UBA – Facultad de Ciencias Exactas y Naturales – Departamento de Computación

Algoritmos y Estructuras de Datos I

Primer cuatrimestre de 2022

Versión 1.2 (20 de abril de 2022)

Trabajo Práctico de Especificación

Esperando el Bondi

Cambios versión 1.2

1. Ejercicio 8: Cambia nombre **recorridoCubierto** por **recorridoNoCubierto**.
2. Ejercicio 9: Se quita la restricción de la cuadratura de las celdas.

1. Modalidad de Trabajo

- El Trabajo Práctico se realiza en grupo de 4 personas. Todos deben pertenecer a la misma comisión.
- Para aprobar el trabajo se necesita:
  - Que todos los ejercicios estén resueltos.
  - Que las soluciones sean correctas.
  - Que la solución sea declarativa.
  - Que el lenguaje de especificación esté bien utilizado.
  - Que las soluciones sean prolijas: evitar repetir especificaciones innecesariamente y usar adecuadamente las funciones y predicados auxiliares.
- Fechas de Entrega:
  - **Entrega**: *Viernes 29 de Abril - 23:55.*
  - **Recuperatorio**: *Viernes 27 de Mayo - 23:55.*
- Entregable:
  1. Completar la solución de los ejercicios usando LaTeX y siguiendo el *template* dado por la cátedra.
  2. Entregar la versión digital de la resolución en formato **.pdf** a través del campus de cada comisión por cualquiera de los miembros del grupo (realizar **una** sola entrega por grupo).
  3. Cada comisión puede pedir una entrega de medio término (Primera Parte) del TPE con el objetivo de hacer un seguimiento de la resolución del trabajo por parte de cada grupo.

2. Introducción

El punto 11 de los objetivos de desarrollo sostenible definidos a nivel mundial por la ONU en 2015[1] involucra el desarrollo de ciudades sostenibles. En este sentido, un campo de vital importancia para lograrlo es una mejor planificación del transporte urbano, y principalmente, el desarrollo de un sistema de servicios públicos sustentable.

Las llamadas ciudades inteligentes tienen así un interesante objetivo para el diseño de estrategias de mejora que permita reducir el impacto tanto en emisiones de CO2 como en contaminación sonora. Para el primer objetivo, el análisis del estado presente del sistema de transporte constituye un primer paso hacia conocer el problema y poder definir alternativas de mejora.

Con este objetivo, se ubicaron dispositivos GPS en cada uno de los colectivos de línea con el objetivo de mejorar el servicio público de transporte urbano provisto por las empresas privadas. Estos dispositivos registran la posición en longitud y latitud de los colectivos cada una cantidad de segundos T fija ($T = 20$). Estas posiciones son almacenadas localmente y, en caso de tener señal de red disponible, son transmitidas a un control central encargado de registralos en la base de datos respectiva para su posterior procesamiento.

Los equipos de investigación y desarrollo del Departamento de Computación han sido contratados para el análisis de estos datos. En este trabajo, nos proponemos resolver problemas relacionados al mundo del transporte, utilizando los datos de las unidades GPS, con las herramientas que iremos aprendiendo en la materia.

3. Definiciones

Un punto **GPS** estará representado como un par de números reales ($latitud \times longitud$) medidos en grados. Un valor de *latitud* es válido si se encuentra en el rango $[-90.0, 90.0]$; un valor de *longitud* es válido si se encuentra en el rango $[-180.0, 180.0]$.

El tiempo será determinado por un número real que mide la cantidad de segundos transcurridos desde la medianoche (UTC/GMT) del 1/1/1970.

Utilizaremos la palabra **Recorrido** para referirnos a una secuencia de puntos GPS que indican por donde debería ir un colectivo, incluyendo paradas y otros puntos registrados (en orden) por la compañía cada 20 metros aproximadamente, desde el principio hasta el final del trayecto. Por ejemplo, un recorrido es el de la línea `160-ramal-A-ida`.

Cuando hablemos de **Viaje** nos referiremos al trayecto en puntos GPS que indica por dónde fue un colectivo según el dispositivo propio. Estos puntos **no necesariamente están ordenados** pero contienen una marca temporal asociada a cada punto que indica en qué momento fue registrado. Cada viaje está asignado a un recorrido.

Para nuestra especificación utilizaremos los siguientes renombres de tipos:

type $Tiempo = \mathbb{R}$

type $Dist = \mathbb{R}$

type $GPS = \mathbb{R} \times \mathbb{R}$

type $Recorrido = seq\langle GPS \rangle$

type $Viaje = seq\langle Tiempo \times GPS \rangle$

type $Nombre = \mathbb{Z} \times \mathbb{Z}$

type $Grilla = seq\langle GPS \times GPS \times Nombre \rangle$

Para los ejercicios pueden utilizar sin definir el auxiliar **aux dist(p1: *GPS*, p2: *GPS*) : *Dist*** que dados dos puntos GPS calcula la distancia que los separa en metros (ver, por ejemplo, `https://es.wikipedia.org/wiki/F%C3%B3rmula_del_semiverseno`). En caso de utilizar esta auxiliar con valores de GPS fuera de rango, la función se indefine.

4. Ejercicios: especificar los siguientes problemas

4.1. Parte 1

Ejercicio 1. proc viajeValido(in v: *Viaje*, out res : *Bool*) Que devuelve verdadero si los puntos GPS del viaje y los tiempos están en rango.

Ejercicio 2. proc recorridoValido(in v: *Recorrido*, out res : *Bool*) Que devuelve verdadero si los puntos GPS del recorrido están en rango.

[1] `https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/`

Ejercicio 3. proc enTerritorio(in v: *Viaje*, in r: *Dist*, out res : *Bool*) Que chequea que todos los puntos registrados en un viaje válido se encuentren dentro de un círculo de radio r kilómetros.

Ejercicio 4. proc tiempoTotal(in v: *Viaje*, out t : *Tiempo*) Que dado un viaje válido, determine el tiempo total que tardó el colectivo. Este valor debe ser calculado como el tiempo transcurrido desde el primer punto registrado y hasta el último.

Ejercicio 5. proc distanciaTotal(in v: *Viaje*, out d : *Dist*) Que dado un viaje válido, determine la distancia recorrida en kilometros aproximada utilizando toda la información registrada en el viaje, es decir, utilizando la información registrada de todos los tramos.

Ejercicio 6. proc excesoDeVelocidad(in v: *Viaje*, out res : *Bool*) Que dado un viaje válido devuelva verdadero si el colectivo superó los 80 km/h en algún momento del viaje.

4.2. Parte 2

Ejercicio 7. proc flota(in v: $seq\langle$ *Viaje*$\rangle$, in t_0 : *Tiempo*, in t_f : *Tiempo*, out res : $\mathbb{Z}$) Que dada una lista de viajes válidos, calcule la cantidad de viajes que se encontraban en ruta en cualquier momento entre t_0 y t_f inclusives. Por ejemplo, si un viaje comenzó a las 13:30 y terminó a las 14:30 y la franja es de 14:00 a 15:00, el viaje debería estar considerado. Lo mismo ocurre si el viaje comenzó a las 14:10 y terminó a las 14:15 o si comenzó a las 13:30 y terminó a las 16:00.

Ejercicio 8. proc recorridoNoCubierto(in v: *Viaje*, in r: *Recorrido*, in u : *Dist*, out res : $seq\langle$ *GPS*$\rangle$) Que dado un viaje v válido, un recorrido r válido y un umbral u (en kilometros), devuelva todos los puntos del recorrido que no fueron cubiertos por ningún punto del viaje. Se considera que un punto p del recorrido está cubierto si al menos un punto del viaje está a menos de u kilometros del punto p.

Ejercicio 9. proc construirGrilla(in esq1: *GPS*, in esq2: *GPS*, in n: $\mathbb{Z}$, in m: $\mathbb{Z}$, out g: *Grilla*) Que dados dos puntos GPS, construye una grilla de $n \times m$. Estas grillas están conformadas por celdas contiguas rectangulares. Los lados latitudinales (respectivamente longitudinales) de todas las celdas miden la misma cantidad de grados. Cada celda está caracterizada por sus puntos superior izquierdo e inferior derecho (coordenadas GPS) y un *nombre*, que es un par ordenado de enteros que representa la posición de la celda en la grilla. Estos pares ordenados van desde $(1, 1)$ en el punto que se encuentre en la celda que comienza en la posición $esq1$ y hasta (n, m) en la posición en donde se encuentra la celda con esquina $esq2$. La latitud de $esq1$ debe ser mayor a la latitud de $esq2$ y la longitud de $esq1$ debe ser menor a la longitud de $esq2$ Por ejemplo, el panel izquierdo de la Figura 1 muestra un ejemplo de grilla de (3×5).

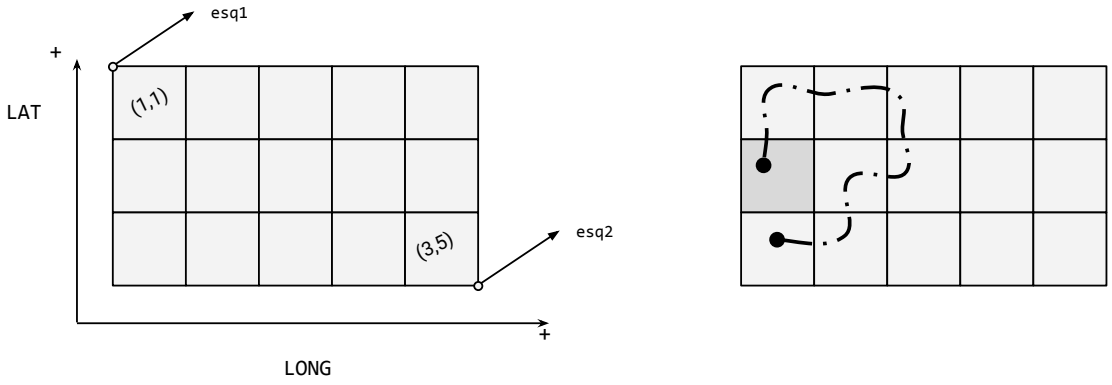
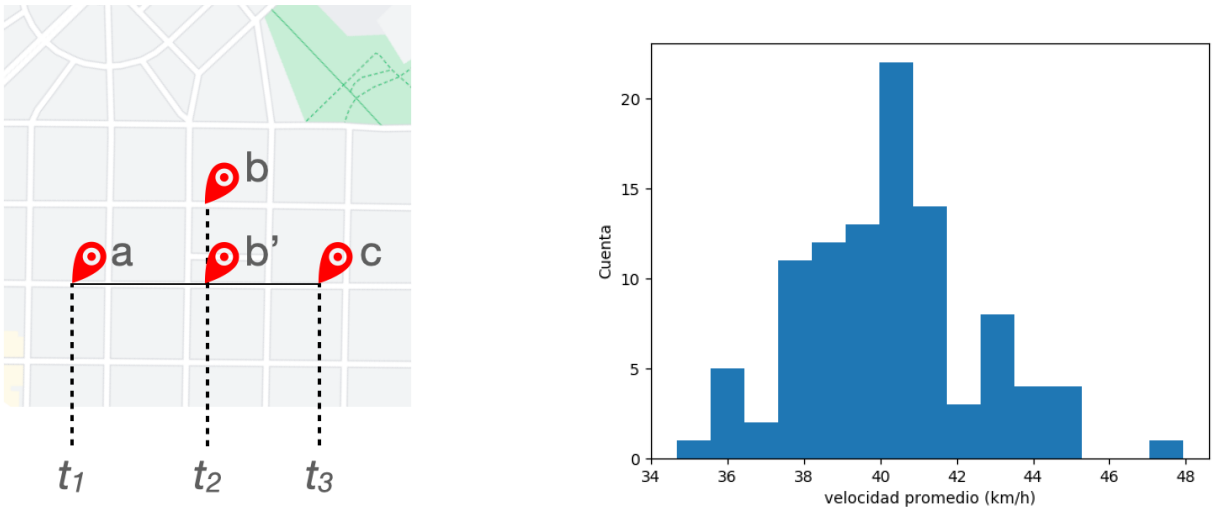


Figura 1: En el lado izquierdo de la figura, un ejemplo de grilla de 3×5. En el panel derecho, un ejemplo de recorrido en donde cada punto simboliza un registro GPS del viaje con comienzo en la celda sombreada.

Ejercicio 10. proc regiones(in r: *Recorrido*, in g: *Grilla*, out res: $seq\langle$ *Nombre*$\rangle$) Que dado un recorrido, devuelve la secuencia ordenada de regiones visitadas por el colectivo[2]. Por ejemplo, en el panel derecho de la Figura 1, puede verse un recorrido que genera la siguiente secuencia: $< (2, 1), (1, 1), (1, 1), (1, 2), (1, 3), (2, 3), (2, 2), (2, 2), (3, 2), (3, 1) >$

[2]Ejemplo real de utilización de grillas: "Real-Time Detection of Anomalous Taxi Trajectories from GPS Traces" `https://www.semanticscholar.org/paper/Real-Time-Detection-of-Anomalous-Taxi-Trajectories-Chen-Zhang/77b7a8fea9ac604af487d262ec21b15e96062713`



(a) *(b)*

Figura 2: *(a)* Ejemplo de viaje corregido (Ejercicio 13). *(b)* Histograma del ejemplo del Ejericio 14.

Ejercicio 11. proc cantidadDeSaltos(in g: *Grilla*, in v: *Viaje*, out res : $\mathbb{Z}$) Que dado un viaje válido y una grilla, determine cuántos saltos hay en el viaje. Diremos que hay un *salto* si dos puntos del viaje consecutivos temporalmente se encuentran a dos o más celdas de distancia.

Ejercicio 12. proc corregirViaje(inout v: *Viaje*, in errores: $seq\langle$*Tiempo*$\rangle$) Los viajes cargados en el sistema pueden contener valores de GPS erróneos debido al ruido electromagnético de algunos puntos del recorrido, que bloquean o hacen que el sistema devuelva localizaciones inválidas. Esto ocurre sobre todo en zonas del micro-centro de CABA, donde existe una altísima densidad de señales electromagnéticas. En estos casos, un operador puede generar una lista de los puntos únicos del viaje donde ocurrió esto, identificados por los valores de tiempo correspondientes a las localizaciones erróneas, ya que el tiempo es siempre bien medido y guardado.

Para este ejercicio se cuenta con un viaje válido de más de 5 puntos, y la lista **errores** que indica cada momento para el cual el valor registrado por el GPS fue erróneo y que debe ser corregido automáticamente. Para la corrección, se buscan los dos puntos más cercanos temporalmente (y correctos), que permiten calcular la velocidad media del vehículo en ese tramo del viaje. Luego, esos dos puntos definen una recta, sobre la cual se va a definir el punto GPS corregido, de acuerdo a la distancia recorrida, usando para ello la velocidad media. Por ejemplo, si originalmente una parte del viaje es $\langle \cdots, (t_1, a), (t_2, b), (t_3, c), \cdots \rangle$ (ver Figura 2) y la lista **errores** sólo contiene a t_2, esa parte del viaje resultante debería ser $\langle \cdots, (t_1, a), (t_2, b'), (t_3, c), \cdots \rangle$.

Se debe tener en cuenta que la cantidad de puntos a corregir en la lista **errores** no puede superar el 10 % del largo del viaje y que la lista **errores** puede indicar cualquier punto del viaje.

Ejercicio 13. proc histograma(in xs: $seq\langle$*Viaje*$\rangle$, in bins: $\mathbb{Z}$, out cuentas : $seq\langle\mathbb{Z}\rangle$, out limites : $seq\langle\mathbb{R}\rangle$) Que dada una lista de viajes válidos, calcule el histograma de velocidades máximas registradas entre todos los viajes.

Dado un conjunto de números (en este caso velocidades máximas), un *histograma* puede calcularse dividiendo el rango en que se mueven dichos números en intervalos de igual ancho (también llamados *bins*) en los que para cada uno se contabiliza la cantidad de valores incluidos en ellos.

- el parámetro **cuentas** deberá corresponderse con la cantidad de elementos en cada bin.
- el parámetro **limites** deberá contener los límites de cada uno de los bins en orden como se muestra en los siguientes ejemplos:
  - si las velocidades máximas de 4 viajes son $\langle 0, 1, 2, 10 \rangle$ y $bins = 5$, el resultado debería ser $cuentas = \langle 2, 1, 0, 0, 1 \rangle$, $limites = \langle 0, 2, 4, 6, 8, 10 \rangle$
  - si las velocidades máximas de 10 viajes son $\langle 33, 24, 24, 1, 62, 88, 94, 79, 25, 24 \rangle$ y $bins = 4$, el resultado debería ser $cuentas = \langle 4, 2, 1, 3 \rangle$, $limites = \langle 1.00, 24.25, 47.50, 70.75, 94.00 \rangle$.

Aclaración: como puede verse en el ejemplo, los intervalos se consideran cerrado-abierto, salvo el caso del último intervalo que se considerará cerrado-cerrado.