IN1 Informatica 1 (Fondamenti)

A.A. 2005/2006

Prof. Marco Liverani

Programmazione, algoritmi e strutture dati

1. Problemi ed algoritmi.

Introduzione alle caratteristiche del calcolatore ed al rapporto programmatore/esecutore; compiti ed abilità del programmatore; principali caratteristiche ed abilità dell'esecutore, operazioni di base (logiche, aritmetiche e di confronto).

Modelli di macchina calcolatrice: la macchina RAM, cenni sulla macchina di Turing.

Linguaggi di programmazione: linguaggi imperativi e dichiarativi. Istruzioni fondamentali di un linguaggio di programmazione procedurale generico. Algoritmi e programmi; diagrammi di flusso. Regole della programmazione strutturata, teorema di Jacopini-Böhm; approccio *top–down* alla soluzione di un problema.

2. Il linguaggio C.

Organizzazione della memoria di un calcolatore, indirizzi, parole, puntatori. Codifica binaria. Tipi di dato, strutture dati (*array*, matrici, pile, code, code di priorità, liste, alberi, grafi).

Linguaggio macchina, linguaggi di alto livello; compilatori ed interpreti. Il linguaggio C: scopi e principali caratteristiche.

La struttura di un programma C, l'inclusione degli *header*, dichiarazione delle variabili; le librerie.

Tipi di dato elementari in linguaggio C: interi, *floating point*, *double*, *char*. Puntatori; aritmetica sui puntatori. *Array* e matrici e loro rappresentazione in memoria. Strutture dati complesse: liste, alberi, grafi; l'istruzione "`struct`".

Operatore di assegnazione, operatori aritmetici in C in forma estesa e compatta. Strutture di controllo: "`if` ... `else` ...", "`while` ...", "`do` ... `while`", "`for` ...".

Funzioni: funzioni di libreria e funzioni definite dall'utente. Passaggio di parametri per valore e per indirizzo alle funzioni. Funzioni ricorsive.

Funzioni di input/output: "`printf`", "`scanf`"; funzioni per la gestione della memoria: "`malloc`", "`free`", "`sizeof`"; gestione di liste di record collegati tramite puntatori.

3. Algoritmi di ordinamento

Algoritmi di ordinamento elementari: *Insertion sort*, *Selection sort*, *Bubble sort*; l'approccio "divide et impera", l'algoritmo *Quick sort*. Strutture di tipo LIFO (*Last In First Out*), le pile; strutture di tipo FIFO (*First In First Out*), le code; code di priorità, gli *heap*. Algoritmi ottimi per l'ordinamento: *Heap sort*, *Merge sort*. Complessità

di un algoritmo, la notazione "O grande", analisi della complessità degli algoritmi di ordinamento.

4. Algoritmi elementari sui grafi

Definizioni principali: grafo, grafo orientato, grafo connesso; sottografo, sottografo indotto; passeggiata, cammino, ciclo, grafo aciclico; alberi, foreste, *spanning tree* di un grafo. Strutture dati per la rappresentazione di grafi mediante un calcolatore: liste di adiacenza e matrici di adiacenza. Algoritmi di visita di un grafo: visita in ampiezza (BFS), visita in profondità (DFS), ordinamento topologico di un grafo. Problemi di cammino minimo su un grafo, l'algoritmo di Dijkstra. Il problema del minimo albero ricoprente: l'algoritmo di Kruskal. Alberi binari di ricerca: definizione e algoritmi per l'ordinamento degli elementi, la ricerca di un elemento, la ricerca del massimo e del minimo, l'inserimento e la cancellazione di un vertice dell'albero.

Analisi della complessità degli algoritmi presentati. Cenni sulle classi dei problemi P, NP, NP-completi. Il problema "P=NP".

Testi consigliati

[1] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduzione agli algoritmi e strutture dati*. McGraw–Hill, (2005).
[2] H. Schildt, *Linguaggio C - La guida completa*. McGraw–Hill, (1995).
[3] M. Liverani, *Programmare in C*. Esculapio, (2001).

Bibliografia supplementare

[4] A.V. Aho, J.E. Hopcroft, J.D. Ullman, *Data structures and algorithms*. Addison-Wesley, (1987).
[5] C. Demetrescu, I. Finocchi, G.F. Italiano, *Algoritmi e strutture dati*. McGraw–Hill, (2004).
[6] B.W. Kernighan, D.M. Ritchie, *Linguaggio C*. Gruppo Editoriale Jackson, (1985).
[7] M. Liverani, *Qual è il problema? Metodi, strategie risolutive, algoritmi*. Mimesis, (2005).

Modalità d'esame

- valutazione in itinere ("esoneri")		■ SI	□ NO
- esame finale	scritto	■ SI	□ NO
	orale	■ SI	□ NO
- altre prove di valutazione del profitto (meglio descritte sotto)		□ SI	■ NO

L'esame si articola su due fasi successive: una prova scritta e una prova orale.

Per accedere alla prova orale è necessario superare con un voto sufficiente la prova scritta, o le due prove di esonero previste. La prova scritta consiste nella soluzione di due esercizi: viene richiesto di presentare una codifica in linguaggio C di un algoritmo che risolva il problema dato. La prova orale consiste nella discussione di alcuni tra i principali argomenti trattati durante il corso (algoritmi di ordinamento, algoritmi di visita di grafi, strategia risolutiva, analisi della complessità, esempi, diagrammi di flusso; aspetti importanti del linguaggio C: allocazione della memoria, puntatori, ecc.).