Tarea 2: RadixSort y QuickSort

Profesores: Benjamín Bustos
Gonzalo Navarro
Auxiliares: Diego Salas
Asunción Gómez

1 Introducción

En esta tarea se estudiarán dos algoritmos para ordenar arreglos de enteros: radix sort y quick sort. Se experimentarán ambos algoritmos sobre un arreglo de n elementos en un universo de tamaño u. El objetivo es ir viendo cómo se comportan ambos algoritmos a medida que crece el tamaño del universo.

2 Algoritmos

2.1 Radix Sort

El algoritmo usa una serie de Bucket Sort. Por lo cual, deberán programar también este algoritmo. Toda la información de estos dos algoritmos se encuentra bien detallada en el apunte.

El Radix Sort puede ordenar de a k bits por ronda. Por lo cual, nos interesará ir variando este número de bits para ver cómo se comporta mejor el algoritmo.

2.2 Quick Sort

Este conocido algoritmo de ordenamiento usa la técnica "dividir para conquistar" y divide el arreglo en subarreglos con la ayuda de un pivot. De acuerdo a este pivot, los agrupa en menor o mayor y se repite este proceso en cada sub-arreglo hasta que cada sub-arreglo se encuentre ordenado.

Los detalles sobre el algoritmo se pueden encontrar aquí. Si usan código de internet, deben referenciarlo!

Para la implementación del algoritmo usen el del **pivot aleatorio**.

3 Objetivos

Para esta tarea, se deberá implementar:

1. El algoritmo del Radix Sort (y del Bucket Sort).
2. El algoritmo del Quick Sort.

Además, queremos evaluar el tiempo en cada algoritmo.

4 Experimentación

Debe comparar los tiempos de ordenamiento de los algoritmos sobre el mismo arreglo desordenado. Se deberá probar los algoritmos sobre un arreglo con estas características:

- El tamaño n del arreglo será fijo de 100 millones.
- Se debe crear los arreglos de forma aleatoria con números que pertenezcan al rango $[1, u]$ con $u \in [2, 2^2, 2^3, ..., 2^{64}]$. Use unsigned long long para guardar los números.
- Realice al menos 100 repeticiones por cada tamaño del universo. Cada repetición debe ser con un arreglo distinto, y se debe usar el mismo arreglo para ambos métodos.

En el experimento, se pide comparar:

- El tiempo de ordenamiento promedio en el Radix Sort según el número de bits k y el tamaño del universo.
- El tiempo de ordenamiento promedio por tamaño del universo entre el Radix Sort y el Quick Sort.

Para comparar el Radix Sort con el Quick Sort, deberá ver por cada tamaño del universo u, el k óptimo, y con ese k comparar ambos algoritmos. Para esto, deberá ejecutar previamente el Radix Sort tomando en cuenta $k \in [1, \log_2(u)]$ y de esta forma, determinar el k óptimo para cada u.

Además, determinen el valor del universo hasta donde conviene usar Radix Sort o Quick Sort, y entreguen alguna recomendación para el k a usar. Por último, grafiquen sus resultados, exponiendo los tiempos de ejecución de ambos algoritmos y cómo se comportan al modificar las variables k y u.

5 Entrega

- La tarea puede realizarse en grupos de a lo más 3 personas.
- Para la implementación puede utilizar C, C++ o Java.
- Para el informe se recomienda utilizar Latex.
- Escriba un informe claro y conciso, la entrega debe contar con el código de su implementación y todas las indicaciones necesarias para su ejecución explicadas de forma clara.
- La nota del informe se calculará de la forma:

  - Introducción: 0.8 ptos.
  - Desarrollo: 0.8 ptos.
  - Resultados: 2.4 ptos.
  - Análisis: 1.2 ptos.
  - Conclusión: 0.8 ptos.

- La nota final de la tarea es el promedio del informe y del código.