Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Laurea Magistrale in Ingegneria e Scienze Informatiche

Algoritmi

Complessità

Riassunto dei principali argomenti

Candidati:
Davide Bianchi
Matteo Danzi

Anno Accademico 2017/2018

Indice

1 Introduzione

1.1 Cos'è la complessità computazionale

Nella teoria della complessità ci si pone la seguente domanda:

> *Come scalano le risorse necessarie per risolvere un problema all'aumentare delle dimensioni del problema?*

La teoria della *complessità computazionale* è una parte dell'informatica teorica che si occupa principalmente di classificare i problemi in base alla quantità di *risorse computazionali* (come il tempo di calcolo e lo spazio di memoria) che essi richiedono per essere risolti. Tale quantità è detta anche *costo computazionale* del problema.

1.2 Problemi *facili* e *difficili*

Vediamo quattro esempi di problemi che classificheremo come facili o difficili:

1. (**Eulerian Cycle**) Esiste un modo per attraversare ogni arco di un grafo una e una sola volta?

   - Il problema si può vedere anche nella forma più piccola del problema dei *sette ponti di Königsberg*:
     A Königsberg ci sono 7 ponti, esiste un percorso che attraversa tutti i ponti una e una sola volta per poi tornare al punto di partenza?
     Se avessi n ponti e su ogni riva partissero 2 ponti avrei 2^n possibili percorsi.
   - La **soluzione di Eulero** dice che un grafo connesso non orientato ha un percorso che parte e inizia esattamente nello stesso vertice e attraversa ogni arco esattamente una volta se e solo se ogni vertice ha grado dispari (grado = numero di archi uscenti).
     Se ci sono esattamente due vertici v e u, di grado dispari, allora esiste un percorso che parte da u e attraversa ogni arco esattamente una volta e finisce in v.
   - Seguendo quindi la soluzione di Eulero, *quanto costa decidere* se un grafo G ha un tour Euleriano?

     ```
     odd-vertex-num = 0;
     For each vertex v of G
         if (deg(v) is odd)
             increment odd_vertex-num
     If(odd-vertex-num is neither 0 nor 2)
         output no Eulerian tour
     output Eulerian
     ```

     Questo algoritmo ha complessità: $O(|E| + |V|)$
     Il costo e l'algoritmo sono gli stessi se vogliamo *provare* che G *non* ha un tour Euleriano.

2. (**Hamiltonian Cycle**) Esiste un modo per attraversare ogni nodo di un grafo una e una sola volta?

   Esistono diverse soluzioni:

   - Provo tutte le possibilità ogni volta, costo: $O(2^n)$
   - Provo tutte le possibili permutazioni, costo: $O(n!)$
   - La soluzione migliore ad oggi è: $O(1.657^n)$

   Alla domanda: *Quanto costa decidere se un grafo ha un tour hamiltoniano?* Non sappiamo rispondere. Non sappiamo dire se il problema ha una soluzione non esponenziale. Per quanto ne sappiamo meglio di $O(1.657^n)$ non sappiamo fare.

   Non sappiamo nemmeno dire se Hamiltonian Cycle è più difficile di Eulerian Cycle.

3. N è un numero primo?

   Il migliore algoritmo conosciuto per decidere se N è un numero primo impiega $O((\log N)^{6+\epsilon})$

4. Quali sono i fattori primi di un numero?

   Ad oggi non conosciamo una procedure per fattorizzare un numero molto grande nei suoi divisori, che non sia provare tutte le possibilità.

1.3 Risolvere vs Verificare

La seguente tabella riassume in modo generico quanto detto nella sezione precedente riguardo alla difficoltà di risolvere problemi e verificare tali problemi su un istanza.

Tabella 1: Risolvere vs Verificare

Problema	**Risolvere**	**Verificare**
Eulerian Cycle	*facile*	*facile*
Hamiltonian Cycle	*difficile?*	*facile*
N è primo?	*facile*	*facile*
N ha un numero piccolo di fattori?	*difficile?*	*facile*

2 Problema computazionale

Un problema computazionale è una semplice relazione p che mappa l'insieme *infinito* di possibili input (domande o istanze) con un insieme *finito* (non vuoto) di output, cioè di risposte o soluzioni alle istanze.

$$\text{p} : \text{ istanze infinite } \mapsto \text{ soluzioni finite alle istanze}$$

Un problema computazionale non è una singola domanda, ma è una **famiglia di domande**:

- Una domanda per ogni possibile istanza
- Ogni domanda è dello stesso tipo (appartiene alla stessa classe)