Complessità computazionale

Introduzione al corso

Piero A. Bonatti

Università di Napoli Federico II

Laurea Magistrale in Informatica

Introduzione

- Questo corso tratta la *teoria della complessità*
  - una branca dell'informatica teorica che studia con rigore matematico la *complessità computazionale degli algoritmi* e la *complessità intrinseca dei problemi*

- Estende significativamente le nozioni di base trattate in Algoritmi e Strutture Dati
  - analisi della complessità degli algoritmi, notazione O, Ω, Θ

e in Ricerca Operativa e Ottimizzazione Combinatoria
  - classi **P**, **NP** e riduzioni

Introduzione

- Questo corso tratta la *teoria della complessità*
  - una branca dell'informatica teorica che studia con rigore matematico la *complessità computazionale degli algoritmi* e la *complessità intrinseca dei problemi*

- Estende significativamente le nozioni di base trattate in Algoritmi e Strutture Dati
  - analisi della complessità degli algoritmi, notazione O, Ω, Θ

  e in Ricerca Operativa e Ottimizzazione Combinatoria
  - classi **P**, **NP** e riduzioni

Utilizzi della teoria della complessità

- Analisi degli algoritmi
  - uso delle risorse, come tempo di CPU e memoria richiesti
    - ma si possono concepire altre misure...
  - quindi previsione del comportamento / analisi di scalabilità
  - confronti tra algoritmi diversi, che possono risultare
    - uno migliore dell'altro
    - sostanzialmente equivalenti
    - non confrontabili tra loro, ad es. uno potrebbe impiegare meno tempo ma più memoria dell'altro

Altri utilizzi della teoria della complessità

- Conoscenza strutturale dei *problemi*
  - e non più dei singoli algoritmi che li risolvono
  - quali sono le risorse *minime* necessarie a risolvere il problema?
  - sono necessarie ricerche articolate in spazi grandi e non strutturati?
  - quante?
  - mutuamente indipendenti oppure no?
  - con queste e altre analisi capisco *quante risorse mi servono davvero per risolverlo*

Ulteriori utilizzi della teoria della complessità

- Combinando le analisi dei singoli algoritmi e quelle della complessità intrinseca dei problemi posso giudicare l'ottimalità di un algoritmo che lo risolve
- Si ha una guida nelle nuove soluzioni algoritmiche
  - ho trovato un algoritmo; lo posso migliorare o passo a fare altro?
  - dico al committente: quello che mi chiedi è *impossibile* (o mi farebbe vincere il premio Nobel se ce la facessi)
  - attenzione: fuori da **P** non c'è solo **NP**
    - e **NP** $\neq$ **ExpTime** !!!

Applicazioni della teoria della complessità alla crittografia

- Fornisce garanzie di robustezza del cifrario
  - violare crittosistemi come RSA, Diffie-Hellman ecc. si dimostra essere un problema *inerentemente complesso*
    - non sono noti algoritmi polinomiali
    - trovarli ci darebbe il Nobel (dimostreremmo $\mathbf{P} = \mathbf{NP}$)
  - in gergo: sono cifrari *computazionalmente sicuri*
    - cambiando le chiavi abbastanza spesso si ha la ragionevole certezza che il cifrario non sarà violato
    - gli algoritmi utilizzabili per farlo richiedono troppo tempo

Applicazioni meno note: Complessità descrittiva

- Consideriamo i problemi di *query answering*
  - sulle basi di dati e di conoscenza (inferenza, ragionamento)
  - semantic web, standard OWL
- Quando progetto un linguaggio di query come lo valuto?
  - quante query riesce a formulare? quanta conoscenza?
  - *infinite*
- Allora tutti i linguaggi di query sono uguali?
  - no, la differenza è misurabile con le classi di complessità
  - i programmi logici (con una piccola aggiunta) possono esprimere *tutte le query calcolabili in tempo polinomiale*
  - l'algebra relazionale (ovvero SQL) ne esprime solo alcune: ad es. non può dire se un nodo A è raggiungibile da B in un grafo rappresentato con una tabella relazionale
- Ci aiuta anche quando non sappiamo dimostrare la differenza
  - ad es. due linguaggi che esprimono esattamente le query in **P** e **NP**, rispettivamente; ci aspettiamo che uno sia più potente, a meno che qualcuno non vinca il Nobel...

Informazioni utili

- Libro di testo:
  - Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley. 1995 (o versioni successive)
- Per ulteriori informazioni sulle classi **NP**/**coNP** (compresa una lunga lista di problemi "rappresentativi") potete cercare su
  - Garey, Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co. 1979
- Una terrificante panoramica online delle classi di complessità si può trovare su *Complexity Zoo*
  - https://complexityzoo.uwaterloo.ca/Complexity_Zoo
  - noi esploreremo una piccola parte di quella gerarchia
- Esame: prova scritta e orale
- Ricevimento: per appuntamento (via email)