

Laboratorio di Algoritmi e Strutture Dati

Prof. Aniello Murano

Ordinamento, Ricorsione e Code di Priorità

Corso di Laurea Codice insegnamento Email docente Anno accademico	Informatica 13917 murano@na.infn.it 2007/2008	Lezione numero: 3 Parole chiave: Ordinamento, Insertion sort, Heapsort,
--	---	---

[next](#)




16/10/2008



La Ricorsione

Il concetto di ricorsione nasce dalla possibilità di eseguire un compito applicando lo stesso algoritmo ad un dominio ridotto rispetto a quello originale **fondendo i risultati.**

Le funzioni C possono essere usate ricorsivamente, cioè una funzione può chiamare se stessa sia direttamente che indirettamente.

Nella ricorsione è importante la condizione di uscita.

Il problema deve poter essere suddiviso in sotto-problemi più piccoli fino ad arrivare ad un sotto-problema banale di cui si conosce immediatamente la soluzione.

[back](#)

[next](#)

Federica 16/10/2008 3 Facoltà di Scienze Matematiche Fisiche Naturali

Soluzione ricorsiva di un problema

PROBLEMA: Voglio lavare una pila di 15 piatti.

Ho un sistema che riesce a lavare i piatti se ha in input una pila più piccola:
(intuitivamente, il calcolatore è in grado di lavare 14 piatti)

Algoritmo *RICORSIVO*: *un po' più semplice del problema intero*
 prendo un piatto,
 resta una pila di 14 piatti:
 il calcolatore li lava
 lavo il mio piatto *facilissimo*

back X next

Federica 16/10/2008 4 Facoltà di Scienze Matematiche Fisiche Naturali

Esempio

```
void lava_piatti(n)
{
  if (nessun_piatto_da_lavare) // caso base
    riposati!
  else // passo ricorsivo
  {
    Prendi un piatto
    Lavalo
    lava_piatti(n-1) // Chiamata Ricorsiva
  }
}
```

Condizione di terminazione

Una funzione ricorsiva:

- Ha una o più condizioni di terminazione (casi base)
- Chiama se stessa ricorsivamente.
- Ad ogni chiamata ci si avvicina alla condizione di terminazione

il numero di piatti da lavare decresce: la terminazione e' garantita!

back X next

Federica 16/10/2008 5 Facoltà di Scienze Matematiche Fisiche Naturali

Stampa dei primi N interi positivi

Sia $N=3$.
Assumiamo che il calcolatore sappia fare la stampa di $N-1$ interi (I primi due interi).
Il mio risultato è dato da:
dopo aver stampato i primi $N-1$ interi (2 interi)
stampa l'ennesimo intero (l'intero 3)
Attenzione: Gli elementi devono essere stampati in ordine crescente!

Considerazioni:

- Assumiamo sempre che la chiamata ricorsiva svolge sempre il suo compito correttamente.
- Bisogna concentrarsi solo sull'ultimo passo di elaborazione per ottenere la soluzione corretta.

back X next

Federica 16/10/2008 6 Facoltà di Scienze Matematiche Fisiche Naturali

```
void StampaN (int n)
{if (n==0)
  return;
 else
  { StampaN(n-1);
    Printf(" %d ",n);}
}
```

Stampa(3)
 N=3;
 Stampa (3-1);

Stampa(2)
 N=2;
 Stampa (2-1);

Stampa(1)
 N=1;
 Stampa (1-1);

Stampa(0)

back X next

Federica 16/10/2008 7 Facoltà di Scienze Matematiche Fisiche Naturali

Esempio di ricorsione

Calcolo del fattoriale di un numero:
 $n! = n \cdot (n-1) \cdot (n-2) \cdot \dots \cdot (n - (n-1))$

<pre>int fact(int num) { int product=1; for(; num>1; --num) product*=num; return product; }</pre>	<pre>int fact(int num) { if (num <= 1) return 1; else return (num* fact(num -1)); }</pre>
--	---

back X next

Federica 16/10/2008 8 Facoltà di Scienze Matematiche Fisiche Naturali

Esempio di ricorsione

Calcolo del numero di Fibonacci

- $f(n) = f(n-1) + f(n-2)$;
- $f(0) = 0$; $f(1) = 1$;

<p>MAIN:</p> <pre>#include <stdio.h> int num; main() { printf("\n numero? "); scanf("%d",&num); printf("ris. = %d",fib(num)); }</pre>	<p>ITERATIVO:</p> <pre>int fib(int num) { int tmp=0, ris=1, prec=0; switch (num) { case 0: return 0;break; case 1: return 1;break; default: { for (; num>1;--num) { tmp=ris; ris+=prec; prec=tmp; } return ris; } } }</pre>	<p>RICORSIVO:</p> <pre>int fib(int num) { switch (num) { case 0: return 0;break; case 1: return 1;break; default: return (fib(num-1) + fib(num-2)); } }</pre>
--	---	--

back X next

Federica 16/10/2008 9 Facoltà di Scienze Matematiche Fisiche Naturali

Algoritmi di ordinamento



Insertion Sort

- L'Insertion Sort è un algoritmo di ordinamento molto efficiente per ordinare un piccolo numero di elementi
- L'idea di ordinamento è quella che potrebbe usare una persona per ordinare le carte nella propria mano.
- Si inizia con la mano vuota e le carte capovolte sul tavolo
- Poi si prende una carta alla volta dal tavolo e si inserisce nella giusta posizione
- Per trovare la giusta posizione per una carta, la confrontiamo con le altre carte nella mano, da destra verso sinistra.
- Ogni carta più grande verrà spostata verso destra in modo da fare posto alla carta da inserire.

back ✖ next

Federica 16/10/2008 10 Facoltà di Scienze Matematiche Fisiche Naturali

Funzione Insertion Sort

```

void insertion(int interi[20],int tot)
{
  int temp;      /* Indice temporaneo per scambiare elementi */
  int prossimo;
  int attuale;
  for (prossimo=1; prossimo<tot; prossimo++)
  {
    temp=interi[prossimo];
    attuale=prossimo-1;
    while ((attuale>=0) && (interi[attuale]>temp))
    {
      interi[attuale+1]=interi[attuale];
      attuale=attuale-1;
    }
    interi[attuale+1]=temp;
  }
}

```

2	4	5	3	6	9	8	1
---	---	---	---	---	---	---	---

↑

back ✖ next

Federica 16/10/2008 11 Facoltà di Scienze Matematiche Fisiche Naturali

Programma per Insertion Sort

```

#include <stdio.h>
#define MAX 20
int i,j; /* Indici di scorrimento dell'array */
void insertion(int interi[MAX],int tot);
main()
{
    int interi[MAX] /* Array che contiene i valori da ordinare */
    int tot; /* Numero totale di elementi nell'array */
    printf("\n Quanti elementi deve contenere l'array: ");
    scanf("%d",&tot);
    while (tot>20)
        { printf("\n max 20 elementi: "); scanf("%d",&tot); }
    for (i=0;i<tot;i++)
        { printf("\nInserire il %d° elemento: ",i+1); scanf("%d",&interi[i]); }
    insertion(interi,tot);
    printf("\nArray Ordinato:");
    for (i=0; i<tot; i++)
        printf(" %d",interi[i]);
}

```

back X next

Federica 16/10/2008 12 Facoltà di Scienze Matematiche Fisiche Naturali

Ingegneria del software...

Autore: **Nome, Cognome, matricola ...**
 Titolo: **nome della funzione (o modulo) implementata.**
 Scopo: **obiettivi dell'algoritmo implementato(sintetico)**
 Specifiche: **Nomi di funzioni, array, variabili importanti**
 Descrizione: **Informazioni sull'algoritmo implementato.**
 Lista dei Parametri: **Parametri input e output**
 Complessità di Tempo e Di Spazio
 Altri parametri eventualmente vuoti:

- **Indicatori di Errore:**
- **Routine Ausiliarie**
- **Indicazioni sull'utilizzo**

Implementazione

back X next

Federica 16/10/2008 13 Facoltà di Scienze Matematiche Fisiche Naturali

Documentazione per Insertion Sort

- **Scopo** : Ordinamento di un array di numeri interi
- **Specifiche:**
 - array di interi "interi[MAX]"
 - void insertion(int interi[MAX], int tot);
- **Descrizione** : L'insertion sort è un algoritmo molto efficiente per ordinare pochi numeri. Questo algoritmo è simile al modo che si potrebbe usare per ordinare un mazzo di carte...
- **Lista dei Parametri**
- *Input:*
 - interi[]: vettore contenente gli elementi da ordinare
 - tot numero degli elementi contenuti nel vettore
- *Output:* array interi[] ordinato in ordine crescente.

back X next

Federica 16/10/2008 14 Facoltà di Scienze Matematiche Fisiche Naturali

Documentazione per Insertion Sort

Complessità di Tempo

- L'algoritmo inserisce il componente **interi[i]** nel vettore già ordinato di componenti **interi[0]...interi[i-1]** spostando di una posizione tutti i componenti che seguono quello da inserire.
- Ad ogni passo la procedura compie nel caso peggiore (vettore ordinato al contrario), N-1 confronti, essendo N la lunghezza del vettore corrente e i-1 spostamenti.
- Le operazioni nel caso peggiore sono dunque $(1+2+\dots+N-1)$, cioè, nel caso peggiore la complessità asintotica di tempo è $O(n^2)$. Nel caso migliore (vettore ordinato) bastano N-1 confronti.

back X next

Federica 16/10/2008 15 Facoltà di Scienze Matematiche Fisiche Naturali

Documentazione per Insertion Sort

Complessità di Spazio:
La struttura dati utilizzata per implementare l'algoritmo è un ARRAY monodimensionale, contenente i valori da ordinare, di conseguenza la complessità di spazio è $O(n)$.

Esempi di esecuzione:

- Dati in ingresso i numeri: 20 11 45, si ottiene in uscita: 11 20 45

Risorse:
[Documentazione Insertion Sort](#)
[QuickSort](#)

back X next

Federica 16/10/2008 16 Facoltà di Scienze Matematiche Fisiche Naturali

Heapsort

L'Heapsort è un algoritmo di ordinamento molto efficiente:
Come l'insertion Sort e il Quicksort, l'Heapsort ordina sul posto
Meglio dell'Insertion Sort e del Quicksort, il running time dell'Heapsort è $O(n \log n)$ nel caso peggiore
L'algoritmo di Heapsort basa la sua potenza sull'utilizzo di una struttura dati chiamata Heap, che gestisce intelligentemente le informazioni durante l'esecuzione dell'algoritmo di ordinamento.

back X next

Federica 16/10/2008 17 Facoltà di Scienze Matematiche Fisiche Naturali

Heap

La struttura dati Heap (binaria) è un array che può essere visto come un albero binario completo

Proprietà fondamentale degli Heap è che il valore associato al nodo padre è sempre maggiore o uguale a quello associato ai nodi figli

Un Array A per rappresentare un Heap ha bisogno di due attributi:

- Lunghezza dell'array
- Elementi dell'Heap memorizzati nell'array

back X next

Federica 16/10/2008 18 Facoltà di Scienze Matematiche Fisiche Naturali

Organizzazione dell'array

La radice dell'Heap è sempre memorizzata nel primo elemento dell'array.

Dato un nodo i, il suo nodo padre, figlio sx e dx possono essere calcolati nel modo seguente:

- `int left(int i) { return 2*i+1; }`
- `int right(int i) { return 2*i+2; }`
- `int parent (int i) { return (i-1)/2; }`

N.B. Nel C il primo elemento di un vettore A è A[0]. Nel libro di testo "Algoritmi e Strutture Dati", il primo elemento è invece A[1] e i valori precedenti sono dati dalle seguenti pseudo-funzioni:

- `Parent (i) return [i/2]`
- `Left (i) return 2i`
- `Right(i) return 2i+1`

back X next

Federica 16/10/2008 19

Facoltà di Scienze
Matematiche
Fisiche Naturali

Esempio di Heap

Heap con 10 vertici

0	1	2	3	4	5	6	7	8	9
16	11	9	10	5	6	8	1	2	4

back X next

Federica 16/10/2008 20

Facoltà di Scienze
Matematiche
Fisiche Naturali

Heapify

Una subroutine molto importante per la manipolazione degli Heap è Heapfy.

Questa routine ha il compito di assicurare il rispetto della proprietà fondamentale degli Heap. Cioè, che il valore di ogni nodo non è inferiore di quello dei propri figli.

Di seguito mostriamo una funzione ricorsiva Heapify che ha il compito di far scendere il valore di un nodo che viola la proprietà di Heap lungo i suoi sottoalberi.

back X next

Implementazione di Heapify

```

void Heapify(int A[MAX], int i)
{
    int l,r,largest;
    l = left(i);
    r = right(i);
    if (l < HeapSize && A[l] > A[i])
        largest = l;
    else largest = i;
    if (r < HeapSize && A[r] > A[largest])
        largest = r;

    if (largest != i) {
        swap(A, i, largest);
        Heapify(A, largest);
    }
}

```

back ✖ next

Example of heapify

```

graph TD
    16((16)) --- 4((4))
    16 --- 10((10))
    4 --- 14((14))
    4 --- 7((7))
    14 --- 2((2))
    14 --- 8((8))
    7 --- 1((1))
    10 --- 9((9))
    10 --- 3((3))

```

back ✖ next

Federica 16/10/2008 23

Facoltà di Scienze
Matematiche
Fisiche Naturali

Example of heapify

Fix the heap

4 < 14

back next

Federica 16/10/2008 24

Facoltà di Scienze
Matematiche
Fisiche Naturali

Example of heapify

4 < 8

back next

Federica 16/10/2008 25 Facoltà di Scienze Matematiche Fisiche Naturali

Example of heapify

```

graph TD
    16((16)) --- 14((14))
    16 --- 10((10))
    14 --- 8((8))
    14 --- 7((7))
    8 --- 2((2))
    8 --- 4((4))
    7 --- 1((1))
    10 --- 9((9))
    10 --- 3((3))
  
```

back X next

Federica 16/10/2008 26 Facoltà di Scienze Matematiche Fisiche Naturali

Costruire un Heap

La seguente procedura serve a costruire un Heap da un array:

```

void BuildHeap(int A[MAX])
{
  int i;
  HeapSize = ArraySize;
  for (i=ArraySize/2; i>=0; i--)
    Heapify(A, i);
}
  
```

Risorse:
[Esempio di Costruzione dell'heap](#)

back X next



Federica

16/10/2008



27

Facoltà di Scienze
Matematiche
Fisiche Naturali

Funzione HeapSort

```
void HeapSort(int A[MAX])
{
  int i;
  BuildHeap(A);
  for (i=ArraySize-1; i>=1; i--) {
    swap(A, 0, i);
    HeapSize--;
    Heapify(A, 0);
  }
}
```

Risorse:

[Esempio di HeapSort](#)

back
✖
next



Federica

16/10/2008



28

Facoltà di Scienze
Matematiche
Fisiche Naturali

Simulazione

back
✖
next

Federica 16/10/2008 29 Facoltà di Scienze Matematiche Fisiche Naturali

Algoritmo di HeapSort

```

#include <stdlib.h>
#define MAX 20
int ArraySize, HeapSize, tot;
int left(int i) { return 2*i+1;}
int right(int i) { return 2*i+2;}
int p(int i) {return (i-1)/2;}
void swap(int A[MAX], int i, int j)
    {int tmp = A[i];
     A[i] = A[j];
     A[j] =tmp;}
void Heapify(int A[MAX], int i);
void BuildHeap(int A[MAX]);
void HeapSort(int A[MAX]);

```

back ✖ next

Federica 16/10/2008 30 Facoltà di Scienze Matematiche Fisiche Naturali

Main di HeapSort

```

main(){
int A[MAX], k;
printf("\ nQuanti elementi deve contenere l'array: ");
scanf("%d",&tot);
while (tot>MAX)
    { printf("\ n max 20 elementi: "); scanf("%d",&tot);}
for (k=0;k<tot;k++) {
    printf("\ nInserire il %d° elemento: ",k+1);
    scanf("%d",&A[k]); }
HeapSize=ArraySize=tot;
HeapSort(A);
printf("\ nArray Ordinato:");
for (k=0;k<tot;k++)
    printf(" %d",A[k]);
}

```

back ✖ next

Federica 16/10/2008 31 Facoltà di Scienze Matematiche Fisiche Naturali

Complessità

- Il running time di Heapify è $O(h)$ dove h è l'altezza dell'Heap. Siccome l'heap è un albero binario completo, il running time è $O(\log n)$. Più in dettaglio la sua complessità è la soluzione della ricorrenza $T(n) \leq T(2n/3) + \Theta(1)$ utilizzando il master method (caso 2).
- BuildHeap fa $O(n)$ chiamate a Heapify. Per cui il running time di BuildHeap è sicuramente $O(n \log n)$. Si noti che le chiamate a Heapify avvengono su nodi ad altezza variabile minore di h . Da un'analisi dettagliata, risulta che il running time di Heapify è $O(n)$.
- Heapsort fa $O(n)$ chiamate a Heapify. Dunque il running time di Heapsort è $O(n \log n)$.
- La complessità di spazio di Heapsort è invece $O(n)$, visto che oltre il vettore di input necessita solamente di un numero costante di variabili per implementare l'algoritmo.

back X next

Federica 16/10/2008 32 Facoltà di Scienze Matematiche Fisiche Naturali

Code di Priorità

Le code di priorità rappresentano una delle applicazioni più efficienti della struttura dati Heap.

Una coda di priorità è una struttura dati utilizzata per mantenere un insieme S di elementi, a ciascuno associato un valore chiamato "chiave".

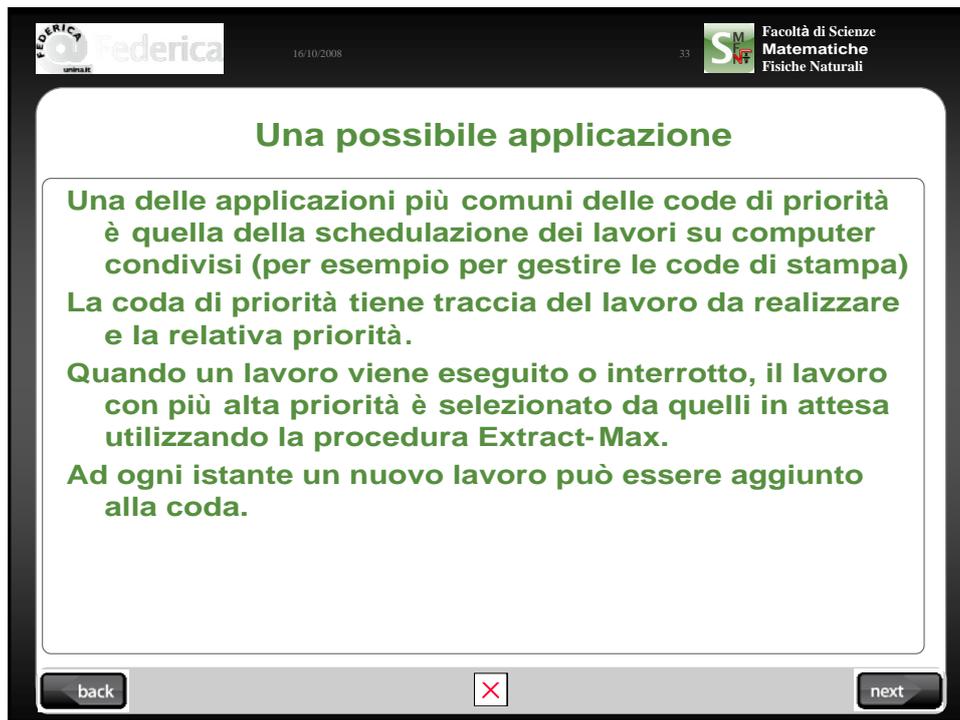
Una coda di priorità supporta le seguenti operazioni

Insert(S, x): Inserisce l'elemento x nell'insieme S .

Maximum(S): Restituisce l'elemento di S con la chiave più grande.

Extract-Max(S): Rimuove e ritorna l'elemento di S con la chiave più grande.

back X next



The image shows a presentation slide with a black border. At the top left, there is a logo for 'Federica' with the URL 'www.federica.it' below it. In the center top, the date '16/10/2008' is displayed. At the top right, there is a logo for 'S.M.F.' (Scienze Matematiche e Fisiche Naturali) and the text 'Facoltà di Scienze Matematiche Fisiche Naturali'. The slide title is 'Una possibile applicazione' in green. The main text is in green and describes the application of priority codes in shared computer work scheduling. At the bottom, there are three buttons: 'back', a red 'X' icon, and 'next'.

Federica
www.federica.it

16/10/2008

S.M.F. Facoltà di Scienze
Matematiche
Fisiche Naturali

Una possibile applicazione

Una delle applicazioni più comuni delle code di priorità è quella della schedulazione dei lavori su computer condivisi (per esempio per gestire le code di stampa)

La coda di priorità tiene traccia del lavoro da realizzare e la relativa priorità.

Quando un lavoro viene eseguito o interrotto, il lavoro con più alta priorità è selezionato da quelli in attesa utilizzando la procedura Extract-Max.

Ad ogni istante un nuovo lavoro può essere aggiunto alla coda.

back X next

This document was created with Win2PDF available at <http://www.win2pdf.com>.
The unregistered version of Win2PDF is for evaluation or non-commercial use only.
This page will not be added after purchasing Win2PDF.