Fondamenti dell'Informatica

A.A. 2002/2003

Corsi A e B

Seconda prova di esonero: 3/6/2003 ore 9.00 – 11.00 (fac simile)

1. Riportare l'algoritmo di ordinamento per inserzione. Scegliere una operazione dominante, motivandone la scelta. Analizzare la complessità in tempo dell'algoritmo nel caso medio. Qual è l'assunzione che si effettua sulla distribuzione di probabilità? (*13 punti*)
2. Fornire la definizione di *schema di codifica* per l'insieme delle istanze di un problema e spiegare quando e perché uno schema di codifica si dice ragionevole. (*8 punti*)
3. Illustrare, fornendo anche esempi, il concetto di completezza per una classe di complessità (*8 punti*)
4. Definire le classi LOGSPACE, P e NP e illustrare le relazioni fra esse esistenti (*8 punti*)

Esercizio n. 1

La procedura Pascal-like che realizza un ordinamento per inserzione è la seguente:

```
procedure insertsort(var a: nelements; n: integer);
var i, j, first, p, x: integer;
begin
        first:=a[1]; p :=1;
        for i:=2 to n do
                if a[i]<first then
                        begin
                            first := a[i]; p:=i
                        end;
                a[p] := a[1]; a[1] := first;
        for i:= 3 to n do
                begin
                        x :=a[i]; j := i;
                        while x<a[j-1] do
                                begin
                                    a[j] := a[j-1]; j:=j-1;
        end;
                        a[j] :=x
        end
end
```

Esso opera posizionando inizialmente l'elemento minimo in prima posizione (primo ciclo for). A questo punto, il segmento dell'array a costituito dai primi DUE elementi risulterà ordinato (sicuramente il minimo è minore del secondo elemento). Si procede, quindi, selezionando il primo elemento del segmento da ordinare (inizialmente esso sarà il terzo elemento) e confrontandolo con gli elementi che lo precedono, finché non si individua la posizione che dovrebbe occupare nel primo segmento ordinato dell'array. Si piazza l'elemento dopo aver provveduto a effettuare una copia in posizione k+1 degli elementi confrontati che occupavano posizione k.

L'operazione dominante scelta è quella di confronto, e più precisamente le operazioni a[i]<first e x<a[j-1]. La scelta ricade su di esse perché sono coinvolte nel primo ciclo for e nel ciclo più interno al secondo ciclo for. Inoltre, il meccanismo del confronto è fondamentale ai fini dell'ordinamento.

Per valutare la complessità dell'algoritmo di ordinamento indichiamo con *n* il numero di elementi dell'array a e stabiliamo che la dimensione dei dati in ingresso sia *n*. Quindi cerchiamo una funzione di complessità temporale $f_{med}(n)$ che esprima il numero medio di confronti effettuati dalla procedura riportata sopra, assumendo quanto segue: la probabilità che a[i], i=3, 4, …, *n* venga inserito in posizione k-esima, k=2, 3, …, i, è la stessa, indipendentemente da i e k.
Osserviamo innanzitutto che il numero di confronti nel primo ciclo è *n*-1, indipendentemente dalla configurazione dei dati in ingresso.
La complessità media per inserire correttamente *a[i]*, *i*=3, 4, …, *n* è data da:

$$\sum_{k=2}^{i} \text{costo}(E_k) P(E_k) = \sum_{k=2}^{i} (i-k+1)\frac{1}{i-1} =$$

$$= \frac{1}{i-1}\left[\sum_{k=2}^{i} i - \sum_{k=2}^{i} k + \sum_{k=2}^{i} 1\right] = \frac{1}{i-1}\left\{i(i-1) - \left[\frac{i(i+1)}{2} - 1\right] + (i-1)\right\} =$$

$$= \frac{1}{i-1}\left(i^2 - i - \frac{i^2}{2} - \frac{i}{2} + 1 + i - 1\right) = \frac{1}{i-1}\frac{i^2 - i}{2} = \frac{i}{2}$$

La complessità nel caso medio è quindi data da:

$$f_{med}(n) = n - 1 + \sum_{i=3}^{n} \frac{i}{2} = n - 1 + \frac{1}{2}\left(\frac{n(n+1)}{2} - 3\right) =$$

$$= n - 1 + \frac{n^2 + n - 6}{4} = \frac{n^2 + 5n - 10}{4}$$

cioè è dell'ordine $O(n^2)$.
L'ordinamento per inserzione è di solito considerato il migliore degli algoritmi con complessità $O(n^2)$ per ordinare piccoli insiemi con scarso preordinamento.

Esercizio n. 2

Dato un problema $\mathcal{P}$, indichiamo con $I_{\mathcal{P}}$ con insieme delle possibili istanze. Dato un alfabeto Σ, definiamo *schema di codifica* di $\mathcal{P}$ una funzione e: $I_{\mathcal{P}} \mapsto \Sigma^*$ che mappa ogni istanza di $\mathcal{P}$ in una corrispondente stringa di simboli in Σ^*.
Uno schema di codifica si dice *ragionevole* se non ha istanze la cui descrizione è artificiosamente lunga. Infatti la complessità di un algoritmo viene definita in funzione della dimensione di una istanza (cioè dell'input) rispetto a qualche schema di codifica, per cui una descrizione artificiosamente lunga può far apparire la complessità di un algoritmo più bassa.

Uno schema di codifica ragionevole è uno che:

- considera codifiche in base $k \leq 2$ dei valori numerici interessati;
- rappresenta insiemi mediante enumerazione delle codifiche dei loro elementi;
- rappresenta relazioni e funzioni mediante enumerazione delle codifiche dei relativi elementi (coppie e *n*-ple);
- rappresenta grafi come coppie di insiemi (di nodi ed archi).

Per gli schemi di codifica ragionevole vale la seguente condizione di correlazione polinomiale:
Due schemi di codifica *e* ed *e'* per un problema $\mathcal{P}$ sono *polinomialmente correlati* se esistono due polinomi *p* e *p'* tali che, per ogni $x \in I_{\mathcal{P}}$:

- $|e(x)| \leq p'(|e'(x)|)$
- $|e'(x)| \leq p(|e(x)|)$

Pertanto la complessità di un algoritmo risulta invariante (a meno di un polinomio) al variare dello schema utilizzato.

Ad esempio, un algoritmo che decide se un numero naturale x è primo dividendolo per tutti i naturali $i<x$ e controllando il resto della divisione, eseguirà $O(x)$ passi (assumendo, per semplicità, che una divisione sia eseguita in tempo costante rispetto al valore dei relativi operandi).
Se una codifica è ragionevole allora $|e(x)| = \log_k x$ per qualche $k \geq 2$. La complessità, espressa in funzione della codifica dell'input x, sarà quindi $O(x)=O(k^{\log_k x})=O(k^{|e(x)|})$ e risulterà quindi esponenziale. Al contrario, se si codificasse l'input mediante una codifica *unaria*, utilizzado un alfabeto di un solo carattere "|" e rappresentando l'intero x per mezzo di una stringa di n caratteri "|", l'algoritmo precedente risulterebbe avere complessità $O(x) = O(|e(x)|)$, cioè lineare.

Esercizio n. 3
Il concetto di completezza è legato alla riducibilità tra problemi. Informalmente un problema $\mathcal{P}_1$ è riducibile a un problema $\mathcal{P}_2$ (si scrive $\mathcal{P}_1 \leq_r \mathcal{P}_2$) se disponendo di un algoritmo per risolvere $\mathcal{P}_2$ siamo in grado di risolvere $\mathcal{P}_1$.

Fissata una relazione di riducibilità $\leq_r$ tra problemi (ve ne sono diverse, come quella di Karp, polinomiale o logspace), e fissata una classe di complessità $\mathcal{C}$ si può dare la seguente definizione:

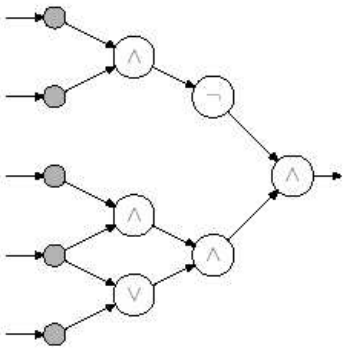
Per ogni classe di complessità $\mathcal{C}$, un problema di decisione $\mathcal{P}$ viene detto *difficile* in $\mathcal{C}$ ($\mathcal{C}$-hard) rispetto a $\leq_r$ se e solo se per ogni altro problema di decisione $\mathcal{P}_1 \in \mathcal{C}$ si ha che $\mathcal{P}_1 \leq_r \mathcal{P}$.

Si ha la *completezza* di $\mathcal{P}$ in $\mathcal{C}$ (si dice che $\mathcal{P}$ è $\mathcal{C}$-completo) se oltre a essere difficile in $\mathcal{C}$ si ha che $\mathcal{P} \in \mathcal{C}$.

Un esempio di problema completo per la classe **P** dei linguaggi decidibili da una macchina di Turing deterministica in tempo polinomiale nella dimensione dell'input è quello del valore calcolato da un circuito, che risulta appunto **P**-completo rispetto ad una riduzione log-space.
Per definirlo occorre dare la definizione di circuito.

Definizione 2 *Un circuito booleano è un grafo orientato aciclico in cui ogni nodo ha $d_{in} \leq 2$ archi entranti. In particolare:*
1. *i nodi con $d_{in} = 0$ sono detti nodi di input;*
2. *i nodi con $d_{in} = 1$ sono detti porte NOT;*
3. *i nodi con $d_{in} = 0$ sono detti porte AND o porte OR.*

Tutti i nodi hanno $d_{out} > 0$, eccetto un solo nodo di output.



Sia $I(C)$ l'insieme di n nodi di input del circuito C; data una assegnazione di n valori booleani, il circuito calcola i valori booleani in uscita da ogni nodo, applicando su ogni porta l'operatore ad essa associato.

Il circuito calcola la funzione $f_C : \{0,1\}^n \longmapsto \{0,1\}$ tale che $f_C(b_0,\ldots,b_{n-1}) = 1$ sse il circuito restituisce il valore 1 per input $b_0,\ldots,b_{n-1}$.

Definizione 3 *Dato C, con $|I(C)| = n$, il linguaggio deciso da C è l'insieme delle stringhe $w \in \{0,1\}^n$ tale che $f_C(w) = 1$*

VALORE CALCOLATO DA UN CIRCUITO
ISTANZA: circuito booleano C con nodi di input $|I(C)|$; stringa $w \in \{0,1\}^n$.
PREDICATO: si ha $f_C(w) = 1$?

Esercizio n. 4
Le classi **P**, **NP** e **LOGSPACE** sono definite come segue:

$P = \cup_{k=0}^{\infty} DTIME(n^k)$ è la classe dei linguaggi decidibili da una macchina di Turing deterministica in tempo polinomiale nella dimensione dell'input;

$LOGSPACE = \cup_{k=0}^{\infty} DSPACE(\log n)$ è la classe dei linguaggi decidibili da una macchina di Turing deterministica in spazio logaritmico nella dimensione dell'input;

$NP = \cup_{k=0}^{\infty} NTIME(n^k)$ è la classe dei linguaggi decidibili da una macchina di Turing non deterministica in tempo polinomiale nella dimensione dell'input;

Dalla semplice osservazione che ogni macchina di Turing deterministica è un caso particolare di macchina di Turing deterministica discende il seguente teorema:

Teorema 22 *Per ogni funzione $f : \mathbb{N} \longmapsto \mathbb{N}$ valgono le due proprietà:*
$DTIME(f(n)) \subseteq NTIME(f(n))$;
$DSPACE(f(n)) \subseteq NSPACE(f(n))$.

e quindi il corollario che **P** $\subseteq$ **NP**.

Inoltre vale il seguente teorema:

Teorema 37 *Per ogni funzione space constructible* $f : \mathbb{N} \longmapsto \mathbb{N}$ *tale che* $f(n) = \Omega(\log n)$ *si ha che* $DSPACE(f(n)) \subseteq DTIME(2^{f(n)})$.

Corollario 38 $LOGSPACE \subseteq P$.

Quindi valgono le inclusioni: **LOGSPACE** $\subseteq$ **P** $\subseteq$ **NP**.

Le inclusioni sono verosimilmente strette, a meno che non si dimostra che il problema **P**-completo del valore calcolato da un circuito è in **LOGSPACE** e il problema **NP**-completo SODDISFACIBILITÀ è in **P**.