# Min Young Tableau 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

## 1 SOMMARIO

# 2 DESCRIZIONE DEL PROBLEMA

Si richiede di creare una libreria per la gestione di Tableaux di Young e, sfruttando le proprietà di quest'ultimo, fornire all'utente della libreria un'ulteriore algoritmo di ordinamento.

## 2.1 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.

# 3 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 $\frac{n*m}{2}$ elementi (quindi fino all'antidiagonale maggiore) è possibile ricavare un albero binario pieno definendo le seguenti relazioni :

## 3.1 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.

## 3.2 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.

## 3.3 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.

## 3.4 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 $\frac{n*m}{2}$ 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.

## 3.5 ESTRAZIONE ELEMENTO MINIMO

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

Verrà infatti restituito l'elemento in quella posizione e successivamente sostituito con 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.

## 3.6 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.

## 3.7 DESCRIZIONE DELLA STRUTTURA DATI RAPPRESENTANTE IL TABLEAU

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

```c
typedef struct {
    int properties[6];
    int **data;
} tableau;
```

### 3.7.1 TABLEAU.DATA

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

### 3.7.2 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 in cui verrà inserito il prossimo elemento

# 4 DETTAGLI IMPLEMENTATIVI

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

## 4.1 FUNZIONI OFFERTE DALLE LIBRERIA

Le funzioni offerte dalla libreria sono le seguenti

```c
tableau* createTableau(int *data, int n,int m, int tot);
int extractMin(tableau *t);
void insert(tableau *t,int k);
void printTableau(tableau *t);
void freetableau(tableau *t);
int youngSort(int *sequenza, int n);
int isEmpty(tableau *t);
int size(tableau *t);
```

### 4.1.1 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.

#### 4.1.2 extractMin

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

Parametri:

- t : puntatore ad un tableau allocato in precedenza

Restituisce l'elemento minimo presente nel tableau.

Postcondizione : elimina l'elemento dal tableau, ne decrementa il numero degli elementi presenti, ne ripristina le proprietà e aggiorna gli indici per individuare la posizione del prossimo elemento da inserire.

#### 4.1.3 insert

Funzione per l'inserimento di un elemento nel tableau.

Parametri:

- t : il puntatore al tableau nel quale inserire l'elemento
- k : l'elemento da inserire nel tableau

Precondizione : l'elemento da inserire non deve essere già presente nel tableau.

Postcondizione : l'elemento è inserito nel tableau rispettandone le proprietà, aggiorna la dimensione e gli indici per individuare il prossimo elemento da inserire.

#### 4.1.4 printTableau

Funzione che permette di stampare l'intero tableau sullo standard output.

Parametri:

- t : il puntatore tableau da stampare

#### 4.1.5 freetableau

Funzione che dealloca il tableau e il suo contenuto dalla memoria statica.

Parametri:

- t : il puntatore al tableau da deallocare

#### 4.1.6 isEmpty

Funzione che verifica se nel tableau c'è almeno un elemento

Parametri:

- t : il puntatore al tableau

Ritorna un intero rappresentante un booleano che assume il valore 1 se il tableau è vuoto, 0 altrimenti.

#### 4.1.7 size

Funzione che ritorna un intero rappresentate il numero di valeri effettivamente inseriti nel tableau

Parametri:

- t : il puntatore al tableau

#### 4.1.8 youngSort

Funzione che ordina una sequenza di interi utilizzando un Tableau di Young

Parametri:

- sequenza : la sequenza da ordinare
- n : il numero degli elementi della sequenza

Precondizione : gli elementi della sequenza sono tutti distinti.

Postcondizione : la sequenza data in ingresso risulta ordinata.

### 4.2 LA STRUCT COORDINATES

La struttura coordinates contiene due interi che rappresentano l'indice di riga e di colonna di una posizione all'interno del tableau.

Viene utilizzata dalle funzioni d'appoggio e offre una maggiore leggibilità al codice oltre che a ridurre il numero di parametri delle funzioni.

### 4.3 FUNZIONI DI SUPPORTO

Le funzioni di supporto utilizzate nell'implementazione della libreria sono le seguenti

```
int Dimension(int n);
coordinates* position(int i,int j);
coordinates* parent(tableau *t,coordinates* i);
coordinates* left(tableau *t,coordinates *i);
coordinates* right(tableau *t,coordinates *i);
void tableaufy(tableau *t,coordinates *cs);
void climbTableau(tableau *t,coordinates *cs);
```

#### 4.3.1 Dimension

Ritorna la dimensione ottimale per costruire un tableau in base al numero di elementi.

Parametri:

- n : numero di elementi

### 4.3.2 position
Alloca una struct di tipo coordinates e ne ritorna il puntatore.

Parametri :

- i : indice di riga
- j : indice di colonna

### 4.3.3 parent
Funzione utilizzata per ricavare il padre di un elemento del tableau.

Parametri:

- t : il puntatore al tableau
- i : coordinate dell'elemento del quale si vuole conoscere il padre

Ritorna un puntatore a NULL se non esiste padre o un puntatore ad una struct di tipo coordinates contente gli indici di riga e colonna del padre

### 4.3.4 left / right
Funzioni simili a parent che ritornano le coordinate del figlio destro/sinistro se esiste altrimenti NULL

### 4.3.5 tableaufy
Funzione utilizzata per ripristinare le proprietà del tableau usando il metodo ricorsivo dal padre ai figli

Parametri:

- t : puntatore al tableau da ripristinare
- cs : coordinate del padre dal quale far partire l'algoritmo ricorsivo

Postcondizione : le proprietà del tableau sono ripristinate

### 4.3.6 climbTableau
Funzione simile a tableaufy solo che ripristina le proprietà del tableau usando il metodo ricorsivo dai figli al padre

Parametri:

- t : puntatore al tableau da ripristinare
- cs : coordinate del figlio dal quale far partire l'algoritmo ricorsivo

## 4.4 TABLERROR
È una costante utilizzata all'interno della libreria per indicare un errore attraverso un codice di tipo intero. Di seguito un elenco dei possibili codici d'errore :

- 0 : nessun errore
- -1 : si è provato ad inserire un elemento in un tableau già completamente pieno
- -2 : si è provato ad estrarre il minimo da un tableau vuoto

### 4.5 INT_MAX

La costante INT_MAX presente nella libreria limits.h è stata utilizzata per codificare un elemento vuoto all'interno del tableau.

# 5 DESCRIZIONE COMPLESSITÀ

Analisi delle complessità di tempo e spazio dalle funzioni offerte dalla libreria

### 5.1 SIZE

La complessità in ordine di tempo è costante in quanto il numero di elementi presenti nel tableau è raggiungibile accedendo al secondo elemento dell'array properties della struttura tableau.

### 5.2 ISEMPTY

La complessità in ordine di tempo è costante in quanto confronta se il ritorno di size è 0 o meno

### 5.3 PRINTTABLEAU

La complessità in ordine di tempo è un Θ(n*m) dove n è il numero di righe e m è il numero di colonne allocate per il tableau.

L'algoritmo scorre tutta la matrice stampando i valori degli elementi presenti al suo interno

### 5.4 TABLEAUFY/CLIMBHEAP

Come notato in precedenza è stato possibile suddividere il tableau in partizioni ognuna delle quali rappresenta un livello dell'albero (dove per livello si intende tutti i nodi ad una certa altezza).

Avendo nozione dell'altezza dell'albero rappresentato dal tableau è possibile dimostrare che tableaufy/climbHeap ha un costo asintotico pari ad un Θ(log(n)) dove n è il numero di elementi del tableau.

La dimostrazione segue quella di heapify (vista a lezione) in quanto , definite le relazioni di padre, figlio sinistro e destro, tableaufy/climbHeap segue un unico percorso all'interno dell'albero rappresentato dal tableau.

### 5.5 INSERT

L'algoritmo di inserimento accede in tempo costante alla posizione indicata dagli indici appositi e richiama climbHeap per ripristinare le proprietà del tableau a partire dall'elemento inserito verso la radice.

Inoltre attraverso una serie di operazioni costanti aggiorna gli indici di posizione nell'array properties presente nella struct tableau.

L'esecuzione di insert, quindi, ha un costo asintotico pari a Θ(log(n)) dove n è il numero degli elementi presenti nel tableau.

### 5.6 EXTRACTMIN

L'algoritmo in oggetto estrae il minimo dal tableau in tempo costante dato che è in una posizione nota della matrice, più precisamente nella posizione di coordinate 0 ; 0.

Infine calcola la posizione dell'ultimo elemento inserito in precedenza tramite gli indici appositi, lo sostituisce con il minimo e aggiorna gli indici. Il peso di queste operazioni ha un costo asintotico costante.

Infine, per poter ripristinare le proprietà del tableau dalla radice ai figli, richiama tableaufy che aumenta il costo asintotico ad un Θ(log(n)) dove n è il numero degli elementi presenti nel tableau.

### 5.7 CREATETABLEAU

Il costo asintotico di questa funzione è data dalla chiamata di insert n volte (dove n è il numero degli elementi presenti nel tableau) e dall'inizializzazione degli elementi dell'array properties.

Il suo costo asintotico è quindi un Θ(n*log(n)).

### 5.8 YOUNGSORT

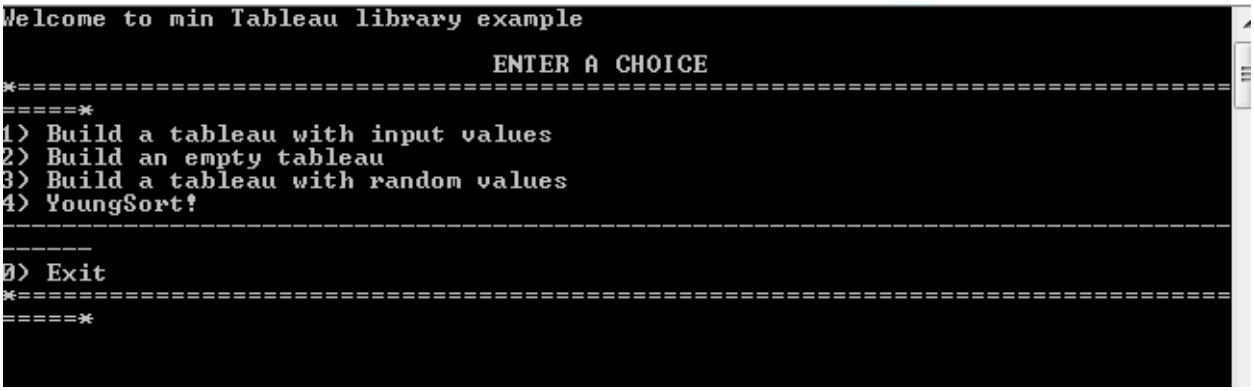
Analizzando i passi dell'algoritmo è facile notare che richiama tante volte createTableau quanti sono gli elementi della sequenza, dopodiché estrae il minimo e lo immette della sequenza finché il tableau non risulta vuoto.

Il suo costo asintotico è quindi di un Θ(n*logn + n*logn), dove n è il numero degli elementi della sequenza.

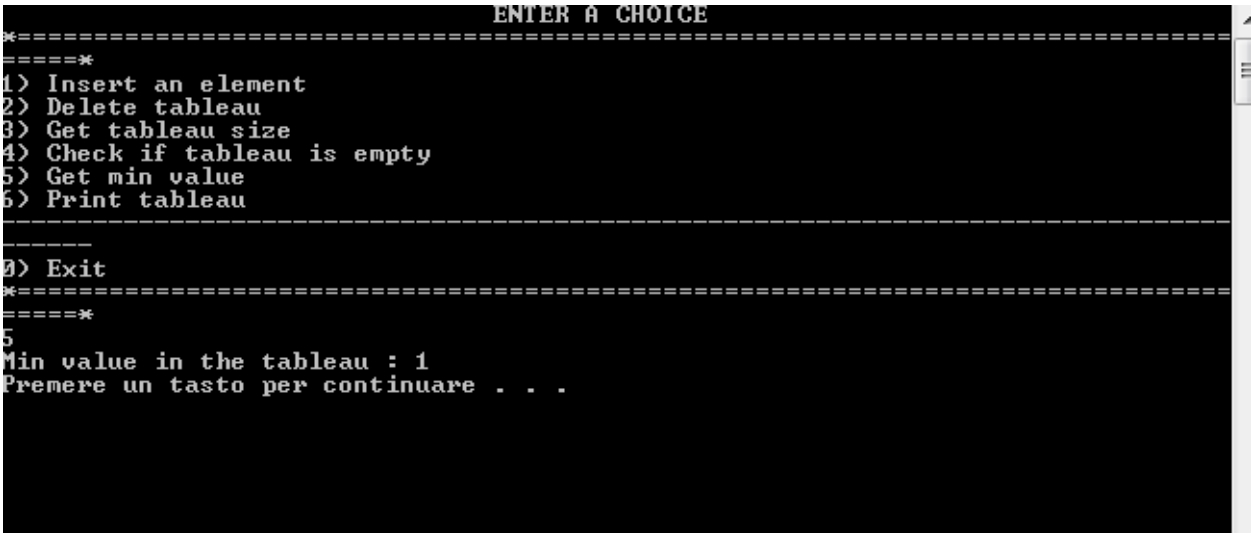
## 6 MANUALE D'USO APPLICAZIONE

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 tableau e da un menù secondario che permette di svolgere operazioni su di esso.

1) Menú principale dove si potrá scegliere la modalitá di creazione del tableau oltre ad effettuare il sort di una sequenza di numeri con l'utilizzo del Tableau di Young come struttura di dati



2) Menú operazionale nel quale è possibile selezionare ed effettuare varie operazioni sul tableau appena creato




# 7 ISTRUZIONI PER LA COMPILAZIONE

Il comando da eseguire per la compilazione del progetto è:

*gcc main.c tableau.h tableau.c -lm*