**MÉTODO DE LA INGENIERÍA**

**Fase 1: Identificación del problema:**

La Bolsa de Valores de Colombia (BVC) es una bolsa multi-producto y multi-mercado que administra los sistemas de negociación y registro de los mercados de acciones, renta fija, derivados, divisas, OTC y servicios de emisores en Colombia. Actualmente la BVC no permite transar con acciones internacionales ni trabajar con el mercado de divisas o de derivados.

* **Definición del problema:**

Se ha requerido por parte de la BVC junto con el gobierno nacional el desarrollo una herramienta que permita conocer los datos de algunos mercados de divisas y de acciones internacionales. Por otro lado debe permitir realizar análisis sobre dichos datos y de esta forma conocer patrones de movimientos de los mercados y que criterios toman más fuerza.

**Requerimientos Funcionales:**

La solución del problema :

**R1 :** Requiere consultar el precio más alto de una acción o mercado de divisas en un rango de tiempo, dada una fecha inicial y una fecha final.

**R2 :** Requiere consultar el precio más bajo de una acción o mercado de divisas en un rango de tiempo, dada una fecha inicial y una fecha final.

**R3 :** Requiere consultar el periodo de tiempo donde una acción / mercado de divisas tuvo su mayor crecimiento.

**R4 :** Requiere mostrar una gráfica del estado de los precios de una acción/mercado de divisas, en la que debe ser posible agregar como máximo 3 acciones/mercado de divisas , donde cada indicador deberá tener un color diferente.

**R5 :** Requiere visualizar cuáles acciones/mercados de divisas superar un valor en un rango de tiempo.

**R6 :** Requiere visualizar cuales son las 3 acciones/mercados de divisas que presentaron mayor crecimiento en un rango de tiempo

**Fase 2: Recopilación de la Información**

**Mercado financiero:**

En economía, un mercado financiero es un espacio (físico o virtual o ambos) en el que se realizan los intercambios de instrumentos financieros y se definen sus precios. En general, cualquier mercado de materias primas podría ser considerado como un mercado financiero si el propósito del comprador no es el consumo inmediato del producto, sino el retraso del consumo en el tiempo.

Los mercados financieros están afectados por las fuerzas de oferta y demanda. Los mercados colocan a todos los vendedores en el mismo lugar, haciendo así más fácil encontrar posibles compradores. A la economía que confía ante todo en la interacción entre compradores y vendedores para destinar los recursos se le llama economía de mercado, en contraste con la economía planificada.

Los mercados financieros, en el sistema financiero, facilitan:

* El aumento del capital (en los mercados de capitales).
* La transferencia de riesgo (en los mercados de derivados).
* El comercio internacional (en los mercados de divisas).

Son usados para reunir a aquellos que necesitan recursos financieros con aquellos que los tienen.

**Acción financiera:**

Es un título emitido por una sociedad que representa el valor de una de las fracciones iguales en que se divide su capital social. Las acciones, generalmente, confieren a su titular (llamado accionista) derechos políticos, como el de voto en la junta de accionistas de la entidad y económicos, como participar en los beneficios de la empresa.

**Mercado o Acción de divisas:**

Es un mercado mundial y descentralizado en el que se negocian divisas. Este mercado nació con el objetivo de facilitar el flujo monetario que se deriva del comercio internacional.

El mercado de divisas es único debido a:

* El volumen de las transacciones
* La liquidez extrema del mercado.
* El gran número y variedad de los intervinientes en el mercado
* Su dispersión geográfica
* El tiempo en que se opera - 24 horas al día (excepto los fines de semana).
* El cambio de horarios trading.
* La variedad de factores que generan los tipos de cambio.



**Fase 3: Búsqueda de soluciones creativas**

La generación de las ideas se desarrollaron de forma conjunta planeando cuales serían las posibles soluciones al problema, donde aceptamos todo tipo de idea, pero con el fin de que su uso fuese lógico en la solución y además lo más eficiente posible , ya que por la gran cantidad de datos a utilizar es necesario que se cumplan las condiciones anteriores para evitar contratiempos e inconvenientes.

**1**.Utilizar árboles AVL cómo estructuras de datos para almacenar la información que se nos esta brindando y es elemental para el desarrollo de los requerimientos.

**2.** Implementar las estructuras de datos nosotros mismos desde el comienzo y adecuarlas al problema mediante las vamos construyendo.

**3.** Mediante el uso de algoritmos de ordenamiento organizar cada una de las divisas para que pueda ser efectivo cada acción que se emplee sobre ellas.

**4.** Buscar videos sobre como diseñar algoritmos para dibujar gráficas y diseñar el algoritmo nosotros mismos.

**5**. Utilizar árboles rojinegros (ARN)

**6**. Mediante el uso de listas enlazadas organizar las divisas.

**7**. Organizar cada una de las divisas en arreglos para así llevar toda su información y realizar la gráfica con esta.

**Fase 4: Transición de las Ideas a los Diseños Preliminares**

Las siguientes ideas las descartaremos tomando como base de criterio el desarrollo por completo y de forma efectiva lo que se nos está pidiendo en este caso con el desarrollo de la factorización de polinomios para hallar sus raíces, no obstante, en cada una de las ideas descartadas que daremos a continuación argumentaremos por qué se tomó la decisión.

Alternativa 4: Ver vídeos sobre algoritmos para gráficas

-Aunque esta alternativa puede considerarse ya que las gráficas hacen parte de la solución, no es necesario ver vídeos sobre esto ya que poseemos todas las herramientas para hacerlo.

Alternativa 3: Uso de algoritmos de ordenamiento

-El uso de métodos de ordenamiento queda totalmente obsoleto en este caso ya

que las divisas no tienen forma alguna de ordenarse específicamente, si no que van variando acorde el tiempo.

Alternativa 6: Uso de listas enlazadas

-El uso de Listas es totalmente absurdo y aunque puede llegar a ser interesante no se empleará su uso como fundamento del desarrollo de la solución.

Alternativa 7: Uso de arreglos

-El uso de arreglos es totalmente absurdo para el desarrollo de la solución y aunque puede ser interesante no se empleara su uso.

**DISEÑOS PRELIMINARES:**

|  |
| --- |
| **TAD AVL** |
|  |
| **Invariantes:**   * La altura de la rama izquierda no difiere en más de una unidad de la altura de la rama derecha o viceversa. * Cada nodo debe tener los dos punteros a los árboles derecho e izquierdo. * El subárbol izquierdo de cualquier nodo, contiene valores menores que el que contiene dicho nodo * El subárbol derecho de cualquier nodo, contiene valores mayores que el que contiene dicho nodo. |
| **Operaciones:**  Insert() → AVL,Nodo x AVL → Constructor  Delete() → AVL,Nodo x AVL → Destructor  LeftRotate() → Nodo, AVL x AVL → Modificador  RightRotate() → AVL, NODO x AVL → Modificador  isRoot() → Nodo, AVL x Boolean → Consultor  Add() → Nodo,AVL x AVL → Constructor  Higher() → AVL x Integer → Consultor  Weight() → AVL,Nodo x Integer → Consultor |
| **Insert()**  **“**Inserta un nodo al árbol AVL”  **{pre:}**  **{post:**Retorna el árbol AVL con el nuevo nodo agregado.**}**  **Delete()**  “Elimina un nodo del árbol AVL”  **{pre:}**  **{post:** El árbol sin el nodo previamente dado.**}**  **LeftRotate()**  **“**Retorna el árbol rotado hacia la izquierda, a partir de un nodo dado**.”**  **{pre:** El nodo dado debe encontrarse en el árbo**l}**  **{post:** Null si el árbol está vacío, de lo contrario el árbol se rota.**}**  **RightRotate()**  **“**Retorna el árbol rotado hacia la derecha, a partir de un nodo dado**.”**  **{pre:** El nodo dado debe encontrarse en el árbo**l}**  **{post:** Null si el árbol está vacío, de lo contrario el árbol se rota.**}**  **isRoot()**  “Verifica si un nodo es raiz.”  **{pre:** El nodo dado debe encontrarse en el árbo**l}**  **{post:** Null si el árbol está vacío, de lo contrario el árbol se rota.**}**  **Add()**  “Agrega un nodo al árbol AVL”  **{pre: }**  **{post:** El árbol AVL con el nodo agregado y balanceado.**}**  **Higher()**  “Retorna la altura del árbol AVL”  **{pre: }**  **{post:** El valor de la altura del árbol.**}**  **Weight()**  “Retorna el peso de un nodo”  **{pre:** El árbol no está vacío.**}**  **{post:** El peso del nodo ingresado.**}** |

|  |
| --- |
| **TAD ROJINEGRO** |
|  |
| **Invariantes:**   * La raíz es negra. * Todo nodo es o bien rojo o negro. * Todas las hojas(NULL) son negras. * Todo nodo rojo debe tener dos hijos negros. * Cada camino desde un nodo dado a sus hojas descendientes contiene el mismo número de nodos negros. |
| **Operaciones:**  LeftRotate() → Nodo,ARN x ARN → Modificador  RightRotate() → ARN,NODO x ARN → Modificador  Black-height() → ARN x Entero → Consultor  TreeInsert() → ARN,Nodo x ARN → Constructor  Delete() → ARN,Nodo x ARN → Destructor  DeleteFixup() → ARN,Nodo x ARN → Modificador |
| **LeftRotate()**  **“**Retorna el árbol rotado hacia la izquierda, a partir de un nodo dado**.”**  **{pre:** El nodo dado debe encontrarse en el árbo**l}**  **{post:** Null si el árbol está vacío, de lo contrario el árbol se rota.**}**  **RightRotate()**  **“**Retorna el árbol rotado hacia la derecha, a partir de un nodo dado**.”**  **{pre:** El nodo dado debe encontrarse en el árbo**l}**  **{post:** Null si el árbol está vacío, de lo contrario el árbol se rota.**}**  **Black-height()**  **“**Retorna la altura negra del árbol.**”**  **{pre:}**  **{post:** El valor de la altura negra del árbol.**}**  **TreeInsert()**  “Inserta un nuevo nodo al árbol.”  **{pre:}**  **{post:** El árbol con el nuevo nodo agregado.**}**  **Delete()**  “Elimina un nodo del árbol.”  **{pre:}**  **{post:** El árbol sin el nodo previamente dado.**}**  **DeleteFixup()**  “Elimina un nodo del árbol y rota el árbol según corresponda.”  **{pre:}**  **{post:** El árbol sin el nodo dado a eliminar, y rotado.**}** |

**Fase 5: Evaluación y Selección de la Mejor Solución**

**Criterios**

Estos son los principios por los cuales serán evaluadas las ideas y donde escogeremos las apropiadas para el desarrollo de la solución del problema de forma definitiva.

**Criterio A: Complejidad**

• [8] Complejidad constante

• [7] Complejidad logarítmica

• [6] Complejidad raíz

• [5] Complejidad lineal

• [4] Complejidad nlogn

• [3] Complejidad polinómica

• [2] Complejidad exponencial

• [1] Complejidad factorial

**Criterio B: Eficiencia**

• [3] Muy eficiente

• [2] Eficiente

• [1] Nada eficiente

**Criterio C: Facilidad implementación**

• [2] Fácil

• [1] Difícil

Evaluación

Evaluando los criterios anteriores en las alternativas que se mantienen, obtenemos la siguiente tabla:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Criterio A | Criterio B | Criterio C | Total |
| Alternativas 5: Árboles Rojinegros | Logarítmica - 7 | Muy eficiente - 3 | Fácil - 2 | 12 |
| Alternativa 1: Árboles AVL | Logarítmica - 7 | Muy eficiente - 3 | Fácil - 2 | 12 |
| Alternativa 2: Algoritmos propios | Lineal - 5 | Nada eficiente - 1 | Difícil - 1 | 7 |

Selección

Teniendo en cuenta la tabla anterior, las alternativas 1 y 5 deben ser escogidas ya que tuvieron mayor puntuación dados los criterios de evaluación.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **DISEÑO DE CASOS PARA PRUEBAS UNITARIAS** | | | | |
| **Clase** | **Método** | **Escenario** | **Entrada** | **Salida** |
| RedBlackTree() | stageOne() | setUp1() | 6 elementos de tipo market, que serán agregados al árbol rojinegro. | El método retorna true si el tamaño de nodos del árbol corresponde a la cantidad de elementos ingresados. |
| RedBlackTree() | stageTwo() | setUp2() | Un elemento de tipo market, que será eliminado del árbol. | El método retorna true si el número de nodos restantes, corresponde a la cantidad de nodos inicial menos 1. |
| RedBlackTree() | stageThree() | setUp3() | Árbol rojinegro y sus elementos de tipo market. | El método retorna true si el último nodo del árbol, corresponde al último elemento de tipo market que haya sido agregado. |
| RedBlackTree() | stageFour() | setUp4() | Árbol rojinegro y sus elementos de tipo market. | El método retorna el valor de la profundidad del árbol. |
| RedBlackTree() | stageFive() | setUp5() | Árbol rojinegro con 6 elementos de tipo market. | Él método retorna true si el árbol contiene uno de los nodos ingresados por parámetro. |
| RedBlackTree() | stageSix() | setUp6() | Árbol rojinegro con 6 elementos de tipo market. | Él método retorna true si el árbol está vacío, retorna false en caso de contener elementos. |

**BIBLIOGRAFÍA**

https://es.wikipedia.org/wiki/Mercado\_de\_divisas

https://docs.google.com/document/d/1yvI4PvTrjuYewCfXQtNeI4JlcfqPh\_QdnGMowV4jdkQ/edit