**Min Heap Library**

Libreria per la gestione e l’utilizzo di Tableau di Young 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

Sommario

[2 Descrizione del problema 1](#_Toc479162488)

[2.1 Tableau di Young 2](#_Toc479162489)

[3 Strategie di risoluzione del problema e descrizione delle strutture dati utilizzate 2](#_Toc479162490)

[3.1 Relazione di figlio destro 2](#_Toc479162491)

[3.2 Relazione di figlio sinistro 2](#_Toc479162492)

[3.3 Relazione di padre 2](#_Toc479162493)

[4 Dettagli implementativi 2](#_Toc479162494)

[4.1 Funzioni semplici 2](#_Toc479162495)

[4.1.1 Descrizione funzioni semplici 3](#_Toc479162496)

[4.2 Funzioni complesse 3](#_Toc479162497)

[4.2.1 Descrizione funzioni complesse 3](#_Toc479162498)

[4.3 HeapSort 3](#_Toc479162499)

[4.3.1 Descrizione di heapSort 3](#_Toc479162500)

[5 Descrizione Complessità 4](#_Toc479162501)

[6 Manuale d’uso applicazione 4](#_Toc479162502)

# Descrizione del problema

Si richiede di creare una libreria per la gestione di Tableaux di Young e, sfruttando le sue proprietà, fornire all’utente della libreria un’ulteriore algoritmo di ordinamento basato su questa struttura.

## Tableau di Young

Un Tableau di Young è una griglia (matrice) di dimensioni n x m dove n è il numero delle righe e m è il numero delle colonne che compongono la tabella.

Per ogni riga e per ogni colonna gli elementi sono ordinati in senso crescente rispettivamente da sinistra verso destra e dall’alto verso il basso.

È facile notare che il minimo tra gli elementi inseriti si trova sempre nella posizione in alto a sinistra.

# Strategie di risoluzione del problema e descrizione delle strutture dati utilizzate

Analizzando la struttura di un Tableau, si è giunti alla conclusione che per i primi elementi (quindi fino all’antidiagonale maggiore) è possibile ricavare un albero binario pieno dal Tableau definendo le seguenti relazioni :

## Relazione di figlio destro

Dati x, y elementi del Tableau, y è figlio destro di x se e solo se x si trova a sinistra rispetto ad y e sono adiacenti.

## Relazione di figlio sinistro

Dati x, y elementi del Tableau, y è figlio destro di x se e solo se x si trova in alto rispetto ad y e sono adiacenti.

## Relazione di padre

Dati x, y elementi del Tableau, x è padre di y se e solo se si trova in alto o a sinistra di y e sono adiacenti.

Si noti che tutti gli elementi che non si trovano sulla prima riga o sulla prima colonna hanno due padri. Considerando una porzione del Tableau tale che uno degli elementi abbia due padri si ha una situazione del tipo

|  |  |
| --- | --- |
| A | B |
| C | D |

Dove A < D, D < B, D < C e B < C

L’unico elemento scambiabile con D al fine di ripristinare le proprietà del Tableau è C , poiché se si scambiasse D con B la seconda riga sarebbe ordinata in senso decrescente da destra verso sinistra non rispettando la definizione di Tableau.

È possibile quindi definire un unico padre per ogni elemento.

## Inserimento di un elemento

Ai fini di poter sfruttare la stessa idea di un inserimento in un heap è necessario stabilire un ordine di posizioni nelle quali inserire l’elemento, in modo da tenere traccia della prossima posizione nella quale inserire e quindi accedervi in tempo costante.

È possibile dividere il tableau in partizioni definendo un indice che indica la riga dal quale tracciare l’antidiagonale, chiamato indice antidiagonale. Questo indice viene incrementato ogni qual volta si riempie una partizione e, una volta superato il numero di righe, indicherà da quale cella dell’ultima riga si tracciano le antidiagonali per la parte triangolare inferiore della matrice. La cella è calcolabile in maniera costante in quanto è rappresentata dalla differenza tra l’indice antidiagonale e il numero di righe totali.

Adesso è possibile sfruttare un algoritmo simile a quello dell’heap, che inserisce un elemento all’ultima foglia e, confrontandolo e scambiandolo con il padre se necessario.

## Estrazione elemento minimo

# Dettagli implementativi

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

## Funzioni semplici

Sono funzioni ordinarie per la gestione di un heap:



### 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

## Funzioni complesse

Funzioni utili per la gestione di un heap:



### 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

## HeapSort

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

Sfrutta le proprietà dell’heap ed è definita come segue



### 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.

# Descrizione Complessità

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.

# Manuale d’uso applicazione