Min Heap Library

Libreria per la gestione e l'utilizzo di min Heap sviluppata dal gruppo 11 composto da Francesco Borrelli, Alessandro Grieco e Camilla Zampella durante il corso di Laboratorio di Algoritmi e Strutture Dati dell' a.a. 2017/2018

1 SOMMARIO

2 Definizione di HEAP 1
2.1 Definizione di MIN HEAP 2
3 Operazioni di base 2
4 Funzioni offerte dalla libreria 2
4.1 Funzioni semplici 2
4.1.1 Descrizione funzioni semplici 2
4.2 Funzioni complesse 3
4.2.1 Descrizione funzioni complesse 3
4.3 heapSort 3
4.3.1 Descrizione di heapSort 3
5 Esempio d'utilizzo della libreria 3

2 DEFINIZIONE DI HEAP

Un heap è un albero binario pieno parzialmente ordinato in quanto , definita una relazione d'ordine, il padre è in relazione con i figli ma i figli possono non essere in relazione tra loro.

In questo modo si ha sempre il massimo della relazione d'ordine come radice.

Per ottenere un tempo d'accesso ottimale si definisce un attributo (heapsize) che contiene il numero di elementi attualmente presenti nell'heap e ,avendo nozione dell'altezza dell'albero, un visita lungo un percorso costa $\Theta(log(heapsize))$.

Un heap è rappresentabile come albero binario o come array dato che è possibile definire un indice di posizione di un elemento al suo interno.

Viene utilizzato in genere per le code di priorità (es. code di stampa).

2.1 DEFINIZIONE DI MIN HEAP

Un min heap è un heap che ha come relazione d'ordine la relazione **<** (N.B. non **≤** poiché all'interno di una coda di priorità non possono esserci due elementi con la stessa priorità), quindi la sua radice sarà sempre l'elemento con minore priorità e, per ogni nodo, il padre sarà sempre minore dei figli.

Paradossalmente l'operazione max (si veda il capitolo successivo) ritornerà l'elemento con la minima priorità e dunque il minimo.

3 OPERAZIONI DI BASE

Le operazioni basilari che permettono il corretto funzionamento dell'heap sono definite di seguito:

- Max : ritorna il massimo della relazione d'ordine (accessibile in tempo costante per definizione)
- Left : ritorna l'indice di posizione del figlio sinistro del nodo preso in esame
- Right : ritorna l'indice di posizione del figlio destro del nodo preso in esame
- Parent : ritorna l'indice di posizione del padre del nodo preso in esame

Queste operazioni sono utilizzate all'interno delle funzioni offerte dalla libreria ma non utilizzabili dall'esterno.

4 FUNZIONI OFFERTE DALLA LIBRERIA

Per semplicità si distinguono le funzioni offerte dalla libreria in semplici e complesse come segue

4.1 FUNZIONI SEMPLICI

Sono funzioni ordinarie per la gestione di un heap:

```
int isEmpty(heap *h);
int size(heap *h);
int min(heap *h);
void insert(heap *h,int k);
void delete(heap *h,int k);
```

4.1.1 Descrizione funzioni semplici

- isEmpty: ritorna 0 se l'heap è vuoto, 1 altrimenti
- size: ritorna un intero che rappresenta il numero di elementi presenti nell'heap

- min: ritorna l'elemento di minima priorità (nel caso di min heap il massimo dell'ordinamento) oppure 0 se l'heap è vuoto. Si consiglia di verificare prima se l'heap contiene almeno un elemento utilizzando il metodo isEmpty
- insert: inserisce l'intero k all'interno dell'heap
- delete: rimuove l'intero k dall'heap

4.2 FUNZIONI COMPLESSE

Funzioni utili per la gestione di un heap:

```
heap* buildHeap(int* data, int dim);
void freeheap(heap *h);
void printHeap(heap *h);
```

4.2.1 Descrizione funzioni complesse

- buildHeap: data una collezione di interi in ingresso e la sua dimensione, costruisce un heap e lo ritorna
- freeheap: libera la memoria dinamica allocata dall'heap cancellandolo in maniera corretta
- printHeap: stampa tutti gli elementi dell'heap in base alla loro posizione all'interno dello stesso

4.3 HEAPSORT

È una funzione di ordinamento che non spreca memoria aggiuntiva e che ha complessità asintotica pari a $\Theta log(N)$ dove N è la dimensione della sequenza di ordinare data in ingresso.

Sfrutta le proprietà dell'heap ed è definita come segue

```
void heapSort(int *data, int dim);
```

4.3.1 Descrizione di heapSort

Ha come parametri d'ingresso la sequenza di interi da ordinare e la sua dimensione.

Dato che la sequenza è passata per riferimento la funzione in oggetto permette di modificare gli elementi della sequenza in modo da ordinarli in senso crescente.

Non c'è bisogno di creare un heap per poterla utilizzare in quanto questa operazione è svolta stesso all'interno del metodo e questo permette di poter utilizzare l'ordinamento tramite heap senza averne alcuna nozione.

5 ESEMPIO D'UTILIZZO DELLA LIBRERIA

Nella libreria proposta vi è anche un esempio di utilizzo della stessa, composto da un menù principale che guida l'utente alla creazione di un heap e da un menù secondario che permette di svolgere operazioni su di esso.