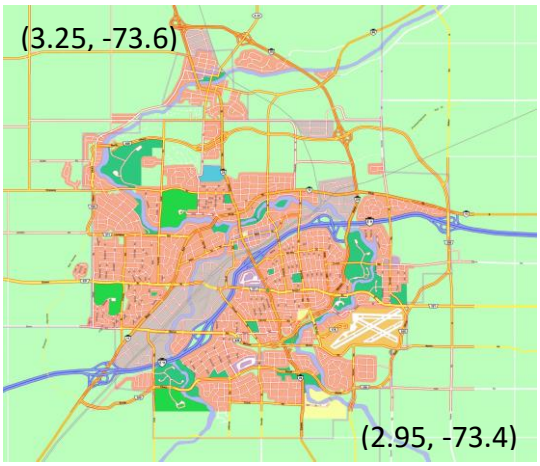
Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Estructuras de datos
Proyecto del curso, 2016-30

1 Descripción del problema

El turismo es un fenómeno económico y social, que contribuye al bienestar de las comunidades en países en desarrollo. Este sector económico ha presentado un rápido crecimiento en los últimos años, incluso superando los niveles de las exportaciones de petróleo y otros productos tradicionales[1].

El problema a tratar en este proyecto de curso es la implementación de un sistema que permita organizar en una estructura de fácil búsqueda los diferentes atractivos turísticos de una ciudad. Adicionalmente, se desea la generación de rutas turísticas que permita unir varios sitios según el tipo de atractivo (ie. Ruta de Arquitectura). En este caso, se trabajará con la ciudad de San Juan de Manacal, la cual se puede ubicar mediante coordenadas en grados decimales (Latitud, Longitud), en el rectángulo formado por los puntos (3.25, -73.6) y (2.95, -73.4).



2 Descripción del proyecto

2.1 Requerimientos del Sistema

Se debe tener en cuenta que los atractivos turísticos están ubicados en un mapa utilizando el sistema de grados decimales, por lo tanto, para calcular una distancia será necesario realizar la conversión de unidades a metros. Adicionalmente, los caminos entre atractivos se construyen usando el algoritmo de *Bowyer-Watson*, del cual se pueden encontrar varias implementaciones.

2.2 Entradas del Sistema

El sistema cuenta con un archivo de texto como entrada que tiene la siguiente estructura:

C	T		
Sitio 0	T_0	Lat_0	Lon_0
Sitio n	T_M	Lat_n	Lon_n

[1] http://www2.unwto.org/es/content/por-que-el-turismo

Donde:

- **C** es el número de lugares turísticos disponibles.
- **T** es el tipo de atractivos disponibles (M).
- **Sitio** (Sitio 0) es el nombre del sitio. Puede ser cualquier cadena de caracteres.
- **Tipo** (T_0) es un código numérico que indica el tipo de atractivo, $0 \leq T < M$
- **Lat** y **Lon** representan las coordenadas del atractivo, ubicadas dentro del rectángulo antes mencionado.

2.3 Interacción con el Sistema

La interfaz de la aplicación a construir debe ser una consola interactiva donde los comandos correspondientes serán tecleados por el usuario. El indicador de línea de comando debe ser el carácter $. Para cada comando, se debe incluir una ayuda de uso que indique la forma correcta de hacer el llamado (i.e. el comando 'ayuda <comando>' debe existir). Cada comando debe presentar en pantalla los mensajes de resultado especificados antes, además de otros mensajes que se consideren necesarios (i.e. errores). Los comandos de los componentes deben ensamblarse en un único sistema (es decir, funcionan todos dentro de un mismo programa, no como programas independientes por componentes).

Componente 1: Configuración

1. **Comando:** cargar "archivoEntrada"
   Salida: "archivoEntrada" [cargado/fallo]
   El Sistema lee el archivo "archivoEntrada" con los datos de entrada.

2. **Comando:** cantSitios [T]
   Salida: El sistema tiene [n] sitios de tipo [T]
   El Sistema informa la cantidad de sitios de un tipo particular. Si no se entrega el parámetro T, retorna la cantidad total de sitios.

3. **Comando:** obtenerSitio x y
   Salida: El sitio turistico mas cercano se encuentra a [n] metros. Su nombre es [Nombre], es de tipo [t] y se ubica en ([Lat], [Lon]).
   El Sistema informa sobre el sitio turístico más cercano a unas coordenadas $\langle x, y \rangle$. Calcula además la distancia en metros lineales entre las coordenadas $\langle x, y \rangle$ y el sitio.

4. **Comando:** crearSitio Nombre t Lat Lon
   Salida: El sitio de tipo [t] en ([Lat, Lon]) ha sido creado.
   El Sistema crea un sitio turístico en las coordenadas dadas.

5. **Comando:** modificarSitio x y NuevoNombre NuevoTipo NuevaLat NuevaLon
   Salida: La informacion del Sitio [NombreAntiguo] se ha actualizado a: [NuevoNombre] de tipo [NuevoTipo], con coordenadas ([NuevaLat], [NuevaLon]).
   El Sistema busca el sitio más cercano a las coordenadas $\langle x, y \rangle$ y modifica sus datos. Los parámetros NuevoNombre, NuevoTipo, NuevaLat y NuevaLon son obligatorios.

6. **Comando:** eliminarSitio x y
   Salida: El sitio [Nombre] en las coordenadas ([Lat], [Lon]) ha sido eliminado.
   El Sistema busca el sitio más cercano a las coordenadas $\langle x, y \rangle$ y lo elimina.

Componente 2: Búsqueda mediante árboles

7. **Comando:** buscarVecinos x y
   Salida: Los vecinos del sitio [Nombre] son: al NE [VecinoNE], al NO [VecinoNO], al SE[VecinoSE] y al SO [VecinoSO].
   El Sistema verifica cuáles son los sitios más cercanos en las 4 coordenadas NE, NO, SE y SO. Se debe tener en cuenta los grados decimales

8. **Comando:** buscarXUbicacion x_{max} y_{max} x_{min} y_{min}
   Salida: Entre las coordenadas ([x_{max}, y_{max}]) y ([x_{min}, y_{min}]) se encuentran los siguientes sitios:
   - [Nombre1] de tipo [t], ubicado en ([Lat1], [Lon1]).
   - [Nombre1] de tipo [t], ubicado en ([Lat2], [Lon2]).
   - **…**

   El Sistema retorna el listado de sitios ubicados entre unas coordenadas particulares.

9. **Comando:** crearPuntoInfo n
   Salida: [N] puntos de información han sido creados:
   - Punto [0] ubicado en ([Lat0], [Lon0]).
   - Punto [1] ubicado en ([Lat1], [Lon1]).
   - …

   El Sistema, teniendo en cuenta una variable *n* (cantidad de puntos de información), crea la mejor distribución de puntos de información posible. Estos puntos de información deben estar ubicados en el centro de gravedad de los sitios turísticos que abarque.

Componente 3: Aplicaciones de Grafos

10. **Comando:** construirCaminos [t]
    Salida: Los caminos han sido creados.
    El sistema usa el algoritmo de construcción de caminos dado (algoritmo de *Bowyer-Watson*). Asigna además una distancia a cada camino.

11. **Comando:** construirRuta x1 y1 x2 y2 [t]
    Salida: La ruta a seguir es:
    - Punto Inicial: El sitio [Nombre] de tipo [t].
    - Camine [M] metros, y llegara al sitio [Nombre2] de tipo [t]
    - …..
    - Punto final: El sitio [Nombre] de tipo [t].

    El Sistema muestra la ruta más corta entre dos puntos dados e imprime los sitios a recorrer. Si el parámetro t está disponible, la ruta sólo puede pasar entre sitios de tipo t.

12. **Comando:** calcularTiempos Nombre1 Nombre2
    Salida: El tiempo promedio entre [Nombre1] y [Nombre2] es:
    - [A] minutos, si se desplaza caminando.
    - [B] minutos, si se desplaza en bicicleta.
    - [C] minutos, si se desplaza en automóvil.

    Suponiendo que las distancias son lineales, y calculando una velocidad promedio caminando de 4.8 km/h, en bicicleta de 21,43 km/h y en carro 23 km/h, el Sistema retorna el tiempo promedio de desplazamiento entre 2 puntos de la ciudad, según su nombre.

3 Evaluación

Las entregas se harán en la correspondiente actividad de UVirtual, hasta la media noche del día anterior al indicado para la entrega. Se debe entregar un archivo comprimido (únicos formatos aceptados: .zip, .tar, .tar.gz, .tar.bz2, .tgz) que contenga dentro de un mismo directorio (sin estructura de carpetas interna) los documentos (único formato aceptado: .pdf) y el código fuente (.h, .hxx, .hpp, .c, .cxx, .cpp). Si la entrega contiene archivos en cualquier otro formato, será descartada y no será evaluada, es decir, la nota definitiva de la entrega será de 0 (cero) sobre 5 (cinco).

3.1 Entrega 1 (miércoles 28 de septiembre de 2016)

Componente 1 completo y funcional. Esta entrega tendrá una sustentación durante la correspondiente sesión de clase. Esta entrega se compone de:

- **(30 %) Documento de diseño**. Debe seguir las pautas de ingeniería: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares. Para la descripción de los TADs, debe seguirse la plantilla definida en clase. Además, se exigirán un(os) esquemático(s) que resuma(n) el(los) problema(s) y su(s) solución(ones) gráficamente.
- **(55 %) Código fuente compilable** en el compilador gnu-g++ (versión 4.0.0, como mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- **(15 %) Sustentación** con participación de todos los miembros del grupo.

3.2 Entrega 2 (viernes 28 de octubre de 2016)

Componente 2 completo y funcional. Esta entrega tendrá una sustentación durante la correspondiente sesión de clase. Esta entrega se compone de:

- **(15 %) Completar la funcionalidad** que aún no haya sido desarrollada de la primera y segunda entregas. Se debe generar un acta de evaluación de la entrega anterior que detalle los elementos faltantes y cómo se completaron para la tercera entrega.
- **(25 %) Documento de diseño**. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares. Para la descripción de los TADs utilizados, debe seguirse la plantilla definida en clase. Además, se exigirá un(os) esquemático(s) que resuma(n) el(los) problema(s) y su(s) solución(ones) gráficamente.
- **(45 %) Código fuente compilable** en el compilador gnu-g++ (versión 4.0.0, como mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- **(15 %) Sustentación** con participación de todos los miembros del grupo.

3.3 Entrega 3 (viernes 25 de noviembre de 2016)

Componente 3 completo y funcional. Esta entrega tendrá una sustentación durante la correspondiente sesión de clase. Esta entrega se compone de:

- **(15 %) Completar la funcionalidad** que aún no haya sido desarrollada de la primera y segunda entregas. Se debe generar un acta de evaluación de la entrega anterior que detalle los elementos faltantes y cómo se completaron para la tercera entrega.
- **(25 %) Documento de diseño**. El documento de diseño debe seguir las pautas de ingeniería que usted ya conoce: descripción de entradas, salidas y condiciones para el procedimiento principal y las operaciones auxiliares. Para la descripción de los TADs utilizados, debe seguirse la plantilla definida en clase. Además, se exigirá un(os) esquemático(s) que resuma(n) el(los) problema(s) y su(s) solución(ones) gráficamente.
- **(45 %) Código fuente compilable** en el compilador gnu-g++ (versión 4.0.0, como mínimo). Este porcentaje de la entrega será un promedio de la evaluación de cada comando.
- **(15 %) Sustentación** con participación de todos los miembros del grupo.