|  |  |  |
| --- | --- | --- |
|  | **MaxColaCP** | **MaxHeapCP** |
| ¿Cuándo ocurre el peor caso de agregar()? | Si debo recorres todos los datos para agregar un elemento. | Cuando es necesario redefinir el tamaño del arreglo. |
| Complejidad peor caso de agregar() | O(N) | O(N) |
| ¿Cuándo ocurre el mejor caso de agregar()? | Cuando los datos que se agregan a la cola de prioridad ya están ordenados. | Cuando los datos están parcialmente ordenados. |
| Complejidad mejor caso de agregar() | O(1) | O(logN) |
| Tiempo promedio de agregar() (Promedio de agregar 200.000 datos aleatorios a una cola vacía) | 99000 milisegundos | 12 milisegundos |
| ¿Cuándo ocurre el peor caso de sacarMax ()? | Siempre es un buen caso ya que es solo sacar al primero y definir el segundo nodo como primero. | Cuando es necesario redefinir el tamaño del arreglo. |
| Complejidad peor caso de sacarMax () | O(1) | O(N) |
| ¿Cuándo ocurre el mejor caso de sacarMax ()? | Siempre es un buen caso. | Cuando la cantidad de ordenamientos no son demasiados. |
| Complejidad mejor caso de sacarMax () | O(1) | O(logN) |
| Tiempo promedio de sacarMax () (Promedio de sacar 200.000 datos de una cola con 200.000 datos) | 75 milisegundos | 90 milisegundos |

DOCUMENTO DE ANÁLISIS

En el siguiente documento se realizará un análisis comparativo entre una MaxColaCP y una MaxHeapCP, estructura de datos que se vieron en clase. Se mirarán casos óptimos y no óptimos de las funciones de agregar y sacar el máximo en cada una de estas estructuras.

Como se podrá evidenciar, con los datos más adelante, podemos ver que en una MaxColaCP es mas eficiente sacando al máximo que ingresando elementos, por este motivo podemos decir que es bueno usar esta estructura si tiene los datos parcialmente ordenados. Por otro lado una MaxHeapCP es eficiente en ambos casos, como sacar y agregar, pero es mas lenta sacando que la MaxColaCP.