Instituto Tecnológico de Costa Rica, Análisis de Algoritmos
Prof. Rodrigo Núñez, Caso #1 – 5%, individual
Objetivo: Replantear soluciones de algoritmos tomando en cuenta cantidad de iteraciones, operaciones o comparaciones buscando reducirlas todo lo posible.

Proceda a escribir en C++ dos algoritmos para los primeros tres ejercicios, buscando que uno de ellos sea por mucho mejor que el otro. En los comentarios de cada implementación de cada ejercicio haga una explicación donde deje claro la diferencia entre las dos implementaciones que hace a una mejor que la otra.

1. Compare the Triplets | HackerRank
2. Time Conversion | HackerRank
3. Subarray Division | HackerRank

Escriba en C++ un algoritmo lo más optimizado posible para los siguientes tres problemas de hackerrank.

4. The Minion Game | HackerRank
5. Cipher | HackerRank
6. Pairs | HackerRank

Deberá entregar el ejercicio al asistente, al correo kevinqj.2002@gmail.com el día 19 de febrero a las 6pm. Incluyendo:

- Link al repositorio de git
- Un solo archivo .cpp que tenga todas las implementaciones de los ejercicios y un main que haga las llamadas de prueba, haciendo hasta 2 pruebas por ejercicio
- Un Makefile para el compilado del caso
- Si hay algún detalle que el asistente deba saber anotarlo en el readme.md
- Deberá usar gcc o c++ para linux, en caso de usar Windows use un c++ de ansi, eso excluye al compilador default que trae c-lion, visual studio, c++ buider

Instituto Tecnológico de Costa Rica, Análisis de Algoritmos
Prof. Rodrigo Núñez, Caso #2 – 10%, individual
Objetivo: Aprender y calcular el O grande de diversos algoritmos para que funcionen como referencia al comparar los rendimientos de algoritmos en el futuro.

El algoritmo de Quicksort podría ser en algunos casos O(n log(n)) y en otros es O(n^2), mismos casos que podrían verse cambiados cuando se trata de un pivote fijo a uno seleccionado aleatoriamente.

En el mismo sentido, el insertion sort podría ser en algunos casos O(n) y en otros casos O(n^2).

Otro problema importante para analizar es lo que resuelven muchos editores de texto cuando se hacen búsquedas del tipo "free text". Es decir, dado un texto de larga extensión, si se hace la búsqueda de "cal" por ejemplo, puede decir la cantidad de apariciones de "cal" y dichas apariciones, por ejemplo: cal, cala, calor, calentura, caléndula, calzado… Este problema fácilmente puede programarse en O^2 o incluso un caso peor.

Proceda a programar los 3 algoritmos anteriores demostrando programáticamente, con los resultados y análisis obtenidos de experimentos, debidamente documentados en el código; todos los casos anteriores:

- Quicksort logarítmico
- Quicksort cuadrático
- Influencia del pivote fijo y aleatorio
- Insertion sort lineal
- Insertion sort cuadrático
- Lograr un freetext search en tiempo logarítmico o lineal. Se recomienda usar un texto de mínimo 10MB.

Deberá entregar al asistente al correo kevinqj.2002@gmail.com los siguientes entregables:

1. Link al repositorio de git de la solución
2. Un archivo cpp separado para cada uno de los 3 algoritmos
3. Un archivo Makefile para compilar
4. Deberá usar c++, g++ de linux, en el caso de usar Windows no utilice compiladores que no sean ANSI, es decir, evite los que trae por default clion, visual studio y similares
5. En el repositorio de git debe tener un readme.md que explique debidamente los experimentos y resultados obtenidos, la forma en que justifica que logró demostrar que los algoritmos programados y las pruebas hechas cumplen los tiempos algorítmicos que se indican en los supuestos
6. Entrega será el día 8 de marzo a las 9pm