UNIMORE
UNIVERSITÀ DEGLI STUDI DI
MODENA E REGGIO EMILIA

Dispense del Corso di Laboratorio di Fondamenti di Informatica II e Lab

Esercitazione 10: Heap

Ultimo aggiornamento: 22/05/2019

Introduzione 1/2

- Definito il tipo `element`: `typedef int element;`
- Sono date le implementazioni delle seguenti primitive:

```
int Left(int i);
int Right(int i);
int Parent(int i);
int Compare(const element *e1, const element *e2);
void Swap(element *e1, element *e2);
void MoveUpMaxHeap(element *h, int i);
void MoveDownMaxHeap(element *h, int i);
element *CreateHeap();
void FreeHeap(element *h);
```

Introduzione 2/2

- Trovate le dichiarazioni e le rispettive implementazioni delle primitive sopra elencate nei file `heap_int.h` e `heap_int.c` su dolly.
- Al link http://imagelab.ing.unimore.it/heap/heap__int_8h.html trovate la loro documentazione, che vi aiuterà nella risoluzione degli esercizi.

HEAP: Heapify

- Esercizio 1 (Heapify):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
void HeapifyMaxHeap(element *h);
```

La funzione `Heapify` prende in input un vettore `h` di element e lo converte in una coda (max)heap. Si testi la funzione con un opportuno `main` di prova.

HEAP: Primitive Ricorsive 1/2

- Esercizio 2 (MoveUpRec):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
void MoveUpMaxHeapRec(element *h, int i);
```

La funzione prende in input una coda (max)heap `h` e l'indice di un nodo `i` e deve spostare il nodo i-esimo verso l'alto, ovvero scambiarlo con il padre, fino a quando l'ordinamento non è rispettato. A differenza della primitiva `MoveUpMaxHeap`, la funzione deve essere ricorsiva. Si testi la funzione con un opportuno `main` di prova.

HEAP: Primitive Ricorsive 2/2

- Esercizio 3 (MoveDownRec):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
void MoveDownMaxHeapRec(element *h, int i);
```

La funzione prende in input una coda (max)heap `h` e l'indice di un nodo `i` e deve spostare il nodo i-esimo verso il basso, ovvero scambiarlo con il figlio maggiore, fino a quando l'ordinamento non è rispettato. A differenza della primitiva `MoveDownMaxHeap`, la funzione deve essere ricorsiva. Si testi la funzione con un opportuno `main` di prova.

HEAP: Push

- Esercizio 4 (Push):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
element *PushMaxHeap(const element *new_e, element *h);
```

La funzione PushMaxHeap prende in input un nuovo elemento `new_e` e una coda (max)heap e aggiunge il nuovo elemento alla coda heap, allocando opportunamente la memoria e assicurandosi che la proprietà della (max)heap siano rispettate. La funzione deve restituire il puntatore alla nuova heap. Si testi la funzione con un opportuno `main` di prova.

HEAP: Pop

- Esercizio 5 (Pop):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
element *PopMaxHeap(element *h, element *popped);
```

La funzione PopMaxHeap prende in input una coda (max)heap e un puntatore ad `element` (popped). La funzione deve estrarre l'elemento massimo dalla heap, deallocando opportunamente la memoria e assicurandosi che la proprietà della (max)heap siano rispettate. La funzione deve restituire il puntatore alla nuova heap e salvare l'elemento estratto in popped. Si testi la funzione con un opportuno `main` di prova.

HEAP: Heapsort

- Esercizio 6 (Heapsort):

Nel file `heap.c` si implementi la definizione della seguente funzione:

```
void HeapsortMaxHeap(element *h);
```

La funzione `HeapsortMaxHeap` prende in input una coda (max)heap e la trasforma in modo tale che al termine dell'esecuzione l'array sia ordinato in maniera decrescente. Si testi la funzione con un opportuno `main` di prova.

N.B. Non bisogna usare algoritmi di ordinamento ma sfruttare le proprietà della heap.

Al termine dell'esecuzione il vettore h sarà ancora una coda heap?

HEAP: Min Heap

- Esercizio 7 (MinHeap):

Si modifichino le primitive fornite per le (max)heap di interi in modo tale che diventino primitive per le (min)heap di interi. Si risolvano quindi gli esercizi precedenti facendo riferimento alle (min)heap. In questo caso la `Pop` dovrò estrarre l'elemento minimo dalla heap, l'`Heapify` dovrà costruire una (min)heap, e la `Heapsort` dovrà ordinare in ordine crescente.