Universidad del Rosario.
Organizador de Currículo.
Camilo Gomez, Santiago ortiz

Repositorio: https://github.com/CamiloGomezV/algoritmos

La problemática principal para un estudiante con una cierta malla curricular, se presenta al momento de cancelar, perder, adelantar una materia, pues es cuando el tiempo que toma realizar sus estudios se puede ver afectado de no tener una buena organización. otro caso es cuando el estudiante quiere realizar doble programa o mención academia, puesto que, el ideal es repartir las materias de tal forma que se aproveche la cantidad de créditos por semestre, se avance en ambas carreras y realizar esto en el menor tiempo posible.

Justamente en esto se enfoca nuestra solución computacional, optimizar el tiempo que toma realizar una carrera o cumplir con una malla curricular en factor de las materias por ver, la solución, consiste en dado un arreglo de materias, sin importar el semestre en que se encuentre y teniendo en cuenta los requisitos de cada una de estas, realizar una distribución óptima de las mismas en los distintos semestres que sean necesarios, cada materia cuenta con ciertos parámetros, los cuales son: nombre de la materia, número de créditos, prerrequisito y etiqueta, este último parámetro se utiliza para diferenciar el tipo de materia, para nuestra implementación utilizamos dos conjuntos, básica y complementaria, en las básicas, se encuentra toda materia relacionada directamente con el programa y en las complementarias, se encuentran todas las electivas y materias de desarrollo personal, tales como, ética, constitución política, entre otras.

Para que una materia pueda ser vista en un cierto semestre, tiene que cumplir que su requisito ya se haya visto, y que al ver esta materia en el semestre, no se exceda el número de créditos matriculables, por otra parte se plantea una distribución arbitraria de las materias complementarias al no contar con un pre requisito.

Nuestro trabajo para poder llevar a cabo lo anteriormente planteado, con cada una de las pautas, fue implementacion de dos clases, la clase materia y la clase curriculum, que contiene a la estructura semestre, a partir de estas dos clases, se pensó y analizo los métodos pertinentes para cada una de ellas.

para la estructura de datos materia, se utilizan métodos principalmente informativos, es decir, métodos utilizados solamente para obtener información sobre la materia, estos métodos son:

- **get_priority**: este método requiere el uso de uno de los métodos privados de la clase con el fin de definir su prioridad dentro de todas las materias, esta prioridad se define según una función hash, que se basa en utilizar sus caracteres y la aritmética modular para definir su código o prioridad.
- **get_name:** como su nombre lo indica, con este método simplemente obtendremos el nombre de la materia.
- **get_request:** retorna un puntero tipo materia a el prerrequisito.
- **get_num_credits:** retorna el número de créditos de la materia.

- **get_label:** retorna el tipo de la materia, es decir, si la materia es básica, complementaria, electiva u opción de grado, estas se identifican mediante las letras B, C, E, D respectivamente.

es importante resaltar que todas estas funciones toman tiempo constante en su ejecución. adicional a ello, los operadores que se sobrecargaron fueron:

- **>** este operador tiene como objetivo poder realizar el correcto ordenamiento del currículo, la comparación se realiza mediante las prioridades de cada materia y, con este número, se determina si es mayor o no la materia.
- **==** esta comparación se realiza simplemente con los nombres de cada materia.
- **<<** utilizado para poder visualizar una materia, lo hace mediante su nombre.
- **()** este operador es de gran importancia, dado que, gracias a la utilización de los priority_queue, la organización y evaluación que hace, la tiene que hacer mediante algún comparador, ya sea menor que o mayor que, para poder acceder a este comparador, lo hace "consultado" el operador ()

los métodos principales de la clase curriculum son:

- **insert:** evalúa en qué semestre puede meter una materia evaluando si el prerrequisito ya está dentro de algún semestre anterior y si =, al insertar la materia, no se exceda la cantidad de créditos totales por semestre, en caso de que no exista un semestre para insertar la materia, crea uno nuevo
- **search:** busca una materia dentro del currículo, esto lo hace mediante una función auxiliar que recorre un semestre buscando la materia.
- **display:** este método, se utiliza para visualizar las materias en cada uno de los semestres dentro del currículo.

Otros métodos que se encuentran definidos en privado y son auxiliares o de ayuda para los principales, son:

- **create_new_semester:** como su nombre lo indica, crea un nuevo semestre y lo pone al final de la lista enlazada.
- **set_insert_control:** dentro de la estructura de datos, contamos con un puntero de nombre "insert_control" que, apunta al semestre al cual aún hay espacio hablando de la cantidad de créditos, en que se pueda insertar, el objetivo de esta función es, cuando el semestre cumple con la cantidad de créditos, es decir, no se puede insertar más en este, el puntero avanza al siguiente semestre, la función, evalúa si existe un semestre al cual avanzar o no, si no existe recurre a create_new_semester y luego el punto apuntará a este, de lo contrario, solo avanza
- **aux_insert:** función auxiliar utilizada para insertar una materia que tiene prerrequisito, esta, cumple con la condición de que una materia y su prerrequisito no pueden estar en el mismo semestre.
- **aux_search:** busca en un semestre la materia dada.

por último, hablaremos de los algoritmos implementados para esta solución intentando describir desde esta perspectiva la descripción detallada de la herramienta en conjunto. Dado que nuestro problema y solución constan de optimizar el tiempo, implementamos un algoritmo voraz de nombre maker, que consta de distintas partes, en primera medida se

utilizan cuatro priority queues para clasificar en ellos cada tipo de materia según su etiqueta y ordenarlos, con la comparación sobrecargada en la clase materia, estos, crean cuatro vectores, de ellos, se recorre cada uno para ir insertando en el currículo, este algoritmo, cumple con las características de un algoritmo voraz ya que los candidatos a ser insertados son cada una de las materias y en el conjunto solución se maximiza el número de créditos y se minimiza el número de semestres, cumpliendo así con las especificaciones del problema planteado. para entender la complejidad computacional de este algoritmo, es necesario evaluar primero los métodos que requiere.
para esto tendremos en cuenta dos magnitudes, N como el número de materias en el arreglo y M cantidad de datos en el curriculum. partiendo de ello, comenzando a analizar el search, este método en el peor caso se demora O(N) y en el mejor caso es O(1). Insert, en el peor caso se demora O(M) y en el mejor caso es O(1), los métodos create_new_semester y set_insert_control toman tiempo constante y los métodos aux_insert toman tiempo constante y aux_search en el peor caso toma O(M) y en el mejor caso demora O(1)
ahora, analizando nuestro algoritmo implementado, O(N+NM)