Lucia Pardo - 99999
Federico del Mazo – 100029
12 de junio del 2017

Facultad de Ingeniería de la Universidad de Buenos Aires

Año 2017 - 1er Cuatrimestre

Algoritmos y programación II - 75.41

Cátedra Wachenchauzer

Trabajo Práctico N.° 2

El trabajo consiste en dos programas, que a su vez hacen uso de un TDA Min Count Sketch y una extensión del heap, ademas de los TDAs ya desarrollados, Heap, Hash y la librería Strutil.

Programas:

- Procesar Tweets: Con el objetivo de calcular los trending topics (hashtags mas twiteados) en una serie de tweets, el programa lee el archivo (invocado como stdin) por linea, hace uso de split (de strutil.c) en las lineas, guarda cada hashtag en un contador (TDA Min Count Sketch), una vez guardado cada hashtag guarda el (tag,valor) en un hash. Luego se recorre con el iterador externo este hash y se pasa todo a un arreglo de nodos de heap, al cual se llama a la función 'top k' para imprimir los TTs.
- Procesar Usuarios: Esta función procesa todo un archivo recibido como parámetro e imprime por salida estándar a todos los usuarios agrupándolos por cantidad de hashtags twiteados. Para esto, recorre el archivo linea por linea, guardando en un hash cada hashtag (haciendo uso de split). Una vez lleno el hash, se crea un arreglo de nodos de heap (recorriendo el hash con un iterador externo) y después de un heap_sort se imprimen de menor a mayor todos los usuarios con sus respectivas cantidades de hashtags twiteados

Herramientas auxiliares:

- TDA Min Count Sketch: El TDA tiene un funcionamiento muy similar al hash, con la diferencia de que ahora en vez de procesar solo una vez cada clave por

una función de hashing, habrá varias funciones de hashing (en nuestro caso, modificables por constante, tres). Una vez hasheada una clave (tag) se sumara en uno la cantidad en cada fila de la matriz de arrays, que es una matriz de k (cantidad de hashings) * tamaño de cada array (capacidad por parámetro, primo). Al no importarnos las colisiones y simplemente sumando ++ en todo guardado, es importante saber que, aunque muy aproximado, este TDA no es exacto.

Por lo tanto, este TDA tiene 4 primitivas:

- Crear: Inicializa la matriz en ceros (con calloc)
- Destruir: Libera cada fila de la matriz
- Guardar: Recibe una clave, la hashea k veces, y le suma +1 en cada posición respectiva de las filas
- Obtener: Recorre toda la matriz y calcula el menor valor de todos los posibles en cada fila, con el fin de ser el mas aproximado

- heap_aux: Esta librería externa le ofrece a los dos programas las siguientes herramientas:
  - struct nodo_heap: Esta estructura de (clave,dato) sirve para poder insertar tuplas al heap y así no perder referencia a las claves una vez encolados/desencolados los datos.
  - cmp_nodo_heap_inv: Esta es la función de comparación necesaria para que el heap correctamente llame a downheap/upheap una vez lleno de nodos. Al ser invertida, en nuestro heap implementado como si fuese de máximos, esta función de comparación crea un heap de mínimos, necesario para la función top_k
  - arreglo_imprimir_top_k: Esta función recibe un arreglo de nodos, y con los conceptos de un heap de mínimos, imprime los mayores k números del arreglo. Llama a heapify y heapsort y hace uso de una función de comparación inversa.