# Tarea Nº1

## Estructura de Datos y Algoritmos

Universidad Austral de Chile

2018

**Profesor**

Héctor Ferrada

**Integrante**

Eduardo Hopperdietzel

# Problema 1

### Método posAprox

Este método retorna una índice aproximado de la ubicación de un número x en el arreglo ordenado A.  Para esto utiliza proporcionalidad. Divide el tamaño del arreglo por la diferencia entre su último y primer valor y lo multiplica por x más el primer valor, obteniendo así una posición aproximada, que se torna más exacta a medida que el rango de aleatoriedad con el cual se construye A disminuye.

### Método isXinA

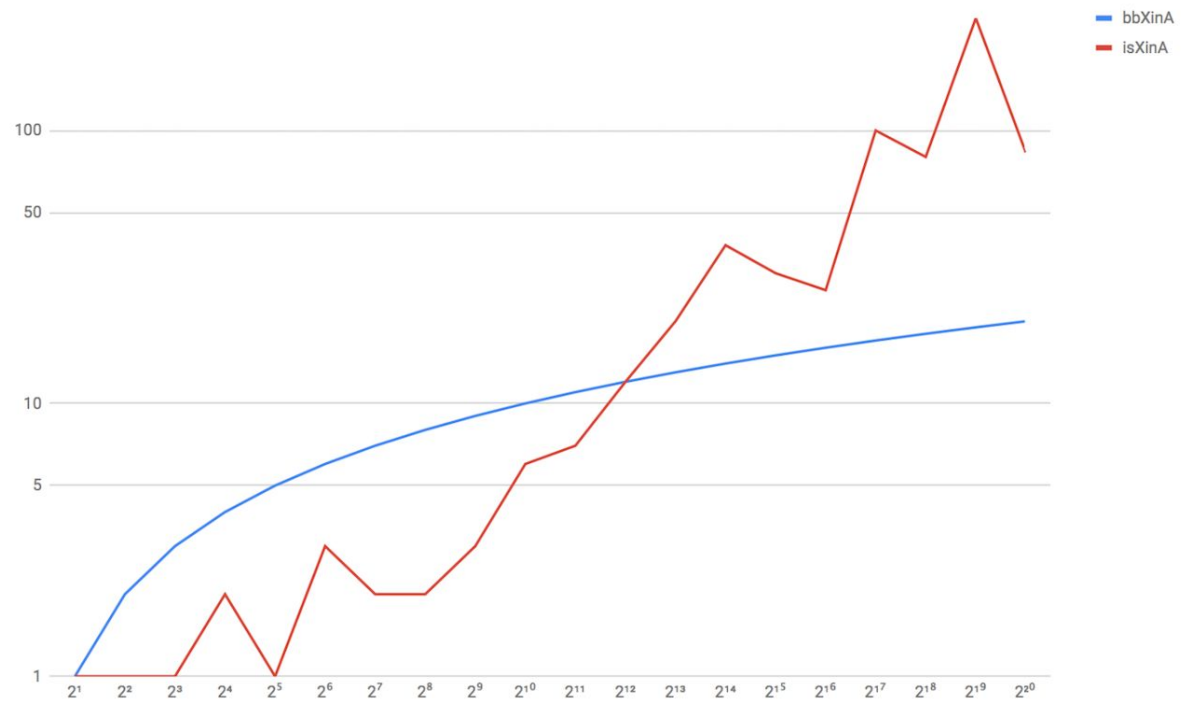
Este método comprueba si un elemento x existe en A. Para esto, comienza utilizando el método anterior para encontrar un índice aproximado de x. Si el valor asociado al índice es igual a x, retorna verdadero, si es menor recorre el arreglo hacia arriba hasta encontrar una coincidencia, y en caso de encontrar un número menor a si mismo o si llega al final del arreglo retorna falso.
Lo mismo ocurre si el número del índice es es mayor, en cuyo caso recorre el arreglo en la otra dirección.

### Método bbXinA

Este método tiene la misma función que el anterior, pero su sistema de búsqueda es distinto. Utiliza el algoritmo de búsqueda binaria, el cual consiste en dividir el tamaño del arreglo en dos para obtener el índice de la mitad, luego compara este valor con x, si son iguales retorna verdadero, si es menor descarta la primera mitad del arreglo, y vuelve a dividir por dos la segunda mitad. De la misma forma ocurre así si el valor es mayor, pero descarta la otra mitad.
Se realiza este paso hasta que el intervalo se reduzca a uno, en cuyo caso, si no se encontró el valor, retorna falso.

**Eficiencia isXinA vs bbXinA**

Tras realizar varios benchmarks, se pudo concluir que el método **isXinA**, aumenta su eficiencia, cuando la diferencia entre cada elemento del arreglo y su antecesor tiende a ser constante, y resulta ser más efectivo que **bbXinA** cuando el tamaño del arreglo es relativamente pequeño ( Menor a 4096 elementos ).




(X = Tamaño del arreglo, Y = Nº de iteraciones)

Esto debido a que, a pesar de que la búsqueda utilizando proporcionalidad es más efectiva que dividir por dos directamente, se utiliza solo en la primera iteración, después debe recorrer el arreglo linealmente, lo cual es muy ineficiente cuando el arreglo es muy grande.
Esta es la gran diferencia con **bbXinA** , el cual nunca recorre A linealmente, sino que lo divide consecutivamente en dos, consumiendo un total de log2 n iteraciones, muy eficiente arreglos de gran tamaño.

# Problema 2

Para el segundo problema, se intentó utilizar la menor cantidad de memoria extra posible para ordenar los elementos, lo cual generó un aumento de iteraciones.

El procedimiento planteado comienza con un doble for de n iteraciones cada uno, los cuales ordenan los pacientes, de menor a mayor edad, utilizando el algoritmo de burbuja. Mientras va ordenando la cola, elimina los pacientes de prioridad 3 y los almacena en Q3, ya que esta cola no requiere estar ordenada por edad.

Teniendo a los pacientes de prioridad 1 y 2 ordenados por edad, el siguiente paso utiliza otro for del tamaño de la cola actual, el cual elimina a los pacientes de prioridad 2 y los agrega a Q2. A su vez, va almacenando la cantidad de personas de prioridad 1 según su rango de edad en un arreglo de  enteros de tamaño 6 ( Los 6 intervalos de edad requeridos ).

El paso anterior termina con la cola llena solo con pacientes de prioridad 1 pero con orden invertido.
Por lo tanto teniendo el número de pacientes de cada uno de los 6 intervalos de edad, se implementan 6 for separados, con una cantidad de iteraciones igual a la cantidad de pacientes de cada intervalo de edad. Los cuales eliminan a los pacientes restantes de la cola y los almacenan en sus respectivos stacks.
Siendo el primer for, el del stack P6, ( ya que la cola quedó invertida )
y el stack P1 el último.

Como conclusión, se puede observar que la parte que genera mayor deficiencia en el programa, es el algoritmo de la burbuja, el cual utiliza un total de $n^2$ iteraciones para ordenar a los pacientes. Este podría haber sido reemplazado por otro algoritmo que utilice menos iteraciones, pero que probablemente requiera de mayor uso de memoria externa para realizar las operaciones.