Estructura

15 de junio de 2024

1. Estructura

Clase Trie
var raiz : nodoTrie< T > : // vamos a definir algun tipo de nodo con su iterador bidireccional

Metodos:
proc nuevoTrie() : Trie
.. Creamos el trie vacio ..

proc esta(t:Trie, clave:string) : bool
.. devuelve true si la clave esta en el trie ..

proc definir y **proc** borrar hacen lo que se espera jaja

Clase Materia
var alumnos: ListaEnlazada< LU >
var docentes: Array< $\mathbb{N}$ >
var iguales: ListaEnlazada< $infoIguales$ >

- **infoIguales** es otra clase que va a tener como atributos el diccionario de la otra carrera que tiene la misma materia , y la clave para encontrarla en ese diccionario , despues explico por que hacer esto en el metodo borrar lol.

Metodos:

proc nuevaMateria(): Materia
this.alumnos = listaVacia()
this.docentes = new Array< $\mathbb{N}$ >(4)
this.iguales = listaVacia()

proc inscribir(m:Materia, LU:string)
.. aca hacemos m.alumnos.agregarAtras(LU) y cuesta O(1)

proc inscribirDocente(m:Materia,cargo:string)
.. aca si cargo es "PROFE"sumamos 1 en m.docentes[0] , si es "JTP"sumamos 1 en m.docentes[1] y asi etc , cuesta O(1) ..

proc agregarIgual(m:materia, t:Trie, c:Clave)
var nuevo = new nuevaInfo(t,c)
m.iguales.agregarAtras(nuevo)

cuesta O(1)

Clase infoIguales
var direccionCarrera: Trie< $Materia$ >
var claveMateria: string

Metodos
proc nuevaInfo(t:Trie< $Materias$ >, clave:string) : infoIguales
this.direccionCarrera = t
this.claveMateria = clave

Lo que hay que codear:

Modulo SIU **implementa** sistemaSIU
var carreras: Trie< $Trie < Materias \gg$
var estudiantes: Trie< $\mathbb{N}$ >

proc abrirSistema(infoMaterias:$seq\langle infoMateria\rangle$, Libretas:$seq\langle LU\rangle$): SIU
res = new SIU (no es necesario esta linea la pongo para claridad por ahora)
res.carreras = new Trie()
res.estudiantes = new Trie()

```
for (infoMateria info : infoMaterias)
    string[] carreras = info.carreras
    string[] materias = info.nombresEnCarreras
    Materia materia = new nuevaMateria()
    for (j = 0 ; j < carreras.length ; j ++ )
        if (res.carreras.esta(carreras[i]) == false) do
            nuevo = new Trie()
            nuevo.definir(materias[i],materia)
            res.carreras.definir(carreras[i],nuevo)
        else
            Trie mat = res.carreras.obtener(carreras[i])
            mat.definir(materias[i],materia)
        endif
        Trie direccion = res.carreras.obtener(carreras[i])
        materia.agregarIgual(direccion,materias[i])
```

- Vamos paso a paso con la complejidad.
- Cuando creamos el sistema tenemos que definir las claves de "*carrera*" y definir las claves de cada subtrie de "*carreras*", que vendrian a ser un subtrie de materias.
- La complejidad del for anidado se divide en 3:
  - Cuantas veces chequeo la guarda del if.
  - La asignacion adentro del if.
  - La asignacion cuando termino el if.

- Cuantas veces chequeo la guarda del if y la ultima asignacion tienen la misma complejidad , y la complejidad de la asignacion adentro if va a ser la misma a las otras 2 sumada la complejidad de definir cada subtrie de materias de cada carrera.
- Entonces por regla de complejidad quedaria O(F) + O(F + G) + O(F) = O(max(F,F) + O(F + G) = O(max(F,F+G) = O(F + G).
- Por lo tanto vamos paso a paso por la asignacion dentro del if.
- Suponiendo que tenemos C = [c1,c2,c3].Cada c_n me cuesta $|c_n|$ entrar al if.Cuantas veces voy a querer entrar if con cada c_n? voy a entrar con cada c la cantidad de veces que tenga materias en infoMaterias.
- Osea si c1 tiene 3 materias , entro al if 3 veces , si c2 tiene 2 materias , entro 2 veces y si c3 tiene 6 materias entrare al if 6 veces.
- Si $|Mc_n|$ es la cantidad de materias que tiene cada $|c_n|$ entonces con c1 entre al if $|c1| * |Mc_1|$ veces , con c2 entre $|c2| * |Mc_2|$ veces y con c3 $|c3| * |Mc_3|$ veces.
- Entonces en total entre al if $|c1| * |Mc_1| + |c2| * |Mc_2| + |c3| * |Mc_3|$ que es equivalente a haber entrado $\sum_{c \in C} |c| * |Mc|$ veces.

- Ahora calculare cuanto me cuesta asignar materias a los subtries de carreras.
- Cada materia sabemos que tiene su nombre especifico para cada carrera , osea cada materia m que pertenece al conjunto de materias M de infoMaterias tendra su conjunto Nm = [n1,n2,n3].
- Definir una clave en los subtrie "*materias*" me cuesta $|n|$.Cuantas veces voy a definir una clave en cada subtrie? Tantas veces tenga nombres en Nm.
- si Nm = [n1,n2,n3] , en el primer j = 0 defino la materia con su nombre n1 y me cuesta $|n1|$.Cuando j sea 1 , defino la materia con su nombre n2 y me cuesta $|n2|$.Cuando j sea 2 defino la materia con su nombre n3 y me cuesta $|n3|$.
- En total una vuelta entera del segundo for j me abra costado $|n1|+|n2|+|n3|$.Si tuvieramos una cantidad arbitraria de nombres entonces cada ciclo for j me va a costar $\sum_{n \in Nm} |n|$.
- Y cuantas veces voy a entrar al segundo for j? Tantas veces hayan m en M.Si M tiene 2 elementos , me costara j + j.Si M tiene 4 elementos me costara j + j + j + j.
- Si M tiene m elementos hare m vueltas con j osea $\sum_{m \in M} j$.Pero ya habiamos calculado cuanto nos costaba cada j.
- Reemplazando con sus valores la complejidad queda $\sum_{m \in M} \sum_{n \in Nm} |n|$.
- Sumando las complejidades hasta aca el proc tiene complejidad $0(\sum_{c \in C} |c| * |Mc| + \sum_{m \in M} \sum_{n \in Nm} |n|)$.
- Faltaria agregar la parte de agregar los estudiantes pero eso es agregar claves acotadas a un trie que en total tiene costo O(E) y ahi queda la complejidad del enunciado.

proc cerrarMateria(sistema:SIU, carrera:string, nombreMateria:string)

Trie materias = sistema.obtener(carrera) $\longrightarrow$O($|c|$)

Materia mat = materias.obtener(nombreMateria) $\longrightarrow$O($|m|$)

infoIguales mismas = mat.listaIguales() $\longrightarrow$O(1)

for (infoIguales iguales : mismas) do
    iguales.direccionCarrera.borrar(iguales.claveMateria)

- cada vuelta del for cuesta $|n|$ y voy a iterar tantas veces como nombres tenga la materia osea si tiene 3 hare $|n1| + |n2| + |n3|$, ergo $\sum_{n \in Nm} |n|$

ListaEnlazada< LU > alumnos = mat.alumnos() $\longrightarrow$O(1)

for (LU estudiante : alumnos) do
    $\mathbb{N}$ alumne = sistema.estudiantes.obtener(estudiante)
    alumne -= 1

- Esto cuesta en total O(EM).

2. Como corno funciona infoMaterias

- Segun los tests que nos dieron infoMateria funciona asi:
- la clase infoMateria tiene dos atributos
- carreras y nombresEnCarreras.
- a nosotros nos dan infoMaterias = [infoMateria1,infoMateria2,infoMateria3].
- si hacemos infoMateria1.carreras tenemos [Compu,Datos,Biologia] ponele.
- si hacemos infoMateria1.nombresEnMaterias tenemos [AED,AED2,AED] osea como se llama la misma materia en las distintas carreras.
- infoMateria1.carreras[0] corresponde a infoMateria1.nombresEnMaterias[0] , en el ejemplo seria que para compu AED se llama AED.
- Si aumentamos el indice y hacemos infoMateria1.carreras[1] y infoMaterias1.nombresEnCarreras[1] tenemos que para Datos AED se llama AED 2.
- AHora si hacemos infoMateria2.carreras tenemos [Compu,Datos,Biologia] y infoMateria2.nombresEnCarreras tenemos [Algebra,Algebra,Caballo].
- Ahora para la materia algebra , en Compu se llama algebra , en Datos algebra y Biologia Caballo.