Obligatorio de Estructura de Datos y Algoritmos 2

- Nombre: Pedro Benjamín Chialanza Arrieta
- Número de Estudiante: 302782

Ejercicio 01

Solución

La solución de este ejerció es indicado por la misma letra que indica el uso de la estructura AVL.

Justificación de Orden de Cada Acción

ADD

La accion ADD creo el objeto Book y lo inserto en el AVL con el metodo clasico de insercion con O(log(n)), ademas hay un metodo que es llamado cuando la insercion sucede para mantener la constancia de libros habilitaodos y deshabilitados

FIND

La accion FIND consigue la key y busca en el AVL con su respectivo algoritmo de busquedao asegurando la un O(log(N))

TOGGLE

La accion TOGGLE consigue la key y busca en el AVL con su respectivo algoritmo de busquedao asegurando la un O(log(N)) y ademas de eso le modifica los datos pedidos tomando constancia de los libros habilitaods y deshabilitados

COUNT

La accion COUNT consigue el count directamente de la estructura que lleva cuenta de cuantos nodos tiene permitiendo que su O(1).

Ejercicio 02

Solución

La solución de este ejerció es indicado por la misma letra que indica el uso de la estructura tabla hash cerrado con solucion de colisión doble hash.

Justificación de Orden de Cada Acción

ADD

La accion ADD arma el objeto y la inserta en el HashTable con el metodo de seteado de orden constante e igual que el ejericio anterios hay una funcin que mantiene el estado de libros habilitados o no;

FIND

La accin FIND utiliza el metodo clasico de HashTable permitiendo una busqueda de orden constante promedio.

TOGGLE

La accion TOGGLE consigue la key y busca en el HashMap con su respectivo algoritmo de busquedao asegurando la un O(1) promedio y ademas de eso le modifica los datos pedidos tomando constancia de los libros habilitaods y deshabilitados

COUNT

La accion COUNT es de orden constante tambien porque el HashTable toma constancia de la cantidad de elementos que tiene guardados.

Ejercicio 3

Solucion

Al tener varios objetos repetidos y elegir el que tiene menor precio se usuaria un minheap y un hashtable para no comprar obejetos repetidos.

Justificación de Orden de Cada Acción

Primera Etapa

Utilizo el metodo de insercion del heap siendo o(1) promedio y peor caso O(log(N))

Segunda Etapa

Utilizo el metodo pop de Heap para extraer el elemento con la mayot prioridad y seteo ese ebjeto en el HashTable para saber que ya lo compre.

Ejercicio 4

Solucion

Se utilizo el heap para mantener el un order y un hashmap para ver si el pedido ya fue entregado

Justificacion de Orden de Cada Accion

I

La accion I utiliza el metodo del HashHeap que seria O(1) promedio ademas de setear con el metodo de HashTable el indice que esta ubicado manteniendo la constancia del indice despues de hundir y flotar. Ademas setea esa key en un HashTable para saber si ese pediddo ya fue entregado

E

La accion "E" elmina el peidido entregado del HashTable con su metodo de eliminacion con O(1) promedio

C

La accion C consigue el elemento del HashHeap con el metodo get con un orden constante porque usa la HashTable dentro de la misma para conseguir su indeice, y cambia sus valores y rebalance esa celda con la nueva prioridad con flotar y hundir. ademas consigo el mismo objeto y lo cambio en el HashTable con el mismo metodo get con un orden costante.

Resolucion

El print utiliza get de HashTable y el pop de HashHeap que es my similar a el de Heap simplemente elimina la key del HashTable dentro del HashHeap.

Ejercicio 5

Solucion

Se armo dos SparseGraph (Grafo Disperso) de misiones y cuidades. Se hace un dijkstra de de las ciudades con el origen siendo el origen brindado, y el dijkstra de cada cuidad para despues de hacer un order topologico de las misiones.

Justificacion de Ordenes

Dijkstra

Algoritmo clasico con un O((V + E)log(V))

Orden Topologico

Algoritmo clasico con un O(V + E) reteo el heap y inserto los mismos valor con prioridad actualiza y se puede consider O(1) porque la cantidad de elementos que hay es

despreciable, ademas de inprimir el camino para llegar que tambies es despreciable

findVertex

Es orden constante utiliza un HashTable para conseguir el indice de un vertice en el grafo.

Ejercicio 6

Solucion

Uso el dijkstra mencionado anteriormente y entre medio de cada uno llamo una funcion que actualiza los caminos tomados duplicando su costo. Creo dos grafos una para desactivar primero la Entidad y el otro buscar el equipo, por ultimo los comaparo y los imprimo como las pruebas lo pide.

Justificacion de Ordenes

El orden del dijkstra es el mismo que el ejercicio 6 siendo O((V + E)log(V)) y la la funcion de updatePath que seria el peor caso O(E) (porque puede recorrer todo los aristas). Siendo el orden del ejercicio completo O(2((V + E)log(V) + E) + 2(V + E)log(V)), siendo O((V + E)log(V)).

Ejercicio 7

Solucion

La solucion consiste en un Divide and Conquer y un Greedy, si consideramos una barra de tiempo total de las canciones, se toma uno de base valido (que se escuchan todas las canciones) y se hace el divide se prueba de los dos lados y actualizo la base (con el de menor tiempo) y se repite, pero tambien con cada repeticion se achica la distacia la cual resta o se uma la base. Luego tenemos la parte Greedy que es la ejecuccion de los estudiantes escuchan canciones y este trata de hacer que el estudiante escuche todas las canciones posibles sin superar el maximo tiempo este siendo establecido por el divide.

Justificacion de Ordenes

Si imaginamos la barra como un array ordenado y se hace un binary search, nada mas que con cada repeticion se prueba valores cada vez mas cercano de la base siendo O(log(T)) T siendo el tiempo total de todas las canciones, y con cada repeticion se ejecuta el parte Greedy que intenta eschar las N canciones siendo de O(N). Entonces el O(log(T)N).

Ejercicio 8

Solucion

Teniedo todas las cuidades se ordenan por la coordenada y y se hace un divide and conquer tomando la mitad del area y se evalua cada lado hasta que haya una cantidad manejable obteniedo la mejor distancia efectiva en esa zona y caso, luego se compara el de ambos lados y se toma el mas chico, esto se repite hasta conseguir un resultado. Este no conseguira el mejor resultado porque cerca de la linea divisoria puede haber dos cuidades mas cerca (esas dos cuidades son separadas por la linea divisoria).
El metodo de resolucion de la linea divisoria es aumentar el low y reducir el high achicando la zona, este cumpla la condicion que las de la zona se encuentran cierta distancia de la ciudad central. Paso 2, al principio lo que hicia es recorrer devuelta la funcion con esta nueva zona, pero hay caso borde que seria que generas una nueva linea y no logras achicar la zona de la forma mencionada anteriormente (generado un bucle infinito). Esta forma fue descartada, se siguo achicando el area de la misma forma pero en vez de pasarlo de vuelta por la funcion se pasan a otro array y se ordena por x y se prueba cada pareja para encontrar una pareja de ciudades mas cercana, para que no sea tan lento se aplican ciertas podas.

Justificacion de Ordenes

El orden seria O((N log(N)) + (2N * N log(N)))

Ejercicio 9

solucion

tomo todos los datos de los jugadores, creo la matriz de cuadro dimension sindo la primera coordena los jugadores considerados, la seunda el presupuesto, cantidad de extrajeros habilitados y por ultimo cuantos jugadores iban a estar en la formacion final.
primeri inserto el primer jugador para no tener que preocuparnos de el acceso i - 1, tom encueanta que no pueden aceptar mas jugadores extranjeros y no pueden pagarle el sueldo.

Justificacion de Ordenes

orden de tiempo y espacio siend j *p* e * 11

Ejercicio 10

solucion

Lo primero que hago es ordenar las flores de mayor restriccion primero porque las quiero insertar primero para limitar mis opciones al ejecutar el backtracking, ademas inserto primero en los lugares que se pueden insertar menos flores por la misma razon dicha anteriormente, tambien tengo una matriz que me dice cuantas restricciones me impide plantar una en ese en vez de fijarme alrededor, y por ultimo si ya plante 5 flores y me qudan

vistar 3 celdas y mi maximo de flores planteadas son 10 corto esa rama completamente porque no es posible superaro.

La matriz con el numero de restricciones que me impide plantar esa flor es precarcado con las restricciones de la fila de las flores y cuando planto una flor marco alrededor que no se pudede plantar la flor sumandole 1 y hago lo contrario de enamrcadola.

Justificacion de Ordenes

El orden seria O(N*Nlog(N* N) + F + N*N*2F + N*Nlog(N*N) + N * F)

N dimension de la matriz y F cantidad de flores

Problema	Resultado
1	Completo
2	Completo
3	Completo
4	Completo
5	Completo
6	Completo
7	Completo
8	Completo
9	Completo
10	Completo