Esame di Algoritmi e Strutture di Dati (A)
Lunedì 2 Settembre 2002

Nome:
Cognome:
Matricola:

Per un buon esito dell'esame, si consiglia di:

- scrivere in forma leggibile il proprio nome, cognome e matricola sul testo del compito e su ogni foglio consegnato;
- provare gli esercizi prima in brutta copia. Ricopiarli in bella copia e consegnare quest'ultima oltre al testo del compito;
- non fatevi prendere dal panico, e neppure dalla noia. Un giusto livello di tensione è ciò che serve;
- svolgere il compito individualmente. Passare copiando è dannoso per voi, e irrilevante per il docente.

Esercizio 1 (*Punti 24*)

Si consideri la struttura di dati insieme. *Si rappresenti un insieme di numeri mediante una lista in cui il campo chiave di ogni oggetto della lista contiene un elemento dell'insieme. Si considerino i seguenti due casi:*

1. *la lista che rappresenta l'insieme non è ordinata (cioè gli oggetti nella lista sono inseriti in ordine qualsiasi);*
2. *la lista che rappresenta l'insieme è ordinata (cioè gli oggetti nella lista sono inseriti in ordine crescente di chiave).*

In entrambi i casi, si scriva la procedura $SubSet(A, B)$ *che verifica se l'insieme contenuto nella lista* A *è un sottoinsieme di quello contenuto nella lista* B*. Si calcoli in entrambi i casi la complessità computazionale della procedura.*

Soluzione

Nel caso della lista non ordinata abbiamo la seguente procedura:

Algoritmo 1 $SubSet(A, B)$

```
SubSet(A, B)
1: x ← head(A)
2: while x ≠ NIL do
3:    if ListSearch(B, key[x]) = NIL then
4:       return FALSE
5:    else
6:       x ← next[x]
7:    end if
8: end while
9: return TRUE
```

Siano n la lunghezza di A e m la lunghezza di B. Dato che la complessità di $ListSearch$ è lineare, la complessità di $SubSet$ risulta $\Theta(nm)$.

Nel caso della lista ordinata abbiamo la seguente procedura:

Algoritmo 2 $SubSet(A, B)$

```
SubSet(A, B)
 1: x ← head(A)
 2: y ← head(B)
 3: while x ≠ NIL do
 4:    while (y ≠ NIL) and (key[x] ≠ key[y]) do
 5:       y ← next[y]
 6:    end while
 7:    if y = NIL then
 8:       return FALSE
 9:    else
10:       x ← next[x]
11:       y ← next[y]
12:    end if
13: end while
14: return TRUE
```

Siano n la lunghezza di A e m la lunghezza di B. La complessità di $SubSet$ risulta $\Theta(n + m)$.

Esercizio 2 (*Punti 6*)

1. *Siano S una sequenza di numeri distinti e i un indice di S. Si dia la definizione di elemento di ordine i della sequenza S;*

2. *siano A un vettore di numeri distinti e i un indice di A. Si consideri la procedura $Select(A, i)$ che restituisce l'elemento di ordine i di A. Si scriva una procedura $SelectSort(A)$ che risolve il problema dell'ordinamento usando la procedura $Select$.*

Soluzione

L'elemento di ordine i di una sequenza S è quell'elemento si S tale che esistono $i - 1$ elementi in S più piccoli di esso.

Algoritmo 3 $SelectSort(A)$

```
SelectSort(A)
1: for i ← 1 to length(A) do
2:    B[i] ← Select(A, i)
3: end for
4: for i ← 1 to length(A) do
5:    A[i] ← B[i]
6: end for
```

Esame di Algoritmi e Strutture di Dati (B)
Lunedì 2 Settembre 2002

Nome:
Cognome:
Matricola:

Per un buon esito dell'esame, si consiglia di:

- scrivere in forma leggibile il proprio nome, cognome e matricola sul testo del compito e su ogni foglio consegnato;
- provare gli esercizi prima in brutta copia. Ricopiarli in bella copia e consegnare solo quest'ultima;
- non fatevi prendere dal panico, e neppure dalla noia. Un giusto livello di tensione è ciò che serve;
- svolgere il compito individualmente. Passare copiando è dannoso per voi, e irrilevante per il docente.

Esercizio 3 (*Punti 24*)

Si consideri la struttura di dati insieme. *Si rappresenti un insieme di numeri mediante una lista in cui il campo chiave di ogni oggetto della lista contiene un elemento dell'insieme. Si considerino i seguenti due casi:*

1. *la lista che rappresenta l'insieme non è ordinata (cioè gli oggetti nella lista sono inseriti in ordine qualsiasi);*
2. *la lista che rappresenta l'insieme è ordinata (cioè gli oggetti nella lista sono inseriti in ordine crescente di chiave).*

In entrambi i casi, si scriva la procedura $SetEqual(A, B)$ *che verifica se gli insiemi contenuti nelle liste* A *e* B *sono uguali. Si calcoli in entrambi i casi la complessità computazionale della procedura.*

Soluzione

Nel caso della lista non ordinata, usiamo la seguente procedura ausiliaria $SubSet(A, B)$ che verifica se l'insieme contenuto nella lista A è un sottoinsieme di quello contenuto nella lista B (vedi soluzione Compito A).

Algoritmo 4 $SetEqual(A, B)$

$SetEqual(A, B)$

1: **return** $SubSet(A, B)$ **and** $SubSet(B, A)$

Dato che la complessità di $SubSet$ è $\Theta(nm)$, la complessità di $SetEqual$ risulta $\Theta(2nm) = \Theta(nm)$.

Nel caso della lista ordinata abbiamo la seguente procedura:

Algoritmo 5 $SetEqual(A, B)$

```
SetEqual(A, B)
 1: x ← head(A)
 2: y ← head(B)
 3: while (x ≠ NIL) and (y ≠ NIL) do
 4:    if key[x] ≠ key[y] then
 5:       return FALSE
 6:    else
 7:       x ← next[x]
 8:       y ← next[y]
 9:    end if
10: end while
11: if (x = NIL) and (y = NIL) then
12:    return TRUE
13: else
14:    return FALSE
15: end if
```

Siano n la lunghezza di A e m la lunghezza di B. La complessità di $SetEqual$ risulta $\Theta(min\{n, m\})$.

Esercizio 4 (*Punti 6*)

1. *Siano S una sequenza di numeri distinti e i un indice di S. Si dia la definizione di elemento di ordine i della sequenza S;*

2. *siano A un vettore di numeri distinti e i un indice di A. Si consideri la procedura $Select(A, i)$ che restituisce l'elemento di ordine i di A. Si scriva una procedura $SelectSort(A)$ che risolve il problema dell'ordinamento usando la procedura $Select$.*

Soluzione

L'elemento di ordine i di una sequenza S è quell'elemento si S tale che esistono $i - 1$ elementi in S più piccoli di esso.

Algoritmo 6 $SelectSort(A)$

```
SelectSort(A)
1: for i ← 1 to length(A) do
2:    B[i] ← Select(A, i)
3: end for
4: for i ← 1 to length(A) do
5:    A[i] ← B[i]
6: end for
```