Introduccion

Representar informacion es fundamental en la informatica y la computacion

- El principal proposito de muchos programas computacionales no es ejecutar calculos, sino almacenar y recuperar informacion, generalmente tan rapido como sea posible.
- Por esta razon el estudio de las estructuras de datos y de los algoritmos que las manipulan constituye el nucleo central de la informatica y de la computacion.

Estructuras de datos

En el sentido mas general, una estructura de datos es cualquier representacion de datos y sus operaciones asociadas. Inclusive un numero de punto flotante almacenado en la computadora puede ser visto como una estructura de datos simple.

- Sin embargo, comunmente el termino estructura de datos se utiliza para indicar una organizacion o estructura para una coleccion de items de datos.
- Una lista ordenada de enteros almacenados en un arreglo es un ejemplo de este tipo de estructuras.

Resolucion de problemas

Existen muchas formas de resolver un problema. Como elegir entre ellas? Como eje central del diseno de programas computacionales hay dos metas (algunas veces conflictivas):

1. Disenar un algoritmo que sea facil de entender, codificar y depurar
2. Disenar un algoritmo que haga uso eficiente de los recursos de la computadora

Idealmente, el programa resultante deberia cumplir ambas metas. en ese caso se dice que se obtiene una solucion elegante.

Eficiencia de una solucion

Una solucion se dice que es eficiente si esta resuelve un problema dentro de las restricciones de recursos preestablecidas. ejemplos de restricciones incluyen:

- el espacio total disponible para almacenamiento
- el tiempo permitido para ejecutar cada subtarea

Eficiencia de una solucion

A veces se dice que una solucion es eficiente si utiliza menos recursos que otras soluciones conocidas, en lugar de si esta cumple con una restriccion particular.

- El costo de una solucion es la cantidad que recursos que una solucion consume.
- A veces el costo se mide en terminos de algun recurso clave como el tiempo, con la suposicion de que cumple con todas las otras restricciones de recursos.

Elegir el diseno adecuado

Se debe empezar realizando un analisis del problema para determinar las metas de rendimiento que se deben cumplir. esto permitira seleccionar la estructura de datos mas adecuada para el trabajo que se va a realizar.

- Un programa mal disenado ignora este analisis y utiliza aquella estructura de datos mas familiar pero posiblemente inapropiada para el problema.
- El resultado generalmente son programas lentos e ineficientes.
- De igual manera no existe necesidad de adoptar una representacion compleja cuando un programa puede brindar un buen rendimiento utilizando un diseno simple.

Pasos del analisis del problema

Cuando se selecciona una estructura de datos para resolver un problema, se deben seguir los siguientes pasos:

1. Analizar el problema para determinar las operaciones basicas que deben soportarse. ejemplos son: agregar, borrar, modificar, consultar, etc.
2. Cuantificar las restricciones de recursos para cada operacion
3. Seleccionar la estructura de datos que mejor cumple con estos requerimientos

Seleccion de la estructura de datos

Las restricciones de recursos en ciertas operaciones claves tales como la busqueda, insercion y borrado; generalmente conducen el proceso de seleccion de la estructura de datos. Se deben tener en mente las siguientes cuestiones:

- Todo los datos seran agregados al desde el inicio, o las inserciones iran intercaladas con otras operaciones ? Las aplicaciones estaticas (donde se cargan los datos desde el inicio y estos nunca cambian) generalmente requieren estructuras de datos simples.

Seleccion de la estructura de datos

- Pueden ser borrados los items de datos ? esto puede hacer la implementacion mas complicada.
- Son todos los datos procesados en algun orden bien definido, o se permite la busqueda para items de datos especificos ? el acceso aleatorio requiere generalmente estructuras de datos mas complejas.

Costo y beneficio

Cada estructura de datos tiene asociados costos y beneficios. Una estructura de datos requiere cierta cantidad de espacio para cada item de datos que almacena, cierta cantidad de tiempo para ejecutar un operacion simple, y cierta cantidad de esfuerzo de programacion.

- Cada problema tiene restricciones en el espacio y tiempo disponibles
- Cada solucion a un problema utiliza algunas operaciones basicas en cierta proporcion, y en la seleccion de la estructura de datos se debe tomar en cuenta esto.

Tipos de datos abstractos y algoritmos

Aunque los terminos tipos de datos, estructura de datos y tipo de datos abstracto parecen semejantes, su significado es diferente:

- el tipo de datos de una variable es el conjunto de valores que esta puede tomar
- un tipo de datos abstracto (TDA) es un modelo matematico, con varias operaciones definidas sobre ese modelo
- para representar los TDAs se emplean estructuras de datos, que son conjuntos de variables, quiza de tipos distintos, conectadas entre si de diversas formas.

Tipos de datos

- Un tipo es una coleccion de valores. Por ejemplo, el tipo booleano consiste de los valores TRUe y FALSe. Se dice que un tipo es simple si este no contiene subpartes. Un registro por ejemplo, se considera un tipo compuesto o agregado.
- Un tipo de datos es un tipo junto con una coleccion de operaciones para manipular el tipo. Por ejemplo, una variable entera cuenta con operaciones de suma, resta, multiplicacion y division.

Clasificacion de tipos de datos

Los lenguajes de programacion proporcionan tipos de datos para clasificar clases de datos. Los tipos de datos se pueden clasificar en:

- Tipos simples o primitivos
- Tipos compuestos o agregados