Metodos de ordenamiento

Ordenar una determinada información consiste en disponer un conjunto de datos en algún determinado orden respecto de sus valores o de algún campo que compone una determinada estructura.

El orden tiene que ver con la relación de una cosa con otra y puede ser ascendente o descendente.

Hay numerosos métodos de ordenamiento que difieren en eficiencia. Si bien el análisis de la eficiencia es más complejo está orientado a la cantidad de comparaciones realizadas por cada uno. Las comparaciones serán en función de "n". Siendo "n" el tamaño del vector a ordenar.

Los lenguajes de programación modernos incluyen funciones de biblioteca para ordenar arreglos , pero a veces es necesario implementar alguno de los muchos algoritmos de ordenamiento que se conocen.

Ordenamiento método de la burbuja.

Este algoritmo va comparando cada elemento del arreglo con el siguiente; si un elemento es mayor que el que le sigue, entonces se intercambian; esto producirá que en el vector quede como su último elemento, el más grande. Este proceso deberá repetirse recorriendo todo el vector hasta que no ocurra ningún intercambio. Los elementos que van quedando ordenados ya no se comparan.

Algoritmo de la burbuja

```
...
    int aux;   //Auxiliar para hacer el intercambio
    int datos[] = {81, 22, 35, 18};
    system("cls");
    for (int i = 0; i < n - 1; i++)
    {
        for (int j = 0; j < n - i-1; j++)
        {
            if (datos[j] > datos[j + 1])
            {
                aux = datos[j];
                datos[j] = datos[j + 1];
                datos[j + 1] = aux;
            }
        }
    }
...
```

Ordenamiento método de selección

El algoritmo consiste en encontrar el menor de todos los elementos del vector e intercambiarlo con el que está en la primera posición. Luego el segundo más pequeño, y así sucesivamente hasta ordenarlo todo.

Algoritmo de selección

```
...
int n = 4;
    int datos[] = {19, 35,34, 18};
    int i, j;
    int indiceMenor, aux;
    for (int i = 0; i < n - 1; i++)
    {
        indiceMenor = i;
        for (int j = i + 1; j < n; j++)
        {
            if (datos[j] < datos[indiceMenor])
                indiceMenor = j;
        }
        if (i != indiceMenor)
        {
            aux = datos[i];
            datos[i] = datos[indiceMenor];
            datos[indiceMenor] = aux;
        }
}
...
```

Ordenamiento método de inserción

El ordenamiento por inserción consiste en tomar uno por uno los elementos de un arreglo y colocarlo en su posición correcta dentro de una lista ordenada. Así empieza con el segundo elemento y lo ordena con respecto al primero. Luego sigue con el tercero y lo coloca en su posición ordenada con respecto a los dos anteriores, así sucesivamente hasta recorrer todas las posiciones del arreglo.

Algoritmo de inserción

```
...
 int n = 4;
    int datos[] = {81, 22, 35, 18};
    int i, j, aux;

    for (i = 1; i < n; i++)
    {
        j = i;
        aux = datos[i];
        while (j > 0 && aux < datos[j - 1])
        {
            datos[j] = datos[j - 1];
            j--;
        }
        datos[j] = aux;
    }
...
```

Métodos de búsqueda

La búsqueda de un elemento dentro de un array es una de las operaciones más importantes en el procesamiento de la información,también se conoce como recuperación o searching y permite la recuperación de datos previamente almacenados.

El tipo de búsqueda se puede clasificar como interna o externa, según el lugar en el que esté almacenada la información (en memoria o en dispositivos externos).

La finalidad de estos algoritmos es: determinar si el elemento buscado se encuentra en un conjunto determinado y si el elemento está en el conjunto, hallar la posición en la que se encuentra.

Existen diversos algoritmos o técnicas de búsqueda, pero la elección dependerá de la forma en que se encuentran almacenados los datos. Dichas técnicas se conocen con el nombre de búsqueda secuencial o lineal y búsqueda binaria o dicotómica.

Búsqueda Secuencial o lineal

Es la forma más sencilla de búsqueda. Consiste en comparar cada elemento del listado contra el valor deseado.

Consiste en recorrer y examinar cada uno de los elementos del array hasta encontrar el o los elementos buscados, o hasta que se han mirado todos los elementos del array, por lo tanto , tiene dos condiciones de finalización, cuando se encuentra el valor o cuando se termina de leer por completo los datos del array.

Las ventajas de este método es que es sencillo de implementar ya que únicamente se utilizan estructuras repetitivas. No requiere que el listado esté ordenado o en algún estado especial. Las desventajas son por un lado la poca eficiencia ya que se debe recorrer por completo el arreglo, por otro lado el tiempo excesivo de procesamiento en arrays muy grandes.

Algoritmo de búsqueda secuencial

```
//Utilizando un ciclo while
...
  int n = 10;
  int datos[] = {56, 22, 30, 18, 67, 18, 99, 74, 16, 40};
  int valor = 18;
  int pos = 0;
  while (pos < n && datos[pos] != valor)
    {
        pos++;
    }
...
```

El algoritmo realiza dos comparaciones en cada iteración, una para comparar si se encuentra el elemento, la otra para evaluar si ya se llegó al final del vector.

Búsqueda Binaria o dicotómica

Si los elementos sobre los que se realiza la búsqueda están ordenados, entonces podemos utilizar un algoritmo de búsqueda mucho más rápido que el secuencial, la búsqueda binaria.

Es eficiente para conjuntos grandes de datos. Requiere que los datos estén ordenados. Se basa en la división sucesiva del array en sucesivas mitades, es decir que está basado en la técnica de "divide y vencerás"

El algoritmo consiste en reducir paulatinamente el ámbito de búsqueda a la mitad de los elementos, comparar el elemento a buscar con el elemento que se encuentra en la mitad del intervalo y en base a esta comparación se tienen en cuenta las siguientes consideraciones:

- Si el elemento buscado es menor que el elemento central, entonces estamos seguros que el elemento está en la mitad inferior de la lista.
- Si es mayorestamos seguros que el elemento está en la mitad superior de la lista.
- Si es igual al elemento central, se procede a finalizar la búsqueda ya que se ha encontrado el elemento con éxito.

Algoritmo de búsqueda binaria

```
//REQUIERE QUE LOS DATOS ESTÉN ORDENADOS
...
  int n = 9;
    int datos[] = {-8, 4, 5, 9, 12, 18, 25, 40, 60};
    int valor = -8;
    int izq = 0;
    int der = n - 1;
    int pos, p=-1;
    while (izq <= der && p==-1)
    {
        pos = (izq + der) / 2;
        if (valor == datos[pos])
        {
            p = pos;
        }
        else
        {
            if (valor < datos[pos])
            {
                der = pos - 1;
            }
            else
            {
                izq = pos + 1;
            }
        }
    }
...
```