del nodo stesso per tutti i nodi)

21->23 (27/01/2022)

Teoria (Cabodi)

- Funzione Partition nel Quick Sort e la sua complessità
- Equazione alle ricorrenze del Quick Sort (spiegare il significato dei vari termini)
- Activity selection con paradigma Greedy

Programmazione (Palena)

- Implementare l' ADT di I classe STACK con funzione di push

- Risoluzione di un problema con i modelli del calcolo combinatorio (dati n interi, determinare le sequenze lunghe k in cui:

  - un numero pari è sempre seguito da un numero divisibile per 3

  - sono presenti almeno T numeri dispari.

23->24 (08/02/2022)

Teoria (Camurati) :x

- Codici prefissi
- Paradigma Greedy
- LIS (Longest increasing sequence)

Pratica:

- grafo trasposto lista adiacenze
- inserzione in testa
- disposizioni ripetute con pruning

Commento: La prima domanda di Camurati non la sapevo, lui super comprensivo mi ci ha fatto arrivare con il ragionamento. Patti per programmazione mi ha dato tutto il tempo del mondo e non è stato per nulla pignolo.

22->25 (27/01/2022)

Teoria (Cabodi):

- Partition di un BST
- Delete di un bst e quali possono essere i problemi legati alla cancellazione
- Teorema degli archi sicuri, spiegando un po' gli algoritmi di Prim e Kruskal

Pratica (Pasini):

- Data una lista, rimuovere tutti gli elementi di posizione pari
- Dato un grafo G, rappresentato come lista delle adiacenze o matrice delle adiacenze, generare un bipartizionamento, se esiste, e prendere quello con differenza di elementi dell'insieme minima

  Il bipartizionamento di un grafo è dato da due insiemi di vertici che non hanno vertici in comune con quelli dell'insieme stesso.

  Per risolverlo si poteva usare un powerset con disposizioni ripetute in modo da assegnare a tutti i vertici del grafo una partizione e poi era richiesto di implementare la funzione di controllo per vedere se i due insiemi davano una bipartizione del grafo

6+15->18 (14/09/2021)

*Teoria (Camurati) :

- Hash su stringhe
- Hash su numeri razionali (formula dell'intervallo con min e max)
- Fattore di carico (alfa=N/M) e quanto deve essere in open addressing (alfa=0.5 perché M deve essere grande almeno il doppio di N per evitare tante collisioni)
- Algoritmi in loco/stabili
- Complessità minima algoritmi di confronto: O(n*logn) (richiede la dimostrazione)

*Pratica:

- Code
- Ricerca su tabelle di hash con linear chaining

15->18 (+1,6 punti di laboratorio, 13/09/2022)

- *Programmazione (CABODI)*
  - Dato un BST con nodi contenenti un tempo nel formato (hh:mm) e un contatore c, inserire un nodo nel BST e aggiornare il contatore.
  - Dato un DAG e un vettore di interi, verificare che il vettore rappresenti un ordinamento topologico del DAG.

NOTA: se hai difficoltà, cerca di aiutarti e di farti arrivare alla soluzione con il ragionamento. Se sai come applicare le varie nozioni non è tosto, ed inoltre è più umano di quanto mi aspettassi.

- *Teoria (CAMURATI)*
  - Tipologie di archi
  - Programmazione dinamica
  - Interval BST

NOTA: Same as above. E come aveva detto qualcuno qui, "monofacciale ma comprensivo".

15->18 (14/09/2021)

Teoria (Cabodi):

- Quick find e quick union, differenze, funzionamento e complessità
- Hashing (utilizzo e nozioni generali): cos'è un cluster, come vengono gestite le collisioni, quali metodi convengono per gestirle e perchè
- etichette di un arco, spiegare quali sono e come si assegnano (tree, back, forward e cross), un albero con soli archi Tree forma un minimum spanning tree?

Programmazione (Pasini):

- funzione che inserisce in una lista dei punti cardinali ordinati per distanza dall'origine
- dato un grafo e un suo vertice di partenza determinare tutti i cammini semplici di lunghezza k che partono da quel vertice dato (da fare con la ricorsione)

15 -> 19

Teoria (Cabodi):

- Interval BST
- Come si vede un ciclo in un grafo
- Zaino discreto e continuo con algoritmo greedy

Programmazione (Patti):

- Dato un vettore di interi, dividerlo in pari e dispari in un vettore di liste ordinate
- Dato un vettore di interi, generare tutti i terzetti con determinate proprietà (primo e terzo pari, secondo dispari e che il primo sia maggiore del terzo e il secondo minore del primo). Sono le disposizioni semplici dove dentro l'if (pos >= k) c'è l'if di check della condizione

Voto 15->18 (01/09/2021)

Teoria:

- cos'è un heap e sue proprietà, spiegare l'heapify, proprietà di un albero per cui può essere un heap
- Cos'è un grafo completo, un grafo connesso e fortemente connesso e quanti archi ha.

Programmazione:

- ADT 1 Classe list (strutture dati) con valore intero + dire cosa metti nel .c e .h e perché + funzione che cancella tutti i nodi con valore dispari.
- Strutture dati per dichiarare un albero binario (Tree) (non per forza adt) + funzione ricorsiva per contare tutti i nodi completi (con entrambi i figli)

15 -> 19(+2pt Lab) (27/02/2022)

Programmazione (Palena):

- Definisci il tipo lista ADT ed una funzione void filter_odd(list l) che elimina tutti i nodi della lista i cui valore è dispari.

- Data una BST che memorizza valori di tipo carattere in [0-9,+,*], scrivi una funzione int eval(list l) che ritorni il valore delle operazioni di calcolazione effettuate sull'intero albero.

  Es. se il parent ha + e i child 1 e 5, output 1 + 5

Teoria (Camurati):

- MST, cosa sono, che algoritmi si usano, su che grafi si applicano e cosa sono Teorema e Corollario
- Codici senza prefisso, cosa sono e a cosa servono
- Come funzionano gli algoritmi greedy

be15 -> 18

Teoria:

- DAG
  -
  - Directed Acyclic Graph: Grafo orientato aciclico. Utilizzato, ad esempio, per operazioni di scheduling (precedenze tramite archi orientati): dati compiti (tasks) e vincoli di precedenza (constraints), come programmare i compiti in modo che siano tutti svolti rispettando le precedenze. Nei DAG esistono 2 classi di nodi: i nodi sorgente (source) che hanno indegree=0 e i nodi pozzo o scolo (sink) che hanno out-degree=0
- Cammino di Hamilton
  - Dato un grafo non orientato, cammino semplice che contiene tutti i vertici. Esempio: tour del cavallo. Complessità: la verifica di un cammino di Hamilton è polinomiale, ma la ricerca è di complessità esponenziale a causa del backtrack.
- Partition di BST
- Riorganizzazione dell'albero avendo l'item con la k-esima chiave più piccola nella radice (codice partR
- Struttura dati per BFS
  - Grafo non pesato come matrice delle adiacenze
  - Coda Q dei vertici grigi (esterna al grafo)
  - Vettore st[] dei padri nell'albero di visita in ampiezza
  - Vettore pre[] dei tempi di scoperta dei vertici
  - <Contatore *time* del tempo>

Programmazione:

- Data una stringa [ciao,bello,come,stai\0] individuare le sottostringhe tra le virgole e inserirle in una lista ordinata
- Dato un BST nel quale per ogni nodo è presente una lista di stringhe, liberare il tutto(free)

15 -> 18

Teoria:lista

- Equazione alle ricorrenze
    - T(n) viene espressa in termini di:
        - D(n): costo della divisione
        - Tempo di esecuzione per input più piccoli (ricorsione)
        - C(n): costo della ricombinazione
        - Si suppone che il costo della soluzione elementare sia unitario Teta(1)
        -
    - a è il numero di sottoproblemi che risulta dalla fase di Divide
    - b è il fattore di riduzione, quindi n/b è la dimensione di ciascun sottoproblema l'equazione alle ricorrenze ha forma: T(n) = D(n) + a T(n/b) + C(n)
- Spiegare parametri e caso di ricerca dicotomica
    - l indice sinistro vettore
    - r indice destro vettore
    - k chiave da ricercare
    - m indice di metà del vettore
    - ad ogni passo: confronto k con elemento centrale del vettore:
        - =: terminazione con successo
        - <: la ricerca prosegue nel sottovettore di SX
        - >: la ricerca prosegue nel sottovettore di DX
- Caratteristiche di un albero: altezza, profondità, grado e rango
    - Profondità di un nodo: lunghezza del cammino fino alla radice
    - Altezza: massima profondità (massimo cammino radice-foglia)
    - Grado: numero massimi dei figli di un qualunque nodo
    - Chiave di rango r (r-esima più piccola chiave)
    -

- Differenza grafo-albero
    - Un albero è un grafo non orientato, connesso e aciclico, ovvero un grafo nel quale due vertici qualsiasi sono connessi da uno e un solo cammino
- Differenza BST-heap
        - Un heap garantisce solamente che il nodo padre sia maggiore o uguale a tutti i discendenti, garantisce quindi un ordine “dall’alto verso il basso” o viceversa. In un BST, invece, tutti i nodi nel sottoalbero sinistro contengono una chiave inferiore a quella contenuta nel nodo in esame (e viceversa), garantisce quindi un ordine “da sinistra a destra”.
- Hash Table ricerca e "mantiene l'ordine?"
- Differenze Dijkstra e Bellman-Ford
    - Dijkstra potrebbe non restituire la soluzione ottima in presenza di archi negativi e ha un risultato senza significato in presenza di cicli negativi; esegue la relaxation una sola volta per ogni arco; strategia greedy
    - Bellman-Ford rileva cicli negativi e garantisce la soluzione ottima anche in presenza di archi negativi; esegue la relaxation |V| - 1 volte per ogni arco; strategia programmazione dinamica
    - Cambia inoltre l’ordine in cui si rilassano gli archi
- Differenze tra paradigma greedy e programmazione dinamica
    - Il paradigma greedy ha alla base l’idea che ad ogni passo si scelga la soluzione che è *localmente* ottima, cioè che abbia appetibilità (che è definita a seconda dei casi, in generale è un costo minimo o un valore massimo) massima in quel momento. In seguito non c’è backtrack, cioè le scelte precedenti non vengono riconsiderate. I vantaggi sono che gli algoritmi greedy sono semplici e veloci, lo svantaggio è che non è sempre garantita la soluzione ottima.
    - La programmazione dinamica vuole “segnarsi” i risultati di sottoproblemi già risolti in modo da non risolvere più volte lo stesso problema. Il processo seguito dalla programmazione dinamica prevede prima la definizione della *struttura* della soluzione ottima. In seguito si definisce ricorsivamente il *valore* della soluzione ottima (non la soluzione in sè, ad esempio nel problema della catena di montaggio il valore della soluzione è il tempo di costruzione, mentre la soluzione vera e propria è la sequenza di stazioni) e si calcola tale valore

bottom-up. Infine si costruisce la soluzione ottima vera e propria

Programmazione:

- Dato un vettore di stringhe dividere in due sottovettori le stringhe che iniziano per voc e cons (ti dà 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 da 0-9 lunghe k per cui vengono rispettati dei vincoli (vuole il pruning)

15 -> 18 programmazione il (4/03/2021) teoria il (11/03/2021) (+1,7 lab)

Programmazione (Palena):

- definire una lista di liste di interi come ADT di 1° classe.
- definire un BST “base” (solo puntatori ai figli) di stringhe dinamiche come quasi ADT.
    - scrivere una free per deallocarlo.

Teoria (Cabodi)

- Scrivere e spiegare l’equazione alle ricorrenze del merge sort.
- Codici di Huffman.
- Grafo bipartito.
- Cos’è un grafo completo, quanti archi ha e perchè.

15 -> 18 (15/07/21) (+0,8 lab)

Teoria (Camurati):

- rappresentazione grafi e complessità
- grafo sparso e denso
- quicksort caso peggiore: equazione delle ricorrenze (T(n)=T(n-1)+n=O(n^2))

Programmazione (Palena):

- quasi adt lista singola di stringhe
- concatenazione stringhe dati lista e separatore
- ricerca dicotomica e sua complessità

15 -> 18 (15/07/21) (senza punti lab)

Teoria (Cabodi):

- Fase del Merge nel merge sort e complessità. è stabile? è in loco?

- colorabilità di un grafo nella visita in profondità (parlare dei vari archi B,T,F,C)
- interval BST
- Approccio Greedy e interval selection

Programmazione (Patti)

- Dato un vettore di interi inserire i numeri in una lista pari e in una lista dispari seguendo l'ordine crescente
- Dato un vettore di interi da 0 a 9 generare tutte le terne possibili tali per cui E1 != E2 != E3 e E1 + E2 + E3 = 15 (poteva essere fatto sia con il modello delle disposizioni sia combinazioni)

15 -> 19 (26/01/21)

Teoria Cabodi:

- Insertion sort e selection sort, complessità caso peggiore e caso migliore.
- Grafo trasposto. Complessità matrice e lista di adiacenza.
- Tabella ad accesso diretto.

Programmazione Patti:

- Dato un vettore, salvare in una lista gli elementi pari e in un'altra quelli dispari, conservando l'ordine nelle liste.

15 -> 18 no punti extra

teoria:

- inserzione heap, cancellazione massimo e complessità
- heapify come funziona e complessità.
- def. relaxation
- relaxation in dijkstra e bellman-ford

programmazione:

- data matrice crearne un'altra float delle stesse dimensioni all'interno della funzione e renderla visibile al main

- void funzione(————————)
- liberazione bst con stringa come nodo

15 -> 18

Teoria:

- Heap
    - In un heap il nodo padre è maggiore o uguale a tutti i discendenti; è un albero binario quasi completo (tutti i livelli completi, tranne eventualmente l'ultimo, riempito da SX a DX), il che implica che sia anche quasi bilanciato
- Huffman
    - Codice: stringa di bit associata ad un simbolo; servono a codificare le lettere dell'alfabeto ad esempio. Approccio greedy per rappresentare i simboli più utilizzati su sequenze di bit di lunghezza minima

Programmazione:

- Merge di due stringhe ordinate in una sola sempre ordinata
- Inserimento in hash con linear chaning senza duplicati del valore in input

15 ->18

Teoria:

- Limite inferiore lasco degli algoritmi di ordinamento basati sul confronto
    - Algoritmi basati sul confronto: confronto a_i e a_j
    - Esito: decisione ($a_i > a_j$ o $a_i <= a_j$), riportata su un albero delle decisioni
    - Per n interi distinti: numero di ordinamenti = numero di permutazioni n!
        - Complessità: numero h di confronti (altezza dell'albero)
        - Ogni soluzione = foglia
        - Numero di foglie = 2^h
        - Approssimazione di Stirling: $n! > (n/e)^n$ quindi $2^h >= n! > (n/e)^n$
        - $h > lg(n/e)^n = n \lg n - n \lg e = (n \lg n)$
- Grado, altezza degli alberic
    - Grado: numero massimi dei figli di un qualunque nodo
    - Altezza: massima profondità

- Componenti fortemente connesse
    - Una componente fortemente connessa di un grafo diretto G è un sottografo massimale di G in cui esiste un cammino orientato tra ogni coppia di nodi ad esso appartenenti

Programmazione:

- Inserimento in un BST, con funzione ricorsiva InsertR

15 ->18

Teoria:

- Tabella di simboli
    - ADT che supporta operazioni di:
        - insert: inserisci un elemento (STinsert)
        - search: ricerca dato con certa chiave (STsearch)
        - delete: cancella il dato con una certa chiave (STdelete)
    - Talora la tabella di simboli è detta dizionario
- Tabella di hash
    - ADT con occupazione di spazio O(|K|) e tempo medio di accesso O(1). La funzione di hash trasforma la chiave di ricerca in un indice della tabella
    - Tabella di hash usata per inserzione, ricerca, cancellazione, non per ordinamento e selezione
- Merge Sort (equazione alle ricorrenze)
    - $T(n) = 2T(n/2) + n$
- Quick sort (caso peggiore)
    - Efficienza legata al bilanciamento delle partizioni. A ogni passo partition ritorna:
        - caso peggiore: un vettore da n-1 elementi e l'altro da 1
        - caso migliore: due vettori da n/2 elementi
        - caso medio: due vettori di dimensioni diverse
    - Bilanciamento legato alla scelta del pivot
    - Caso peggiore: pivot = minimo o massimo (vettore già ordinato)
    - $T(n) = T(n-1) + n$

Programmazione:

- Dato un vettore di stringhe dividerlo e ritornare due vettori. Il primo conterrà le stringe inizianti per carattere, il secondo quelle inizianti con numeri. (Tutto in una funzione)
- Dato un grafo implementare la funzione di GRAPHEdges (dato un grafo ritorna il vettore degli archi)

15 -> 19

Teoria:

- Costo della DFS
    - Matrice delle adiacenze T(n) = ($|V|^2$)
    - Lista delle adiacenze: T(n) = O(|V|+|E|)
- Rappresentazioni dei grafi (matrice o lista di adiacenze) con vantaggi e svantaggi
    - Matrice:
        - Complessità spaziale S(n) = ($|V|^2$) vantaggiosa SOLO per grafi densi. No costi aggiuntivi per i pesi di un grafo pesato
        - Accesso efficiente (O(1)) alla topologia del grafo
    - Lista:
        - Grafi non orientati: elementi complessivi nelle liste = 2|E| Grafi orientati: elementi complessivi nelle liste = |E| Complessità spaziale S(n) = O(max(|V|, |E|)) = O(|V+E|) vantaggioso per grafi sparsi
        - verifica dell'esistenza di arco (v,w) mediante scansione della lista di adiacenza di v. Uso di memoria per i pesi dei grafi pesati
- Tabelle di simboli ad accesso diretto
    - Insieme universo U con M = card(U) = maxN elementi
    - Corrispondenza biunivoca tra ciascuna delle chiavi k e gli interi tra 0 e M-1. L'intero funge da indice in un vettore
- Cancellazione in una tabella di simboli con open addressing (solamente teorico, no codice)
    - Operazione complessa che interrompe le catene di collisione. L'open addressing è in pratica utilizzato solo quando non si deve mai cancellare
    - Soluzioni:
        - sostituire la chiave cancellata con una chiave sentinella che conta come piena in ricerca e vuota in inserzione
        - reinserire le chiavi del cluster sottostante la chiave cancellata

Programmazione:

- Disposizioni semplici (Considerato un vettore di caratteri con tutte le lettere dell'alfabeto, generare tutte le parole di 10 lettere considerando che in ogni parola una stessa lettera può essere ripetuta massimo 2 volte)
- Selection sort su un vettore di puntatori a stringhe (la matrice di stringhe ha lunghezza variabile di riga in riga, a secondo della lunghezza della stringa che vi è su una data riga) (visto che sono puntatori lo scambio può essere fatto direttamente usando un puntatore char* tmp di appoggio, senza dover usare strdup o altro) -Data una matrice di caratteri (vettore di stringhe) allocare dinamicamente un vettore di puntatori a stringhe in cui ogni riga ha la lunghezza che serve e non una lunghezza generica.

15 ->19

Teoria:

- Come si trasformano le stringhe in interi per le tabelle di hash
- Il limite inferiore degli algoritmi basati sul confronto

Programmazione:

- 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 (ho usato una struct con due puntatori, al fratello destro e al figlio sinistro, e mi hanno detto che avrei potuto usare tranquillamente un vettore)

15 -> 19

Teoria:

- Funzione partition su BST
- Tabelle di Hash

Programmazione:

- Date due liste ordinate di stringhe, fonderle in una terza mantenendo l'ordinamento
- huffman
- Determinazione delle scc su un grafo.

15 -> 19

Teoria:

- Come funziona la ricorsione (salvataggio nello stack, indirizzo di ritorno).
- Dov'è lo stack fisicamente e com'è strutturato.
- Algoritmi greedy.
- Backtrack.

Programmazione:

- Vettore di puntatori a struct di tipo punto (non vettore di struct!!!), definire la struct punto, leggere da file i punti
- Data una struttura FIFO (array e vettore) passata per parametro ritornare un counter che indica il numero di elementi e stampare

15 -> 19 (26/01/2021) (+Lab)

Teoria (Cabodi):

- Shellsort. Nel caso migliore, quale fra selection e insertion ha complessità minore? Perché?
- Cammini minimi lunghezza max

Programmazione (Patti)

- Due liste ordinate dove nella prima ci vanno i num. pari e nell'altra i dispari, ritornarli alla funzione chiamante entrambi (usi i puntatori)
- Liberare un BST dove il valore è una stringa (funzione ricorsiva del freeBST)

15 -> 19 (3/03/2021) scritto (16/02/2021)

Teoria(Cabodi):

- Perché la complessità migliore che si può avere negli algoritmi di ordinamento basati sul confronto è linearitmica?

  la migliore possibilità che abbiamo per ordinare un vettore è dividerlo ricorsivamente, ottenendo così un albero di altezza logaritmica da dimostrare(), ed essendo che per ogni nodo devo eseguire un ordinamento in qualche modo, e che i migliori sono O(n), il costo di ricombinazione sarà n, dunque dall'eq alle ricorrenze avrò O(N lg N). In poche parole ho generalizzato un po' le dimostrazioni del mergesort o del quicksort.

  Ha detto che andava bene, anche se penso si possano fare dimostrazioni molto più precise.
- Cos'è un DAG e perché è utile (ordinamento topologico e topologico inverso e brevi cenni sui cammini minimi).
- Come faccio a generare un grafo non orientato che abbia tutti i vertici di grado 1 ?

  Parto da due vertici connessi da un solo arco, e aggiungo solo componenti connesse.

15 -> 19 con teoria il (10/03/2021) e programmazione il (4/03/2021) scritto (16/02/2021)

Teoria (Camurati)

- Cos'è un rango in un BST, come trovare una chiave di rango con la SELECT e cosa aggiungere al BST per questa operazione.
- Costo della rotazione in un BST
- Caso peggiore Quicksort e sua eq alle ricorrenze

  Programmazione (Vendraminetto)
- Creare struttura BST e fare una funzione che conti il numero di foglie

- data una parola con lettere ripetute, generare tutti i suoi anagrammi

Programmazione(Marco Palena):

- Generare un ADT di I classe Lista e creare una funzione che scandisce un vettore di interi e riporta in una nuova lista gli elementi del vettore con lo stesso ordine.
- Generare un BST di I classe con item stringa allocata dinamicamente, e fare la free del bst.

15->19 (Appello del 16/02/2019) (+2 LAB)

Teoria (Cabodi)

1. Tabelle ad accesso diretto, vantaggi e svantaggi di liste e vettori (ricerca e inserimento, discutere i vari casi)
2. Grafo bipartito

Programmazione (Vendraminetto)

1. Determinare l'altezza di un BST
2. Data una stringa, determinare il numero di sottostringhe separate da un carattere separatore ed inserirle in un vettore di stringhe passato by reference alla funzione (è possibile fare due letture, la prima per contare e la seconda per allocare).

16 -> 18 (26/01/2021)

Teoria (Camurati):

- Heap:definizione, come viene implementato, perché è vantaggioso usare un vettore per l'heap rispetto a BST, complessità heap
- Definizione grafo completo

Programmazione (Patti):

- Data una matrice di interi, copiarla in un vettore di liste ignorando gli zeri
- free di un BST

16 -> 18 (26/0 1/2021)

Teoria (camurati):

- Multigrafo
- Complessità merge sort
- merge sort e stabilità/in loco

Programmazione:

data una stringa contenente lettere e virgole. Prendere la virgola come separatore di sottostringhe e mettere ognuna di esse in una lista ordinata

- implementare la bstFree di un bst. Ogni nodo del bst aveva un campo che era una lista

16 -> 18 (26/01/2021)

Teoria(Cabodi):

- codice Huffman: perché i codici creati sono di dimensioni diverse e a cosa è riferita la priorità di ogni elemento
- collisioni nelle tabelle di hash: come gestirle

Programmazione:

- date due liste trasformare la prima nell'intersezione delle due iniziali
- dato un albero n-ario in cui ogni nodo contiene un vettore con i suoi figli calcolare quanti figli ci sono al livello k dell'albero -

16 -> 18 (26/01/2021)

Teoria (Cabodi):

- Complessità della ricerca di un elemento in vettore ordinato (O(logN), dicotomica), vettore non ordinato (O(N)), lista ordinata e non ordinata (entrambe O(N)).
- Perchè non posso usare la ricerca dicotomica nelle liste ordinate (perchè non ha gli indici).
- Algoritmo di Primm, come funziona in generale e quale struttura dati adottare (Coda a priorità).
- Cammini minimi e alberi ricoprenti minimi, parlare della loro relazione e se ad uno corrisponde l'altro (studiare bene le definizioni di ciascuno, è una domanda di ragionamento, non c'è sulle slide).

Programmazione (Patti):

- Praticamente mi ha chiesto di allocare una matrice dentro una funzione (non il main) e far si che il main la vedesse, quindi sostanzialmente dovevo passare per riferimento un altra matrice alla funzione e poi far coincidere i puntatori. Poi mi ha chiesto di fare operazioni semplici (inserimento) sulla matrice originale.
- Dato un BST dentro i cui nodi c'era una stringa allocata dinamicamente, deallocare tutto (albero, nodi e stringhe). (Visita in post order a partire da un wrapper in cui libero l'albero dopo aver chiamato la funzione di visita ricorsiva).

16 -> 18 (16/02/2021)

TEORIA (Camurati):

- Tabella ad accesso diretto cos'è, complessità di inserzione, ricerca, cancellazione, limiti della tabella e come mai è poco implementabile nella realtà (perchè chiavi possono essere teoricamente infinite)
- stack overflow

PROGRAMMAZIONE (Patti):

- Dato vettore di interi (scritto il vettore) creare due liste ordinate contenenti una gli elementi pari e una quelli dispari
- FreeBST

16 -> 18

Teoria:

- Cos'e' la partition e a cosa serve
- Cos'e' il rango di un BST
- Cosa sono gli IBST (e come funzionano)

Programmazione:

- Merge di un vettore di stringhe; Stackpop(stack *s) con stack implementato come ADT1a categoria (Vettore) Find della QuickUnion

16 -> 18

Teoria:

- Codice di Huffman
- Permutazioni

Programmazione:

- 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 BST
- Trovare il grado di un vertice in un grafo implementato con matrice di adiacenze

16 -> 18

Teoria:

- Tabelle di simboli ad accesso diretto, con costi delle operazioni, pro e contro
- Tabelle di hash ed occupazione di memoria.

Programmazione:

- Estrazione da buffer circolare
- FIFO come adt di 1 categoria ed estrazione
- Ricerca in BST
- Svuotare un vettore dai numeri negativi e ricompattarlo (non farlo con due cicli perché lo vuole lineare!)

16 -> 18

Teoria:

- Metodi per implementare una coda a priorità
- Domande su Bst e altezza heap
- Confronto BST e Heap

Programmazione:

- Compattare matrice di interi in un vettore di puntatori a interi eliminando tutti gli zeri
- Problema simile a quello dello zaino (powerset)

16 -> 18

Teoria:

- Codice che calcola l'altezza di un albero binario con paradigma divide et impera
- Shellsort
- Quando un ordinamento si dice stabile e in loco

Programmazione:

- Data una matrice passata come riferimento la media dei valori locali per ogni punto riga colonna e restituire tutto in una nuova matrice anch'essa passata come riferimento
- Implementare funzione heapify.

16->18 (1/07/21)

Teoria (Camurati):

- Alberi ricoprenti minimi, kruskal, prim
- Equazione delle ricorrenze Merge sort
- Select BST

Programmazione (Palena):

- Heapify
- Funzione ricorsiva su un problema di lancio di dadi, con pruning

16->19(15/06/2021, noLab)

Teoria(Camurati):

-Componenti connesse e fortemente connesse

-Cammini minimi e cammini di Hamilton

-Differenza principale tra il cammino di Hamilton e il problema del commesso viaggiatore(sostanzialmente il cammino di Hamilton si calcola su un grafo non pesato, mentre il problema del commesso viaggiatore si applica ad un grafo pesato)

Programmazione(Pasini):

-Codice della Heapify

-Dato un albero binario scrivere una funzione che ritorni il numero di nodi completi

16->19 (30/06/2021, NoLab)

Teoria (Cabodi):

-Cosa è HeapSort, qual è la sua complessità e perché è sconveniente rispetto al QuickSort anche se hanno la stessa complessità.

-Cosa è Relaxation e come e dove viene utilizzata

-Come trovare dei cicli all'interno di un grafo, si possono avere cicli sia in un grafo orientato che non orientato?

Programmazione (Patti):

-Data una stringa aabb;ccbb;ddbb; si faccia una funzione che permetta di dividere le sottosequenze comprese tra i ' ; ' e si inseriscano le liste risultanti in una lista ordinata.

-Free di un BST i quali nodi contengono delle stringhe allocate dinamicamente.

16 -> 20

Teoria:

- Cos'e un MST
  - Minimum Spanning Tree: albero che copre tutti i vertici con i cammini minimi
- Cosa vuol dire greedy
- Bellman Ford
- Dijkstra
- Perché Dijkstra è greedy?

Programmazione:

- Trasformare una lista di stringhe in un vettore di stringhe, poi ordinarlo con un sort a piacere
- Dato un BST e un intero k, scrivere una funzione che elimini a tutti i figli sinistri a partire dal livello k-esimo in giù.

16 -> 20 (26/01/2021)

Teoria (Cabodi):

- HEAPify e Heapsort, complessità e proprietà.
- Teorema degli archi sicuri (MST). Domanda se passo passo si va da un sottoinsieme di MST a un insieme più grande di MST. Risposta: si.

Programmazione (Patti):

- Data una matrice RxC, trasformarla in un vettore di liste (una per riga), in cui vengono salvati tutti i valori non nulli (inserzione in coda) e il relativo indice colonna.
- Dato il vettore delle 10 cifre {0,1,2,3,4,5,6,7,8,9} determinare tutti gli insiemi di 4 numeri in cui:
  - sol[0] pari
  - sol[1] dispari
  - sol[2] dispari
  - sol[3] pari

Inoltre deve valere la condizione sol[3]>sol[0] && sol[2]>sol[1]. Possibilità di implementare sia pruning che check.

16 -> 21

Teoria:

- Grado di un vertice
- DAG
- Punti di articolazione
- Cos'è una tabella di simboli e qual è l'operazione che va ottimizzata al meglio?
    - La ricerca
- Disegnare un grafo composto da un'unica componente fortemente connessa
- Complessità di Heapsort
- Dijkstra e Bellman-Ford.

Programmazione:

- Dato un vettore di stringhe, ritornare 2 vettori di stringhe contenenti uno quelle che iniziano per vocali e l'altro per consonanti
- Ricerca iterativa in un BST.

16 -> 22

Teoria

- Minimum Spanning Tree
- Archi sicuri
- Tagli
- Archi leggeri
- Teorema e corollario degli archi sicuri

TEOREMA: Dato un grafo, connesso, pesato e non orientato. Preso A, un sotto-insieme di E (A incluso in E), inizialmente supposto vuoto, preso un taglio (S, V-S) che rispetta A (cioè che nessun arco contenuto in A attraversa il taglio) e preso un arco leggero (u,v) che attraverso il taglio, allora (u,v) è un arco sicuro per A.

COROLLARIO: Dato un grafo, connesso, pesato e non orientato. Preso A, inizialmente supposto vuoto, preso C (albero della foresta Ga = (V,A)), preso (u,v) un arco leggero che collega C ad un altro albero contenuto in Ga, allora (u,v) é sicuro per A.

- Algoritmi greedy

Programmazione:

- Inversione di lista
- BST: funzione che ritorna il numero di nodi che hanno esattamente un figlio

16 -> prossimo appello (orale 14/09/21)

Teoria (Cabodi):

- Cosa fa la funzione di merge nel Merge Sort
- L'algoritmo di Bellman Ford e considerazioni sulla complessità
- Search nel BST. Ci possono essere nodi con chiavi uguali in un BST? No, perché altrimenti non saprei se devo scendere a dx o a sx.
- Cosa è un DAG?

Programmazione (Vendraminetto):

- Implementare la ricerca dicotomica, a scelta se in maniera ricorsiva o iterativa
- Implementare la DFS dopo aver definito il grafo (non necessariamente come ADT)

Considerazione: è stata onesta come cosa, non ero proprio preparata perché non mi aspettavo di aver passato lo scritto e ho avuto tempo giusto di rileggere le slides.

17 -> 18

Teoria:

- Tabelle di hash
    - Tabelle ad accesso diretto dove la chiave funge da indice di un array; occupazione di spazio O(|K|) e tempo medio di accesso O(1). La funzione di hash trasforma la chiave di ricerca in un indice della tabella. Non usate per ordinamento e selezione
- Valore alfa
    - Alfa: fattore di carico = numero di elementi memorizzati / dimensione tabella
- DAG
    - Directed Acyclic Graph: Grafo orientato aciclico. Utilizzato, ad esempio, per operazioni di scheduling (precedenze tramite archi orientati): dati compiti (tasks) e vincoli di precedenza (constraints), come programmare i compiti in modo che siano tutti svolti rispettando le precedenze. Nei DAG esistono 2 classi di nodi: i nodi sorgente (source) che hanno indegree=0 e i nodi pozzo o scolo (sink) che hanno out-degree=0
- Componente connessa

- Sottoinsieme massimale dei nodi mutuamente raggiungibili

Programmazione:

- Creare una coda a priorità, in cui la priorità è la distanza dall'origine di un punto nel piano cartesiano

17->18 (23/02/2022) (no lab)

Teoria (Camurati):

1)Tra quali estremi può variare l'altezza di un BST di n elementi?

2) Tabelle di hash gestite con linear probing

3) La Bfs è un algoritmo ricorsivo o iterativo?

Programmazione (Cabodi)

1) data una lista di stringhe implementata come adt di prima classe, scrivere una funzione che ritornasse una stringa enorme con tutte le stringhe della lista al suo interno
2) contare i nodi di un BST che avessero valore maggiore di una data chiave k.

considerazioni: nella parte di teoria con Camurati ho avuto qualche incertezza sulla prima e sull' ultima domanda, mi ha portato a ragionarci ma ho sparato qualche cazzata qua e la comunque. (la risposta alla prima domanda è n<altezza dell'albero< logn) (la risposta alla terza domanda è: iterativo).

nella parte di programmazione non ho praticamente fatto la prima richiesta perche mi sono confuso tutto il tempo (purtroppo lo shock di avere cabodi che mi interrogava in programmazione ha avuto la meglio su di me), mentre alla seconda domanda ho risposto subito e in maniera precisa anche se mi ha detto che poteva essere implementata meglio la funzione per non scandire tutto l'albero. (ha comunque detto che capisce che in un esame si adotti un approccio conservativo e che piuttosto che strafare preferisce chi fa cose anche un po' meno ottimizzate ma che funzionano di sicuro).

17 -> 18 (16/02/2021) (no Lab)

Teoria:

- Heap sort, funzionamento e complessità
- Linear chaining
- Definizione di grafo completo, sparso, denso
- Come si individuano i cicli all'interno di un grafo, se si possono individuare quelli di grado k e se possono essere enumerati

Programmazione

- Data una matrice di RxC elementi inserire gli elementi diversi da 0 in un vettore di liste
- Dato un vettore di interi da 0 a 9 trovare tutte le possibili terne aventi un numero pari all'indice 0 e 2, dispari all'indice 1 e valore all'indice 0 maggiore di quello all'indice 1

17 -> 18

Teoria:

- Heapbuild (complessità)
    - Trasforma un albero binario memorizzato in vettore in uno heap: le foglie sono heap; applica HEAPify a partire dal padre dell'ultima foglia o coppia di foglie fino alla radice
    - O(n)
- Heapify (complessità)
    - Trasforma in heap i, LEFT(i), RIGHT(i), dove LEFT(i) e RIGHT(i) sono già heap:
        - assegna ad A[i] il max tra A[i], A[LEFT(i)] e A[RIGHT(i)]
        - se c'è stato scambio A[i] <-> A[LEFT(i)], applica ricorsivamente HEAPify su sottoalbero con radice LEFT(i)
        - se c'è stato scambio A[i] <-> A[RIGHT(i)], applica ricorsivamente HEAPify su sottoalbero con radice RIGHT(i)
    - O(log(n))
    - PQinsert (complessità)
    - Aggiunge una foglia all'albero (cresce per livelli da SX a DX, rispettando la proprietà strutturale)
    - Risale dal nodo corrente (inizialmente la foglia appena creata) fino al più alla radice. Confronta la chiave del dato contenuto nel padre con la chiave del dato da inserire, facendo scendere il dato del padre nel figlio se la chiave da inserire è maggiore, altrimenti inserisce il dato nel nodo corrente
    - O(1) per vettore e lista non ordinati; O(n) per vettore e lista ordinati; O(log(n)) per heap di item
- Algoritmo delle disposizioni ripetute con relativa equazione alle ricorrenze
    - Ogni elemento può essere ripetuto; per ognuna delle posizioni si enumerano esaustivamente tutte le scelte possibili

- Equazione alle ricorrenze? T(n) = 1 + nT(n-1)
- void disp_rip (int pos, int *val, int *sol, int n, int k) {

```
int i;
if (pos >= k) {
        for (i=0; i<k; i++)
        printf("%d ", sol[i]);
        printf("\n");
        return;
}
for (i = 0; i < n; i++) {
        sol[pos] = val[i];
        disp_rip (pos+1, val, sol, n, k);
}
return;
```

}

- Cammini di una certa lunghezza per grafi orientati e non orientati (Dijkstra e BFS)
- Complessità di Dijkstra e Belmann-Ford
  - Dijkstra T(n) = O(|E| lg |V|)
  - Bellman-Ford T(n) = O(|V| |E|)
- Codici di Huffman, a cosa servono
  - Codice: stringa di bit associata ad un simbolo; servono a codificare le lettere dell'alfabeto ad esempio
  -
- DAG
  - Directed Acyclic Graph: Grafo orientato aciclico. Utilizzato, ad esempio, per operazioni di scheduling (precedenze tramite archi orientati): dati compiti (tasks) e vincoli di precedenza (constraints), come programmare i compiti in modo che siano tutti svolti rispettando le precedenze. Nei DAG esistono 2 classi di nodi: i nodi sorgente (source) che hanno indegree=0 e i nodi pozzo o scolo (sink) che hanno out-degree=0
- Ordinamento topologico inverso
  - Dato un arco (u, v) il nodo u si troverà più a destra del nodo v con il corrispondente arco che va

da u a v (da destra a sinistra), tutti gli archi devono soddisfare questa proprietà OPPURE nell'ordinamento topologico inverso i nodi sono disposti linearmente in modo tale che ogni nodo venga dopo di tutti i nodi collegati ai suoi archi uscenti, in questa maniera non si presentano archi Backward nella visita in profondità. L'ordinamento topologico di un dag esiste sempre ma può non essere unico.

Programmazione:

- Dato un vettore di int mettere gli elementi pari in una lista e quelli dispari in un'altra. Tenere le liste ordinate
    - If (v[i]%2 == 0) {inserimento in lista pari}
- Dato un BST in cui ogni nodo ha come item liste concatenate implementare una funzione freebst() per l'eliminazione dell'intero BST
    - Attraversamento in post-ordine del bst con eliminazione di ogni nodo

17 -> 18

Teoria:

- Dato un cammino minimo tra due vertici A e B se inserisco un vertice C tra i due vertici. Perché i sotto-cammini che vanno da A a C e da C a B sono considerati dei sotto-cammini minimi?
- Alberi ricoprenti minimi: Che cosa è un taglio?
    - Partizione dell'insieme dei vertici del grafo
- Come si determina un albero ricoprente minimo? Che cosa è?
- Definizione di arco sicuro.

Programmazione:

- Inserimento in una coda a priorità
    - Implementazione ADT I categoria con una lista
- Stampare a video tutti i cammini semplici di lunghezza k partendo da un vertice dato A

17 -> 18

Teoria:

- Heap e funzioni Heapify, Heapbuild, Heapsort
- Costi di tali funzioni
- Approssimazione di Stirling
- BST, vari casi di cancellazione, successore/predecessore, funzione select

Programmazione:

- Inserzione in una lista ordinata
- Dato un albero binario, funzione in grado di calcolare il numero di: nodi con un solo figlio, numero di x\nodi sx, numero di nodi dx, numero nodi totali

17->19 (14/09/2021)

Teoria(Camurati):l.

- DFS complessità;
- DAG: cos'è, come posso rappresentarlo e complessità lista e matrice adiacenza e svantaggi/vantaggi dell'una e dell'altra

Programmazione(Pasini):

- Date due liste di interi ordinate e prive di duplicati, generare una terza lista data dall'intersezione insiemistica delle prime 2;
- Dato un albero binario generico scrivere la funzione che conti il numero di nodi tra due livelli (considerando lo 0 il livello della radice) wrapper int f(T t, int lvl1, lvl2).

bellman

17->19 (16/02/2021)

Teoria:

- Teorema e corollario arco sicuro per gli MST;
- Codici di Huffman, specificatamente cosa vuol dire "prefix-free" e che peculiarità hanno(lunghezza variabile)

Programmazione:

- Da matrice sparsa inserire gli elementi non nulli in un vettore di liste(ogni riga<->una lista), nel nodo della lista valore intero e indice della colonna nella matrice;
- Liberazione bst con stringa nel nodo;

17 -> 19

Teoria:

- Grafo completamente connesso
- Numero di vertici e a quale algoritmo del calcolo combinatorio è possibile associare il numero di questi vertici

Programmazione:

- Data una struttura wrapper con un intero, contenente il numero di stringhe, e un sh, 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 Bst

17 -> 19

Teoria:

- Powerset
- Quante soluzioni generano le permutazioni ripetute di n elementi
- Funzione che generi tutte le permutazioni di 3 bit (quindi 0 o 1) in pseudo codice

Programmazione:

- Data una lista che contiene un interi casuali generare altre due liste: una che contenga solo gli interi pari e l'altra solo gli interi dispari
- Generare tutti i numeri binari di n elementi di cui almeno d siano 1

17->19 (16/02/2021)(+lab)

Teoria (Camurati):

- Equazione delle ricorrenze
- Quicksort caso peggiore
- Dijkstra

Programmazione (Patti):

- Data una stringa “abc,abb,bcc,bdc” trovare le parole singole stringhe separate dalle virgole ed inserirle in una lista ordinata
- Dato un bst calcolare quanti nodi hanno solo il figlio sinistro e calcolare quanti solo il figlio destro con delle variabili passate per riferimento

17 -> 20

Teoria:

- Cosa sono gli algoritmi greedy?
- Cos’é il backtrack?

Programmazione:

- 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 ritorna due puntatori a grafi uno con archi a valore positivo e l'altro con archi a valore negativo

17 -> 20 (26/01/2021)

Teoria(Camurati):

- Weighted quick union
- Heap, definizione e proprietà
- HEAPify, condizioni di applicabilità e dimostrazione della complessità

Programmazione(Pasini):

- Data una lista di interi (da definire come si preferisce), effettuare la cancellazione di tutti i nodi contenenti la chiave “val” tale che val1<= val <= val2 con “val1” e “val2” passati come parametri alla funzione
    - Dato un albero binario (BST), contare tutti i nodi con 2 figli a partire dal livello “lvl” passato come parametro alla funzione

17 -> 21 (26/01/2021)

Teoria (Camurati)

- Equazione alle ricorrenze;
- Modi di rappresentare un grafo (liste di adiacenze e matrice di adiacenze), con costi dal punto di vista dello spazio e tempi di accesso;
- Counting-Sort (descrizione algoritmo e perchè è stabile);

Programmazione (Patti)

- Dato un vettore di interi creare due liste , una per i numeri pari e l’altra per i numeri dispari
- Ricorsione su calcolatore, STACK frame, tail recursive e funzionamenti vari della ricorsione
- Dato un grafo (implementazione a scelta con matrice o lista di adiacenze), contare tutti i cammini semplici di lunghezza k a partire da un vertice v;

17 -> 21 (2 punti lab inclusi) (8/02/22, ASD)

Teoria (Cabodi):

- Strategie di visita preorder, inorder, postorder e complessità (O(n))
- Algoritmo di Dijkstra, PQChange
- Componenti fortemente connesse

Programmazione (Pasini):

- Definire una lista ed una funzione che cancella i nodi che hanno valore minore di un certo target (dato in input)
- Disposizioni ripetute con pruning, dato un vettore di numeri da 0 a 9 contare tutte le sequenze lunghe k (con ripetizioni), con un vincolo: devono esserci max P numeri pari vicini, con P dato in input.

Cabodi molto tranquillo, ho sbagliato la complessità e un paio di cose nella PQChange; mentre di programmazione ho fatto qualche piccolo errore impostando il pruning.

17 -> 23

Teoria:

- Equazione alle ricorrenze generica
- Quick sort caso peggiore, complessità, complessità partition e sua struttura generica

Programmazione:

- Wrapper con puntatori di puntatori a strutture e robe varie, 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.

18 -> ?? (26/01/2021)

Teoria (Camurati):

- Heap: implementazioni, complessità delle varie funzioni e complessità spaziale (max spazio sprecato), differenze con BSTtpit
- Quicksort nel caso peggiore: complessità ed equazione alle ricorrenze

Programmazione(Pasini):

- Eliminazione da una lista di ogni nodo con valore uguale ad uno dato
- Conteggio di tutti i cammini semplici che partono da un vertice dato lunghi k (con k fornito obv)

18 -> ?? (26/01/2021)

Cabodi teoria:

- Equazione alle ricorrenze delle disposizioni semplici senza ripetizioni;

  Palena programmazione:

- Specchiare un albero binario ricorsivamente;
- Cancellare da una lista di interi singolarmente linkata i nodi contenenti dati dispari;

18 -> 19

Teoria:

- Max, min, predecessore, successore nel bst
  - Max: seguire il puntatore al sottoalbero destro finché esiste
  - Min: seguire il puntatore al sottoalbero sinistro finchè esiste
  - Predecessore, nodo h con item con la più grande chiave < della chiave di item:
    - se esiste Left(h) allora pred(key(h)) = max(Left(h))
    - se non esiste Left(h) allora il pred(key(h)) = primo antenato di h il cui figlio destro è anche un antenato di h
- Fattore dimensionamento tabelle di hash
  - Linear chaining: il più piccolo numero primo >= (numero di chiavi max / 5 (o 10)) così che la lunghezza media delle liste sia 5 (o 10)
  - Open addressing: il più piccolo numero primo >= al doppio del massimo numero di chiavi presenti
- Nella visita di profondità è possibile rilevare cicli?
  - Sì, basta trovare un arco Backward
- Equazione ricorrenze merge
  - T(n) = 2T(n/2) + n

Programmazione:

- 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!=0 e colonna.

- Cammino/ciclo lunghezza k in un grafo

18 -> 19 (16/02/2021)

Teoria(Camurati)

- Quick Find/Union cosa fanno e complessità
- Cosa è un DAG e come si riconosce
- Come si fa un ordinamento topologico

Programmazione(Palena)

- Implementazione HEAP(a voce, non scritto)
- Codice HEAPify
- Struttura dati BST
- Codice inserimento in foglia BST

18 -> 20 con 2 punti lab (21/02/2022)

Teoria (Camurati):

- Definizione di rango di una chiave in un BST
- Funzione select e sua complessità

  La complessità è direttamente proporzionale all'altezza dell'albero e quindi la complessità varia tra O(n) se l'albero è completamente sbilanciato (cioè è come una lista) e O(logn) se l'albero è bilanciato
- Definizione di albero bilanciato
- IBST

Programmazione (Cabodi):

- Data una lista non ordinata di stringhe eliminarne i duplicati
- Variante del primo punto in cui i duplicati vanno tenuti se in numero pari (da spiegare solo a voce)
- Dato un grafo g orientato e non pesato, un suo nodo v e un intero d, contare il numero di nodi di g a distanza d da v

  Si risolve con una visita in ampiezza, siccome il grafo non è pesato

18 -> 20

Programmazione:

- Powerset
- inserimento in ordine e in testa in una lista.

Teoria:

- Cos'è un Heap
- Perchè (nei casi in cui si può usare un heap) è più efficiente usare un heap invece di un albero binario. [Dimostrazione sullo "spreco" massimo di spazio in un heap nel caso peggiore -> (al max si spreca la metà del vettore allocato per l'heap)]
- Grafi: numero di archi in un grafo diretto

18 -> 21 (26/01/2021)

TEORIA(Camurati):

- Heap, descrizione della struttura in generale e proprietà (non partite con "Implementiamo con vettore" ma piuttosto spiegate max(h, h->l, h->r) ed altro)
- Cosa si intende per altezza(profondità massima) e profondità(lunghezza del cammino da r a x)
- Alcune differenze con il BT (BST) e lo spreco di memoria massimo
- Quicksort caso peggiore, sua complessità ed equazione ricorrenze

PROGRAMMAZIONE (Pasini)

- Data una lista, eliminare tutti gli elementi che al suo interno hanno un certo valore
  prototipo → LISTdel(List l, int val)
- Funzione che ritorna il numero di possibili sequenze di un vettore lungo n, che abbia al suo interno valori da 0 a 9, e che contenga solo un numero di valori pari consecutivi massimo a un certo valore dato in input (per dire potete scrivere, per max_pari_cons=3, 00012225100 ecc va bene, invece 000004ecc no). Inoltre le sequenze non devono essere stampate, ritornare soltanto il numero. RIchiesto pruning esplicitamente
- Codice che ho scritto all'orale (magari vi può essere di aiuto) https://pastebin.com/xPsj4Mid

18 -> 21

Programmazione:

- Data una funzione che riceve una matrice quadrata e la dimensione, restituire un vettore allocato dinamicamente che abbia in ogni cella la somma delle diagonali principali.

- Poi mi ha modificato la funzione da int *funz a void funz mandando come parametro il vettore e mi ha chiesto come sarebbe cambiato il codice
- Data una funzione che riceve come parametro il link alla testa di un BST, calcolare il numero di figli (io l'ho fatto ricorsivo ma non era una richiesta specifica, si poteva fare anche iterativo)

Teoria:

- Che cosa sono le rotazioni nei BST? Che complessità hanno?
- Che cos'è un I-BST e che proprietà deve avere per calcolare le intersezioni?
- Che cos'è un grafo completo? Qual è il numero di archi di uno non orientato?

18 -> 21

Programmazione:

- Data una stringa e un vettore di stringhe come parametri (dovevi passare tu il puntatore al doppio puntatore), allocare il vettore, scomporre la stringa in sottostringhe (delimitate da carattere #) e inserirle nel vettore di stringhe.
- Dato un grafo non orientato e non pesato con implementazione a scelta e un suo vertice ritornare il numero di vertici raggiungibili dal vertice

Teoria (Camurati):

- Definizione ricorsiva del BST
- Lo stack frame nelle chiamate ricorsive

Notazione O grande e Theta grande, dire quale tra le due è la complessità di un normale ciclo for per i che va da 0 ad N, la risposta a entrambe. Dire come trasformare il ciclo in modo da renderlo di complessità una delle due.

20->24 (26/01/2021)

Teoria:

- Hashing, collisioni e fattore di carico (sia in linear chaining che in open addressing)
- BST: cancellazione nodi

Programmazione:

- Date due liste ordinate, inserire in nuova lista gli elementi di entrambe, mantenendo l'ordine, senza ripetizione di elementi (se ne ho già aggiunto un'istanza alla nuova lista, non ne metto altre). In pratica é un merge di due liste

- Enumerare cammini semplici di lunghezza k dato un nodo di partenza

18 -> 22 (16/02/2021)

Teoria: (cabodi)

- left-child right-sibling descrizione, vantaggi e svantaggi
- weighted quick union
- Prim come funziona

Programmazione:(vendraminetto)

- implementare int binsearch(char **stringhe, int n, char*chiave) con ricerca dicotomica
- scrivere ADT 1 per STACK implementato come vettore di interi e scrive funzione STACKpush

18 -> 22

Teoria:

- Definizione ricorsiva di albero binario di ricerca (BST)
    - Per ogni nodo x vale che:
        - Per ogni nodo y appartenente a Left(x), key[y] < key[x]
        - Per ogni nodo y appartenente a Right(x), key[y] > key[x]
- Interval bst
    - Intervallo chiuso: coppia ordinata di reali $[t_1, t_2]$, dove $t_1 <= t_2$. L'item intervallo $[t_1, t_2]$ può essere realizzato da una struct con campi low = t_1 e high = t_2
- Tabelle di simboli universal hashing
- Polinomio di horner

Programmazione:

- Funzione void che riceve un vettore e ritorna due vettori allocati dinamicamente uno con numeri pari e l'altro con numeri dispari
- Funzione che dato un grafo calcola tutti i cammini che si possono effettuare da ogni singolo vertice e che abbiano lunghezza minima k

18 -> 22 (Appello 16/02/2021), LAB consegnati ma non tutti gli es. fatti.

Teoria (Camurati):

1. Codici con prefisso e senza prefisso
2. Complessità quicksort nel caso peggiore e perché (O(n^2))

3. Definizione di componenti fortemente connesse (praticamente non l'ho saputa e mi ha risposto lui...)

Programmazione (Pasini)

1. Data una lista con formato a piacere (da definire) con ogni nodo contenente un valore intero, cancellare dalla lista gli elementi che hanno un valore passato alla funzione.
2. Dato un Binary Tree (non per forza BST) e dati due interi che rappresentano dei livelli (altezza dell'albero), calcolare il numero di nodi contenuti tra i due livelli (questi compresi).

Commento personale: A questo punto dato il mio orale (quasi penoso direi) credo che sia molto complicato che boccino…

18 -> 22

Teoria:

- Arco sicuro
- Equazione alle ricorrenze del quicksort nel caso peggiore

Programmazione:

- Generare gli anagrammi di una stringa, col vincolo di consonanti e vocali non consecutive
- Dato un vettore di puntatori a stringa ritornare due vettori di puntatori a stringa contenenti rispettivamente le stringhe inizianti per consonante e quelle inizianti per vocale

18 -> 23

Teoria:

- Forma generale dell'equazione alle ricorrenze
    - Spiega il significato di ciascun termine
- Quale può essere uno dei problemi del metodo ricorsivo?
    - Il fatto che ogni passo è una nuova istanza del problema che non ricorda i risultati calcolati ai passi precedenti
- Come può essere risolto?
    - Salvando i risultati calcolati in una opportuna struttura dati
- All'interno di una funzione ricorsiva, in quale punto faresti il controllo sul fatto che il problema sia stato già calcolato?
    - Dopo il controllo sul caso terminale: prima di procedere con il passo i-esimo controllo nella struttura se è già stato risolto
- Quale potrebbe essere una struttura utile a questo fine?
    - Un vettore
- Cosa dovrebbe contenere questo vettore?

-
    - All'i-esimo indice corrispondente al passo della ricorsione, contiene il risultato del problema a quel passo

Programmazione (assistente):

- Data una funzione con un vettore e la sua dimensione, creare un nuovo vettore in cui ciascun elemento è la media tra l'elemento corrispondente del vettore di partenza, quello precedente e quello dopo (i casi estremi vanno gestiti come buffer circolare, però);
- Data una lista, cancellare gli elementi in posizioni dispari (considera la testa come posizione 0).

18 -> 23 (26/01/2021)

Teoria (Camurati)

- Equazione ricorrenze merge sort.
- Cos'è un DAG e applicazione dell'ordinamento topologico (cammini massimi).
- Come si determina se un grafo è aciclico.

Programmazione (Pasini)

- Data una lista cancellare tutti i nodi che contengono un valore val.
- Dato un BT contare tutti i nodi completi tra due livelli l1 e l2.

18 -> 23

Teoria:

- Tabelle di Hash
- Metodo di Horner(Trasforma stringhe alfanumeriche lunghe in interi derivati dalla valutazione di polinomi in una determinata base minimizzando il numero di operazioni)

Programmazione:

- Mergesort con vettore di puntatori a stringa
- Componenti connesse tramite DFS

18 -> 23

Teoria:

- Come possiamo definire un albero tramite un grafo, cos'è un albero, albero bilanciato, completo,
- Tabelle di hash
- Differenza tra cammini minimi e alberi ricoprenti minimi
- Quicksort e perché non consideriamo il caso peggiore
    - Non consideriamo il caso peggiore perché è uno solo

Programmazione:

- Data una lista, caricarla al contrario su un'altra lista
- Dato il vettore delle 10 cifre (0123456789), scrivere tutte le possibili combinazioni da 5 posti, sapendo che ogni numero può essere riutilizzato al massimo 3 volte e prima di essere riutilizzato deve essere interrotta la catena da un altro numero (cioè, esempi validi:12345 11101 11211 33322 ; esempio non valido 75555)

18 -> 23

Teoria:

- Complessità bst non bilanciato e complessità bst bilanciato
- Dire come funziona partizionamento bst
- Spiegare significato di "left child right sibling”
- Powerset

Programmazione:

- Verificare se un bst è contenuto in un altro (di dimensione maggiore) o meno
- Codice visita in ampiezza (BFS)

18 -> 24

Teoria:

- Tipologia di algoritmi p, np etc…
  - Polinomiale: problemi di decisione decidibili e trattabili (risolvibili in tempi ragionevoli n^c dove c in pratica è 2)
  - Non-deterministico Polinomiale: problemi di decisione decidibili per cui conosciamo algoritmi di soluzioni esponenziali, ma non conosciamo algoritmi polinomiali. Non possiamo però escludere che esistanoNP-Completo: ogni altro problema in NP è riducibile ad esso attraverso una trasformazione polinomiale. Se un problema NP-completo fosse risolvibile con un algoritmo polinomiale, allora con trasformazioni polinomiali si troverebbero algoritmi polinomiali per tutti i problemi NP
  - NP-Hard: ogni problema in NP è riducibile ad esso in tempo polinomiale
- DFS e archi T F B C
  - Visita in profondità: a partire da un vertice visita tutti i vertici del grafo etichettandoli con tempi di scoperta e fine elaborazione

    - Tree: archi dell'albero della visita in profondità
    - Backward: archi che connettono un vertice ad un suo antenato nell'albero (cappi compresi)
    - Forward: archi che connettono un vertice ad un suo discendente nell'albero
    - Cross: archi rimanenti, attraversano da un ramo all'altro o collegano due alberi diversi

Programmazione:

- 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

19 -> 19

Teoria:

- Cammini minimi
- Alberi non radicati

Programmazione:

- Liste doppio linkate come ADT
- Inserzione in foglia BST
- Dato un grafo e un suo vertice di partenza determinare un cammino di lunghezza 5

19 -> 20 (26/01/2021)

Teoria(Camurati)

- Cammini massimi su un grafo qualsiasi (non DAG) che approccio seguiresti con prog. dinamica se è possibile, se non è possibile spiega il perché
- Perché nell'algoritmo di BF si usa un approccio di programmazione dinamica, perché la relaxation si effettua esattamente V-1 volte, cosa accade al V-esimo passo.karatsu
- Weighted quick union a cosa serve, differenze con quick find e quick union a quanto corrisponde la somma dell'albero minimo e massimo nel caso peggiore

Programmazione(Patti)

- Dato un Grafo rappresentato con lista di adiacenze trovare il suo trasposto.
  (mi ha visto in difficoltà e mi ha chiesto poi la NewNode e inserimento in lista)
- Dato un BST con chiave una stringa, effettuare una free dell'albero.

(SOLUZIONE) :
```
void BSTfree(BST bst){
        if(bst==NULL) return;
        treefree(bst->root,bst->z);
        free(bst->z); free(bst);
}
static void Treefree(link h,link z){
        if(h==z) return;
        treefree(h->l,z); treefree(h->r,z);
        free(h->string);
        free(h);
}
```

19 -> 20

Teoria

- In loco e stabile ed alcuni esempi di algoritmi tali
- Disuguaglianza di Stirling

Programmazione

- Codice Heapify
- Fare uno stack con un vettore con possibile realloc

19 -> 20

Programmazione:

- Scrivere una funzione per contare i successori in una lista esempio nodoA->nodoB->nodoC. nodoA= 2 successori, nodoB =1 successore nodoC = 0 successori.
- Scrivere una funzione per contare quanti cammini di lunghezza k ci sono in un grafo, con il grafo e l'intero k come parametri.

Teoria:

- Qual è il limite inferiore di complessità per gli algoritmi di ordinamento basati sul confronto e perché.
- Metodi per rappresentare un grafo(lista e matrice) vantaggi/svantaggi

19 -> 21 (26/01/2021)

Teoria:

- Algoritmi ottimi con paradigma Greedy (in particolare quello sulle attività) ;
- Cosa sono gli archi Cross? ;
- Cos'è una foresta? ;
- Programmazione dinamica;

Programmazione:

- Crea una funzione (dato il prototipo) che a partire da un vettore di stringhe restituisce un vettore che contiene le stringhe del primo vettore ordinate alfabeticamente;
- Crea una funzione che esegue la visita in profondità stampando a video i vertici man mano che vengono incontrati;

19 -> 21

Programmazione:

- Data una matrice sparsa (quella "scritta" come vettore di liste, ogni lista contiene i valori diversi da 0 della matrice) creare una funzione che verifichi che ci sia un"quadrato" di valori non nulli nella matrice. Funzione che calcoli l'altezza di un bst.

Teoria:

- Definizioni: arco sicuro, insieme di archi che rispettano un taglio, taglio. Date le scc di un grafo orientato, se per ogni scc si prende un nodo "rappresentante", il grafo ottenuto sarà ciclico o aciclico? aciclico, perchè se fosse ciclico i nodi del ciclo sarebbero dovuti essere della stessa scc.

19 -> 22

Teoria:

- Realloc: come funziona, quando la usiamo, come la usiamo, perché raddoppiamo la dimensione e non aumentiamo semplicemente di uno e costo della realloc
- Tabella di Hash: valore di M nell'open addressing, fattore di carico, costo di inserzione e double hashing (con codice)

Programmazione:

- Dato un vettore di puntatori a stringhe un intero n (numero stringhe) e un intero d, spostare le ultime d

stringhe in cima e far scalare la altre di d posizioni (voleva si utilizzassero gli scambi tra puntatori lavorando sul vettore di partenza senza crearne uno nuovo)

- Ricerca del massimo in un bst

19->22 (30/06/21)

teoria (cabodi):

quicksort e complessità partion

IBST

bellman-ford - djikstra

programmazione:

cancellazione da lista valori pari

cammini minimi

19 -> 22 (26/01/21)

Teoria (Cabodi):

- Complessità di quick find, quick union, quick weighted union
- In quale algoritmo viene usata unionfind? (kruskal)
- Teoremi su cui si basano kruskal e prim
- Numero di archi di albero ricoprente minimo
- Se il grafo è non pesato o pesi tutti uguali, servono kruskal o prim? (no basta costruire albero di profondita o di ampiezza con v-1 archi)

Programmazione (Pasini):

- Inserimento di una stringa in una lista ordinata che ha delle stringhe come valori definendo anche la struttura dati
- Numero di soluzioni di lunghezza len nei quali sono presenti le cifre da 0 a 9, anche ripetute
- I numeri pari non hanno vincoli, i numeri dispari possono essere ripetuti globalmente max k volte

19 -> 23 (26/01/2021)

Teoria (Cabodi)

- Calcolo complessità ricerca dicotomica
- Punti di articolazione e bridge, un bridge implica punti di articolazione, un punto di articolazione implica dei bridge

Programmazione:

- Ricerca del massimo in un albero n-esimo left-child-right-siblings
- Calcolo anagrammi di una parola di lunghezza k con pruning in modo che due lettere vicine non siano uguali

19 -> 23

Programmazione:

- A partire da una lista data, generare una seconda lista che sia la trasposta della prima.
- Scrivere l'algoritmo che calcola il numero di tutti i cammini semplici di un grafo.

Teoria:

- Disposizioni ripetute
- implementazione del powerset

19 -> 23

Teoria:

- Grado di un albero
- Metodi per rappresentare un nodo di un bst
- Analisi della complessità della moltiplicazione tra due interi

Programmazione:

- Trasformazione di una matrice di char NxN in un vettore di stringhe allocate dinamicamente della giusta lunghezza
- Heapsort

19 -> 23

Programmazione:

- Implementare la funzione void R(char **mat, int r, int c, int R, int C,...) che riceve una matrice di interi mat, il numero di righe r, il numero di colonne c e , R e C. Eliminare l'intera riga R e l'intera colonna C dalla matrice mat e restituire la nuova matrice compattata. Gli eventuali puntatori allocati devono essere allocati della dimensione giusta. La funzione può rivere ulteriori parametri a discrezione dello studente
- Implementare la funziona void f(char *str, int n, int maxRip, int maxSeq) che ricevuta una stringa str lunga n, generi tutte le stringhe che rispettano i seguenti vincoli: nella stringa, non ci possono essere più di maxRip lettere ripetute e una stessa lettera non può comparire consecutivamente per più di

maxSeq volte. Generare le stringhe usando l'alfabeto minuscolo ('a' - 'z'). Per esempio, se maxRip = 2 e maxSeq = 2, la stringa "AAAB" e "AABA" non vanno bene. Implementare la funzione senza pruning. Inoltre, implementare la versione con pruning senza considerare il vincolo su maxSeq.

Teoria:

- Cosa sono le componenti fortemente connesse? In che quale ambito si usano?
- Cosa significa "Massimale"?
- Cosa è un grafo trasposto?
- Supponiamo che il grafo sia rappresentato come lista delle adiacenze...Come può essere implementata la lista? Nel caso fosse noto il numero di nodi, come la implementersti? Perché?
- Quali sono le condizioni per applicare la HEAPify ad una terna radice, sotto-albero sinistro, sotto-albero destro?
- Dimmi l'equazione alle ricorrenze di un algoritmo ricorsivo generico

19 -> 23

Teoria:

- Quando dijkstra fornisce una soluzione ottima e quando no
    - In presenza di archi a peso negativo Dijkstra non restituisce la soluzione ottima; in presenza di cicli a peso negativo è inconcludente; in tutti gli altri casi restituisce la soluzione ottima
- Come implementare i cammini massimi
- Cammini massimi nei DAG
- Paradigma Greedy (appetibilità fisse e variabili)

Programmazione:

- Coda a priorità per i punti del piano (la priorità era la vicinanza all'origine)
- Implementare funzioni Init(), Aggiungi(), Estrai() ADT 1 cat
- Problema di ottimizzazione simile al problema dello zaino

19 -> 23

Programmazione:

- date due liste di lunghezza diversa contare quante volte la piccola è contenuta nella grande.
- contare i cammini di lunghezza k per ogni vertice di un grafo orientato.
- contare tutti i cammini semplici di lunghezza k per ogni vertice di un grafo

Teoria:

- altezza albero,profondità nodo.
- rotazione di un BST:come si fa e costo.
- I-BST:cosa sono e cosa devo aggiungere ad un BST normale

- hash:tutti i modi per teoria: trasformare una stringa in un indice della tabella di hash

19 -> 24 (26/01/2021) consegnando i laboratori

Teoria (cabodi)

- Differenze di complessità della quick-union e quick-find e la complessità di quale delle due operazioni si migliora nella weighted quick-union (risposta: l'operazione di find che diventa logaritmica).
- Differenze tra gli algoritmi per la ricerca dei cammini minimi.
- Complessità algoritmo di dijkstra.

Programmazione (Palena)

- Struttura dati lista di interi come quasi ADT e poi eliminare i nodi con numero pari.
- Fare una funzione ricorsiva che specchiasse un albero binario generico (i nodi a destra devono andare a sinistra e viceversa). Si dava il prototipo "void mirror_tree (TREE t)" dove TREE è variabile puntatore.

19 -> 24 (16/02/2021) consegnando i laboratori

Teoria (Cabodi)

- Partition BST, bilanciamento BST->come si fa partendo dalla partition e quante volte si applica
- Grafo bipartito, varie domande basandosi sul grafo in cui bisognava ragionare
- Dijkstra complessità

Programmazione (Vendraminetto)

- Stack ADT 1 classe con funzione push (Ho utilizzato vettore con inserimento al fondo)
- Shiftleft di una stringa, esempio str: CIAO shift:1 str:IAOC

19 -> 24

Programmazione:

- Data una lista linkata semplice contenente numeri interi, creare una funzione separa(list_t *l) che metta i numeri pari in una nuova lista che verrà ritornata e li cancelli da quella passata come parametro.
- Generare un vettore di archi dato un grafo implementato come matrice delle adiacenze.

Teoria:

- Definizioni di albero, cammino, cammino semplice, ciclo di hamilton. Calcolo cammini minimi, se è possibile il calcolo dei cammini massimi su grafi generici, e come farlo.Algoritmo di Dijkstra.

19 -> 25

Teoria:

- Albero completo/bilanciato differenze
    - Albero completo:
        - tutte le foglie hanno stessa profondità
        - ogni nodo o è una foglia o ha 2 figli
    - Albero bilanciato:
        - Tutti i cammini radice-foglia sono di ugual lunghezza
- Scrivere un albero binario che è bilanciato ma non completo
    - Basta disegnare un albero completo alto a piacere e togliervi uno dei due sottoalberi figli del padre (ad esempio)
- Complessità operazioni BST
    - Le operazioni hanno complessità $T(n) = O(\log_2 n)$ per un albero completamente bilanciato e $T(n) = O(n)$ per un albero completamente sbilanciato
- Stack
    - Tipo di dato astratto che supporta operazioni di:
        - Push: inserimento dell'oggetto in cima allo stack
        - Pop: prelievo (e cancellazione) dalla cima dell'oggetto inserito più di recente
    - La strategia di gestione dei dati è detta pila LIFO (Last In First Out)
- Stack Frame
    - Struttura dati che contiene almeno:
        - Parametri formali
        - Variabili locali
        - Indirizzo a cui si ritornerà una volta terminata l'esecuzione della funzione
        - Puntatore al codice della funzione
    - Lo stack frame viene creato alla chiamata della funzione e distrutto al suo termine
- Ricorsione tail recursive e perché serve la tail recursive
    - Tail recursive quando la chiamata ricorsiva è l'ultima operazione da eseguire, escluso il return; maggiore efficienza (solo discesa, no risalita); le chiamate ricorsive non hanno bisogno di stack: la chiamata successiva rimpiazza quella corrente; trasformabili direttamente in iterative

Programmazione:

- 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

19 -> 25 (26/01/2021) (Inclusa nel voto una compensazione per problemi allo scritto)

Teoria (Cabodi)

- Algoritmo di Er
- Cammini di Hamilton
- Codice di Huffman

Programmazione (Vendraminetto)

- Dato un vettore di stringhe, concatenare una stringa data in mezzo a partire da una posizione passata come parametro e ritornare la nuova stringa.
- Definire come ADT 1C il bst e scrivere funzione che ritorna il numero di foglie del bst

19 -> 25 (26/01/2021)

Teoria (Camurati):

- definizione di DAG (grafo orientato aciclico)
- classificazione dei vertici di un DAG (sorgente: in-degree = 0; pozzo: out-degree = 0).
- ordine topologico DAG, definizione e come si fa nella pratica (si ordinano i vertici in modo tale che tutti gli archi vanno “da sinistra a destra”. Per realizzarlo si effettua una visita in profondità e si ordinano i vertici per tempo di fine elaborazione decrescente)
- classificazione degli archi in un grafo orientato, in teoria e “in codice”(archi T: archi della visita in profondità; archi B: collegano un vertice con un suo antenato; archi F: collegano un vertice con un suo discendente; archi C: tutti gli altri. Per identificare un arco Back nella pratica si effettua una visita in profondità. Se il vertice w è connesso tramite un arco uscente a un vertice v già scoperto, l’arco è Back se post[v] = -1).

Programmazione (Pasini):lista

- lista di punti nel piano. Inserire nella lista un punto in modo da rispettare l’ordinamento. La chiave è la distanza dall’origine.
- problema di calcolo combinatorio con pruning basato sul modello delle disposizioni ripetute, con vincoli locali da rispettare.

20 -> 20 (26/01/2021)

Teoria:

- Qual è la complessità della DFS e perchè (caso con grafo rappresentato con lista di adiacenze)
- Parlare della programmazione dinamica, in particolare spiegare cosa si intende per “sottostruttura ottima”
- Dove si trova la programmazione dinamica nell’algoritmo di Bellman-Ford

Programmazione:

- Stack implementato come ADT di I classe, tramite una lista
- Scrivere l’algoritmo della quick-union

20 -> 20 (15/06/2021)

Teoria:

- Dijkstra
- DAG
- Cammini minimi e dimostrazione della sottostruttura ottima. Si può applicare la programmazione dinamica alla ricerca dei cammini minimi? Perchè?
- Cammini massimi e assenza della sottostruttura ottima
- Code a priorità, operazioni sulle code a priorità e implementazioni (heap, vettori ordinati e non, liste ordinate e non)

Programmazione:

- Inserzione in vettore di liste dei valori non nulli di una matrice di interi. Elementi della riga 0 nella lista di indice 0 nel vettore insieme all’indice di colonna e così via. Esempio:

  1 0 3 4

  8 0 0 0

  1 0 3 5

  L[0] -> (1,0) -> (3, 2) -> (4, 3)

  L[1] -> (8,0)
- Cammini minimi di lunghezza k su un grafo (come matrice delle adiacenze o lista delle adiacenze)

20->21 (10/07/2023)

Teoria(Camurati)

- Codici liberi da prefissi cosa sono, codici a lunghezza fissa sono liberi da prefissi (ho detto di sì, domanda simile già postata) e strutture dati huffman
- Istruzioni coda a priorità, come implementarla (lista/vettore ordinato e non) pqchange con heap come funziona e complessità

Programmazione(cabodi)

- Funzione che data lista di interi divide in due liste a seconda se pari o dispari
- Funzione che dato un grafo orientato, determinare se è dag. Se si determinare quali sono nodi source.

Di programmazione sono andato abbastanza nel panico ma è stato molto comprensivo e mi ha aiutato

20 -> 21 (16/02/2021)

Teoria(Camurati)

- Partition BST, complessità, operazione di rotazione
- Shell Sort, complessità

Programmazione(Pasini)

- Inserimento in lista ordinata di punti del piano cartesiano. Chiave di ordinamento, distanza del punto dall'origine.
- Dato grafo (a scelta se su matrice o lista di adiacenze), cicli semplici di lunghezza k uscenti da vertice v.

20 -> 21 (06/2020)

Teoria:

- Algoritmi Bellman Ford: funzionamento, strutture dati utilizzate e complessità.
- Partition di un BST: breve spiegazione del funzionamento e utilizzo della funzione (Occorre spiegare che viene utilizzata nella delete dei BST)

Programmazione:

- Funzione e strutture dati utilizzate per invertire una lista di interi (Utilizzo delle ricorsione in quanto il nodo della lista definito mediante un puntatore al successivo e NON con puntatore al precedente)
- Dato un BST, contare i nodi che hanno k figli

20 -> 21 (06/2021)

Teoria:

- Coda a priorità con Heap.
- Classificazione dei vertici nella visita in profondità.
- Algoritmi greedy, differenza tra greedy statica e dinamica, ed esempio di algoritmo greedy.

Programmazione

- Data una matrice RxC di interi, comprimerla in un ADT lista con inserimento in testa.
  Sono da inserire solamente i valori non nulli.
  Il nodo deve contenere anche il numero della colonna corrispondente.
  Matrice 0,5,6,0
  2,0,0,8
  0,0,0,3
  Lista list[0]->(5/1)->(6/2)
  list[1]->(2/0)->(8/3)
  list[2]->(3/3)
- Dato un vertice v, trovare tutti i cammini semplici di lunghezza k.

20 -> 22

Teoria:

- Cosa è un ciclo di un grafo
  - Un ciclo è un cammino che inizia e finisce allo stesso nodo.
- Proprietà heap
  - In un heap il nodo padre è maggiore o uguale a tutti i discendenti; è un albero binario quasi completo (tutti i livelli completi, tranne eventualmente l'ultimo, riempito da SX a DX), il che implica che sia anche quasi bilanciato
- Puoi salvare un bst in un vettore? (Differenze con heap)
  - Sì, si fa allo stesso modo dell'heap, usando le funzioni LEFT(i) = ((i*2) + 1), RIGHT(i) = ((i*2) + 2) e PARENT(i) = ((i-1) / 2) per muoversi da un nodo ai figli o al padre, la differenza è l'ordinamento dei nodi, nell'heap alla cella di indice 0 c'è il massimo, mentre nel BST c'è la radice (che non corrisponde al massimo)

Programmazione:

- Costruisci un ADT lista. In questa lista ricerca un Item e cancellalo. (Libera scelta sul tipo lista e tipo cancellazione)
- Cammini semplici tra sorgente e destinazione grafo

20 ->23 (26/01/2021)

Teoria (Camurati)

- Weighted quickunion e vantaggi rispetto quickunion
- Etichettatura archi e relative condizioni
- Shellsort

Programmazione (Pasini)

- Dati due vettori ordinati e le loro dimensioni, generare l'intersezione insiemistica di dimensione corretta e passarlo come parametro al client.
- Dato un albero binario contare i nodi completi compresi tra due livelli.

20 -> 23 (16/02/2021)

Teoria (Camurati)

- Definizione di double hashing
- Vantaggi del double hashing rispetto al probing.
- Spiegazione del concetto di clustering.

Programmazione (Patti)

- Dato un vettore di interi, creare due liste ordinate di cui una avente solo elementi pari, l'altra solo elementi dispari.
- Dato un insieme di interi da 0 a 9, generare e stampare sottoinsiemi di cardinalità 3, rispettando dei determinati criteri.

20 -> 23 (01/07/2021) (LAB: 1,1)

Teoria (Camurati):

- Parlare delle classi di problemi. Complessità problemi NP.
  L'insieme dei problemi P è un sottoinsieme proprio di NP? No, non si può dirlo con certezza, è probabile che lo sia.
- DFS ed etichettatura degli archi in base alla visita. Relazione (in tempi di scoperta e fine elaborazione) tra nodi per archi B e F.

- Grafo trasposto: complessità con liste delle adiacenze. Come lo si costruisce a partire da un grafo e quanto costa questa operazione?

Programmazione (Pasini):

- Date 2 liste di interi l1 ed l2 già ordinate (si suppone siano ordinate in ordine crescente) definire il tipo nodo e il tipo lista (L) ed elaborare una terza lista l3 che includa in maniera ordinata gli elementi delle liste l1, l2. Non si devono mantenere i duplicati. Prototipo funzione: L f(L l1,L l2...) (credo bastino questi due parametri).
- Dato un grafo (tipo G), un vertice di partenza e una lunghezza k (intero) contare i cammini semplici di lunghezza k dal vertice v. Prototipo funzione: int f(G g,int v,int k...) (ci dovrebbe andare int *mark).

20 -> 25 (26/01/2021)

Teoria (Camurati)

- Limite inferiore di complessità negli algoritmi di ordinamento basati sul confronto.
- Gestione dell'inserimento di stringhe di caratteri in una tabella di hash.
- Spiegazione del concetto di "fattore di carico"..

Programmazione (Patti)

- Dato un grafo orientato e organizzato tramite lista delle adiacenze, scrivere una funzione che ne generi il trasposto.
- Scrivere una funzione che, attingendo all'alfabeto, generi tutte le stringhe di 4 lettere tali per cui la prima e la quarta siano vocali (con la prima alfabeticamente minore della quarta) mentre la seconda e la terza siano consonanti (con la seconda alfabeticamente maggiore della terza).

20 -> 25 con punti LAB (08/02/2022)

Teoria (Camurati)

- Programmazione dinamica, sottostruttura ottima (es. cammini minimi), complessità di un problema risolto con programmazione dinamica (O(n*m) dove m è il costo di ogni operazione e n è il numero di dati).

Programmazione (Cabodi)

- Sviluppare una funzione che dato in input un BST e un link x ad un nodo del BST stampi tutti i nodi (escluso x) allo stesso livello di x. (Si calcola prima il livello di x sfruttando il puntatore al padre e poi si stampa con una funzione ricorsiva di visita);
- Dato un Grafo orientato con lista di adiacenza scrivere una funzione che conta i cappi;
- Dato un Grafo orientato con lista di adiacenza elencare i cicli di lunghezza 2 (basta scorrere la lista di adiacenza e per ogni arco verificare se c'è l'arco inverso).

20->21 13/04/22

Teoria (Camurati):

- Condizione applicabilità dell'heapify (figlio sinistro e destro devono essere degli heap)

- Equazioni alle ricorrenze decrease and conquer vs divide and conquer, nello specifico cosa rappresentano i 3 fattori in una equazione alle ricorrenze
- Perchè in Bellman-Ford si applica la relaxation al più |V| - 1 volte

Programmazione (Pasini):

- Data una lista di interi singolo puntata del tipo 1->3->10 renderla 1->2->3->4->...->10
- Dato un grafo, un vertice V ed una lunghezza K trovare i cammini ciclici dal vertice V di lunghezza K

20 -> 26

Teoria:

- Parlare del limite lasco inferiore omega grande
  - Limite di complessità minima, non può essere più semplice di così
- Equazione di Stirling
  - Algoritmi basati sul confronto: confronto a_i e a_j
  - Esito: decisione ($a_i > a_j$ o $a_i <= a_j$), riportata su un albero delle decisioni
  - Per n interi distinti: numero di ordinamenti = numero di permutazioni n!
    - Complessità: numero h di confronti (altezza dell'albero)
    - Ogni soluzione = foglia
    - Numero di foglie = 2^h
    - Approssimazione di Stirling: $n! > (n/e)^n$ quindi $2^h >= n! > (n/e)^n$
    - $h > lg(n/e)^n = n \lg n - n \lg e = (n \lg n)$
- Componenti fortemente connesse

Programmazione:

- File di testo con numero righe e a seguire n righe con cognome e nome. Mi ha chiesto di mettere tutto in un vettore di puntatori ad item, allocando stringhe e item dinamicamente
- Codice delle permutazioni semplici

21 -> 25 (+2 lab) (07/02/2023)

Teoria (Camurati):

- Algoritmo di Karatsuba, per cosa si applica e complessità (non ha chiesto la 'formula' completa).
- Cosa si intende per arco sicuro (teorema e corollario).
- Funzioni tail recursive.

Programmazione (Cabodi):

- Dato un albero binario, stampa il cammino dalla radice fino alla foglia con profondità massima.

- Dato un grafo non orientato e un suo vertice, verifica se quel vertice è un punto di articolazione. Inoltre, cosa succede se il grafo era già sconnesso in partenza? (verificare con una dfs prima di ‘rimuovere’ il nodo).

21 -> 22

Teoria:

- Limite inferiore di complessità degli algoritmi basati sul confronto
    - Omega(n*log(n))
- BST
    - Binary Search Tree: Albero di ricerca binario. Ogni chiave nel sottoalbero sinistro è < del nodo ed ogni chiave nel sottoalbero destro è maggiore del nodo in questione
- Che struttura dati usa la BFS?
    - Coda (FIFO)
- Che struttura dati usa Dijkstra e come funziona?
    - S: insieme dei vertici il cui peso di cammino minimo da s è già stato determinato
    - V-S: coda a priorità PQ dei vertici ancora da stimare
    - Termina per PQ vuota:
        - estrae u da V-S (d[u] minimo)
        - inserisce u in S
        - rilassa tutti gli archi uscenti da u

Programmazione:

- Due x: le entrate della seconda matrice sono la media delle caselle adiacenti della prima matrice. La seconda matrice è passata come puntatore (***) quindi deve essere fatta l'allocazione nella funzione
- Dato un vertice di un grafo trovare tutti i cicli di lunghezza k

21 -> 22 30/06/21

Teoria (Cabodi):

- Descrizione di DAG. Un DAG può avere componenti fortemente connesse?
- Hash table con open addressing + gestione delle cancellazioni.

Programmazione (Pasini <3):

- Date due liste (definire la struttura) ordinate, effettuare il merge in una terza lista evitando qualunque duplicato.
    - Aggiungendo un puntatore alla coda nella definizione del tipo lista è possibile evitare di aggiungere i duplicati appartenenti ad una sola lista controllando l'ultimo elemento inserito ogni volta.
- Dato un grafo (a scelta fra matrice e lista delle adiacenze) calcolare il numero di tutti i cicli semplici di lunghezza k che partono da un dato vertice v.

21 -> 22

Teoria (Cabodi)

- Online Connectivity (Operazioni di Union e Find)
- Memoization e programmazione dinamica: differenze
- Algoritmo di Kosaraju e complessità (complessità = theta(V+E))

Programmazione (Palena)

- Spiegare cosa sia un heap, e scrivere la funzione Heapify
- Definire ADT 1^a classe LIST. Date due liste, trovare l'intersezione delle due e salvarla nella prima lista, senza l'utilizzo di liste ausiliarie.

21 -> 24

Teoria:

- Counting sort e complessità. Si può usare per le stringhe?
- Componente connessa, punto di articolazione e ponte. Il DAG può avere punti di articolazione?
- Complessità algoritmi kruskal e prim

Programmazione:

- Coda come adt di prima classe implementato con vettore + funzione get e put
- Void BSTfree(BST root);

  scrivere una funzione che svuoti tutto il BST (può essere considerata il prototipo scritto come wrapper)

21 -> 24 (26/01/2021) (NO LAB)

Teoria (Camurati):

- Problemi P, NP, NP-C e la relazione tra P e NP
- Complessità Counting Sort e quando conviene usarlo
- Cosa si intende con "in loco"

Programmazione (Pasini):

- Prototipo List funz(List l1, List l2, ...)

  l1 e l2 liste di interi restituire in una terza lista l'intersezione insiemistica
- Calcolo combinatorio per costruire una stringa di k caratteri che abbia un certo numero di vocali al suo interno

21 -> 24

Programmazione:

- Dato un vettore di N elementi, scrivere una funzione C (che riceve un vettore di ingresso di interi e uno di float che coincide con il parametro di output) nella quale si calcola la media dell'i-esimo elemento come la media tra l'i-esimo e quello precedente fino a n/2 e da N/2 a N la media dell'i-esimo elemento con l'elemento successivo.ogni valore calcolato deve poi essere salvato nel vettore di output.
- Data una stringa di caratteri lunga N, restituire tutti gli anagrammi tali per cui una lettera sia ripetuta al più due volte e non vi siano lettere uguali vicine

Teoria:

- Discussione abbastanza generale su come si rappresentano i grafi, vantaggi e svantaggi di rappresentazione per matrice e lista di adiacenze

21 -> 25 (26/01/2021) (con lab) (con scritto da 12)

Teoria (Cabodi):

- BST annotato: normale BST con aggiunta di puntatore al padre del nodo e numero di nodi appartenenti al sottoalbero; applicazioni (successivo, predecessore, etc…);
- Tabelle di hash: strategie di gestione dell'eliminazione di un elemento da tabella gestita con open addressing (campo aggiuntivo per segnare che un elemento é stato eliminato oppure aggiornamento delle posizioni degli elementi successivi a quello eliminato).

Programmazione (Palena):

- Definizione di una ADT I classe per la gestione di una coda LIFO;
- Definizione di una funzione per calcolare, su k lanci di un dado, quanti di questi siano maggiori della media ottenibile dai lanci (somma>(k*sol[i])/2); ho usato il powerset generato
- con strategia divide et impera, mi è stato fatto notare che sarebbe stato più comodo usare quella basata su combinazioni.

21 -> 25

Programmazione:

- Data una matrice crearne una nuova dove ogni elemento sia sostituito dalla media tra l'elemento stesso e tutti quelli che gli sono intorno, facendo attenzione agli estremi.
- Dato un qualunque vertice di un grafo orientato determinare tutti i cicli di lunghezza k.

Teoria:

- Tabella di simboli (cos'è e come può essere implementata)
- cos'è la funzione di hash,svantaggi e vantaggi dell'hash universale.
- Cos'è un grafo non radicato, differenza tra grafo semplice ed aciclico

21 -> 25 (26/01/2021)

Teoria (Camurati)

- PQ, introduzione come heap e poi complessità con vettore ordinato/non ordinato
- Counting sort

Programmazione (Patti)

- Data una matrice RxC sparsa, implementare la funzione per ricompattarla sotto forma di vettore di liste in cui ogni nodo contiene valore diverso da zero e colonna
- Dato un grafo non orientato trovare tutti i cammini semplici di lunghezza k

21 -> 25

Programmazione:

- Inserzione in uno heap.

- Implementare la visita in profondità di un grafo rappresentato con lista di adiacenza con un vincolo: se un nodo ha più di tre archi uscenti non andare oltre il terzo.
- Cancellazione da una FIFO implementata come ADT di prima categoria.

Teoria:

- Componente fortemente connessa.
- Cosa significa massimale.
-
- Quale algoritmo si usa per trovare la componente fortemente connessa e dimostrarne la complessità.
- Come si può rappresentare un grafo ed elencare vantaggi e svantaggi di ogni opzione.
- Come si inizializza un grafo leggendo gli archi da un file senza conoscerne a priori la quantità.

21 -> 25 (26/01/2021)(con lab)

Teoria (Camurati)

- Grafi, Prim e Kruskal, mst tramite calcolo combinatorio, complessità operazioni sui grafi (visite)
- Powerset tramite combinazioni semplici

Programmazione (Patti)

- Intersezione tra insiemi (rappresentati tramite vettori) con vettore risultante passato come void da allocare dentro la funzione
- Creare sequenze numeriche particolari tramite disposizioni ripetute, particolare attenzione al pruning

21 -> 26

Teoria:

- Esiste un limite inferiore alla complessità degli algoritmi di ordinamento? Dimostra come si ricava
    - O(n*log(n))
- Qual è la complessità del quicksort? Scrivi l'equazione alle ricorrenze del caso peggiore

Programmazione:

Definisci la struttura dati di una lista come ADT, compresi i singoli nodi, puntatori ecc.Scrivi un programma che riceva come parametro una lista e cancelli un elemento si e uno no fino alla fine. Scrivi un programma che tramite una visita in profondità determini le componenti connesse di un grafo

21->26 (07/02/2022)

Teoria:

- Partition in un BST:
    - con bst annotato (con numero di vertici nel sottoalbero) con relativa complessità:
        - operazione che riordina il bst mantenendone le proprietà con come radice la r-esima chiave più piccola nel bst (r = rango, passato per parametro)
        - T(n)=O (h) = O(log(n)) se bst è bilanciato
    - complessità senza lista annotata:
        - O (n) in quanto dovrei fare una visita INorder del bst per ordinare i vertici
- bridge e punti di articolazione
    - cosa sono e come li posso trovare
        - archi/vertici la cui rimozione sconnette il grafo, posso trovarli provando a togliere uno alla volta i vertici/archi e verificando se il grafo si disconnette
    - la presenza si bridge implica la presenza di punti di articolazione
        - Si, i vertici adiacenti al bridge
    - la presenza di punti di articolazione implica la presenza di bridge
        - No, un punto di articolazione può avere rango maggiore di 3 e quindi la rimozione di un arco adiacente a esso non implica la sconnessione del grafo
- In un grafo si dicono cricche (o qualcosa di simile) dei sottografi completi e massimali (non studiato a teria), dato un grafo nel quale si e a conoscenza di un sottografo completo come posso determinare se questo è anche massimale:
        - soluzione banale: itero su tutti gli altri vertici e verifico se qualcuno di questi è connesso a tutti i vertici del sottografo (consiglio non cercate soluzioni assurde se ve ne viene in mente una banale date quella)
    - Per fare questa operazione meglio matrice adj o lista?
        - In generale matrice per accesso O(1) alla topologia del grafo ma non c'è un migliore in senso assoluto, dipende dal grafo (sparso, denso….)

Programmazione (Pasini)

- Date due liste l1 e l2 ordinate creare una lista l3 ordinata, composta dagli elementi l1 e l2 rimuovendo i duplicati
    - soluzione: Usare una strategia simile a quella del merge (nel merge sort) tenere 3 link uno con cui scorrere l1 uno per l2 e uno per l3, prima di inserire un elemento vedere se è uguale all'ultimo aggiunto, in questo modo l'algoritmo è O(n1 + n2)
- Funzione ricorsiva check di un SumTree, un albero binario con ogni nodo di valore pari alla somma dei due figli
    - soluzione: Simile a una visita in post order:

        int fRicorsiva (link r, link z)

        - Condizione di terminazione: r == z ritorno 1 → un albero "nullo" è un SumTree
        - Ricorsione:

- ricorro su sottoalbero sinistro (memorizzo output)
- ricorro sul destro (memorizzo output)
- se i due sottoalberi sono sumTree e il nodo attuale è la somma di figli dx e sx allora restituisco 1.

21 -> 26 (26/01/2021) (con lab)

Ho fatto il compito da (24)

Teoria(Cabodi), 5-10 min:

- BF e Dijkstra quando si usano(cicli con o senza peso negativo, archi con o senza peso negativo)
- Weighted Quick Union(come funziona e perché la complessità è logaritmica)
- Costo nel caso di un ADT SET (vettore ordinato/disordinato) per l'unione insiemistica e l'intersezione
- Tra Kruskal e Prim quale sceglieresti?(complessità di entrambi)

Programmazione (Patti), 20 min:

- Data una matrice sparsa, ricompattarla in vettore di liste, in cui ogni nodo contiene la colonna e il valore dell'elemento(se è diverso da 0). Mi è stato chiesto come potevo fare se non avessi voluto ritornare nulla(passavo il vettore di liste per riferimento)
- Determinare il numero di cammini semplici di lunghezza k, a partire dal vertice v;

21 -> 26 (26/01/2021) (+2 di lab)

Teoria (Camurati)

- - stack frame e come si implementa la ricorsione
- - powerset e possibili implementazioni (Divide & Impera, Disp. ripetute e Combinazioni semplici)
- - definire un ADT 1C (concettualmente e a livello di codice)

Programmazione (Vendraminetto)

- - Definizione ADT 1C stack (implementato come vettore) e funzione pop
- - Definire come Quasi ADT Grafo e relativi nodi. Scrivere funzione che, dato un grafo G non pesato, non orientato e un vertice V, ritorni il numero di vertici raggiungibili da V (risolto con le componenti connesse)

21->26 (16/02/2021)(+2 di lab)

Teoria (Cabodi):

- Differenza tra linear probing e double hashing
- Selezione di attività. Selezione di attività che si possono intersecare fino a k-volte come cambia l'algoritmo (non c'è sulle slide, è una domanda di ragionamento)

- archi Cross
- La DFS è ricorsiva? E la BFS?

Programmazione (Patti):

- Grafo trasposto con lista di adiacenza;
- cammino semplice: parti da un vertice b e deve essere lungo k

21 -> 27 (26/01/2021)

Teoria (Camurati):

- Definizione di BST
- Albero completo e bilanciato
- Merge Sort
- Grafo completo e bipartito

Programmazione (Pasini):0

- Implementare codici di Huffman considerando la coda a priorità come di libreria, quindi scrivere la parte dove estraggo i 2 priorità minore e li unisco
- Dato un grafo orientato, non pesato, un vertice di inizio e un valore k, calcolare tutti i cammini anche non semplici di al massimo k, dove però k viene decrementato solo se non è la prima volta che si passa sul vertice

22 -> 22 (02/2019)

Teoria:

- Organizzazione chiamate ricorsione, stack frame
- Albero non radicato
- Cos'è un iBST

Programmazione:

- Date due liste ordinate, farne una terza ordinata senza duplicati
- Dato un grafo non orientato non pesato e dato un vertice di partenza, contare tutti i cicli anche non semplici di lunghezza k

22 -> 23 (26/01/2021)

Teoria (Cabodi):

- Counting sort: complessità e quando utilizzarlo;

- Differenze tra mergesort, quicksort e heapsort e perché il quicksort è più utilizzato;
- DAG e ordinamento topologico (normale e inverso).

Programmazione:

- Date due liste non ordinate costruire una terza lista ordinata che contenga gli elementi delle prime due liste, dove gli elementi uguali sono ripetuti solo una volta;
- Implementazione del calcolo delle componenti connesse.

22 -> 23

Teoria:

- Limite inferiore degli ordinamenti basati sul confronto e dimostrazione (formula di Stirling)
    - Algoritmi basati sul confronto: confronto a_i e a_j
    - Esito: decisione ($a_i > a_j$ o $a_i <= a_j$), riportata su un albero delle decisioni
    - Per n interi distinti: numero di ordinamenti = numero di permutazioni n!
        - Complessità: numero h di confronti (altezza dell'albero)
        - Ogni soluzione = foglia
        - Numero di foglie = 2^h
        - Approssimazione di Stirling: $n! > (n/e)^n$ quindi $2^h >= n! > (n/e)^n$
        - $h > lg(n/e)^n = n \lg n - n \lg e = (n \lg n)$
- Gestione degli elementi cancellati in una hash table
- Come si può implementare un ordinamento basato sulla hash table
    - Domanda trabocchetto: non si può

Programmazione:

- Funzione di verifica del gioco Forza4, cioè data una matrice verificare che ci siano 4 "1" consecutivi sulla riga/colonna/diagonali
- Funzione che ritorna il numero di elementi presenti al livello i-esimo di un heap con i dato
    - Non serve una visita basta verificare che l'i-esimo livello non sia l'ultimo

22 -> 24 <u>(26/01/2021)</u>

Teoria (Camurati):

- classe P, NP, NP-C
- complessità ricerca dicotomica

- karatsuba(solo quando si applica)

Programmazione (Patti):

- grafo con lista, fare la funzione reverse
- dato un vettore contenente tutte le lettere (una sola volta), genera una stringa di dimensione 3, dove le prime due lettere sono vocali e l'ultima è una consonante, la seconda vocale deve essere maggiore della prima

22 -> 25 (16/02/2021)

Teoria (Camurati)

- Cos'è una tabella ad accesso diretto e per cosa viene utilizzata
- Come si trasformano le stringhe in indici quando usiamo una tabella di hash
- Cos'è una funzione tail-recursive e se è semplice da implementare

Programmazione (Pasini)

- Cancellazione nodi con valore pari ad un val dato da una lista di interi linkata fatta come ADT di I classe, con definizione del tipo
- Contare tutti i cammini semplici di un grafo di lunghezza pari a k

22 -> 25 (26/01/2021)

Teoria (Camurati):

- Complessità DFS con matrice delle adiacenze e con lista delle adiacenze
- Definire in generale una coda a priorità (ADT di I classe che supporta operazioni di un certo tipo ecc..)
- Cos'è un codice libero da prefisso e descrivere ad alto livello il funzionamento del procedimento per formare i codici di Huffman

Programmazione (Pasini):

- Date due liste ordinate, generare una terza lista che contiene gli elementi delle prime due (tranne i duplicati)
- Dato un grafo orientato, non pesato, un vertice di partenza v e una lunghezza k, contare il numero di cicli semplici a partire da v di lunghezza k

22 -> 25

Teoria:

- Proprietà di un heap

    - In un heap il nodo padre è maggiore o uguale a tutti i discendenti; è un albero binario quasi completo (tutti i livelli completi, tranne eventualmente l'ultimo, riempito da SX a DX), il che implica che sia anche quasi bilanciato
- Matrice e lista di adiacenze, vantaggi e complessità
    - Matrice: complessità O(|V|^2), è vantaggiosa per grafi densi (|E| circa uguale a |V|^2), non presenta costi aggiuntivi in caso di grafi pesati (basta mettere anziché 1 il peso dell'arco), ha un accesso O(1)
    - Lista: complessità O(|V+E|), vantaggiosa per grafi sparsi ($|E| << |V|^2$), richiede costo aggiuntivo per memorizzare il peso in caso di grafi pesati e per verificare l'adiacenza di due vertici devi scorrere le relative liste
- Come implementare la lista di adiacenze quando non si ha l'informazione sul numero di vertici del grafo?
    - Lista di liste
- Cos'è un albero ricoprente minimo?
    - Un minimum spanning tree è un albero aciclico che copre tutti i vertici del grafo. L'albero MST è unico se e solo se tutti i pesi sono distinti
- Esempio di un problema np-c
    - Ciclo di Hamilton
    - Data una funzione booleana, determinare se esiste una qualche combinazione di valori delle variabili di ingresso per cui la funzione risulti vera

Programmazione:

- 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

22 -> 25 (26/01/2021)

Programmazione:

- Lista (semplice descrizione della struttura dati + funzione che date due liste salvi nella prima lista tutti gli item che differiscono tra le due liste)
- Albero di tipo left child - right sibling(semplice descrizione della struttura dati (quindi ogni nodo ha un puntatore al suo figlio sinistro e a suo fratello destro) + funzione che calcola quanti nodi hanno un numero pari di figli)

Teoria (Cabodi (stra simpatico btw)):

- Perché è impossibile per un algoritmo Union-find avere sia una quick find che una Union find
- È possibile in un grafo connesso avere vertici senza archi (si se il grafo è composto da un solo vertice)
- Codice di Huffman

22 -> 25

Programmazione:

- Implementare il codice della funzione "partition" del Quicksort.
- Ricerca e cancellazione di un nodo di un BST avendo nota la chiave Key e sapendo che il nodo ha 1 solo figlio, non sapendo a priori se il figlio sia destro o sinistro
- (soluzione:dopo aver trovato il nodo giusto, vedere se ha un figlio a sx o a dx, quindi salvare con un puntatore il figlio, cancellare il nodo e rimpiazzarlo con il puntatore che punta al figlio).

Teoria:

- Quali sono le caratteristiche degli alberi binari bilanciati, quasi bilanciati, completi (in pratica voleva sapere la differenza del concetto di profondità, altezza e numero di nodi tra le diverse tipologie).
- Cosa comporta l'eliminazione di un arco da un albero binario (comporta la creazione di un nuovo albero e quindi si passa da un albero ad una foresta di alberi).
- Quanti cammini esistono tra la radice e un nodo dell'albero(esiste un unico cammino).
- Cosa sono i cammini di Hamilton e quelli di Eulero e cosa sono i cammini semplici.

22 -> 25

Teoria:

- Cos'è e come si trova una chiave di un dato rango in un BST
    - Uso una visita. La visita però ha un costo lineare. C'è di meglio?
        - Usare la BSTselect
- Formula generale di una equazione alle ricorrenze? Spiega il significato dei simboli
    - T(n) = D(n) + aT(n/b) + C(n)
- Quanto valgono a, b per la ricerca dicotomica?
    - a=1, b=2
- Caso peggiore Quicksort, complessità?
    - O(n^2)
- Quando si verifica?
    - Quando il pivot è il valore massimo o il valore minimo d el vettore
- Scrivi equazione alle ricorrenze per Quicksort
    - Caso peggiore: T(n) = T(n-1) + n

- Cosa rappresenta la “n” nella formula precedente?
    - Il costo della partizione

Programmazione:

- Inversione lista linkata semplice
- Heapify

22 -> 26

Teoria:

- Hashing:
    - Come implementare (Metodo moltiplicativo, modulare e moltiplicativo-modulare);
    - Collisioni: cosa sono e metodi di gestione;
    - Delete in caso di Open Addressing;

Programmazione:

- 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);

22 -> 26

- Programmazione:
- Dati i numeri da 0 a 9 enumerare tutte le soluzioni di 5 elementi anche ripetuti affinché sia rispettata la condizione che non ci siano due elementi dispari adiacenti o due elementi pari adiacenti e che la somma della soluzione sia divisibile per 3.(disposizioni ripetute) Data una lista doppio linkata fare un programma che elimini i nodi dispari eccetto la radice.

Teoria:

- Caratteristiche bst. Date tre chiavi con ordine k1

22 -> 27 (26/01/2021) (con lab)

Teoria (Cabodi)

- Shell sort, come funziona e complessità. Insertion/Bubble/Selection Sort dire se stabili/non stabili e in loco/non in loco
- DAG può avere cicli? Se gli archi non sono più orientati si possono avere cicli?
- Complessità kruskal qual è e perchè è logaritmica nel numero di vertici

Programmazione (Vendraminetto)

- Data una stringa ed un carattere separatore scrivere una funzione in cui si separano le sottostringhe e le si inseriscono in un vettore passato per riferimento, la funzione deve ritornare il numero di sottostringhe
- Scrivere una funzione che calcoli il numero di foglie di un bst (scrivere anche typedef del bst)

23 -> 24

Teoria:

- Complessità algoritmi di ordinamento basati su confronto e dimostrazione
  - Algoritmi basati sul confronto: confronto a_i e a_j
  - Esito: decisione ($a_i > a_j$ o $a_i <= a_j$), riportata su un albero delle decisioni
  - Per n interi distinti: numero di ordinamenti = numero di permutazioni n!
    - Complessità: numero h di confronti (altezza dell'albero)
    - Ogni soluzione = foglia
    - Numero di foglie = 2^h
    - Approssimazione di Stirling: $n! > (n/e)^n$ quindi $2^h >= n! > (n/e)^n$
    - $h > lg(n/e)^n = n \lg n - n \lg e = \Omega(n \lg n)$
- Minimum Spanning Tree
  - Un minimum spanning tree o albero ricoprente minimo è un albero aciclico che copre tutti i vertici del grafo. L'albero MST è unico se e solo se tutti i pesi sono distinti

Programmazione:

- Implementare un set come adt di prima categoria ed implementare una funzione merge che fonde due set in uno

- Funzione che indichi se è presente un ciclo di lunghezza >= k

23 -> 25

Teoria:

- Punti di articolazione
    - Vertice la cui rimozione disconnette il grafo. Rimuovendo il vertice si rimuovono anche gli archi su di esso insistenti
- Componenti fortemente connesse
    - Una componente fortemente connessa di un grafo diretto G è un sottografo massimale di G in cui esiste un cammino orientato tra ogni coppia di nodi ad esso appartenenti
- Complessità del grafo trasposto
    - Quindi la complessità totale è O(n + |E|)

Programmazione:

- Merge di due liste, un programma sulle disposizioni con ripetizione con robe da controllare in backtracking

23 -> 25

Programmazione:

- ADT di prima categoria di fifo con buffer circolare di inserimento
- dato un grafo orientato implementato con lista delle adiacenze trovare tutti i cicli di lunghezza 2 per i vertici

Teoria:

- MST:per quali grafi? su che algoritmi si basa? cos'è un taglio?

23 -> 25

Teoria(Camurati)

1. Ciclo di Hamilton e il problema del commesso viaggiatore
2. Successore e predecessore

Programmazione(Palena)

1. ADT lista di prima specie (implementarlo con il puntatore opaco nel .h) ed eliminare tutti i nodi con chiave pari
2. Modelli delle disposizioni semplici con pruning

23 -> 27 (17/02/2023) (+2 lab)

teoria camurati:

- 1. componenti connesse, componenti fortemente connesse.
- 2 numero di archi di un grafo completo, modello del calcolo combinatorio da usare per generare gli archi a partire dai vertici

programmazione cabodi:

- 1. trovare il numero di elementi compresi fra due valori dati.
- 2. struttura dati per un multigrafo, come modificare opportunamente lista delle adiacenze o la matrice delle adiacenze

23 -> 26

Teoria:

- Modelli di calcolo combinatorio per il problema delle 8 regine
- Problema della colorazione di una mappa (grafo planare) come adattamento del problema di partizionamento di un grafo

Programmazione:

- Inserimento in coda prioritaria implementata come ADT I categoria tramite lista linkata e tipo item ADT I cat
- Problema dello zaino discreto, esaustivo, ricorsivo, con pruning

23 -> 28

Programmazione:

- Definire una lista in grado di contenere sia un intero che una stringa (a seconda che si chiami l'una o l'altra funzione di inserzione) e di tenere traccia per ogni nodo se quel nodo h tenendo poi conto del tipo di dato effettivamente inserito mediante un flag "type" contenente 0 per gli interi e 1 per le stringhe)
- scrivere le due funzioni di inserzione in coda.
- Dato un grafo, scrivere una funzione che determini, a partire da un dato nodo v, tutti i cammini semplici di lunghezza k a partire da v.

Teoria:

- Caratteristiche degli alberi (numero di archi (|E|=|V|-1)
- in un albero, profondità, altezza(definita anche come massima profondità di una foglia)
- tipi di nodi e definizione di albero

- su quale paradigma e teorema si fondano gli algoritmi degli MST.

23 -> 28 (26/01/2021) (+2 lab)

Teoria (Cabodi)

- Differenza tra programmazione dinamica e memoization
- Esempio di algoritmo di programmazione dinamica applicato ai grafi (Bellman-Ford)
- Complessità del trovare l'out degree e l'in degree di uno o tutti i vertici di un grafo implementato con lista di adiacenza. (out degree: O(out) nel caso di 1 vertice, O(E) tutti i vertici; in degree: O(E) nel caso di un vertice, O(E) nel caso di tutti i vertici trasponendo il grafo)

Programmazione

- ADT di I classe STACK implementato con vettori dinamici
- Implementare un algoritmo di calcolo combinatorio per trovare tutte le combinazioni possibili dopo il lancio di k dadi (combinazioni ripetute) con condizioni sulla ripetizione di valori e sulla somma minima, da applicare attraverso pruning

24 -> 25

Programmazione:

- Scrivere una funzione per scorporare una lista di interi in due liste, una contenente i numeri pari e una contenente i numeri dispari. La lista di partenza deve essere distrutta.
- Scrivere una funzione per riconoscere se un grafo diretto è un DAG o un albero.

Teoria:

- Che cos'è una collisione e come gestisce.
- Che cos'è una coda a priorità e costi delle varie operazioni quando la coda è implementata tramite heap.
- Differenze tra un grafo diretto ciclico e aciclico.

24 -> 26

Teoria:19

- Heap: cosa sono, operazioni principali e complessità
- MST (Minimum Spanning Tree) cosa sono, teorema e corollario
- Algoritmi che ne derivano
- Archi sicuri

Programmazione:dato un albero rappresentato come left child right sibling contare i numero di figli di ogni nodo (solo figli diretti)

24 -> 27 (16/02/2021) (con lab)

Teoria (Camurati):

- complessità del counting sort
- perchè è sconsigliato usarlo anche se lineare?
- equazione alle ricorrenze della ricerca dicotomica

Programmazione (Pasini)

- data due liste già ordinate, inserire in una terza lista l'intersezione insiemistica delle due
- dato un albero LCRS (Left- Child Right- Sibling) trovare per ogni nodo il numero di figli tramite funzione ricorsiva

24 -> 27

Programmazione:

- Data una matrice sparsa implementata come ADT di prima categoria, scrivere una funzione che generi la matrice inversa.
- Implementa il problema greedy dello zainetto discreto.

Teoria:

- Quando un algoritmo di ordinamento si dice "stabile" e quando "in loco"?
- fammi alcuni esempi di algoritmi che conosci.
- Quante volte ricorre su ogni arco l'algoritmo di Bellman-Ford ? Parlami della "relaxation".

24 -> 27 (con 2 punti di lab)

Teoria:

- Complessità inferiore degli algoritmi di ordinamento;
- Equazione delle ricorrenze del divide and conquer
- Algoritmo di Bellman-Ford

Programmazione:

- Eliminazione di nodi il cui contenuto è pari data una lista
- Dato un grafo è un vertice di partenza elencare il numero di cammini semplici lunghi esattamente k

24 -> 28 (con 2 punti di lab) 14/09/21

Teoria:

- Predecessore e successore in un BST e complessità dell'algoritmo

- Cammino di Eulero
- Motivo per cui l'algoritmo di Bellman-Ford esegue solo V-1 iterazioni

Programmazione:

- Unione di due liste in ADT di I classe evitando duplicati
- Anagrammi distinti di una stringa con caratteri ripetuti (Permutazioni con Ripetizione)

25 -> 25

Teoria:

- Equazione alle ricorrenze del mergesort
    - $T(n) = 2T(n/2) + n$
- Classificazioni degli archi di un DAG
    - Tree: archi dell'albero della visita in profondità
    - Backward: archi che connettono un vertice ad un suo antenato nell'albero (cappi compresi)
    - Forward: archi che connettono un vertice ad un suo discendente nell'albero
    - Cross: archi rimanenti, attraversano da un ramo all'altro o collegano due alberi diversi
- Complessità visita in profondità
    - Lista adiacenze: $T(n) = (|V|+|E|)$
    - Matrice adiacenze: $T(n) = (|V|^2)$

Programmazione:

- Invertire una lista. Creare la struttura dati di una lista doppio linkata con FIFO

25 -> 26 (26/01/2021)

Teoria (Cabodi):

- IBST: caratteristiche, funzione search
- componente fortemente connessa: definizione, Kosaraju

Programmazione (Pasini):

- Date due liste di interi l1 e l2, restituire una lista che contiene l'intersezione insiemistica tra le due. prototipo: List f(List l1, List l2, …);
- Dato un grafo g, un vertice v e un intero k, restituire un vettore che contiene tutti i vertici a distanza k da v. prototipo: int* f(Graph g, int v, int k, …);

25 -> 26

Teoria (Cabodi):

- Scrivere l'equazione alle ricorrenze delle Disposizioni Semplici
- Bellman Ford: complessità, differenze con Dijkstra
- Come varia la complessità delle operazioni su una PQ a seconda della sua implementazione
- IBST: caratteristiche, qual è la chiave, a cosa servono, quali operazioni si effettuano (intersezioni) - o qualcosa del genere
- Come modificare un BST per facilitare l'operazione di partizione attorno a una chiave data (aggiungendo l'informazione sulla grandezza del sottoalbero dx e sx)

Programmazione (Vendraminetto):

- Scrivere implementazione di int strcmp(char* s1, char* s2) ;
- Scrivere implementazione di una funzione int BSThigh(BST root) che calcola la profondità del BST

25 -> 27

Teoria:

- IBST: descrizione ed estensioni da eseguire al BST di base per implementarlo
- Condizione per cui due intervalli si intersecano
- Order BST: domande su numero di nodi, altezza dell'albero
- Rotazione di BST

Programmazione:

- Dato un file contenente informazioni su persone (nome e età), inserire i dati in una coda implementata come ADT I Categoria salvando i dati letti in una struct persona prima dell'inserzione in coda
- Dato un grafo implementato come lista di adiacenze o matrice di adiacenze (a discrezione dello studente), si parta da un vertice qualunque e si stampi un cammino lungo al più k (k generico)

25 -> 28

Teoria:

- Differenze tra combinazioni, disposizioni e permutazioni
- Dimostrazione complessità
- Omega grande per algoritmi di ordinamento basati sul confronto

Programmazione:

- Algoritmo Quick-union e differenza con weighted quick-union
- Inserimento in FIFO con buffer circolare

- Enumerazione di tutti i cammini semplici tra due vertici di un grafo

25 -> 29 (26/01/2021)

Teoria (Cabodi the best)

- Algoritmi union find
  - Veloce intro
  - Perché avere una veloce implica l'altra lenta
  - Intersezione insiemistica tramite union find (non possibile, ho solo informazioni sul: non hanno intersezioni/intersezione totale, sono lo stesso insieme)
- Algoritmo di Prim

Programmazione (Patti)

- Ordinare una lista singolo linkata (o sulla stessa lista tipo selection sort o su lista parallela, indifferente)
- Algoritmo per determinare grafo bipartito (domanda di teoria, funzione ricorsiva, funzione di check)

25 -> 29 (16/02/2021) + 2 punti di lab

Teoria (Camurati)

- Classi degli algoritmi
- componenti fortemente connesse con relativa spiegazione del kernel
- notazioni asintotiche

Programmazione

- dato un grafo orientato e un vertice, definire tutti i vertici che distano ad un cammino minore o uguale ad un valore k definito in precedenza (visita in ampiezza)
- data una stringa e un valore n, far slittare di n la stringa (es: data la stringa programmazione e n=3 l'output dovrà essere oneprogrammazi)

25 -> 30 (26/01/2021) (+2 Lab)

Teoria

- Tabelle ad accesso diretto. Costo delle operazioni di ricerca, inserzione, eliminazione
- Decrease and conquer. Equazione delle ricorrenze ed esempio

Programmazione

- MST con Kruskal, implementazione union e find a piacere

- Vertici raggiungibili da v con al massimo k passi

25-> 30 (primo appello sessione invernale 2022)

Programmazione:

- Coda a priorità con lista linkata inserimento e cancellazioni
- Tutti i cammini semplici (con calcolo combinatorio viene più semplice che robe strane con hamilton)

Teoria:

- equazioni ricorrenze delle combinazioni semplici(fai attenzione a tenere conto dello start)
- Dag e ordinamento topologico (ho anche spiegato l'algoritmo)
- Se hai un grafo che ha i pesi sia sui vertici che sugli archi e in un cammino minimo fai la somma dei due come puoi fare l'algoritmo dei cammini minimi. Tutto questo dando per scontato che i pesi non sono negativi (o modifichi l'algoritmo con djikstra (forse anche con bellman ford, ma io ho fatto con djikstra) o ti riporti all'algoritmo originale modificando gli archi che entrano in un vertice aggiungendo il valore del peso del nodo)

26 -> ? (26/01/2021)

Teoria (Cabodi):
- differenze problemi di ricerca e ottimizzazione
- calcolo combinatorio: disposizioni e combinazioni quale sia più efficiente a parità di dati e perché (con relative formule matematiche)
- funzione di hash (come tenerla O(1))
- grafi densi/sparsi e quando conviene usare matrice di adiacenza al posto di lista dal punto di vista temporale

Programmazione (Patti):
- ordinamento in lista (a scelta se linkata o doppio linkata ma soluzione più easy con doppio linkata)
- algoritmo per determinare grafo bipartito (disposizioni ripetute sui vertici con relativo check)

26 -> 28

Programmazione:

- inserimento e cancellazione in lista doppio linkata.
- In un bst crescente farlo diventare decrescente, quindi con ordine delle chiavi contrario, quelle più

piccole a destra e quelle più grandi a sinistra.

Teoria:

- domande sulla ricorsione, i suoi limiti e le alternative, come potrei fare per risolvere il problema di indipendenza dei sottoproblemi.

26 -> 29

Teoria:

- Partizione di un insieme

Programmazione:

- Inserimento bst in radice
- Quicksort
- Esercizio su puntatori

26 -> 29

Programmazione:

- Implementare codici di Huffman.
- Stampare a video tutti i cicli di lunghezza L in un grafo.

Teoria:

- Definizione di Heap, heapsort.
- Complessità degli algoritmi di Djikstra e di Bellman-Ford, quando si usa Bellman- Ford?

26 -> 29 (26/01/2021)

Teoria(Camurati)

- Come viene rappresentata la ricorsione a livello di memoria
- Stack frame
- Funzioni tail recursive e nonCome viene rappresentata la ricorsione a livello di memoria tail recursive

Programmazione(Palena)

- Implementazione ADT stack, con funzioni base di inizializzazione,pop e push
- Funzione che dato un vertice di un grafo non orientato implementato con matrice di adiacenze individui se presente un ciclo semplice

26 -> 30 (08/02/2022)

Teoria [Cabodi]

- Equazione alle ricorrenze del modello delle disposizioni semplici
- Tabelle di hash in open addressing
    - quali sono i vantaggi del quadratic probing e double hashing sul linear probing? (domanda simile a quella di Palena, mi sembra la risposta risieda nella gestibilità dei cluster e nei tentativi medi)
    - nella ricerca, posso aspettarmi più tentativi medi nel caso di una search hit o una search miss?
- Ordinamento topologico
    - cos'è?
    - un DAG ne è sempre disposto?
    - come si trova?
    - è vero che ciascun nodo source si trova prima di ciascun nodo sink? (no)

Programmazione [Palena]

- ST con double hashing come ADT I, contenente Item ADT I
    - definizione struct st in ST.h e ST.c
    - prototipo di hash1 e hash2
    - codifica di STinsert(ST st, Item X) utilizzando le funzioni standard di Item ove necessario
    - come mai il double hashing può risultare più efficace del linear probing?
- BST esteso di int
    - definizione struct di un BST esteso
    - come si trova il successivo di un nodo?
    - codifica di BSTsucc(BST bst, int x)

26 -> 30

Teoria:

- Minimum Spanning Tree
- Paradigma greedy
- In che senso gli algoritmi di ricerca dei cammini minimi e dei minimum spanning tree sono greedy?
- Notazione sigma grande e theta grande

Programmazione:

- Inserzione e cancellazione in una lista doppio linkata
- Dato un grafo contare (ed eventualmente stampare) quanti sono i cammini semplici di lunghezza k

26 -> 30

Programmazione:

- Dati 2 bst, di cui il secondo notoriamente più piccolo del primo, ricevuti come parametri di una funzione, contare il numero di ripetizioni del secondo nel primo.
- scrivere una funzione che, ricevuto un grafo come parametro, riconosca se si tratta di un dag o di un albero.

Teoria:

- Ragionamenti di tipo teorico sulle liste e possibile implementazione senza l'uso di dati puntatore

26 -> 30

Programmazione:

- trovare tutti i cammini semplici di un grafo e cos'è' dal punto di vista del calcolo combinatorio

Teoria:

- problema di giuseppe flavio con lista circolare doppia che ad ogni cancellazione inverta il senso (orario-antiorario)
- dato un grafo e due suoi vertici A e B, trovare i vertici che distano al massimo k da A o B

26 -> 30 (26/01/2021) (+2lab)

Teoria (Cabodi):

- Classi P ed NP
- Alberi left-child right-sibling, quali sono i vantaggi
- Costo della ricerca in un albero ternario (BST con 3 figli per nodo), cambia rispetto al BST? (no, ha sempre complessità logaritmica)

Programmazione (Pasini):

- Costruzione di un albero dei codici di Huffman
- A partire dal vertice di un grafo, contare il numero di cicli dove sono ammessi al massimo k "ripassaggi" da ciascun vertice

26 -> 30L (+2 Lab)

Programmazione:

- Partendo da una stringa di n caratteri e preso in input un carattere "separatore" generare una lista in cui in ogni nodo c'è una sottostringa (interrotta dal separatore).

- Implementazione di Union-Find (tramite QuickUnion)
- implementazione dell'algoritmo di Kruskal con relativa stampa del mst.

Teoria:

- Due modi ricorsivi con cui calcolerei l'altezza di un albero binario.
- Equazione alle ricorrenze e relativo sviluppo per unfolding nel caso di albero bilanciato o fortemente sbilanciato (lista).
- Metodo per implementare una lista senza l'uso dei puntatori (utilizzando gli indici).
- Partendo da N dati, quante sono le possibili liste che posso implementare con questo metodo? (Risp: a meno di ordinamenti posso implementare da 1 a N liste distinte).

27 -> 27

Programmazione:

- inserimenti in lista di stringhe ordinate e vettori di liste ordinate
- trovare, per ogni vertice di un grafo con matrice delle adiacenze, tutti i cammini lunghi esattamente k

Teoria:

- espressione matematica delle varie notazioni asintotiche (con grafico)
- equazioni alle ricorrenze di divide et impera e decrease et conquer

27 -> 28

Programmazione:

- Visita in profondità.
- Merge di due liste

Teoria:

- Qualche nozione sui grafi (grafo connesso, completo ecc...)
- Powerset

27+2 (Lab)-> 30 (21/02/2023)

Teoria:

- Kruskal: algoritmo, struttura dati, come fa il controllo dei cicli?
- Codici di Huffman e la sua decodifica

Programmazione

- Inversione in loco ricorsiva di lista singolo linkata

- Dato un left child right sibling trova il numero di figli di ogni nodo

27 -> 30

Teoria:

- Cos'è la relaxation, come viene utilizzata nell'algoritmo di Dijkstra.
- Cos'è il rango in un BST, come modificare l'adt BST per applicare algoritmi di ricerca della chiave con rango k.
- Dal punto di vista insiemistico, gli alberi Bilanciati sono un sotto insieme degli alberi completi, o il contrario?

Programmazione:

- Data una matrice di interi poco densa (ci sono molti 0), 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.

27 -> 30

Programmazione:

- inserire in una coda a priorità con heap dei vettori ed estrazione del massimo (tutte le funzioni praticamente)
- trovare cicli semplici di lunghezza k in un grafo

Teoria:

- alberi left child right sibling, come implementeresti una insert in questi alberi
- problema pqchange senza tabella di simboli
- un po' di complessità di funzioni queue
- complessità in memoria di un grafo

27 -> 30

Programmazione:

- Modulo PQUEUE implementato come ADT I categoria con liste, con relative funzioni ricorsive per inserimento, cancellazione e ricerca
- Ricerca del numero di cammini di lunghezza k da un singolo vertice in un grafo

Teoria:

- Come implementare una coda a priorità quando il dato immagazzinato è troppo grande (non heap ma tabella di simboli con accesso diretto)

27 -> 30

Programmazione:

- Lista doppio-linkata: contare quanti nodi ci sono prima di un nodo.
- Trovare tutti i cicli di lunghezza k che passano al massimo 2 volte per lo stesso nodo.

Teoria:

- PQueue, Perché implementare queue tramite heap e non tramite lista
- Cammino Semplice
- Cammino Hamiltoniano
- Componenti fortemente connesse
- Algoritmo per trovare le componenti fortemente connesse

27 -> 30 (26/01/2021)

Teoria : (Camurati)

- Cancellazione nell'open addressing
- Cos'è un grafo statico e come ti comporteresti se in un problema i vertici venissero aggiunti/eliminati (si utilizza una variabile flag per marcare se un vertice è presente/assente)
- Classi P, NP ecc.. (riguardo le classi NP ha chiesto anche cosa fosse una macchina di Turing non-deterministica)

Programmazione: (Pasini)

- Invertire una lista in loco
- Dato un grafo non orientato ed un vertice di partenza, salvare in un vettore tutti i vertici raggiungibili da quel vertice con cammini di max k passi (l'ho risolto con un algoritmo di costo esponenziale (una specie di dfsr), ma il modo migliore di risolverlo era utilizzare una visita in ampiezza, dato che il grafo è non orientato)

27 -> 30 (26/01/2021)

Teoria (Cabodi)[5-10m]

- Problema greedy di activity selection (descrivere la procedura usata), variante per massimizzare il tempo complessivamente coperto dalle attività al posto del numero di attività svolte: è garantita la soluzione ottima/perchè? (No, non è garantita)
- È possibile rappresentare un multigrafo pesato o meno con un grafo pesato? (si, aggiungendo archi di peso pari alla somma degli archi del multigrafo/peso pari alla somma dei pesi degli archi del multigrafo)

Programmazione (Patti) [20m]

- Data una lista doppio linkata scrivere funzioni di inserzione ordinata e cancellazione di un item
- Dato un grafo scrivere una funzione per determinare se sia o meno bipartito

27-> 30 (appello 16/02)

Teoria (Cabodi, un cucciolo, ti mette stra a tuo agio)

1. Classe NP-C, proprietà e conseguenze
2. Cosa sono le combinazioni di n elementi a gruppi di n? Differenza tra complessità delle permutazioni e del powerset (cosa è meglio)
3. Grafo completo: definizione, numero di archi.
4. A partire da un grafo G e da un insieme di vertici T che formano un sottografo completo, come verificare se i vertici di T formano una cricca (sottografo completo massimale)

Programmazione (Pasini, decisamente meno cucciolo di quanto sembra)

1. Inversione di una lista (in loco)
2. Dato un grafo generico ed un vertice v, scrivere una funzione che ritorna il puntatore ad un vettore contenente i vertici raggiungibili da v in al massimo k passi (vuole una variante della BFS con complessità polinomiale e non una funzione ricorsiva con complessità esponenziale)

27 + 2 -> 30 (28/02/23)

Programmazione (Palena, non aiuta ma lascia il tempo di ragionare)

- Implementazione di un BST e verificare se due BST sono isomorfi (ti da lui la definizione);
- Generare tutte le sequenze di decimali lunghe k tali per cui il numero di cifre pari sia maggiore del numero di cifre dispari e dopo ogni cifra pari sia presente una cifra divisibile per tre (ti chiede il modello del calcolo combinatorio (disp rip) e di usare le condizioni per effettuare il pruning);

Teoria (Camurati, non mette nemmeno il silenzioso al telefono)

- definizione di DAG e topological sorting;
- cos'è il double hashing e perché si applica (mettere l'accento sull'1+);
- quando si verifica il caso peggiore del quick sort e come diventa l'equazione alle ricorrenze;

27 + 2->30L (26/01/21)

Programmazione

- Inversione iterativa ed in loco di lista semplicemente linkata
- Determinare il numero di cicli da un certo vertice passando al più k volte per ogni vertice

Teoria

- Liste implementate come vettore. Come funzionano e come tenere traccia delle caselle libere tramite “availability list” (non visto a lezione, mi ci ha fatto arrivare tramite ragionamento)
- Classi di algoritmi e macchina non deterministica di Turing.

27 -> 30L (26/01/2021)

Teoria(Camurati):

- Algoritmo di Kosaraju (ha chiesto a cosa serve e ho spiegato anche come funziona)
- Ordinamento topologico e tipologia archi nella dfs
- Cammini di Eulero

Programmazione (Patti):

- Data una lista ordinarla con un algoritmo di ordinamento a piacere (senza nessun vincolo in loco o meno)
- Ricavare la partizione per un grafo bipartito

27 -> 30L (26/01/2021)

Teoria (Camurati)

- Algoritmo di Kruskal, definizione di albero ricoprente minimi e teorema degli archi sicuri
- Perché UF serve in Kruskal
- Differenza tra Quick Union e weighted QuickUnion + dimostrazione complessità weighted Quick Union

Programmazione (Patti)

- Inserzione in lista ordinata e cancellazione (lista doppio linkata)
- Ricerca di tutti i cicli di lunghezza k a partire da un vertice dato(Grafo orientato).

27 -> 30L (16/02/2021)

Teoria (Cabodi):

- Cos’è il partition. Complessità. Complessità se il bst non è la versione estesa (non tiene conto del numero di nodi).
- Cos’è un bridge e come lo trovo. Complessità. Come trovare una coppia di archi che insieme sono un bridge (se li togli entrambi si sconnette il grafo)

Programmazione (Pasini):

- Inversione ricorsiva di una lista in loco
- in un grafo trovare il “vicinato” di un vertice: ovvero i nodi distanti al massimo k dal vertice di partenza

27 -> 30L (07/02/2023)

Teoria (Camurati):

- Varie implementazioni della code a priorità con relativa complessità per operazioni di inserzione, stampa del massimo ed estrazione del massimo. Costo della PQchange nell'heap.
-
- Condizioni necessarie per poter applicare la programmazione dinamica. Esempio di problema in cui NON si può applicare (calcolo dei cammini massimi in un grafo).Costo della partition nel quicksort e quanto è importante la scelta del pivot per l'algoritmo.

Programmazione (Pasini):

- È data una lista singolo-linkata, che potrebbe terminare con un puntatore a NULL, oppure potrebbe richiudersi su se stessa in un generico punto (es a->b->c->d->b->…).
  Si vuole scrivere una funzione che ritorni NULL se la lista finisce, o che ritorni il puntatore alla "giuntura" se c'è un ciclo (nell'esempio sopra deve ritornare un puntatore a b).
  Si richiede inoltre che la complessità in memoria sia O(1). (La chiede a tutti lol).
- Vicinato di un vertice in un grafo, ovvero dato un grafo g trovare tutti i nodi a distanza massima k da un vertice v ricevuto in input. Si risolve con una visita in ampiezza.

27 -> 30L (08/02/2022)

Teoria (Camurati):

- Codici prefissi, codici a lunghezza fissa e variabile: com'è Huffman? Che struttura dati usa Huffman?
- Algoritmo di Er
- Come implementare una lista senza poter usare puntatori. Come tenere traccia delle caselle libere tramite "availability list". Quante liste ci possono essere al massimo e al minimo in un vettore di n elementi (al massimo n liste da un elemento, al minimo due: una lista e la availability list vuota)

Programmazione (Pasini):

- È data una lista singolo-linkata, che potrebbe terminare con un puntatore a NULL, oppure potrebbe richiudersi su se stessa in un generico punto (es a->b->c->d->b->…).
  Si vuole scrivere una funzione che ritorni NULL se la lista finisce, o che ritorni il puntatore alla "giuntura" se c'è un ciclo (nell'esempio sopra deve ritornare un puntatore a b).
  Si richiede inoltre che la complessità in memoria sia O(1)
  (Dopo un'idea che mi sono reso conto essere fallimentare, Pasini mi ha fatto notare che la distanza di un nodo dal head della lista cresce sempre, tranne che nel momento in cui si chiude il loop. Ho avuto bisogno di un altro suggerimento per capire che dovessi ricalcolare da capo la distanza ogni volta)
- È data una matrice M RxC di interi. Si vuole trovare il peso del cammino massimo che porta da una cella della prima riga ad una cella della seconda riga, ma le uniche mosse consentite sono verso il

basso (giù, diagonale giù/dx, diagonale giù/sx)
Mi ha chiesto la complessità dell'algoritmo ricorsivo che ho fatto, poi ho detto che potevo fare la memoisation, e mi ha chiesto come. Ho proposto una matrice RxC, mi ha chiesto se fosse necessaria tutta una matrice (alla fine mi ha detto che bastavano due righe, si rilassa una riga e poi la si dimentica)

27->30L (18/02/2023)

Teoria (Camurati):

- Colorabilità di un grafo
- Pruning

Programmazione (Cabodi)

- In un grafo trovare se esiste un ciclo lungo k per cui $v_1 < v_2 < \ldots < v_{k-1}$ (ovviamente l'ultimo vertice non deve rispettare questa condizione perchè uguale al primo)

28->29(12/07/2022)

Teoria(Camurati):

- Classi di problemi (P, NP, NPC, NPH)
- Algoritmo generico, teorema, corollario su cui si basano gli algoritmi per MST
- Approcci negli algoritmi dei cammini minimi
- Perché Bellman-ford impiega V-1 passi + 1 di controllo?

Programmazione(Pasini):

- inversione ricorsiva lista in loco
- Dato un grafo e un indice di un suo vertice, determinare un vettore con l'elenco dei vertici che appartengono al suo "vicinato" dove per vicinato si intende che a distanza (in termini di archi attraversati) minore di un certo k passato alla funzione: Ho pensato di applicare una bfs terminando dopo k passi

28 -> 30L (16/02/2021)

Teoria (Cabodi)

- Algoritmo di Kruskal
- Proprieta' dei BST
- Funzioni aggiuntive per i BST se si aggiunge ad ogni nodo l'etichetta con il numero di nodi del suo codesottoalbero (partition ecc)
- Code a priorita' e in particolare il costo della PQchange se implemento la PQ con heap: qual e' il problema che non permette di avere un costo migliore (nella PQchange devi prima trovare l'elemento a

cui cambiare la priorita', quindi devi effettuare una ricerca lineare, nonostante poi in effetti il cambio di priorita' con conseguente heapify sia solo logaritmico)

Programmazione (Vendraminetto)

- Scrivere la struttura di un grafo non orientato e non pesato (ladj o madj a piacere) e implementare una funzione che, dato un nodo sorgente, trovi tutti i nodi che abbiano una distanza da quel nodo >=k con k passato come parametro:
    - risoluzione: la funzione data puo' essere una funzione wrapper; all'interno viene chiamata una BFS (modificata) a partire dal nodo sorgente.
    - Nella BFS ho usato un vettore *dist in cui segno la distanza di ogni vertice dal vertice di partenza (ovviamente e' inizializzato a zero); quando estraggo un arco dalla coda, cerco tutti i vertici adiacenti nella matrice (o lista) di adiacenze verifico che tale nodo non sia gia' stato esplorato (quindi devo avere dist[v] == 0), poi inserisco l'arco in coda e aggiorno il valore della sua distanza, che in particolare sara' data da dist[w] + 1, in cui w rappresenta il vertice da cui parte l'arco.
- Scrivere una possibile implementazione della strcmp (quindi prototipo int myStrcmp(char *s1, char *s2))
    - In questo caso io ho utilizzato puntatori e aritmetica dei puntatori per scorrere le due stringhe con un solo ciclo for; si puo' utilizzare chiaramente anche un normale ciclo for scandendo i vettori con un indice.

28 -> 29 (26/01/2021)

Teoria(Cabodi)

- Codici di Huffman, ragionamento su cosa cambia se le due stringhe sono isofrequenti, cosa invece non cambia mai. Ha effetto un ordinamento diverso dei codici 1 e 0 nell'albero?
- Complessità PQinsert per codici di Huffman2
- Ragionamento su ordinamento topologico inverso e possibili applicazioni

Programmazione(Pasini)

- Invertire una lista singolo linkata in loco, senza allocazioni nuove.
- Dato un grafo non orientato e non pesato g e un vertice v, memorizzare in un vettore il numero massimo di vertici raggiungibili da v in al massimo k passaggi.

28 -> 30

Programmazione:

- Implementazione di un ADT di I categoria "poligono" (deve memorizzare n vertici.Un vertice è definito da una struttura "punto"), di una lista in grado di memorizzare diversi poligoni letti da un file (una riga contiene il numero di lati e la seguente l'elenco di vertici). Mi ha fermato mentre stavo implementando

la lista per passare direttamente alla seconda domanda.
- Dato un albero binario e due suoi nodi a e b scrivere una funzione in grado determinare il primo (più vicino ad a e b) loro antenato comune.

Teoria:

- Generalizzazione dei BST ad "alberi di ricerca n-ari". Caso specifico di albero di ricerca quaternario: mi ha fatto inserire diversi nodi partendo da un albero vuoto, mi ha chiesto "cosa non mi piaceva" (l'albero non era bilanciato) e me l'ha fatto bilanciare.
- Discussione sulla convenienza di questa struttura dati (vale anche per iBST) in termine di efficienza: conviene nei casi statici (ci si può permettere il costo di bilanciare l'albero una volta dopo l'inserimento dei dati) e non conviene nei casi molto dinamici (andare a bilanciare continuamente l'albero ogni volta che ci sono variazioni nei dati ha un costo tale da vanificare il guadagno dovuto all'utilizzo dell'albero di ricerca).

28 + 2 -> 30 (26/01/2021)

Programmazione (Palena)[20/30 minuti a testa]:

- struttura di un albero left-child right-sibling + struttura nodo. Conta quanti nodi hanno un numero pari di figli in un albero left-child right-sibling partendo da una wrapper tree t
- struttura di un grafo con madj. Verifica che ci sia un ciclo, in questo grafo con madj, che parte da v fino a v, ritornando 1 se c'è o 0 se non c'è. Funzione di partenza: int circheck(Graph G,int v)

Teoria (Cabodi)[7 minuti a testa circa]:

- Huffman descrivere il problema NON come funziona. Perché mi è utile usare Huffman e non uso ASCII che è già tutto bello fatto?
    - Perché grazie al fatto che hai lunghezza variabile da un testo da 5 kbyte arrivi a 3 kbyte
- Cosa è un grafo completo, quanti archi ha, qual'è il costo per determinare se è completo o no?
    - per ladj: O(E) e per madj: O(V^2)

28 + 2-> 30L (26/01/2021)

Programmazione(Pasini) :

- cammini da sorgente distanti al più k
- Codici di Huffman implementati

Teoria (cabodi) :

- Hash table uniformità, chiave, come andare a limitare il più possibile le collisioni
- Cammini minimi multisorgente complessità (fuori programma)
- Bilanciamento bst funzionamento con Partition

28 + 2 -> 30L (26/01/2021)

Programmazione (Patti, un cucciolo):

- Ordinamento su lista singolo linkata (ho fatto insertion sort, ma non mi ha posto nessun limite; avrei potuto fare qualsiasi cosa anche non in loco)
- Trovare grafo bipartito (prima domanda teorica per verificare se sapessi cos'è un grafo bipartito; risolto con ricorsione e matematica combinatoria, non c'è bisogno di sapere teoremi o cose strane coi colori)

Teoria (Camurati):

Manipolazione algebrica di Karatsuba (da 4 sottoproblemi si passa a 3, non vuole sapere la formula)

- Shellsort
- Punto di articolazione
- Forse altro ma son disabile e non ricordo :(onesto socio bruh

28 -> 30 (27/06/2022)

Teoria (Camurati):

- Open addressing, vantaggi e svantaggi delle tre
- Cos'è un MST, quanti archi ha
- Qual è il caso peggiore nel quicksort ed equazione alle ricorrenze di questo caso

Programmazione (Pasini):

- Data una lista singolo linkata, scrivere una funzione che verifichi se una lista si ripiega su se stessa alla terminazione in un qualsiasi nodo precedente e ritorni il nodo in questione, altrimenti NULL
- Grafo orientato non pesato, scrivere una funzione che ritorni un vettore con l'elenco dei vertici distanti al massimo k dal vertice v (bfs)

29 -> 29

Teoria:

- PQ Change
- Heap
- Come abbattere la complessità della pq change
    - usare st con accesso indice chiave diretto

Programmazione:

- Punti distanze strutture e formula distanza origine
- Quicksort
- DFS che rileva cicli, un ciclo puó essere costituito da piú passaggi sullo stesso vertice (prima versione $O(n)^2$, poi mi ha chiesto di migliorarla)

29 -> 30 (26/01/2021)

Teoria(Cabodi) [5-10 minuti]:

- Classi P, NP, NP-C, NP-H
- esempio problema NP (camm. Hamilton)
- Punti Articolazione
- domande di ragionamento:
- "Se ho un grafo ciclico, come faccio a renderlo un DAG ?" (scelgo alcuni archi da togliere e verifico se è DAG)
- "se ho un elemento del powerset e una funzione di verifica che ha costo lineare, allora il problema si risolve con costo lineare?" (NO ! problemi NP hanno verifica lin. e ricerca esp.)

Programmazione (Palena) [un'ora dio c**e] :

- definizione successore BST e codice BSTsucc()
- definizione left-child right-sib, def "grado" in un albero n-ario
- funzione che calcola il num max di figli in un albero left-child...(consiglio : usate 2 funz ricorsive separate, TREEvisit e TREEcount)

29 + 1 punto lab -> 30 (03/03/2023)

Programmazione (Pasini) (30 min):

- Invertire una lista in loco, prima è stata chiesta iterativa e poi ricorsiva
- trovare tutte le possibili partizioni (se esistono) di un grafo bipartito tramite calcolo combinatorio e descrivere quale modello utilizzare (io ho utilizzato le disp_rip e mi ha detto che sarebbe andato bene anche se non avessi fatto il check dei blocchi con occorrenze uguali a zero)

Teoria (Camurati) (10 min):

- Grafo bipartito, grafo completo (matematica discreta)
- Teorema e corollario dei tagli e degli archi sicuri
- Punti di articolazione e la sua definizione

29 + 2 -> 30L

Programmazione:

- Dato un grafo orientato, determinare la centralità di ogni vertice: la centralità di un vertice è il numero di volte con cui esso compare nei cammini semplici di un grafo (fra tutti i vertici).

- Implementazione algoritmo di Kruskal e Union-Find.

Teoria:

- Equazione alle ricorrenze della heapify. T(n) = T(n/2) +1

29 -> 30L (26/01/2021)

Teoria:

- Notazione Θ(theta grande)
- Equazione ricorrenze Fibonacci (senza unfolding, solo scriverla e commentarla, spiegare anche la "sovrastima" per semplicità di calcoli)

Programmazione:

- Inversione lista semplice in loco (non puoi usare altra lista, non puoi allocare altri nodi)
- Dato un grafo generico, un intero k e un vertice v, scrivere una funzione che ritorni un vettore contenente i vertici raggiungibili in al massimo k passi a partire da v

29 -> 30L (26/01/2021)

Programmazione (Pasini)

- salvare l'intersezione di due liste ordinate in una terza lista
- Dato un grafo, trovare ricorsivamente la partizione di cardinalità massima che renda il grafo bipartito

Teoria (Camurati)

- PQchange -> costo con tutte le possibili implementazioni, come si potrebbe migliorare il costo nel caso di una coda a priorità implementata con heap

29 + 2 -> 30L (26/01/2021)

Teoria (Camurati):

- Cammino di Hamilton, ciclo di Hamilton, come risolvere il problema del commesso viaggiatore
- In cosa consistono il backtrack e il pruning
- Programmazione dinamica applicata all'algoritmo di Bellman Ford
- Perchè un cammino minimo è semplice

Programmazione (Palena):

- Dato un BST e una chiave, cercare, se esiste, il successore della chiave nel BST
- Dato un grafo non orientato e gli id di due vertici, trovare un cammino minimo da id1 a id2

(Per entrambi gli esercizi era richiesto di definire le strutture utilizzate come ADT di I categoria)

29 -> 30L (26/01/2021)

Programmazione (Patti)

- Implementazione dei codici di Huffman
- Dato un grafo determinare il numero di cammini ottenibili da un vertice di partenza; si potevano effettuare k ripassaggi su un nodo (ogni volta che un nodo era già stato visitato k decrementava ed era una variabile praticamente globale)

Teoria (Camurati)

- Macchina di Turing, si è concentrato in particolare sul concetto di stato (voleva sapere che era funzione del tempo)
- Descrizione della programmazione dinamica

30 -> 30L (26/01/2021)

Teoria:

- Travelling Salesman (commesso viaggiatore)
- Ciclo di Hamilton con implementazione di calcolo combinatorio
- In un albero binario creare un vettore di liste, ogni lista deve contenere un percorso radice -> foglia
-
- Classi di problemi
- Macchina di Turing

Programmazione:

- Inversione di lista in loco senza allocazione di nuova memoria
- Per ogni nodo di un albero Left-Child Right-Sibling aggiungere al nodo il numero di figli

29->30L (23/02/2022)

Teoria (Camurati):

- Programmazione dinamica applicata a Bellman-Ford: perchè la si può applicare e dove si vede
- Equazione alle ricorrenze disposizioni semplici
- Algoritmo di Er: a cosa serve, quanti rami ricorsivi ha e cosa fanno

Programmazione (Pasini):

- Data una lista singolo linkata, determinare se ad un certo punto il nodo finale di questa invece di terminare a NULL punta ad un nodo della lista (se fa un ciclo in pratica) NB:inizialmente avevo pensato di porre banalmente un flag su ogni nodo e verificare per ogni nodo se veniva visitato più di una volta

ma Pasini ha detto che così avrebbe funzionato ma che voleva un altro approccio ->riadattamento di Bellman Ford

- Dato un grafo e un indice di un suo vertice, determinare un vettore con l'elenco dei vertici che appartengono al suo "vicinato" dove per vicinato si intende che a distanza (in termini di archi attraversati) minore di un certo k passato alla funzione: Ho pensato di applicare una bfs terminando dopo k passi
-

29-> 30L (01/03/2023)

Teoria (Cabodi)

- Come funziona Heapsort, complessità Heapbuild,Heapify.
- Ordinamento topologico normale ed inverso
- In un ordinamento topologico posso avere nodi sink prima di nodi source? (Sì se il DAG è sconnesso)
- Come posso verificare che un grafo sia completo?
    - Matrice adiacenze, deve essere piena tranne che sulla diagonale.
    - Lista adiacenze, V-1 nodi per lista, ma devo assicurarmi che non ci siano cappi o multigrafi

Programmazione (Pasini) Esercizi già segnati

SENZA VOTO

Teoria:

- Cammino semplice
    - Cammino che non passa mai dallo stesso vertice
- Programmazione dinamica
    - Applicata a problemi di ottimizzazione
    - Procedimento:
        - Caratterizzazione della struttura di una soluzione ottima
        - Definizione ricorsiva del valore di una soluzione ottima
        - Calcolo bottom-up del valore di una soluzione ottima
        - Costruzione di una soluzione ottima
- Differenza tra programmazione dinamica e memoization
    - Programmazione dinamica è bottom up mentre memoization è top down
- Algoritmi di ordinamento ricorsivi basati sul confronto; quali di questi sono in loco?
    - Merge sort, non in loco
    - Quick sort, in loco

- Possibili implementazioni di coda a priorità; è possibile implementarla anche con vettore non ordinato?
    - Si, nel vettore si mantiene aggiornato l'indice a cui inserire/estrarre il prossimo elemento (priorità massima)
- Delete in una tabella di hash
    - Operazione complessa che interrompe le catene di collisione. L'open addressing è in pratica utilizzato solo quando non si deve mai cancellare
    - Soluzioni:
        - sostituire la chiave cancellata con una chiave sentinella che conta come piena in ricerca e vuota in inserzione
        - reinserire le chiavi del cluster sottostante la chiave cancellata

Programmazione:

- Vettore di interi: creare 2 liste, una con gli elementi pari e una con gli elementi dispari, mantenendo le liste ordinate
- Dato un vettore con le cifre da 0 a 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

voto?????

Teoria (Camurati)

- Dijkstra (paradigma, strutture dati, condizioni di applicabilità)
- Bellman-Ford (perché si può usare per trovare i cammini minimi e dimostrazione della struttura ottima dei sotto - problemi)
- Coda a priorità (definizione concettuale, possibili implementazioni)

voto?? (26/01/2021) "ipotizzo un 17 18 considerando le domande potrebbe essere anche 16"

Teoria (Camurati)

- Shell sort
- Proprietà di Bellman-Ford
- Quick Union Weighted e perché migliora la quick union

Programmazione (Pasini)

Implementare una lista a piacimento e eliminare tutti i nodi con un certo valore

- Dato un grafo non orientato scrivere una funzione che restituisce il numero di cicli a partire da un vertice v e sia k il numero massimo di volte che i vertici già visitati possono essere percorsi

voto 18-> 21

Programmazione(Palena, tanto buono e caro: ti da tutto il tempo che vuoi e se non riesci in un punto ti aiuta facendoti arrivare all'imprecisione senza farti pesare l'errore)

- Heapify
- Cos'è una PQ e come la implementi (ti fa parlare liberamente, puoi iniziare il discorso come vuoi)
- Implementazione dello Stack come ADT di 1° classe e funzione Stackpush

(Al mio collega ha chiesto: implementazione lista come ADT 1° classe -vuole implementazione tipo LIST e del nodo-, definire un BST -anche qui vuole la struttura dati- e funzione che ti ritorna il numero dei nodi di un BST)

Teoria (Cabodi, è tosto: se studi ma ragioni poco e commetti erroretti si innervosisce parecchio)

- Rappresentazione left child-right sibling e differenza con BST (spoiler: nel nodo è presente un puntatore al primo figlio di sinistra e un puntatore al fratello di destra. Se necessario si può inserire anche un puntatore al padre).
- Equazione alle ricorrenze MergeSort, spiegare cosa significa il +N e qual è il costo della divisione (O(1), se la sbagli si arrabbia)
- QuickSort, equazione alle ricorrenze e cosa significa il +N
- Cosa serve Kosaraju (componenti fortemente connesse)
- Per le componenti connesse invece cosa faresti e con quale costo? Risposta: visita in profondità e O(N) dove N è il numero degli archi

Al mio collega ha chiesto: Come funziona la programmazione dinamica e dove si applica e differenza con Memoization, Ricerca minimo in un BST,che cos'è e come trovare il Successore, rappresentazioni e differenze di un grafo

Voto 19 -> 21 (28/02/23)

Programmazione (Palena)

- Realizza un ADT LIST e una funzione che trova quando una lista si avvolge su se stessa (quando un nodo ha come suo successore un nodo che si trova prima di lui nella lista)
- Realizza un ADT symboltable e una funzione di insert con open addressing e double hashing

Teoria (Camurati)

- Tabelle ad accesso diretto e tabella di hash e loro differenze
- L'heap in generale

17 -> ?? (25/09/2023)

Camurati Teoria

- - Componenti fortemente connesse e kernel, componente massimale
- - BST, rango, ricerca in order, complessità della inorder

Pasini Programmazione

- PQ come lista. Definire solo il nodo e la funzione insert
- Quanti cammini semplici a distanza k partendo da un nodo definito v