Introducción

Representar información es fundamental en la informática y la computación.

- El principal propósito de muchos programas computacionales no es ejecutar cálculos, sino almacenar y recuperar información, generalmente tan rápido como sea posible.
- Por esta razón el estudio de las estructuras de datos y de los algoritmos que las manipulan constituye el núcleo central de la informática y de la computación.

Estructuras de datos

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

- Sin embargo, comúnmente el término estructura de datos se utiliza para indicar una organización o estructura para una colección de ítems de datos.
- Una lista ordenada de enteros almacenados en un arreglo es un ejemplo de este tipo de estructuras.

Resolución de problemas

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

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

Idealmente, el programa resultante debería cumplir ambas metas. En ese caso se dice que se obtiene una solución "elegante".

Eficiencia de una solución

Una solución 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 solución

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

- El “costo” de una solución es la cantidad que recursos que una solución consume.
- A veces el costo se mide en términos de algún recurso clave como el tiempo, con la suposición de que cumple con todas las otras restricciones de recursos.

Elegir el diseño adecuado

Se debe empezar realizando un análisis del problema para determinar las metas de rendimiento que se deben cumplir. Esto permitirá seleccionar la estructura de datos más adecuada para el trabajo que se va a realizar.

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

Pasos del análisis 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 básicas que deben soportarse. Ejemplos son: agregar, borrar, modificar, consultar, etc.
2. Cuantificar las restricciones de recursos para cada operación
3. Seleccionar la estructura de datos que mejor cumple con estos requerimientos

Selección de la estructura de datos

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

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

Selección de la estructura de datos

- Pueden ser borrados los ítems de datos ? Esto puede hacer la implementación más complicada.
- Son todos los datos procesados en algún orden bien definido, o se permite la búsqueda para ítems de datos específicos ? El acceso aleatorio requiere generalmente estructuras de datos más complejas.

Costo y beneficio

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

- Cada problema tiene restricciones en el espacio y tiempo disponibles
- Cada solución a un problema utiliza algunas operaciones básicas en cierta proporción, y en la selección de la estructura de datos se debe tomar en cuenta esto.

Tipos de datos abstractos y algoritmos

Aunque los términos “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 está puede tomar
- un tipo de datos abstracto (TDA) es un modelo matemático, con varias operaciones definidas sobre ese modelo
- para representar los TDAs se emplean estructuras de datos, que son conjuntos de variables, quizá de tipos distintos, conectadas entre sí de diversas formas.
- Tipos de datos
- Un “tipo” es una colección 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.

Clasificación de tipos de datos

Los lenguajes de programación 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