Diario delle lezioni di Algoritmi e Strutture Dati (modulo I), a.a. 2013/14.

1. (7/10/13). Introduzione al corso.

2. (9/10/13). Il problema del calcolo dell'*n*-esimo numero di Fibonacci. Un algoritmo numerico e un algoritmo ricorsivo. Analisi della complessità temporale dell'algoritmo ricorsivo. Un algoritmo iterativo di complessità temporale O(*n*) e di complessità spaziale O(*n*) (Fibonacci3). Portare la memoria a O(1): Fibonacci4. Introduzione informale alla notazione asintotica. Algoritmo con complessità O(log *n*) per il calcolo dell'*n*-simo numero di Fibonacci.

3. (14/10/13) Discussione della complessità spaziale degli algoritmi ricorsivi Fibonacci2 e Fibonacci6. Modello di calcolo RAM. Costi uniformi e logaritmici. Complessità caso peggiore, migliore, medio. Notazioni asintotiche: O-grande, Omega-grande, Theta. O-piccolo, Omega-piccolo. Definizioni e semplici esempi. Proprietà. Usare la notazione asintotica nelle analisi della complessità computazionale degli algoritmi.

4. (16/10/13) Analisi della complessità nel caso medio: un esempio. Il problema della ricerca di un elemento in un insieme: ricerca sequenziale e ricerca binaria. Equazioni di ricorrenza. Metodo dell'iterazione. Metodo della sostituzione. Discussione sull'importanza di lavorare sui Problem Set.

5. (21/10/13) Teorema Fondamentale delle Ricorrenze (Master). Semplici esempi. Quando non si può applicare. Metodo del cambiamento di variabile. Metodo che usa l'albero della ricorsione.

6. (23/10/13) Equazioni di ricorrenza: uno scenario meno comune. Picture-Hanging Puzzles, ovvero come appendere un quadro in modo perverso arrotolando un corda intorno a dei chiodi in modo tale che, rimuovendo uno qualsiasi dei chiodi, il quadro cada. Soluzione per due chiodi. Soluzione ricorsiva per *n* chiodi che usa corda esponenzialmente lunga. E soluzione che usa corda di lunghezza polinomiale.

7. (28/10/13). Problema dell'ordinamento. Selection Sort. Insertion Sort: due varianti. Algoritmo di ordinamento MergeSort.

8. (30/10/13). Esercitazione. Esercizio: mettere in ordine le funzioni per velocità crescenti. Esercizio: dimostrare o confutare una relazione asintotica (Es. 1). Esercizio di progettazione di un algoritmo che, dato un vettore ordinato A di *n* interi distinti e un valore *x*, trovare (se esistono) due elementi di A che sommano a *x*. Soluzione banale con complessità quadratica, soluzione di complessità O(n log n) e soluzione con tempo O(n) (Es. 2).

9. (04/11/13). Algoritmo di ordinamento QuickSort: analisi del caso peggiore, migliore, e intuizioni sul caso medio. Discussione versione randomizzata del QuickSort. Correzione esercizi 3 e 4 del primo problem set.
10. (06/11/13). Algoritmo di ordinamento HeapSort.

11. (11/11/13). Delimitazioni superiori e inferiori di algoritmi e problemi. Un lower bound alla complessità temporale necessaria per ordinare n elementi (con una classe di algoritmi ragionevoli, quelli basati su confronti). Algoritmi veloci per ordinare interi: IntegerSort, BucketSort, RadixSort.

12. (13/11/13). Esercitazione. Primo esercizio: dato un array di n interi compresi fra 1 e k, costruire in tempo O(n+k) un *oracolo* (struttura dati) che sia in grado di rispondere in tempo costante a domande del tipo "quanti interi nell'array sono compresi fra a e b?"(Esercizio e soluzioni a fine delle slide sull'IntegerSort). Secondo esercizio: dati due insiemi di n elementi forniti come vettori ordinati, progettare un algoritmo che in tempo O(n) e memoria costante trova il mediano dell'unione dei due insiemi (Es. 3).

13. (18/11/13). Strutture dati elementari: rappresentazioni indicizzate e rappresentazioni collegate. Implementazione di un dizionario con array ordinato/non ordinato e lista ordinata/non ordinata. Rappresentazioni di alberi. Algoritmi di visita di un albero: profondità versione iterativa, profondità versione ricorsiva (preordine, postordine, ordine simmetrico), ampiezza. Algoritmo per calcolare l'altezza di un albero.

14. (20/11/13). Esercitazione sulle visite di alberi. Ricostruire un albero, dati gli ordini di visita simmetrica e in preordine dei nodi (Problema 3.7 del libro di testo). Dimostrazione che la sequenze di visita in preordine più quella in postordine non sono sufficienti in generale per ricostruire l'albero. Progettazione di un algoritmo che, preso un albero con chiavi e colori (rosso e nero), trova il valore del cammino rosso di tipo nodo-radice di valore massimo (Es 4).

15. (25/11/13). Alberi binari di ricerca. Definizione. Visita in ordine simmetrico di un BST. Ricerca, inserimento, cancellazione (ricerca del massimo, del minimo, del predecessore e del successore di un nodo). Correzione esercizio relativo alla possibilità di usare un vettore posizionale anche per alberi non completi (vedere slide cap. 3).

16. (27/11/13). Alberi AVL: definizione ed esempi. Alberi di Fibonacci. Dimostrazione della delimitazione superiore dell'altezza di un albero AVL. Operazioni sugli alberi AVL: search, insert, delete.

17. (02/12/13). Esercitazione. Dato un array di *n* valori reali. Trovare la coppia di indici *i* e *j* con $i<j$ che massimizza A[j]-A[i] (Es. 5). Esercizio su albero AVL e "distanza" fra chiavi (Es. 6).

18. (04/12/13). Correzione Problem Set 2.

19. (11/12/13). Tecnica della programmazione dinamica. Un primo problema per vederla all'opera: calcolare un insieme indipendente di peso massimo in un cammino. Principi generali della programmazione dinamica.

20. (16/12/13). Ancora sulla tecnica della programmazione dinamica. Calcolo della distanza fra due parole. Esercizio: calcolo della sottosequenza crescente più lunga di un vettore (Es. 7).

21. (18/12/13). Discussione soluzione dell'esercizio sul calcolo della sottosequenza crescente più lunga di un vettore. Insieme indipendente di peso massimo di un grafo: un algoritmo polinomiale per trovare un tale insieme indipendente vale un milione di dollari (e la gloria eterna fra gli scienziati). Un algoritmo di programmazione dinamica per calcolare un insieme indipendente di peso massimo di un albero (Es. 8). Esercizio: il Signor Marche va in vacanza fra Roma e Firenze: aiutatelo a spendere il meno possibile (Es. 9).

22. (13/01/14). Code con priorità. d-Heap, Heap Binomiali, (cenni sugli) Heap di Fibonacci.

23. (15/01/14). Correzione Problem Set 3.

24. (20/01/14). Esercitazione sulla programmazione dinamica. Aiutate il Re Imprenditore (Es. 10). Aiutate Homer a mangiare più donut possibile (Es. 11). Dare il resto usando il minimo numero di monete. Il problema del distributore automatico: minimizzare il numero di monete nel dare il resto (Es. 12). Algoritmo greedy (goloso): non sempre funziona. Algoritmo di programmazione dinamica che risolve il problema.