Analisi e Progetto di Algoritmi

UniShare

Davide Cozzi
@dlcgold

Gabriele De Rosa
@derogab

Federica Di Lauro
@f_dila

Indice

Capitolo 1

Introduzione

Questi appunti sono presi a lezione. Per quanto sia stata fatta una revisione è altamente probabile (praticamente certo) che possano contenere errori, sia di stampa che di vero e proprio contenuto. Per eventuali proposte di correzione effettuare una pull request. Link: `https://github.com/dlcgold/Appunti`.
Grazie mille e buono studio!

Capitolo 2

Introduzione al corso

2.1 Argomenti

Si hanno diversi tipi di problemi:

- **problemi di ottimo** dove si cercano singole soluzioni efficienti (massimi o minimi) tra molte soluzioni possibili. Si usa anche la **programmazione greedy**, dove si sceglie in base ai costi locali per ottenere massimi e minimi senza però guardare i costi complessivi.
- **problemi non risolubili in tempi accettabili**, per i quali si usa la **programmazione dinamica**, che cerca di individuare sottostrutture ottime per risolvere il problema, cercando la soluzione migliore memorizzando le altre soluzioni e utilizzandole. Si cerca comunque la soluzione meno dispendiosa in termini di tempo.
- **problemi NP-completi**, ovvero problemi per cui non si può trovare un algoritmo o non si può trovare un algoritmo con una complessità asintotica polinomiale. Si useranno anche tecniche non deterministiche. Si cercherà di studiare uno dei 10 problemi più difficili della matematica: $P \subseteq NP$?

Studieremo poi i grafi non pesati con gli algoritmi *BFS* (per cercare in ampiezza) e *DFS* (per cercare in profondità). Studieremo anche i grafi pesati con problemi di cammino minimo.

2.2 Ripasso Algoritmi 1

Innazitutto due algoritmi con lo stesso scopo si possono confrontare in base a tempo e spazio, scegliendo anche in base alle esigenze hardware. Per lo spazio si calcola quanto spazio viene richiesto da variabili e strutture dati, soprattutto queste ultime che dipendono dalla dimensione dell'input. Per quanto riguarda il tempo si usano le tecniche di conto soprattutto basate sui cicli e, in generale, su tutte operazioni da effettuare. Il tempo si basa sull'input n e si indica con $T(n)$ e si esprime in forma asintotica, interessandoci quindi unicamente all'ordine di grandezza. Si hanno il caso peggiore, indicato con l'O-grande e quello migliore indicato con l'o-piccolo a seconda di n.
Si ricorda poi la tecnica della ricorsione con algoritmi che si muovono su se stessi mediante dei "passi" arrivando ad un caso base di uscita. Per calcolare i tempi di un algoritmo ricorsivo si ha $T(n) = F(n)+T(n-1)$ con F che rappresenta le istruzioni delle subroutines. Questa equazione di ricorrenza non è facilmente calcolabile ma può essere espansa muovendosi sui passi fino a che non si arriva a qualcosa di calcolabile grazie al caso 0, questo è il metodo iterativo (anche se si ha anche il metodo per sostituzione). Per gli algoritmi ricorsivi si hanno anche i divide et impera (dove il problema P è diviso in sottoproblemi risolti separatemente, con la divide, e poi combinati alla fine, con la combina) dove i tempi non sono sempre calcolabili ma se lo sono si usa il metodo dell'esperto (studiando le tre possibili casistiche). Abbiamo poi visto alcune strutture dati: *array, list, stack, queue, tree (e binary-tree) e heap.*