PIMO - Grupo 1
Hashing y Grafos

26 de abril de 2016 - Prof: Rodrigo López

Suponga que, como resultado de las pruebas de la función de hashing del ejercicio de la semana anterior, se concluye que para una lista específica de 49 ciudades se van a utilizar el `hash_code` y `compress`, con los parámetros

$$(N, a, b, p) = (37, 67, 59, 73)$$

Es decir, la tabla será de tamaño 37 y los factores empleados en la función `compress` son 67, 59 y 73.

Ahora, dado un archivo de datos (como se describe más adelante) hay que hacer lo siguiente:

- Almacenar la información de cada ciudad en una tabla de hashing que se accederá a partir de las coordenadas de la ciudad.
    - Las colisiones se manejarán con la estrategia de *Encadenamiento Separado*. Es decir, cada *bucket* de la tabla contiene una lista de ciudades a las que les corresponde el mismo valor de hashing.
- Utilizar la misma tabla de hashing para añadir a cada ciudad la información de las ciudades con las que está directamente conectada. Es decir, la tabla de hashing se podrá utilizar como la representación de un grafo mediante lista de adyacencias.

En el archivo de datos hay un renglón por cada ciudad, con formato siguiente:

- Inicia con los datos de la ciudad, como en el laboratorio anterior; es decir, nombre, latitud,longitud,población y religión, separados por coma. A continuación, si la ciudad tiene conexiones directas aparecen dos puntos seguidos de una o más parejas de la forma (latitud-longitud), correspondientes a las coordenadas de las ciudades con las cuales está directamente conectada.

  Un trozo de este archivo puede ser:

```
Tad-Dwieli,35|53/04@N,14|28/35@E,6314,Budistas Zen
Tad-Duluri,35|53/16@N,14|29/26@E,5296,Evangelicos
Tad-Dawl,35|51/03@N,14|28/20@E,3246,Testigos de Jeova: (35|53/04@N-14|28/35@E)
Tac-Cawla,35|51/25@N,14|29/22@E,3647,Budistas: (35|53/16@N-14|29/26@E)
Ta' Xbiex,35|53/57@N,14|29/40@E,4585,Bahas: (35|51/25@N-14|29/22@E)
Ta' Sardina,36|02/45@N,14|17/04@E,1941,Bahas: (35|51/25@N-14|29/22@E) (35|51/03@N-14|28/20@E) (35|53/16@N-14|29/26@E)
```

Todas las conexiones son bidireccionales pero en el archivo solamente aparece una dirección, por lo que en la representación en la tabla de adyacencias hay que completar la dirección faltante.

Finalmente, su programa debe incluir la función

```
def path(coord1,coord2):
```

en la que `coord1` y `coord2` son tuplas de la forma (latitud,longitud), con el formato de latitud y longitud utilizado en este ejercicio. Una invocación típica sería

```
path(('35|53/57@N','14|29/40@E'),('35|51/03@N','14|28/20@E')}.
```

Como resultado de `path(coord1,coord2)`:

- Si hay un camino entre `coord1` y `coord2`, este se imprime como la sucesión de nombres de las ciudades separados por coma.
- Si no hay una ciudad en `coord1` se imprime: `"No hay ciudad en coord1"` (sin las dobles comillas).
- Si no hay camino, se imprime el mensaje `"No hay camino"` (sin las dobles comillas).

Para calcular el camino se debe utilizar el algoritmo de búsqueda por amplitud en el grafo, a partir de `coord1`.

Condiciones de entrega

Esta tarea se puede desarrollar por grupos de hasta 2 personas y tendrá sustentación.