COMPLESSITÀ COMPUTAZIONALE DEGLI ALGORITMI

Fondamenti di Informatica a.a.2006/07
Prof. V.L. Plantamura
Dott.ssa A. Angelini

Bontà dei dati in ingresso

- Il costo puo' dipendere dal *valore* dei dati in ingresso
- Un tipico esempio è dato da:
  if (condizione) {corpo istruzioni};
  if (condizione) {corpo 1} else {corpo 2};

Esempio

Problema: ricerca di un elemento in un vettore A (array) di numeri. **Algoritmo risolutivo**: ricerca lineare nel vettore.

```
i=0;     //istruzioni della procedura di ricerca
while ((i< n) && (A[i] <> elem))
```

3	-2	0	7	5	4	0	8	-3	-1	9	12	20	5

Supponiamo di avere questo array di 14 elementi. Definiamo come dimensione dell'input la lunghezza dell'array: **n=14**

1) Cerchiamo il numero 8: 1 (assegnaz.) + 8 (test) = **9** passi base;
2) Cerchiamo il numero 3: 1 (assegnaz.) + 1 (test) = **2** passi base;
3) Cerchiamo il numero 6: 1 (assegnaz.) + n+1 (test) = **2 + n** passi base (16 nell'esempio).

Bontà dei dati in ingresso

Supponendo di sapere che l'elemento cercato è presente nell'array, si possono distinguere tre casi:

•*Caso migliore*: l'elemento cercato è il primo. Costo: **2**;

•*Caso peggiore*: l'elemento cercato è l'ultimo. Costo: **2+n**;

•*Caso medio*:

- •Ipotesi di distribuzione uniforme (disp. casuale);
- •Probabilità di trovare l'elemento x cercato in posizione i nell'array è: P(*x*) = (1/*n*)*i
- •In totale il numero medio di confronti da effettuare è:

$$\sum_{i=1}^{n} P(x) = (n+1)/2$$

Bontà dei dati in ingresso

- Nella seconda situazione, in cui non si sa se l'elemento cercato è presente nell'array, non si può assegnare un valore di probabilità come fatto prima:
  - In questa situazione il caso migliore è la presenza dell'elemento nell'array, il caso peggiore è l'assenza;
  - Il caso medio si può ricavare, se si dispone di una stima della probabilità di presenza, con una somma pesata del caso medio della prima situazione e del caso di assenza, che implica il costo di scansione dell'intero array.

Bontà dei dati in ingresso

- *Caso migliore*: quelle configurazioni della struttura dei dati che danno luogo ad un minimo della funzione *T(n)*;
- *Caso peggiore*: quelle configurazioni che danno luogo ad un massimo;
- *Caso medio*: corrisponde al comportamento medio.

Oss: L'algoritmo può avere tempo di esecuzione *T(n)* indipendente dai valori assunti dai dati. In tal caso i tempi impiegati nei diversi casi coincidono.

Modello di calcolo

- Per ottenere una valutazione del costo che dipende solo dalla dimensione dell'input e non dalla bontà si fa riferimento al ***caso peggiore***;
- Anche la nozione di caso peggiore dipende dall'algoritmo considerato.
  - Ad esempio per un dato algoritmo di ordinamento il caso peggiore potrebbe verificarsi quando tutti gli elementi da ordinare sono disposti in maniera decrescente. Tuttavia per un altro algoritmo questa potrebbe essere una condizione favorevole.

Programmi Strutturati

- Nel calcolo della complessità di un programma strutturato si deve:
  - calcolare la complessità di ogni procedura e funzione;
  - per ogni chiamata a procedura/funzione, aggiungere la complessità della stessa al costo globale del programma.

Esempio 7

```
void Stampastelle (int ns);
      main () {
         int n,m,j;
         printf("Quante stelle per riga?"); scanf("%d", &n);
         printf("Quante righe di stelle?"); scanf("%d", &m);
         for (j=1; j<=m; j++) Stampastelle(n);
      } \* main *\
void Stampastelle (int ns) {
      int i;
      printf ("\n");
      for (i=1; i <= ns; i++) printf ("*");
      } \* Stampastelle *\
```

Esempio 8

```
void Stampastelle (int ns);
      main () {
         int m,j;
         printf (“Quante righe di stelle?”);
         scanf(“%d”, &m);
         for (j=1; j <= m; j++)
                Stampastelle(j);
      } \* main *\
void Stampastelle (int ns) {
      int i;
      printf (“\n”);
      for (i=1; i <= ns; i++) printf (“*”);
      } \* Stampastelle *\
```

Esempio 9

```
for (i=0; i<n; i++) {
    for (j=1, sum = a[0]; j<=i; j++)
      sum += a[j]; }
```

Esempio…

```
# define MAXN 10000; /* massimo numero di elementi */
int Int_Array[MAXN] /* dichiarazione array */
void CaricaArray(int L); /* Carica L valori in array (L<=MAXN deciso
                            dall’utente) */
int RicercaLineare(int L, int x);
/* restituisce la posizione di un elemento in un array
   Int_Array; limita la ricerca alle prime L posizioni;
   se l’elemento è presente restituisce la posizione, altrimenti
   restituisce -1 */
main () {
    int L, elemento, pos; /* richiesta delle dimensioni reali, L di Int_Array */
    do {  printf(“\n Numero elementi: “);
          scanf(“%d”, &L); }
    while ((L >= MAXN) || (L < 0))
    CaricaArray(L);
    printf(“\n elemento da cercare: “);
    scanf(“%d”, &elemento);
    pos = RicercaLineare (L, elemento);
```

…Esempio

```
    if (pos >= 0)
        printf("L'elemento cercato si trova in posizione", "%d",
    pos)
    else
        printf ("L'elemento cercato non è presente nell'array")
}
void CaricaArray(int Nelem) {
    int i;
    for (i=0; i <= Nelem; i++) {
        printf("Inserisci elemento %d:", i);
        scanf("%d", &Int_Array[i]);} };
int RicercaLineare(int Nelem, int x) {
    int k;
    k =0;
    while ((k <= Nelem) && (Int_Array[k] !=x)) ++k;
    if (k <= Nelem) return k
    else return -1; }
```