**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 2](#_Toc479252387)

[2.1 Tableau di Young 2](#_Toc479252388)

[3 Strategie di risoluzione del problema e descrizione delle strutture dati utilizzate 2](#_Toc479252389)

[3.1 Relazione di figlio destro 2](#_Toc479252390)

[3.2 Relazione di figlio sinistro 2](#_Toc479252391)

[3.3 Relazione di padre 2](#_Toc479252392)

[3.4 Inserimento di un elemento 3](#_Toc479252393)

[3.5 Estrazione elemento minimo 3](#_Toc479252394)

[3.6 Young sort 4](#_Toc479252395)

[3.7 Descrizione della struttura dati rappresentante il tableau 4](#_Toc479252396)

[4 Dettagli implementativi 4](#_Toc479252397)

[4.1 Funzioni semplici 4](#_Toc479252398)

[4.1.1 Descrizione funzioni semplici 4](#_Toc479252399)

[4.2 Funzioni complesse 4](#_Toc479252400)

[4.2.1 Descrizione funzioni complesse 5](#_Toc479252401)

[4.3 HeapSort 5](#_Toc479252402)

[4.3.1 Descrizione di heapSort 5](#_Toc479252403)

[5 Descrizione Complessità 5](#_Toc479252404)

[6 Manuale d’uso applicazione 5](#_Toc479252405)

# 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 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 e che quell’elemento violi le proprietà, si ha una situazione del tipo

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

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

Da queste relazioni risulta evidente che l’elemento D viola le proprietà di D.

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 indichi la riga dalla quale tracciare l’antidiagonale, che verrà detto “indice antidiagonale”. Questo indice verrà incrementato ogni qual volta si riempie una partizione e, una volta superato il numero di righe, indicherà da quale cella dell’ultima riga tracciare le antidiagonali per la parte triangolare inferiore della matrice. Una volta superati gli elementi all’interno del Tableau la posizione dalla quale tracciare l’antidiagonale verrà calcolata in tempo costante in quanto è rappresentata dalla differenza tra l’indice antidiagonale e il numero di righe totali.

È quindi possibile sfruttare un algoritmo di inserimento simile a quello utilizzato per gli heap, che inserisce un elemento all’ultima foglia e, confrontandolo e scambiandolo con il padre se necessario, ripristina le proprietà della struttura.

L’inserimento inoltre garantisce sempre l’aggiornamento degli indici che aiutano nell’individuazione della posizione del prossimo elemento da inserire.

Dalle proprietà elencate precedentemente si evince che ogni partizione individuata su di un’antidiagonale rappresenta un livello dei nodi dell’albero.

## Estrazione elemento minimo

Come detto in precedenza, il minimo si trova nella cella di posizione (0;0) del Tableau.

Verrà infatti l’elemento in quella posizione ed inserito il valore che si trova nella posizione dell’ultimo elemento inserito. Dopodiché verrà chiamata una funzione simile ad heapify per gli heap (chiamata tableaufy) che si occupa di ripristinare le proprietà tra gli elementi del Tableau.

Anche l’estrazione del minimo garantisce l’aggiornamento degli indici utili per individuare la posizione del prossimo elemento da inserire.

## Young sort

È l’algoritmo di ordinamento che sfrutta una tabella di Young costruita utilizzando proprietà, relazioni e operazioni descritti in precedenza.

Creando una tabella di Young da una sequenza ed estraendo il minimo da essa fino a svuotarla, è possibile riordinare la sequenza in senso crescente.

## Descrizione della struttura dati rappresentante il tableau

Di seguito la struttura dati utilizzata per rappresentare il tableau di Young



### TABLEAU.DATA

Questo campo della struttura è un puntatore ad una matrice di interi e consiste nella rappresentazione degli elementi del tableau.

### TABLEAU.PROPERTIES

Array di interi che rappresentano le caratteristiche del tableau ai fini di ottenere una complessità asintotica efficiente. Di seguito un elenco che descrive le componenti di tale array:

1. properties[0] : rappresenta il numero di righe del tableau
2. properties[1] : rappresenta il numero di colonne del tableau
3. properties[2] : rappresenta il numero di elementi presenti attualmente nel tableau
4. properties[3] : indice antidiagonale
5. properties[4] : indice di riga della posizione in cui verrà inserito il prossimo elemento
6. properties[5] : indice di colonna della posizione i cui verrà inserito il prossimo elemento

# Dettagli implementativi

Si elencano di seguito le funzioni offerte dalla libreria (e quindi presenti nel file header) e le relative funzioni di supporto.

## Funzioni offerte dalle libreria

Le funzioni offerte dalla libreria sono le seguenti



### createTableau

Funzione utilizzata per la creazione di un Tableau.

Parametri:

* data : la sequenza di elementi da inserire nel tableau
* n : il numero di righe del tableau
* m : il numero di colonne del tableau
* tot : il numero di elementi da inserire inizialmente nel tableau

Ritorna un puntatore ad una struttura di tipo tableau allocata durante la sua esecuzione.

Precondizione : la sequenza deve essere composta da tutti elementi distinti

### extractMin

Funzione utilizzata per l’estrazione dell’elemento minimo dal tableau.

Paramentri:

* t : puntantore ad un tableau allocato in precedenza

Restituisce l’elemento minimo presente nel tableau.

Postcondizione : elimina l’elemento dal tableau e ne ripristina le proprietà.

### insert

### printTableau

### freetableau

### isEmpty

### size

## 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