Algoritmi & Strutture Dati

Andrea Comar

October 2024

Contents

Part I
Algoritmi

1 Algoritmi di ordinamento

PROBLEMA Problema data una sequenza a1, a2, ..., an di numeri, trovare una permutazione tale che $a1 \leq a2 \leq ... \leq an$.
Soluzioni:

1.1 Insertion sort

Algorithm 1: InsertionSort

```
Data: A array, i indice, j indice
for i ← 2 to A.length do
    key ← A[i];
    j ← j - 1;
    while j > 0 && A[j] > key do
        A[j+1] ← A[j] ;
        j ← j - 1 ;
    A[j+1] ← key;
```

Complessità Spaziale : $\theta(1)$ in richiede unicamente 3 interi (i, j, A.length) per memorizzare i valori.

Complessità Temporale :
- nel caso migliore: $\theta(n)$ vettore già ordinato
- nel caso peggiore: $\theta(n^2)$ vettore ordinato al contrario

Correttezza :

1.2 Merge

Procedura che unisce due vettori ordinati in un unico vettore ordinato. I due vettori di input non devono necessariamente avere la stessa lunghezza.

Algorithm 2: Merge

```
Input: Array A, indices p, r, q
Output: Merged array A[p..q]
i ← p;
j ← r + 1;
B ← new array of size q − p + 1;
k ← 1;
while i < r + 1 and j < q + 1 do
    if A[i] ≤ A[j] then
        B[k] ← A[i];
        i ← i + 1;
    else
        B[k] ← A[j];
        j ← j + 1;
    k ← k + 1;
if i > r then
    for l ← j to q do
        B[k] ← A[l];
        k ← k + 1;
else
    for l ← i to r do
        B[k] ← A[l];
        k ← k + 1;
```

1.3 Merge sort

Idea: divide et impera. Divido il vettore in due parti, ordino le due parti e poi le unisco.

Algorithm 3: MergeSort

```
Input: Array A, indices p, q
Output: Sorted array A[p..q]
if p < q then
    r ← ⌊(p+q)/2⌋;
    MergeSort(A,p,r);
    MergeSort(A,r+1,q);
    Merge(A,p,q,r);
```

MergeSort è un algoritmo basato su ricorsione. Necessita della procedura Merge per unire i due vettori.

Complessità Spaziale : $\theta(n)$ in quanto richiede un vettore di appoggio di dimensione n.

Complessità Temporale : $\theta(n \log n)$ in quanto il vettore viene diviso in due parti e ogni parte viene ordinata in $\log n$ passi.

Part II
Strutture Dati

2 strutture dati lineari

2.1 array

struttura dati **statica** (= suo spazio di memoria non varia) di n elementi. Sono a **indirizzamento diretto** e l'accesso ha un costo fisso di $\theta(1)$

2.2 lista

2.3 code di priorità