

## Resumen AED CAP 16

Antes de desarrollo, Software es esencial revisar los objetivos del proyecto, que pueden variar según la organización. Estos objetivos proporcionan una idea de lo que se debe construir o cambiar, pero no indican cómo lograr la velocidad, costo y facilidad de mantenimiento deseado.

### ¿Qué tipo de datos?

Los datos en un problema informático varían según el dominio (como transacciones empresariales o grabaciones de animales). Se construyen a partir de tipos primitivos y se agrupan en formas como cadenas de texto o arrays. Al elegir estructuras de datos, es clave considerar cómo se accede a los y si hay claves o índices únicos.

### ¿Cuántos datos?

Después de definir los tipos de datos y su manipulación, la clave es determinar cuantos grupos de datos existen y cuál importa la elección de la estructura de datos. Anticipar el crecimiento de la cantidad de datos es crucial, porque los proyectos comienzan con un alcance pequeño.

## BUSCAR LIMITACIONES EXTERNAS

Si los datos tienen restricciones es probable que el código no necesite procesar más allá de cierta cantidad. Por ejemplo, si el sistema solo almacena controles conocidos de una persona, es improbable que supere los 1000 elementos. Un sistema para registrar uno o dos elementos por persona podría manejarlo con éxito hasta alcanzar miles (o miles de millones) de elementos.

¿Qué genera los datos?

Datos manuales, limitados por tiempo y participación humana, contrastan con datos automáticos como información astronómica, que pueden ser mucho mayores. Las subestimaciones son comunes ya que proxetas iniciales o menores no anticiparon completamente la necesidad de escalar para grandes cantidades de dato.

¿Qué operaciones y con qué frecuencia?

La planificación del sistema debe considerar operaciones de mantenimiento como la actualización de modelos de predicción meteorológica y la adaptabilidad ante la variabilidad de la demanda en sistema ejecutado bajo demanda.

El diseño de un sistema implica listar las operaciones estimar su frecuencia y requisitos de rendimiento. La elección de estructura de datos se basa en la frecuencia relativa de operaciones, como inserción, búsqueda, etc.

eliminaciones, búsquedas y recorridos. Las restricciones algorítmicas y la necesidad de operaciones rápidas ayudan en la elección de estructuras de datos especializadas.

### • Quién manejará el software?

La elección de estructuras y algoritmos depende de la habilidad del personal. En entornos técnicos las opciones son amplias, pero en entornos menos técnicos, las elecciones deben ser más simples para facilitar el mantenimiento, cumplimientos, cumpliendo siempre con los requisitos de rendimiento.

### Estructura de Datos fundamentales

Las estructuras de datos fundamentales como arrays, lista enlazadas, árboles tablas hash, se construyen sobre tipos primitivos y formas de organizarlos. Véase cuán como se abordan los elementos y cómo se asigna dicha asignación memoria. La elección dependerá de las características del problema arrays indexadas, listas y árboles para referencia y tablas hash para abordar por clave.

### Velocidad y Algoritmo

Estructuras de datos varían en velocidad: matrices y listas son lentas, árboles y diccionarios rápidos, tablas hash muy rápidas. Los más rápidos pueden ser más complejos de programar y más eficientes en memoria.

## Velocidad de programación en un objetivo en movimiento

El rápido avance tecnológico se ve contrarrestado por el software más complejo, afectando el rendimiento de las estructuras de datos. Se sugiere empezar con soluciones simples y evaluar el rendimiento antes de optar por opciones más complejas adaptando el diseño según sea necesario.

## Librerías

Librerías comerciales ofrecen estructuras de datos predefinidas en varios lenguajes, reduciendo la programación necesaria. Analizar sus características vitales para garantizar adaptabilidad al problema específico.

## Arrays

Los arrays son eficientes para datos predicibles de tamaño razonable y acceso frecuente por índices fijos. Mújor. con tamaño asignados una vez, pero su eficiencia disminuye con crecimiento.

## LinkedList

Considera una lista enlazada para cantidades de datos no predicibles, minimizando el uso de memoria y con frecuentes inserciones/eliminaciones que mantienen el orden. Son rápidas para inserciones.

## Binary Search Trees

Un árbol de búsqueda binaria es útil para ordenar y acceder a datos por clave con inserciones diferentes y frecuentes y bajo acceso a la memoria. Ofrece operaciones rápidas  $O(\log N)$  pero evita su uso.

## Balanced Search Trees

Árboles de búsqueda equilibrados (2-3-4-rojo-negro, AVL) ofrecen rendimiento  $O(\log(N))$  con datos ordenados. Difieren en que programan óptimo para inserciones / eliminaciones frecuentes.

## Hash Tables

Hash Tables tienen el mejor rendimiento  $O(1)$  para datos referenciados por clave, ideal cuando el orden no importa y la memoria no es crítica. No son sensibles al orden de insercción y son muy simples que árboles balanceados pero pueden usar más memoria.

## Stack

Un stack es una estructura de datos de último en entrar primero en salir (LIFO) que proporciona acceso rápido solo al último insertado. Se implementa comúnmente como array o LinkedList.

## cola

Una cola se utiliza cuando solo se necesita acceder al primer elemento de datos insertando y quitando una estructura de tipo primero en otros, primero en salir (FIFO). Las colas pueden implementarse como matrices o listas vinculadas ambas con inserción y eliminación eficiente en O(1)

## cola de prioridad

Una cola de prioridad se utiliza cuando el único acceso deseado es el elemento de datos con la mayor prioridad determinado por la clave más grande. Pueden implementarse como una matriz ordenada o como un montículo. La inserción en una matriz ordenada es lenta, pero la eliminación es rápida.

## sorting

para datos en memoria, miso con el ordenamiento por inserción, y si es necesario, considera algoritmos más complejos como shellsort o quicksort. Para situaciones complejas

## Graphs

para modelar situaciones del mundo real son esenciales, como en redes de transporte o algoritmos como PageRank. La elección clave es entre matriz, díadiagonal para grafos densos y lista díadiagonal o tablas hash para grafos dispersos. Operaciones como búsqueda en profundidad o amplitud y encontrar árboles y subgrafos varían en complejidad según la representación elegida. La elección dependerá de la densidad del grafo y se debe estimar el número de vértices y orígenes para tomar la decisión adecuada.