Diario delle lezioni di Algoritmi e Strutture Dati (modulo I), a.a. 2016/17.

1. (3/10/16). Introduzione al corso. Motivazioni e concetti fondamentali. Un primo esempio: il problema di trovare una moneta falsa (più pesante) fra n monete usando una bilancia a due piatti.

2. (05/10/16). 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. Discussione della complessità spaziale degli algoritmi ricorsivi Fibonacci2 e Fibonacci6.

3. (12/10/16) 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. (17/10/16) 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 che usa l'albero della ricorsione.

5. (19/10/16) Ancora sulle equazioni di ricorrenza. Metodo della sostituzione. Teorema Fondamentale delle Ricorrenze (Master). Semplici esempi. Quando non si può applicare. Metodo del cambiamento di variabile. (introdotto il primo Problem Set.)

6. (24/10/16) 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. (26/10/16). Il Problema dell'ordinamento. Un algoritmo semplice ma inefficiente: il Selection Sort. Un algoritmo migliore: il MergeSort. Un altro algoritmo che usa la tecnica divide et impera: il QuickSort: analisi del caso peggiore, migliore, e intuizioni sul caso medio. Discussione versione randomizzata del QuickSort e differenza fra complessità nel caso medio e tempo atteso di un algoritmo randomizzato.

8. (2/11/16). Progettare algoritmi efficienti attraverso la progettazione di strutture dati efficienti. Un esempio: l'HeapSort - che ordina in loco n elementi in tempo O(n log n) nel caso peggiore.

9. (7/11/16). Esercitazione. 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*, trova (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). Discussione Esercizio 2 primo Problem Set.

10. (9/11/16). Delimitazioni superiori e inferiori di algoritmi e problemi. Un lower bound alla complessità temporale necessaria per ordinare n elementi (per una classe di algoritmi ragionevoli, quelli basati su confronti). Algoritmi veloci per ordinare interi: IntegerSort, BucketSort, RadixSort.

11. (16/11/16). 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: dato un vettore A di n numeri, progettare un algoritmo che in tempo O(n) trova due indici i e j con i<j che massimizzano A[j]-A[i] (Es. 3).

12. (21/11/16). 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.

13. (23/11/16). 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). Altro esercizio: progettare un algoritmo che, preso un albero e in intero h, restituisce il numero di nodi dell'albero di profondità almeno h (Es 5).

14. (28/11/16). 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 di progettazione di un algoritmo per ri-radicare un albero (vedere slide cap. 3).

15. (30/11/16). 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.

16. (05/12/16). Esercitazione. Progettare un algoritmo che, dato un vettore ordinato A[1:n] di n bit, trova il numero k di zero presenti in A. Algoritmo con complessità O(log n). Un miglior algoritmo con tempo O(log k) (Es. 6).

Progettare un algoritmo con complessità lineare che, dato un vettore A[1:n] di n bit, trova l'indice k tale che il numero di zeri in A[1:k] è uguale al numero di uni in A[k+1:n] (Es. 7).

17. (07/12/16). Code con priorità. d-Heap, Heap Binomiali, (cenni sugli) Heap di Fibonacci.

18. (12/12/16). 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.

19. (14/12/16). Ancora sulla tecnica della programmazione dinamica. Esercizio: aiutate il Re Imprenditore (Es. 8). Altro esercizio: vendere al meglio una stecca di cioccolata (Es. 9). Il problema della sottosequenza crescente più lunga di un vettore (Es. 10).

20. (19/12/16). Ancora sulla tecnica della programmazione dinamica. Calcolo della distanza fra due parole. Esercizio: il Signor Marche va in vacanza fra Roma e Firenze: aiutatelo a spendere il meno possibile (Es. 11).

21. (21/12/16). Esercitazione sulla programmazione dinamica. Il signor Marche e il suo lavoro a cottimo (Es. 12). Il problema del distributore automatico: minimizzare il numero di monete nel dare il resto (Es. 13). Algoritmo greedy (goloso): non sempre funziona. Algoritmo di programmazione dinamica che risolve il problema (discussione).

22. (9/01/17). Esercitazione sulla programmazione dinamica. Algoritmo di programmazione dinamica per il problema del distributore automatico (Es. 13). Aiutate Homer Simpson a mangiare più donut possibile (Es. 14).

23. (11/01/17). Correzione esercizi 2 e 4 del Problem Set 2.

24. (16/01/17). Correzione Esercizio 3 Problem Set 2.