UNIVERSITÀ DEGLI STUDI DI PALERMO

DIPARTIMENTO	Ingegneria
ANNO ACCADEMICO OFFERTA	2020/2021
ANNO ACCADEMICO EROGAZIONE	2021/2022
CORSO DILAUREA	INGEGNERIA INFORMATICA
INSEGNAMENTO	ALGORITMI E STRUTTURE DATI
TIPO DI ATTIVITA'	B
AMBITO	50289-Ingegneria informatica
CODICE INSEGNAMENTO	01175
SETTORI SCIENTIFICO-DISCIPLINARI	ING-INF/05
DOCENTE RESPONSABILE	LO PRESTI LILIANA Professore Associato Univ. di PALERMO
ALTRI DOCENTI	
CFU	9
NUMERO DI ORE RISERVATE ALLO STUDIO PERSONALE	144
NUMERO DI ORE RISERVATE ALLA DIDATTICA ASSISTITA	81
PROPEDEUTICITA'	
MUTUAZIONI	
ANNO DI CORSO	2
PERIODO DELLE LEZIONI	1° semestre
MODALITA' DI FREQUENZA	Facoltativa
TIPO DI VALUTAZIONE	Voto in trentesimi
ORARIO DI RICEVIMENTO DEGLI STUDENTI	**LO PRESTI LILIANA** Martedì 16:00 17:00

DOCENTE: Prof.ssa LILIANA LO PRESTI

PREREQUISITI	Buona conoscenza del linguaggio C
RISULTATI DI APPRENDIMENTO ATTESI	- Conoscenza e capacita' di comprensione Al termine del corso, lo studente avra' conoscenza delle problematiche inerenti gli algoritmi e le strutture dati. In particolare, lo studente conoscera' nel dettaglio le metodologie di analisi degli algoritmi, le strutture dati elementari, gli algoritmi di ricerca e di ordinamento e conoscenze di base relative ai grafi e ai principali algoritmi sui grafi. Per il raggiungimento di questo obiettivo il corso comprende: lezioni frontali; analisi e discussione di frammenti di programmi. - Capacita' di applicare conoscenza e comprensione Lo studente sara' in grado di valutare le caratteristiche, i vantaggi e le limitazioni dei principali algoritmi e strutture dati. Sara' in grado di progettare, analizzare e valutare le soluzioni software a problemi di media complessita. Sara' anche in grado di sviluppare nuove soluzioni software, valutandone la qualita' in termini di semplicita, efficacia ed efficienza. Per il raggiungimento di questo obiettivo il corso comprende esercitazioni da svolgere in aula relative all'analisi di frammenti di codice/pseudo-codice e alla progettazione di algoritmi efficienti. Per la verifica di questo obiettivo l'esame comprendera' l'analisi di algoritmi gia' esistenti, l'applicazione di tecniche note e la progettazione di un algoritmo a partire dalla descrizione testuale del problema da risolvere. - Autonomia di giudizio Lo studente sara' in grado sia di effettuare l'analisi di un problema che di progettare, a partire da precise specifiche, una opportuna soluzione software. Sara' in grado di valutarne la qualita' di una soluzione software in termini di semplicita, leggibilita, efficienza e possibilita' di riutilizzo. Per il raggiungimento di questo obiettivo il corso comprende: analisi e discussioni di frammenti di codice/pseudo-codice; lezioni relative alle tecniche di analisi degli algoritmi; discussioni su possibili vantaggi e svantaggi derivanti dall'utilizzo di particolari strutture dati. Lo studente verra' incoraggiato inizialmente a trovare e valutare autonomamente soluzioni ai problemi posti, al fine di potere comprendere la qualita' e l'utilita' delle soluzioni proposte durante il corso. Per la verifica di questo obiettivo l'esame comprendera' la progettazione di un algoritmo. Lo studente dovra' effettuare delle scelte progettuali in autonomia quali, per esempio, la scelta della struttura dati idonea all'implementazione di una soluzione efficiente. - Abilita' comunicative Lo studente acquisira' la capacita' di comunicare ed esprimere problematiche inerenti l'oggetto del corso. Sara' in grado di sostenere conversazioni su tematiche relative alla implementazione software di algoritmi e strutture dati efficienti. Sara' in grado di utilizzare un linguaggio semplice e chiaro per la descrizione dei processi di analisi e di sintesi di soluzioni software a problemi di media complessita'. Per il raggiungimento di questo obiettivo il corso comprendera' esercitazioni in aula durante le quali gli studenti proporranno soluzioni agli esercizi proposti e discuteranno le eventuali difficolta' riscontrate. Per la verifica di questo obiettivo l'esame comprendera' la discussione orale sugli argomenti trattati nel corso. - Capacita' d'apprendimento Lo studente dovra' sviluppare la capacita' di apprendere i processi di analisi e di sintesi relativi alla codifica di algoritmi di media complessita' e alla relativa implementazione di librerie e strumenti software. Per il raggiungimento di questo obiettivo il corso comprende: esercitazioni da svolgere autonomamente; discussione delle eventuali difficolta' riscontrate. Per la verifica di questo obiettivo l'esame comprendera' la discussione su alcuni argomenti introdotti a lezione e il cui approfondimento e' lasciato agli studenti.
VALUTAZIONE DELL'APPRENDIMENTO	Sono previste due prove: una prova scritta ed una orale. La prova scritta richiede lo svolgimento di esercizi atti ad accertare la capacita' del candidato di analizzare il costo computazione di un algoritmo, comprendere il funzionamento di un algoritmo, progettare un algoritmo utilizzando le tecniche di programmazione apprese durante il corso, applicare gli algoritmi studiati durante il corso. I candidati che raggiungono la sufficienza (18/30) nella prova scritta sono ammessi a sostenere la prova orale. La prova orale consiste della discussione della prova scritta e di domande atte ad accertare la conoscenza del candidato degli argomenti trattati nel corso. Il voto finale in trentesimi, nell'intervallo 18/30-30/30 con Lode, e' ottenuto come media della valutazione complessiva della prova scritta e di quella orale.

	In accordo con i descrittori di Dublino, la formulazione delle prove permette una valutazione dei risultati attesi in relazione al voto finale come segue: - da 18/30 a 20/30: mediocre o sufficiente conoscenza e capacita' di comprensione degli argomenti trattati, parziale capacita' di applicazione delle conoscenze acquisite per la risoluzione dei problemi proposti; parziale autonomia di giudizio, abilita' comunicative e capacita' di apprendere. - da 21/30 a 23/30: sufficiente o discreta conoscenza e capacita' di comprensione degli argomenti trattati, sufficiente capacita' di applicazione delle conoscenze acquisite per la risoluzione dei problemi proposti, sufficiente autonomia di giudizio, abilita' comunicative e capacita' di apprendere. - da 24/30 a 26/30: discreta conoscenza e capacita' di comprensione degli argomenti trattati, discreta capacita' di applicazione delle conoscenze acquisite per la risoluzione dei problemi proposti, sufficiente autonomia di giudizio, abilita' comunicative e capacita' di apprendere. - da 27/30 a 30/30 e lode: buona o eccellente conoscenza e capacita' di comprensione degli argomenti trattati, buona o eccellente capacita' di applicazione delle conoscenze acquisite per la risoluzione dei problemi proposti, buona o eccellente autonomia di giudizio, abilita' comunicative e capacita' di apprendere Requisito minimo per il superamento dell'esame e' la dimostrata conoscenza, nelle due prove, delle nozioni di base relative a: analisi e comprensione di algoritmi, calcolo della complessita' di un algoritmo, gestione di strutture dati elementari (array, pile, code, alberi e tabelle hash) e di strutture complesse (code con priorita, union-find e grafi), algoritmi di ordinamento e di ricerca.
OBIETTIVI FORMATIVI	Il corso trattera' in maniera approfondita lo studio degli algoritmi e delle strutture dati. Lo studente acquisira' una buona conoscenza degli aspetti relativi l'analisi dell'efficienza di un algoritmo, specifici algoritmi di ricerca e di ordinamento dei dati, l'implementazione di strutture dati quali pile, code, alberi, tabelle hash e code con priorita. Il corso puntera' ad evidenziare i vantaggi e gli svantaggi delle tecniche presentate in base al problema che si intende risolvere. Il corso mira ad introdurre i concetti relativi a particolari tecniche di programmazione quali divide et impera, programmazione dinamica e strategie greedy. Infine, verranno trattate problematiche a algoritmi relativi ai grafi e i concetti di base dell'ottimizzazione numerica.
ORGANIZZAZIONE DELLA DIDATTICA	Lezioni frontali ed esercitazioni
TESTI CONSIGLIATI	C. Demetrescu, I. Finocchi, G. F. Italiano (2008). Algoritmi e strutture dati - seconda edizione. McGraw-Hill T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, L. Colussi (2010). Introduzione agli algoritmi e strutture dati - terza edizione. McGraw-Hill

PROGRAMMA

ORE	Lezioni
5	Introduzione agli algoritmi. Algoritmi iterativi e ricorsivi. Occupazione di memoria ed efficienza di un algoritmo. Analisi di algoritmi. La notazione asintotica O-grande, Omega-grande, Theta-grande. Analisi del caso peggiore, migliore e medio. Analisi di algoritmi ricorsivi.
6	Definizione di struttura dati; operazioni tipiche per la gestione di strutture dati (inserimento, cancellazione e ricerca); rappresentazioni indicizzate e collegate; array e loro ridimensionamento; strutture collegate: record e puntatori; lista semplice, lista doppiamente concatenata, lista circolare; pile e code; alberi: definizioni di base; tecniche di rappresentazione di alberi in memoria; algoritmi di visita di alberi e loro complessita' computazionale.
8	Il problema dell'ordinamento di n oggetti; albero delle decisioni e altezza in funzione delle foglie; dimostrazione del lower-bound degli algoritmi di ordinamento nel caso peggiore; algoritmi di ordinamento con upper-bound quadratico: insertionSort, selectionSort, bubbleSort; algoritmi di ordinamento in tempo n log n: heapSort, mergeSort e quickSort; definizione di heap e sue proprieta; ordinamento in loco; randomizzazione; ordinamento di interi e oggetti con chiave intera: integerSort, bucketSort, radixSort.
3	Il problema della selezione, calcolo del minimo e del secondo minimo; albero del torneo e dimostrazione del costo computazionale per la ricerca del secondo minimo; heapSelect; quickSelect; mediano dei mediani.
5	Alberi binari di ricerca e operazioni di ricerca, inserimento e cancellazione di elementi; alberi AVL e fattore di bilanciamento, alberi di Fibonacci, inserimento e cancellazione di elementi in alberi AVL, operazioni di rotazione SS, DS, SD, DD; analisi della complessita' degli algoritmi; alberi Red-Black
2	Tabelle ad accesso diretto e fattore di carico; tabelle hash; definizione di collisione; funzioni hash perfette e non perfette; uniformita' delle funzioni hash; costruzione di funzioni hash: metodo della divisione, metodo del ripiegamento, metodo della moltiplicazione, approccio polinomiale; risoluzione delle collisioni: liste di collisione e indirizzamento aperto, metodo di scansione lineare e scansione quadratica, hashing doppio; analisi dei costi per le tabelle hash.
4	Code con priorita' e operazioni di inserimento e cancellazione, d-heap e sue caratteristiche; alberi binomiali e loro proprieta; heap binomiali; ristrutturazione di un heap binomiale; analisi del costo delle operazioni su heap binomiali.
2	Definizione di union-find e operazioni di interesse; quickFind e quickUnion; euristiche di bilanciamento delle operazioni union; euristiche di bilanciamento per il quickUnion; union by rank e union by size; euristiche di compressione nell'operazione find; analisi dei costi computazionali per le union-find con e senza euristiche.

PROGRAMMA

ORE	Lezioni
4	Tecniche algoritmiche: divide et impera; programmazione dinamica; strategie greedy; applicazioni di programmazione dinamica: sotto-vettore di massimo valore; cammini di valore minimo; distanza tra stringhe (edit distance); massima sotto-sequenza comune.
12	Grafi e rappresentazione di grafi in memoria: lista di archi, liste di adiacenza, liste di incidenza, matrici di adiacenza, matrici di incidenza; algoritmi di visita dei grafi; grafi connessi e calcolo delle componenti connesse di un grafo non orientato; connettivita' forte in grafi orientati e calcolo delle componenti fortemente connesse in grafi orientati; definizione di minimo albero ricoprente; algoritmi di Prim e Kruskal; definizione di cammino minimo; distanza tra nodi; algoritmo di Bellman-Ford e algoritmo di Dijkstra.
4	Teoria della NP-completezza: cenni sulle macchine non deterministiche; classi di complessita: problemi in P, NP, NP-C e NP- H. Esempi.
4	Introduzione all'ottimizzazione numerica. Introduzione a programmazione convessa e metodi di ottimizzazione numerica (interior-point method). Introduzione all'algoritmo di discesa lungo il gradiente.

ORE	Esercitazioni
4	Stima del costo computazionale e dell'occupazione di memoria di algoritmi. Individuazione della classe di complessita' di un algoritmo.
4	Funzionamento di strutture dati: alberi binari di ricerca; tabelle hash, alberi AVL, code con priorita.
8	Progettazione di algoritmi per la risoluzione di problemi di media complessita.
6	Applicazioni inerenti i grafi.