**Algoritmia y Complejidad**

Problemas Tema 2, Algoritmos Voraces

23 de febrero de 2022

Adrián Borges Cano y Marco González Martínez

2ºGII (Laboratorio= 15:00-17:00)

Índice:

**\*\*\*METER UN ÍNDICE AL TERMINAR TODOS LOS EJERCICIOS\*\*\***

**PROBLEMA 3:**

**Se dispone de un vector V formado por n datos, del que se quiere encontrar el elemento mínimo del vector y el elemento máximo del vector. El tipo de los datos que hay en el vector no es relevante para el problema, pero la comparación entre dos datos para ver cuál es menor es muy costosa, por lo que el algoritmo para la búsqueda del mínimo y del máximo debe hacer la menor cantidad de comparaciones entre elementos posible.**

**Un método trivial consiste en un recorrido lineal del vector para buscar el máximo y después otro recorrido para buscar el mínimo, lo que requiere un total de aproximadamente ente 2n comparaciones entre datos. Este método no es lo suficientemente rápido, por lo que se pide implementar un método con metodología Voraz que realice un**

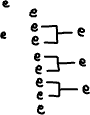
**máximo de 3n/2 comparaciones**

Solución adoptada:

Para poder resolver este problema lo más eficiente posible, hemos optado por implementar un algoritmo que aplica el simple concepto de “rey de la pista” (el elemento “ganador” mantiene su puesto, mientras que el perdedor queda eliminado). Dividimos los elementos del vector original en dos nuevos vectores, de la mitad de tamaño cada uno de ellos. Una vez tengamos divididos ambos vectores (donde uno almacenará los elementos mayores, y otro, los elementos menores).

Al realizar este procedimiento de “rey de la pista”, el primer elemento de cada vector será comparado con el resto de ellos. Si cumple la condición (que el primer elemento sea el mayor en el primer vector, y que sea el menor en el segundo) se mantendrá, de lo contrario, se eliminará y el elemento que le ha eliminado pasará a ser el rey de la pista. Cuando todas las iteraciones terminen, obtendremos nuestros resultados buscados. Gráficamente, para facilitar la explicación, quedaría algo tal que así:

Siendo ‘e’ los elementos comparados, donde N es el número total de elementos:



Componentes voraces del algoritmo:

1. Conjunto de Candidatos: Todos los elementos de entrada del vector.

2. Conjunto de Decisiones: Calcular el elemento mayor

3. Función que determina la solución del problema:

4. Completable:

5. Función Selección:

6. Función Objetivo:

Ejemplos de ejecución:

\*\*COMPLETAR\*\*

Código del ejercicio:

\*\*COPIAR Y PEGAR\*\*

**PROBLEMA 4:**

**Teniendo como datos la cantidad de nodos n y la matriz de costes M, se pide encontrar el árbol soporte mínimo del grafo G utilizando el algoritmo de Prim, utilizando las siguientes ideas:**

**• A diferencia del algoritmo de Kruskal (que crea el árbol utilizando componentes conexas independientes que se van uniendo entre sí), el algoritmo de Prim se basa en la idea de ir construyendo un árbol cada vez más grande, empezando por un único nodo y acabando por recubrir todo el grafo.**

**• El algoritmo comienza con un árbol de un nodo, al que se le añade un segundo nodo, luego un tercero, etc, hasta tener los n nodos unidos. La forma de escoger un nodo es buscando el nodo más cercano a todo el árbol, sin que se creen ciclos.**

**• A medida que crece el tamaño del árbol la búsqueda del nodo más cercano se complica, por lo que para que el algoritmo sea eficiente (el método debe tener O(n2)) hay que crear una estructura de datos que almacene la mejor distancia de cada nodo al conjunto de nodos del árbol.**

**• Se necesitará almacenar de alguna manera la forma en que se ha creado el árbol, por ejemplo indicando a qué nodo del árbol se está uniendo el nuevo candidato seleccionado.**

Solución adoptada:

\*\*COMPLETAR\*\*

Componentes voraces del algoritmo:

\*\*COMPLETAR\*\*

Ejemplos de ejecución:

\*\*COMPLETAR\*\*

Código del ejercicio:

\*\*COMPLETAR\*\*

PROBLEMA 6: