LAB13

Objetivos:

- Creación de operaciones con listas dinámicas

Lab 13. Apuntadores y Listas dinámicas – Avanzado

13.1. Lista de ejercicios

Varios de los ejercicios que aquí se proponen se han propuesto previamente. En este laboratorio cambian el tipo de datos en el que se almacenan los datos y/o los resultados. Así, la resolución del ejercicio no radicará en los cálculos a realizar sino en la manera de acceder a los datos y/o resultados.

13.1.1. Clonar y concatenar

Implementa la función **Clonar_y_Concatenar** que, dadas dos listas dinámicas L1 y L2, cree una nueva lista L, diferente de las anteriores que tenga una copia de todos los elementos de L1 y, a continuación, una copia de todos los elementos de L2.

13.1.2. Borrar todas las apariciones

Implementa el procedimiento **Borrar** que, dada una lista dinámica y un entero N, modifique la lista eliminando de la lista todas las apariciones del entero N en la lista.

13.1.3. Intersección

Implementa el procedimiento **Intersección** que, dadas dos listas dinámicas L1 y L2, ordenadas ascendentemente, reorganice los elementos en otras dos listas, comunes y no_comunes, también ordenadas ascendentemente, de manera que la primera contenga los elementos que están en ambas listas (dos veces) y la segunda los que solo están en una de ellas. L1 y L2 después de la llamada al procedimiento quedan deshechas. En la implementación de este subprograma, no puede haber instrucciones new.

13.1.4. Listas dinámicas iguales

Implementa la función **Son_iguales** que dadas dos listas dinámicas de enteros, ordenadas o no, indique si ambas listas contienen el mismo número de elementos y en el mismo orden.

13.1.5. Simplificar lista dinámica

Implementa el procedimiento **Simplificar** que, dada una lista dinámica de puntos, la modifique eliminando los puntos que estén a una distancia menor a 0.001 en ambas coordenadas. Los puntos que queden en la lista deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. Se penalizará el uso de listas auxiliares en la resolución del problema.

13.1.6. Simplificar lista de puntos

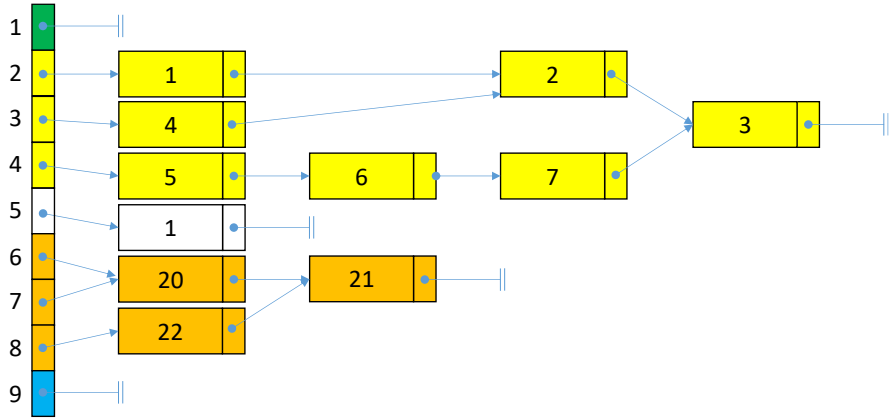
Implementa el procedimiento **Simplificar** que, a partir de una lista de puntos no vacía L, la modifique para que solo tenga un cuarto de los puntos del original, respetando uno sí y tres no. Los puntos que se queden deben mantener el orden de la lista original y los puntos inicial y final siempre se habrán de mantener. También se devolverá el número de puntos resultantes en la lista resultado. (Es el mismo ejercicio que el laboratorio 7 avanzado, pero con listas dinámicas).

13.1.7. Cuántos grupos hay

Dos listas pertenecen al mismo grupo si comparten alguno de sus elementos. Implementa la función **cuantos_grupos** que dado un vector completo de listas dinámicas de enteros que pueden (o no) compartir alguno de sus elementos, devuelva cuántos grupos hay en dicho vector.

Por ejemplo, el vector de la figura tiene 9 listas dinámicas. Las podemos agrupar en cinco grupos. La lista uno no comparte ninguno de sus elementos con ninguna otra lista (básicamente porque no tiene ninguno) y, por ello, forma un grupo. Lo mismo ocurre con la lista 9. Las listas 2, 3 y 4 forman un grupo porque

la lista 2 y 3 comparten dos elementos (el 2 y el 3); por su parte la lista 4 comparte el 3 con las listas 2 y 3.



La lista 5 no comparte ningún nodo con ninguna otra lista, por lo que forma un grupo por sí misma. Fíjate que la lista 5 y la lista 2 comparten un nodo con un 1, pero no se trata del mismo nodo, sino de nodos distintos que tienen dentro el mismo valor.

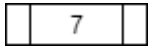
Las listas 6 a 8 forman también un grupo. Las listas 6 y 7 comparten todos sus nodos y éstas dos comparten, a su vez, el nodo 21 con la lista 8.

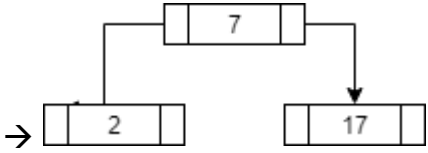
Por todo ello, la respuesta de la función cuántos_grupos con un vector de listas como el anterior, debe devolver 5 (1-234-5-678-9).

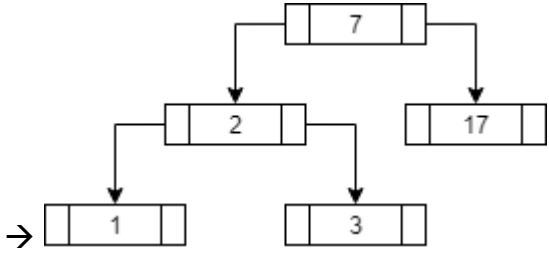
13.1.8. Crear árbol binario

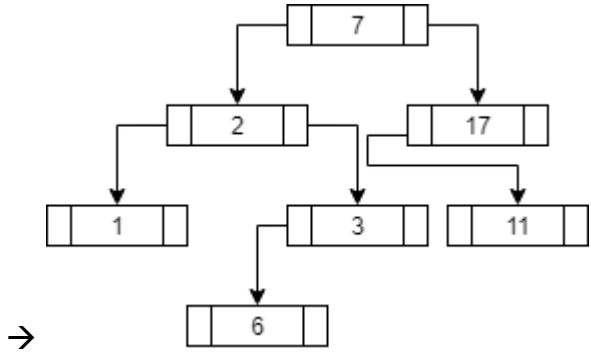
Implementa el procedimiento **Crear_arbol_binario** que, dado un vector de enteros, cree un árbol binario de la siguiente manera. Se mira el valor del entero de la raíz. Si el valor a insertar es menor que él, se introduce en el subárbol izquierdo y si no en el derecho (si es igual, se ignora). Si no hay subárbol izquierdo o derecho, se crea un nodo con el elemento a insertar y si no, se decide en qué rama hay que introducir el nuevo elemento de la misma manera que antes (menores a la izquierda y mayores a la derecha). Por ejemplo, si el vector

contiene los valores (7 2 17 1 3 11 6 16 4 5 18 8). Del árbol vacío se crearía uno con solo el 7. Luego, el 17 se pondría colgando de la rama derecha (porque 17>7) y el 2 colgando de la rama izquierda (porque 2<7). Con los siguientes números se haría parecido: el 1 va por la rama izquierda del 7 y la izquierda del 2. El 3 seguiría la rama izquierda del 7 y la derecha del 2. Y así sucesivamente.

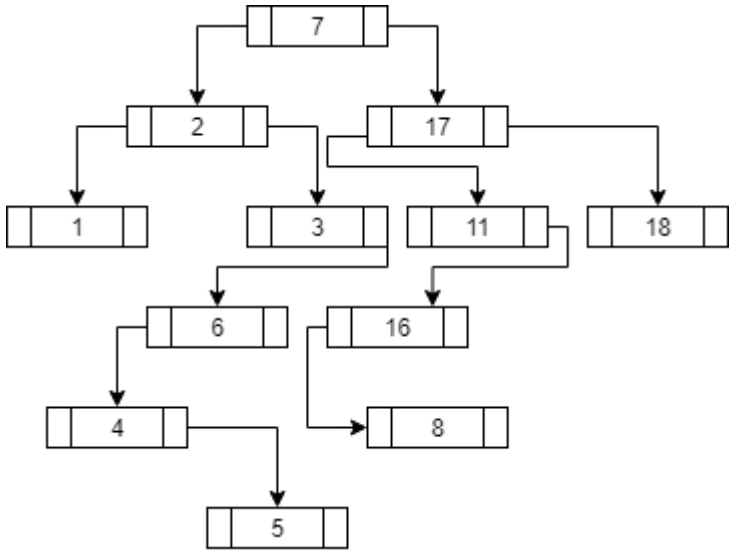







7
2
17
1
3
11
6

→


7
2
17
1
3
11
18
6
16
4
8
5

→