Geometría Computacional
Práctica 02: segunda parte

Profesor: Adriana Ramírez Vigueras
Ayudante: José Emiliano Cabrera Blancas

13 de marzo de 2014

1 Objetivos:

- Que el alumno implemente algunas de las operaciones básicas que tiene una *DCEL* concretamente:
  - Dado un vértice, regresar una lista de las aristas que son incidentes a este.
  - Dada una cara, regresar una lista de las caras adyacentes a esta.
  - Dada una cara, regresar una lista de las aristas que son adyacentes a esta.
- Que el alumno logre dibujar la *DCEL* en*Ruby-Processing*.

2 Descripción

2.1 Directorio fuente

Como es costumbre, explicaré solamente los directorios nuevos y/o los directorios que tuvieron modificaciones.

Archivos

```
|-- Makefile.rb
|-- dcel
|   |-- dcel.c
|   |-- dcel.h
|-- double_linked_list
|   |-- double_linked_list.c
|   |-- double_linked_list.h
|-- example.c
|-- example.h
|-- face
|   |-- face.c
|   |-- face.h
|-- gui
|   |-- dcel.rb
|-- half_edge
|   |-- half_edge.c
|   |-- half_edge.h
|-- lib
|-- main.c
```

```
|-- points
|   |-- 2d_points.c
|   |-- 2d_points.h
|-- red_black_tree
|   |-- rb_tree.c
|   |-- rb_tree.h
|-- shared_libraries
|   |-- dcel.rb
|   |-- double_linked_list.rb
|   |-- face.rb
|   |-- half_edge.rb
|   |-- points.rb
|   |-- rb_trees.rb
|-- tests
|   |-- dcel_tests.rb
|   |-- half_edge_tests.rb
|   |-- list_tests.rb
|   |-- points_tests.rb
|   |-- rb_tree_tests.rb
|   |-- tests.rb
|-- types
    |-- types.h

11 directories, 30 files
```

- **dcel:** En este directorio se encuentra la estructura que representa una *DCEL* junto con las tres funciones que deben implementar. Recuerden leer los comentarios de los archivos que contiene este directorio, en ellos se mencionan datos importantes sobre como implementar las funciones *incident_he_to_v()*, *incident_f_to_f()* y *incident_he_to_f()* que se les pide en esta práctica.

- **face:** Aquí se encuentra la descripción de la estructura que representa una cara de la *DCEL* y contiene los datos que previamente se explicaron en clase.

- **example.h y example.c:** Contiene una sola función para construir de forma manual una *DCEL* de ejemplo que se les proporciona para depurar su código. En la sección que sigue se explica su contenido e interpretación.

- **gui:** Deberán dibujar los vértices y las aristas de una *DCEL*. Para facilitarles las cosas ya viene implementado el método de *Ruby* que obtiene un arreglo de vértices y otro de aristas, por lo que ustedes solamente deben dibujar los puntos y aristas dados.

- **half_edge:** Se agregaron nuevos parámetros a esta estructura. Es importante que entiendan el propósito de estos parámetros que se explican en su correspondiente cabecera.

- **main.c**: La *DCEL* es una estructura estática que no tiene una función que la construya de forma eficiente. Aquí construyo una *DCEL* que ustedes pueden utilizar para probar las funciones antes de pasarlas por los *tests*. Este archivo cuenta con ejemplos y comentarios para depurar su código, leanlos antes de intentar resolver la práctica.

- **points:** Al igual que la estructura *half_edge*, *points* tuvo pequeñas modificaciones que deberán entender antes de comenzar su práctica.

- **red_black_tree:** Se adaptó esta estructura para que trabaje con aristas, vértices y caras, no necesita ser modificada para resolver su práctica. Se agregó al final la función *rb_tree_to_list()*, útil para implementar una de las tres funciones que se les pide.

3 *DCEL* ejemplo

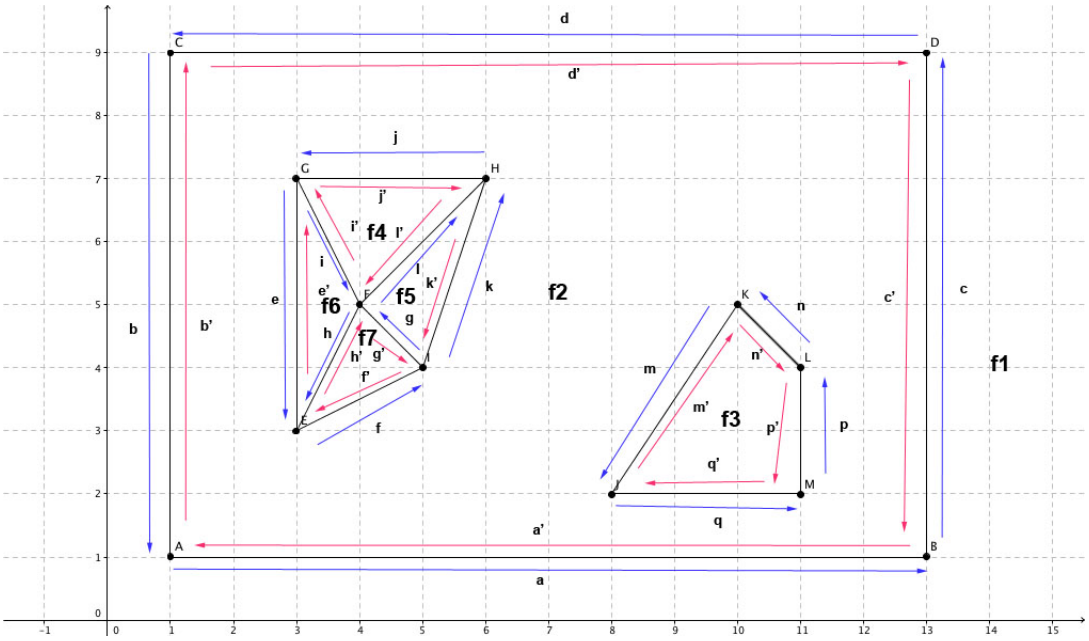

Figura 1

En la figura 1, podemos observar gráficamente la *DCEL* que contruye el archivo *example.c*, y en las tablas siguientes se detalla cada campo que tiene la *DCEL* de ejemplo.

Vértices

name	outer_component	inner_components
$face\ 1$	$NULL$	c
$face\ 2$	c'	$[k, m]$
$face\ 3$	m'	$NULL$
$face\ 4$	j'	$NULL$
$face\ 5$	k'	$NULL$
$face\ 6$	e'	$NULL$
$face\ 7$	f'	$NULL$

Caras

name	x	y	incident_edge
A	1	1	b'
B	13	1	c
C	1	9	d'
D	13	9	d
E	3	3	h'
F	4	5	l
G	3	7	j'
H	6	7	l'
I	5	4	k
J	8	2	q
K	10	5	n'
L	11	4	p'
M	11	2	q'

Recuerden leer el archivo *main.c* para entender como probar sus funciones.

Aristas

name	first	incident_face	twin	next	prev
a	A	$face\ 1$	a'	c	b
a'	B	$face\ 2$	a	b'	c'
b	C	$face\ 1$	b'	a	d
b'	A	$face\ 2$	b	d'	a'
c	B	$face\ 1$	c'	d	a
c'	D	$face\ 2$	c	a'	d'
d	D	$face\ 1$	d'	b	c
d'	C	$face\ 2$	d	c'	b'
e	G	$face\ 2$	e'	f	j
e'	E	$face\ 6$	e	i	h
f	E	$face\ 2$	f'	k	e
f'	I	$face\ 7$	f	h'	g'
g	I	$face\ 5$	g'	l	k'
g'	F	$face\ 7$	g	f'	h'
h	F	$face\ 6$	h'	e'	i
h'	E	$face\ 7$	h	g'	f'
i	G	$face\ 6$	i'	h	e'
i'	F	$face\ 4$	i	j'	l'
j	H	$face\ 2$	j'	e	k
j'	G	$face\ 4$	j	l'	i'
k	I	$face\ 2$	k'	j	f
k'	H	$face\ 5$	k	g	l
l	F	$face\ 5$	l'	k'	g
l'	H	$face\ 4$	l	i'	j'
m	K	$face\ 2$	m'	q	n
m'	J	$face\ 3$	m	n'	q'
n	L	$face\ 2$	n'	m	p
n'	K	$face\ 3$	n	p'	m'
p	M	$face\ 2$	p'	n	q
p'	L	$face\ 3$	p	q'	n'
q	J	$face\ 2$	q'	p	m
q'	M	$face\ 3$	q	m'	p'

4 *Tests*

Para verificar que las funciones que han implementado sirven de forma correcta, debe de pasar el *test* correspondiente a cada función.

Tests

```
dcel.h: incident_he_to_v aristas adyacentes dado un vertice     [FAIL]
dcel.h: incident_f_to_f, caras adyacentes dado una cara         [FAIL]
dcel.h: incident_he_to_f, aristas adyacentes dad una cara       [FAIL]
```

5 ¿Qué debes de programar y dónde comentar tus cambios?

En esta práctica tienes que implementar las funciones *incident_he_to_v()*, *incident_f_to_f()* y *incident_he_to_f()* que vienen declaradas y explicadas en el archivo de cabecera *dcel/dcel.h*. Recuerda primero leer y entender las estructuras *face*, *half_edge* y *vertex* que se agregaron o modificaron para esta práctica. Debes verificar que tu implementación pase los tres *tests* que se te piden y por último terminar de implementar el método *draw_vertex_and_half_edges* declarado en **gui/dcel.rb**, donde este método dibuja los vértices y aristas previamente obtenidos (en el archivo viene explicadas las variables).

Por último recuerda incluir el archivo *README.txt* en la carpeta que contiene al directorio *src*, donde indiquen los cambios que hicieron en la práctica y por qué. Recuerden que es importante que sigan las convenciones de código de *C* que se les pasó al principio del curso. También deben recordar documentar sus estructuras de forma minusiosa y concisa.