Tarea 1 - ELO320 Estructura de Datos y Algoritmos
Prof.: Nicolás Gálvez R.
Ayudantes: Raimundo Gross - Cristóbal Nilo - Matías Soto
Campus San Joaquín, UTFSM. 22 de Abril de 2022

1 En búsqueda del Tesoro Perdido

Después del controvertido `Cerrito` del día 6 de Mayo de 2022, ud. se retiraba por los pasillos exteriores de la universidad, cuando de repente, como por arte de magia, algo golpea su cabeza: una tarjeta SD. Confundido, ud. recoge la tarjeta y la inserta en su computador y se encuentra que el contiene dos archivos: `keys.txt` y `tweets.csv`.

El archivo `keys.txt` contiene, en una sola línea, una lista de números que parecen no indicar nada concreto.

`keys.txt`

```
1600199 1600124 1600102 2200243 46 862666 1600105 1600119 1600164 2200222 1600143 45 1600213 1600062↩
     25 1600171 2200256 2200245 1600193 1600183 1600131 1600187 1600079 1600125 28 16 20727 1600083 ↩
    2200221 2200229 1600175 1600166 1600163 1600139 1600169 40 30 43 1600159 1600180 2200217 1600136↩
     1600080 8 1600100 1600160 1600185 1600142 1600074 1600146 1600089 1600196 1600188 1600156 20 ↩
    2200231 2200237 1600141 39 1600088 54 1600172 2200242 2200226 1600066 1600063 1600192 2200240 55↩
     1600145 1600191 2200263 1600181 41 2200253 1600061 1600177 1600154 1600081 1600209 1600097 ↩
    1600070 1600138 4 1600186 2200248 1600155 1600133 2200239 21 1600060 10 1600101 2200258 31 ↩
    1600178 1600078 1600099 1600134 1600129 48 36 1600161 2200264 1600202 1600212 1600107 1600128 ↩
    1600065 2200216 53 1600158 1600112 2200236 1600150 32 2200247 2200246 1600211 1600094 50 2200238↩
     1600162 7 2200251 1600157 1600190 2200255 6 35 2200215 1600168 2200224 2200262 2200233 1600176 ↩
    2200228 1600067 1600198 1600132 1600170 961566 26 1600151 1600096 34 22 0 17 1600149
```

Mientras que el archivo `tweets.csv`, parece ser un archivo que contiene más de dos millones de mensajes emitidos en la plataforma **Twitter**.

`tweets.csv`

```
...
2019-05-05;18:58:14;TrenesseEssence;"I just want to be home making my own margaritas, cuddled up in ↩
    bed watching game of thrones episodes. Smh"
2019-04-29;14:22:30;CherryAdair;"GOT. Some spoilers. My fav is #29, which is yours? https://t.co/0↩
    xvzamT6yd https://t.co/0xvzamT6yd"
2019-04-15;01:20:02;PeterClines;"Between the LA Festival of Books, Game of Thrones, and tax day ↩
    tomorrow, I'm going to be interested to see how many folks show up for the dystopian book club ↩
    tonight."
...
2019-04-24;16:06:00;WLWT;Could a person ride a dragon? https://t.co/fDtMq6hvR1 https://t.co/↩
    tAFprexlcj
2009-05-31;23:55:46;nadiarasidi;@jacqjacqjacq MML is pretty cool.
2009-06-03;05:59:27;mbrbrry;is on PPA! Doesnt know where to star
...
```

En este archivo, cada linea representa un mensaje en la plataforma o **tweet**, e incluye la siguiente información:

1. La fecha de emisión del tweet con formato `aaaa-mm-dd` (`aaaa`: año, `mm`: mes, `dd`: día).
2. La hora de emisión del tweet con formato `hh:mm:ss` (`hh`: horas, `mm`: minutos:, `ss`: segundos).
3. El usuario que emite el tweet.
4. El tweet emitido.

Note, que cada campo está separado por el delimitador punto y coma (`;`). Además, que el tweet emitido puede incluir este carácter y no es considerado como un delimitador.

Esta situación, tanto como la información de los archivos, no le hace sentido, y acude a Oliver Atom, profesor de la UTFSM experto en el manejo de datos. El profesor Atom, quién aún no se ve convencido de su historia, le

comenta que al revisar rápidamente los archivos, piensa que `keys.txt` es un **vector hash** que indica la posición de ciertos twitts que al combinarlos entrega un mensaje secreto. Sin embargo, le comenta que esto no funciona con el estado actual del archivo `tweets.csv`.
Una de las opciones que el Profesor Atom maneja, es ordenar los contenidos del archivo `tweets.csv` en orden cronológico de emisión y ver que pasa. Sin embargo, el Profesor Atom no tiene más tiempo disponible, por lo que se lo asigna como tarea como alumno de su curso de *ELO320 - Estructura de Datos y Algoritmos*. Utilizando lenguaje C y el siguiente `header` como base:

`tweets.h`

```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

typedef struct twitter{
    char * fecha;
    char * hora;
    char * user;
    char * mensaje;
} tweet;
```

El profesor Atom le solicita:

(a) (15 puntos) Ordenar el archivo `tweets.csv` de forma cronológica utilizando algún algoritmo de ordenamiento de su elección que tenga complejidad de orden cuadrática o superior, o exponencial, vale decir, ($\mathcal{O}(n^2)$, $\mathcal{O}(n^3)$, $\mathcal{O}(2^n)$, etc.).

(b) (15 puntos) Ordenar el archivo `tweets.csv` de forma cronológica utilizando algún algoritmo de ordenamiento de su elección incluya complejidad logarítmica, vale decir, ($\mathcal{O}(\log n)$, $\mathcal{O}(n \log n)$, $\mathcal{O}(n^2 \log n)$, etc.).

(c) (15 puntos) Ordenar el archivo `tweets.csv` de forma cronológica utilizando algún algoritmo de ordenamiento de su elección que posea complejidad lineal, vale decir, ($\mathcal{O}(n)$, $\mathcal{O}(d \times n)$, $\mathcal{O}(n + k)$, etc.).

(d) (15 puntos) Definir un archivo llamado `sorting.c` donde se encuentren las implementaciones de los tres algoritmos seleccionados en los puntos anteriores.

(e) (30 puntos) Generar un archivo llamado `reporte.txt` donde se coloque información relevante para comparar empíricamente la diferencia en la complejidad de los algoritmos seleccionados. Éste puede incluir información como: tiempo de reloj de la ejecución, número de instrucciones realizadas, número de comparaciones realizadas, cantidad de memoria primaria utilizada, cantidad de memoria secundaria utilizada, etc. Sea ingenioso en como comparar los algoritmos, ya que posee la libertad de elegir los criterios que ud. estime conveniente.

(f) (10 puntos) Generar un archivo llamado `mensaje_secreto.txt`, el cual muestre el supuesto mensaje secreto oculto en la información contenida en la tarjeta SD caída desde el cielo.

2 Reglas de entrega y Consideraciones Generales

- La información de esta tarea es real y proviene de la plataforma `Twitter`. Fue obtenida desde distintos dataset disponibles en internet y de aplicaciones como `Tweepy`.
- Este trabajo debe realizarse **individualmente**, vale decir, en **grupos de un (1) estudiante**. No se harán excepciones.
- El programa debe ser desarrollado en lenguaje C, y compilado con la versión de gcc disponible en el servidor Aragorn[1]: `gcc 4.8.5`.
- Los archivos bases se encontrarán disponibles en AULA USM, y deben ser ingresados como parámetros de la ejecución del programa a través de la línea de comando.
- No hay un límite máximo de funciones a realizar, si ud. desea modularizar al máximo, tiene la libertad de hacerlo.
- La tarea debe ser entregada en la plataforma **AULA USM**, el día Viernes 8 de Julio de 2022 hasta las 23:59:59 Hora de Chile Continental (UTC -4).
- Cada minuto de atraso, implicará un descuento siguiendo la sucesión de Fibonacci. Así, diez minutos de atraso implicarían un descuento de cincuenta y cinco (55) puntos ($f(10) = 55$).
- La tarea debe incluirse en un archivo comprimido **.tar.gz**. El nombre del archivo debe seguir la siguiente estructura: `tarea2-eda-nombre-apellido-paralelo.tar.gz`; e.g., `tarea2-eda-oliver-atom-p200.tar.gz`.
- El archivo comprimido debe incluir, al menos, lo siguiente:
  - Cabecera `.h`: Archivo cabecera en el cual se deben incluir todas las bibliotecas a usar en el programa, además de las definiciones de macros, variables globales, tipos de datos personalizados, struct y prototipos de todas las funciones.
  - Código `.c`: Código del programa solicitado. Considere que éste es un desafío de al menos tamaño mediano, por lo tanto es recomendable dividir el código en diferentes archivos agrupados por su finalidad.
  - `README`: Archivo de texto plano en el cual se debe incluir: (1) una pequeña reseña del programa, (2) las condiciones de compilación y ejecución, (3) las instrucciones de compilación y ejecución, y (4) la información del creador del sistema.
  - `Makefile`: Archivo de compilación automática del sistema. Una cápsula de video para su confección se encontrará disponible en AULA USM.
- La revisión de los programas se hará utilizando diferentes archivos de tamaño variable. Por lo tanto, debe programar de forma genérica para cualquier cantidad de información de entrada.
- Cada fuga de memoria será penalizada. Utilice `valgrind` para verificar la correcta asignación, uso y liberación de memoria en la ejecución de su programa.
- Si su programa no compila, su nota será automáticamente **un cero (0)**.
- Cualquier atisbo de copia, será penalizada con máxima severidad.
- No deje la tarea para último momento, planifique bien sus tiempos y constrúyala paso a paso.
- Consulte sus dudas, lo más pronto posible, a través de los diversos medios de la asignatura.

[1] `ssh aragorn.elo.utfsm.cl -l cuentausm`