

Laboratorio di algoritmi e strutture dati

Docente: Violetta Lonati

Heap e code di priorità

1 Esercizio: implementazione di code di priorità con heap

Realizzate la struttura di dati *codice di priorità* dichiarando e implementando questa *interfaccia*:

```
typedef struct pqueue *Pqueue;  
  
/* crea una coda di priorità vuota che potrà contenere al massimo n Item */  
Pqueue pqueue_new( int n );  
  
/* distrugge la coda di priorità */  
void pqueue_destroy( Pqueue );  
  
/* restituisce la lunghezza della coda di priorità */  
int pqueue_length( Pqueue );  
  
/* inserisce l'Item nella coda di priorità */  
void pqueue_insert( Pqueue, Item );  
  
/* estrae dalla coda di priorità l'Item con chiave minima */  
Item pqueue_extractmin( Pqueue );  
  
/* restituisce l'Item con chiave minima nella coda di priorità */  
Item pqueue_min( Pqueue );
```

Implementate queste funzioni usando la struttura dati *heap*.

Dovrete innanzitutto definire il tipo **struct** `pqueue`:

```
typedef Item *Heap;  
  
struct pqueue {  
    Heap h;  
    int size, count;  
};
```

Inoltre sarà utile definire alcune funzioni ausiliarie (ad esempio `heapify_up` o `heapify_down`).

Applicazione: ordinamento tramite codice di priorità

Usate le funzioni dichiarate in per realizzare la funzione

```
pqueue_sort( Item a[], int l, int r );
```

che ordina la porzione compresa tra gli indici `l` e `r` dell'array `a`. Potete usare il seguente algoritmo:

```
crea una nuova coda di priorità Q
inserisci in Q un elemento di a alla volta
finchè Q non è vuota
  estrai il minimo m da Q
  stampa m
```

Applicazione: heapsort

Completate l'esercizio con una funzione `heapsort` che ordina un array di interi usando uno heap e con la funzione `main` che testa il funzionamento di `heapsort`.

Cosa cambia rispetto all'applicazione precedente?

[Facoltativo] Altre implementazioni

Provate ad implementare la coda di priorità in uno dei modi seguenti (meno efficienti!) e a confrontarne il funzionamento:

- array (o lista) + puntatore al minimo;
- lista ordinata;
- array ordinato.

2 Esercizio: merging di k liste ordinate di n/k elementi

Usate la struttura dati *coda di priorità* (implementata efficientemente tramite heap) per risolvere il seguente problema in tempo $O(n \log k)$:

Input k liste ordinate L_0, L_1, \dots, L_{k-1} , ciascuna con n/k elementi;

Output un'unica lista ordinata L contenente gli n elementi delle liste L_0, L_1, \dots, L_{k-1} .

Suggerimento: non è necessario copiare gli elementi delle varie liste, basta modificare man mano i puntatori `next` in modo da costruire la nuova lista. Per mantenere l'ordinamento, è opportuno mantenere un puntatore alla coda delle lista finale, in modo da aggiungere i nuovi nodi in coda.

