Practica 05

Estructura de Datos y Algoritmos 1

Josué Alexis Campos Negrón
josue.campos@cimat.mx
Universidad de Guanajuato
09 de marzo del 2023

Fecha de entrega: Lunes 20 de marzo.

Problemas

Problema 1. Realizar la clase `Fraction` que tendrá dos variables, una que representa el *numerador* y otra que represente el *denominador*. Utiliza la construcción `template <typename T>` para la generalización de tipo de datos `int`, `long`, `float` y `double`. Además realizar sobre carga de operaciones según las operaciones usuales de las fracciones.

- `operator +`: Realiza la suma de
$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}.$$
- `operator -`: Realiza la resta de fracciones
$$\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}.$$
- `operator *`: Realiza el producto de fracciones
$$\frac{a}{b} * \frac{c}{d} = \frac{ac}{bd}.$$
- `operator /`: Reliza la división de fracciones
$$\frac{a}{b} / \frac{c}{d} = \frac{ad}{bc}.$$
- `operator float()`: Retorna el valor de la división en variable de tipo flotante.

También implementa la función `print()` la cual muestra al numerador y denominador de la siguiente manera

$$numerador/denominador$$

seguido de un salto de linea.

Problema 2. Realiza la implementación de *Disjoint Set* utilizando listas enlazadas deacuerdo con lo visto en el libro de *Cormen Introduction to Algorithms Third Edtion* página 564 y utilizando una de las heuristicas *union by rank* o *path compression*.

Problema 3. Soluciona el problema de Números Gordos (link) utilizando únicamente los temas y herramientas vistas en clase y ayudantías.

Entregable

Se entrega un archivo comprimido `.zip` con el siguiente formato `Apellido1Apellido2_Practica01.zip` el cual contiene un folder con los siguientes documentos.

- Un reporte tipo `pdf` con nombre `Apellido1Apellido2_Practica01.pdf`. En la sección **Reporte** se detalla los requisitos del `pdf`.
- Tres archivos `.cpp` con nombres `fraction.cpp`, `DSU_LinkedList.cpp` y `OmegaUp5701.cpp` los cuales tendrán la implementación de la clase `Fraction`, la implementación de DSU utilizando Linked-list y la solución al problema de números gordos.
- Imagenes de evidencia del `output` que demuestre el correcto funcionamiento de las implementaciones. Las imagenes deben tener nombres del estilo `px_IMGn` donde `x` es el número del problema y `n` es el número de imagen. En el caso del **problema 3** se tiene que añadir captura del juez de Omega Up que muestre el resultado AC y aparezca su cuenta.

Reporte

El reporte consta de seis secciones las cuales son:

1. **Fraction:** Explica el funcionamiento de la línea de código `template <typename T>`.
2. **DSU con Linked-list:** Explica la lógica de las implementación de DSU utilizando Linked-list y la diferencia con la implementación utilizando arreglos.
3. **Números Gordos:** Explica la lógica de la solución al problema y que herramientas utilizaste para la solución. Además justifica la elección de la heurística. También da la complejidad del algoritmo con notación O sin utilizar la definición.
4. **Problemas encontrado**: Mención de los problemas encontrados (si es que hubieron) al momento de implementar los algoritmos o comprenderlos.
5. **Conclusión**: Resumen breve de lo aprendido y posibles aplicaciones en la vida cotidiana.
6. **Referencias**: Enunciar las referencias utilizadas para la realización de la practica.

Código

Para el **problema 1** tenemos

Input: En la primera línea recibiremos 4 variables de tipo `T` a, b, c y d los cuales representan dos fracciones v_1 y v_2 tal que

$$v_1 = \frac{a}{b} \quad \text{y} \quad v_2 = \frac{c}{d}.$$

En la segunda línea tenemos un entero Q el cual representa el número de queries de tipo

- `S`: Realiza la suma de fracciones y almacena la respuesta en v_1. Luego utiliza la función `print()` de v_1.
- `R`: Realiza la resta de fracciones y almacena la respuesta en v_1. Luego utiliza la función `print()` de v_1.
- `D`: Realiza la división de fracciones y almacena la respuesta en v_1. Luego utiliza la función `print()` de v_1.
- `M`: Realiza la multiplicación de fracciones y almacena la respuesta en v_1. Luego utiliza la función `print()` de v_1.
- `F`: Imprime el valor al realizar la división en variable de tipo flotante.

Output: El output son Q líneas que representan la impresión de las Q queries.

Para el **problema 2** tenemos

Input: Dos enteros N y M que representa el número de nodos enumerados del 0 al $N-1$ y el número de parejas respectivamente. Seguido de esto tenemos M lineas con dos números p y q tales que $0 \leq p, q \leq N-1$ que representa una conección entre p y q.

Output: Por cada conección de elementos imprimir los representantes de cada nodo, es decir, M líneas presentando los representantes después de la conección de p y q.

Para el **problema 3** tenemos que el *input* y *output* será el la página de OmegaUp indica.