Estructuras y complejidades

Lista Enlazada

Operación	Complejidad
listaVacía	$O(1)$
longitud	$O(1)$
vacía	$O(1)$
agregarAdelante	$O(cp(e))$
agregarAtras	$O(cp(e))$
fin	$O(1)$
comienzo	$O(1)$
primero	$O(1)$
ultimo	$O(1)$
obtener	$O(n)$
eliminar	$O(n)$
modificarPosicion	$O(n + cp(e))$
concatenar	$O(m * cp(e))$

Pila Sobre Lista

Operación	Complejidad
pilaVacía	$O(1)$
vacía	$O(1)$
encolar	$O(cp(e))$
desencolar	$O(1)$
tope	$O(1)$

Cola Sobre Lista

Operación	Complejidad
colaVacía	$O(1)$
vacía	$O(1)$
encolar	$O(cp(e))$
desencolar	$O(1)$
proximo	$O(1)$

Vector

Operación	Complejidad
vectorVacío	$O(1)$
longitud	$O(1)$
vacía	$O(1)$
agregarAdelante	$O(f(n) + cp(e))$
agregarAtras	$O(f(n) + cp(e))$
fin	$O(n)$
comienzo	$O(n)$
primero	$O(1)$
ultimo	$O(1)$
obtener	$O(1)$
eliminar	$O(n)$
modificarPosicion	$O(1)$
concatenar	$O(m)$

Conjunto Lineal

Operación	Complejidad
conjVacío	$O(1)$
tamaño	$O(1)$
pertenece	$O(n * eq(T))$
agregar	$O(n * eq(T) + cp(e))$
agregarRapido	$O(cp(e))$
sacar	$O(n * eq(T))$
unir	$O(n * m * cp(T) * eq(T))$
restar	$O(n * m * eq(T))$
intersecar	$O(n * m * eq(T))$

Conjunto Log

Operación	Complejidad
conjVacío	$O(1)$
tamaño	$O(1)$
pertenece	$O(\log n * eq(T))$
agregar	$O(\log n * eq(T) + cp(e))$
agregarRapido	$O(\log n * eq(T) + cp(e))$
sacar	$O(\log n * eq(T))$
unir	$O((n + m) \log(n + m) * cp(T) * eq(T))$
restar	$O((n + m) \log(n + m) * eq(T))$
intersecar	$O((n + m) \log(n + m) * cp(T) * eq(T))$

Conjunto Digital

Operación	Complejidad
conjVacío	$O(1)$
tamaño	$O(1)$
pertenece	$O(\|e\| * \|a\|)$
agregar	$O(\|e\| * \|a\| * eq(a) + cp(e))$
agregarRapido	$O(\|e\| * \|a\| * eq(a) + cp(e))$
sacar	$O(\|e\| * \|a\| * eq(a))$
unir	$O(m * max(c2) * \|a\| * eq(a))$
restar	$O(n * m * \|a\|)$
intersecar	$O(n * max(c1) * m * max(c2))$

Diccionario Lineal

Operación	Complejidad
dictVacío	$O(1)$
está	$O(n)$
definir	$O(n * eq(K) + cp(k) + cp(v))$
definirRapido	$O(n * eq(K) + cp(k) + cp(v))$
obtener	$O(n)$
borrar	$O(n)$
tamaño	$O(1)$

Diccionario Log

Operación	Complejidad
dictVacío	$O(1)$
está	$O(\log n * eq(K))$
definir	$O(\log n * eq(K) + cp(k) + cp(v))$
definirRapido	$O(\log n * eq(K) + cp(k) + cp(v))$
obtener	$O(\log n * eq(K))$
borrar	$O(\log n * eq(K))$
tamaño	$O(1)$

Diccionario Digital

Operación	Complejidad
dictVacío	$O(1)$
está	$O(\|k\| * \|a\| * eq(a))$
definir	$O(\|k\| * \|a\| * eq(a) + cp(k) + cp(v))$
definirRapido	$O(\|k\| * \|a\| * eq(a) + cp(k) + cp(v))$
obtener	$O(\|k\| * \|a\| * eq(a))$
borrar	$O(\|k\| * \|a\| * eq(a))$
tamaño	$O(1)$

Referencias

Nombre	Explicacion
n	Cantidad de elementos de la estructura
m	Cantidad de elementos de una segunda estructura
$cp(e)$	Costo de copiar el elemento e
$f(n)$	$f(n) = \begin{cases} n & \text{si } n = 2^k \text{para algún k} \\ 1 & \text{en caso contrario} \end{cases}$
$eq(T)$	Costo de comparar elementos de tipo T
$cp(T)$	Costo de copiar elementos de tipo T
$max(c1)$	La clave mas grande del primer conjunto
$max(c2)$	La clave mas grande del segundo conjunto
$eq(K)$	Costo de comparar claves definidas en el diccionario
$cp(k)$	Costo de copiar la clave
$cp(v)$	Costo de copiar el valor

Sorting

Algoritmos de ordenamiento

Nombre	Mejor caso	Caso promedio	Peor caso	Es estable
Selection	$O(n^2)$	$O(n^2)$	$O(n^2)$	No
Insertion	$O(n)$	$O(n^2)$	$O(n^2)$	Si
Merge	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	Si
Quick	$O(n \log(n))$	$O(n \log(n))$	$O(n^2)$	No
Bubble	$O(n)$	$O(n^2)$	$O(n^2)$	Si
Heap	$O(n \log(n))$	$O(n \log(n))$	$O(n \log(n))$	No

TADs

Invariante de representación y Función de abstracción

`pred` *invRep* (*in implementacion* : `ImplementacionDeTad`) {
    Todas las condiciones que las variables deban cumplir para que tu implementacion sea valida
}

`proc` *funAbs*(*in implementacion* : `ImplementacionDeTad`, *out instanciaDeTad* : `Tad`){
    Describir todos los observadores del TAD en base a las variables de tu implementacion (no hace falta usarlas todas, pero si abarcar al TAD completo)
}