El lenguaje de programación C
- Búsqueda – Ordenación

Algoritmos de Búsqueda

- Los procesos de búsqueda involucran recorrer un array completo con el fin de encontrar algo. Lo más común es buscar el menor o mayor elemento (cuando es puede establecer un orden), o buscar el índice de un elemento determinado.
- Para buscar el menor o mayor elemento de un array, podemos usar la estrategia, de suponer que el primero o el último es el menor (mayor), para luego ir comparando con cada uno de los elementos, e ir actualizando el menor (mayor). A esto se le llama Búsqueda Lineal.

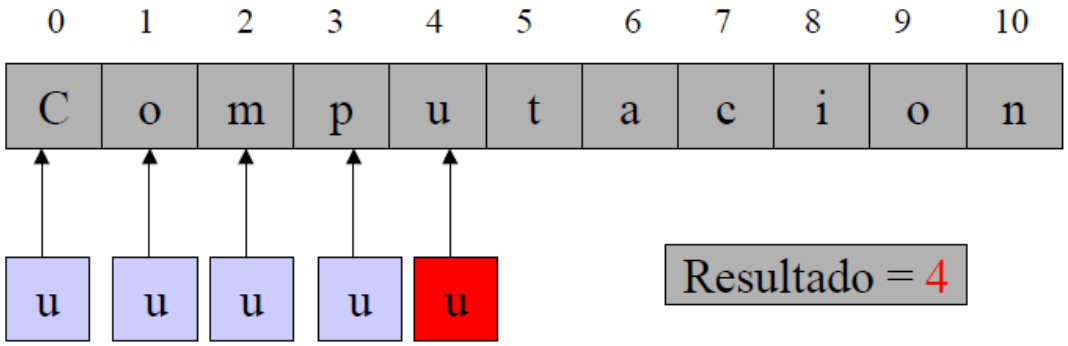
Algoritmos de Búsqueda

- Definición:
  - – Para encontrar un dato dentro de un array, para existen diversos algoritmos que varían en complejidad, eficiencia, tamaño del dominio de búsqueda.

- • Algoritmos de Búsqueda:
  - – Búsqueda Secuencial
  - – Búsqueda Binaria

Búsqueda Secuencial

- Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta cuando se encuentra.
- Busquemos el elementos **‘u’**



Búsqueda Secuencial

- Búsqueda del menor

```
menor = a[0];
        for (i=1;i<n;i++)
            if ( a[i]<menor )
                menor=a[i];
```

Búsqueda Secuencial

- **Búsqueda del menor**

```
mayor = a[n-1];
        for (i=1;i<n;i++)
                if ( a[i]>mayor )
                        mayor=a[i];
```

Búsqueda Secuencial

- **Búsqueda de un elemento**

```
encontrado=-1;
i=0;
While (i<n) && (encontrado==-1) {
                    if ( a[i]==elemento_buscado )
                                encontrado=i;
                                i++; }
```

Ejemplo

- Desarrollar un programa que posea una función que reciba como parámetro un array de 10 enteros, y un entero, y retorne la posición del entero si es que se encuentra, de lo contrario devolver –1.

```
#include <stdio.h>

int encuentra(int A[], int b) {
        int k=1, result=-1;

        do{
                if (A[k]== b)
                        result =k;
                else
                        k++;
        }while ((result==-1)&&(k<10));
        return result;
}

int main() {
        int i, x[10];

        for(i=0;i<10;i++)
                scanf("%d",&x[i]);
        i = encuentra( x, 10);
        printf("resultado %d\n",i);

        return 0;
}
```

Búsqueda Binaria

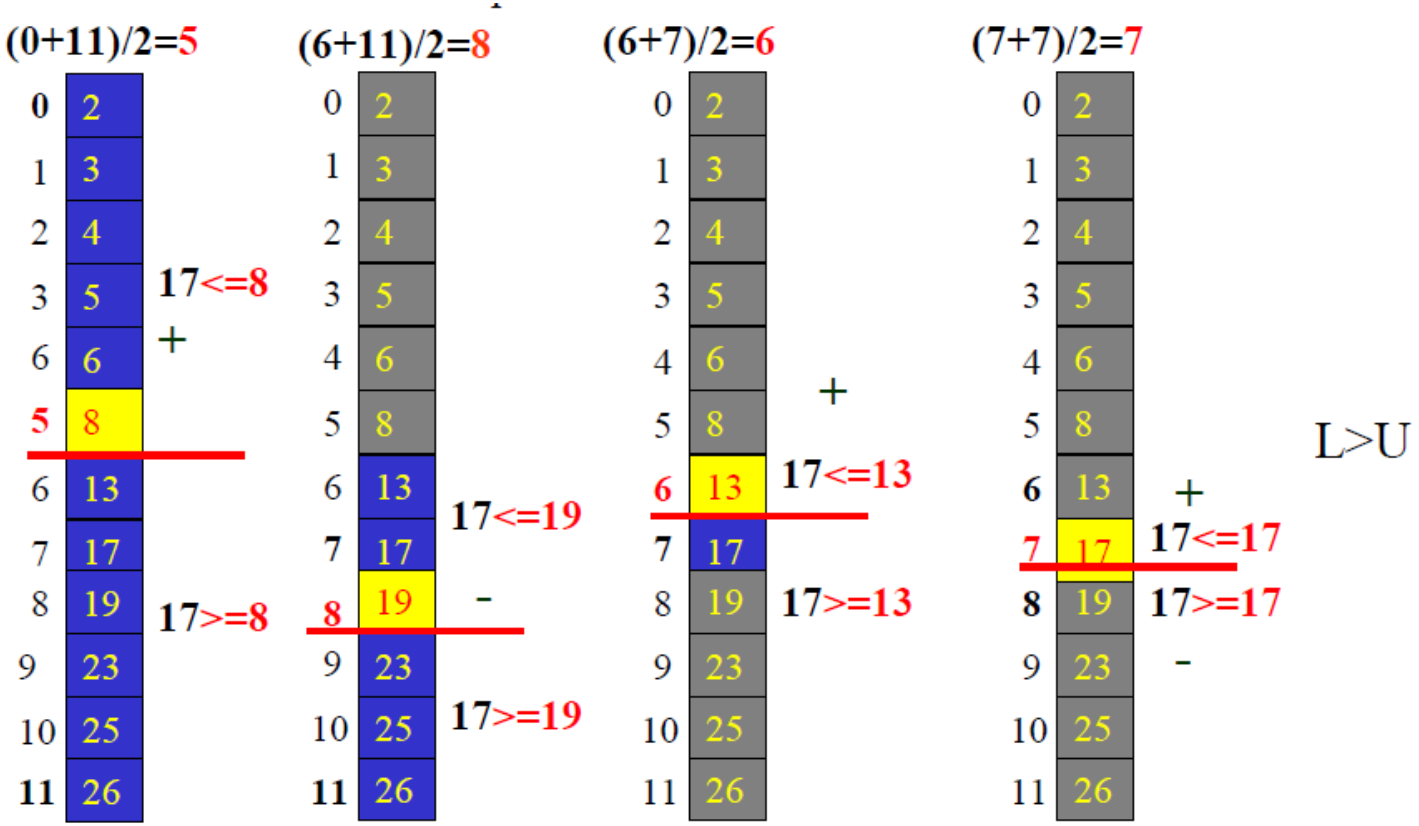
- En el caso anterior de búsqueda se asume que los elementos están en cualquier orden. En el peor de los casos deben hacerse **n** operaciones de comparación.

- Una búsqueda más eficiente puede hacerse sobre un array ordenado. Una de éstas es la Búsqueda Binaria.

- La Búsqueda Binaria, compara si el valor buscado está en la mitad superior o inferior. En la que esté, subdivido nuevamente, y así sucesivamente hasta encontrar el valor.

Búsqueda Binaria

- Supuesto: Array con datos ordenados en forma ascendente: i<k a[i]<a[k]
- Estamos buscando la posición del valor 17



Algoritmo de Búsqueda Binaria

```
#include <stdio.h>

int main() {
      int b,i,j,k, v[12];

      for(i=0;i<12;i++)
            scanf("%d",&v[i]);
      printf("fin del llenado\n");
      printf("ingrese numero a buscar ");
      scanf("%d",&b);
      i= 0;
      j= 12-1;
      do {
            k= (i+j)/2;
            if (v[k]<=b )
                  i=k+1;
            if (v[k]>=b )
                  j= k-1;
      } while (i<=j);
      printf("elemento %d esta en %d\n",v[k],k);
      return 0;
}
```

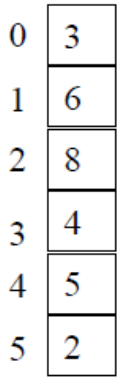
```
i= 0;
j= tamaño-1;
do {
      k= (i+j)/2;
      if (v[k]<=b )
            i=k+1;
      if (v[k]>=b )
            j= k-1;
} while (i<=j);
```

Ordenamiento de Componentes

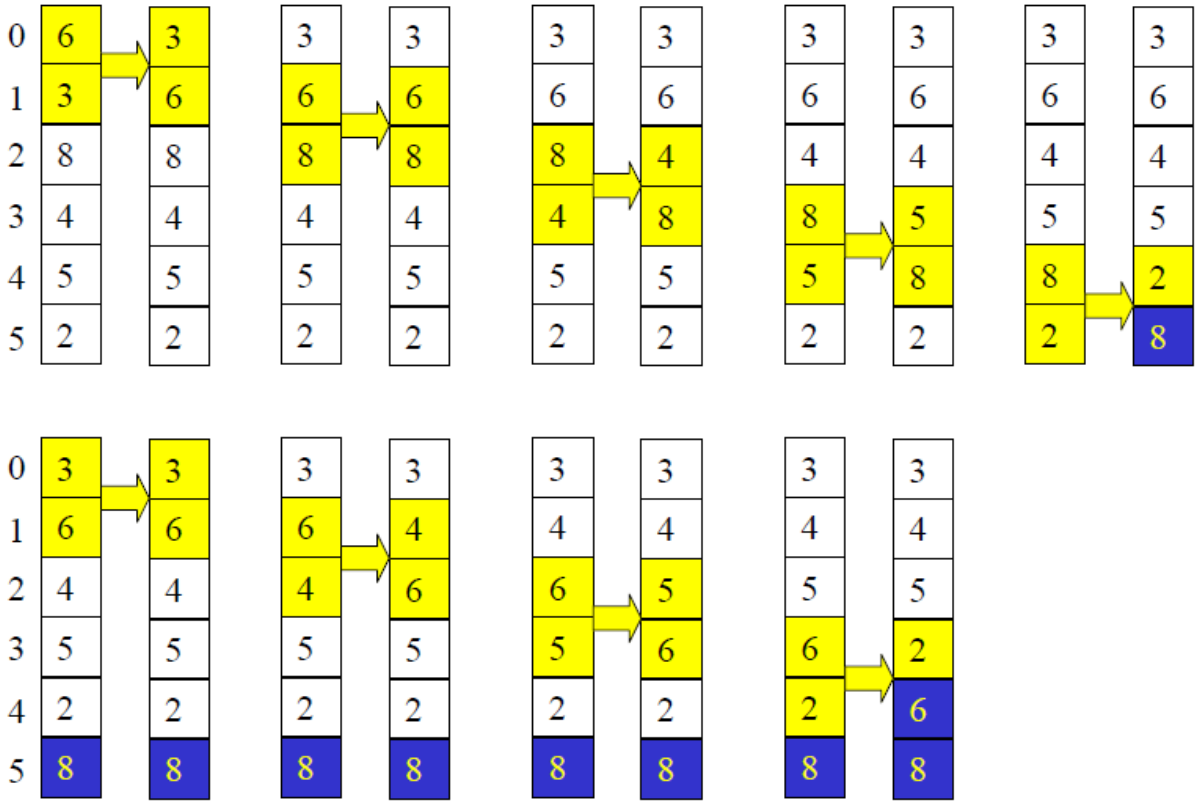
- **Ordenamiento Ascendente**
  - Existen numerosos algoritmos para ordenar. A continuación se verán algunos algoritmos de ordenamiento.

- **Ordenamiento Burbuja (bublesort):**

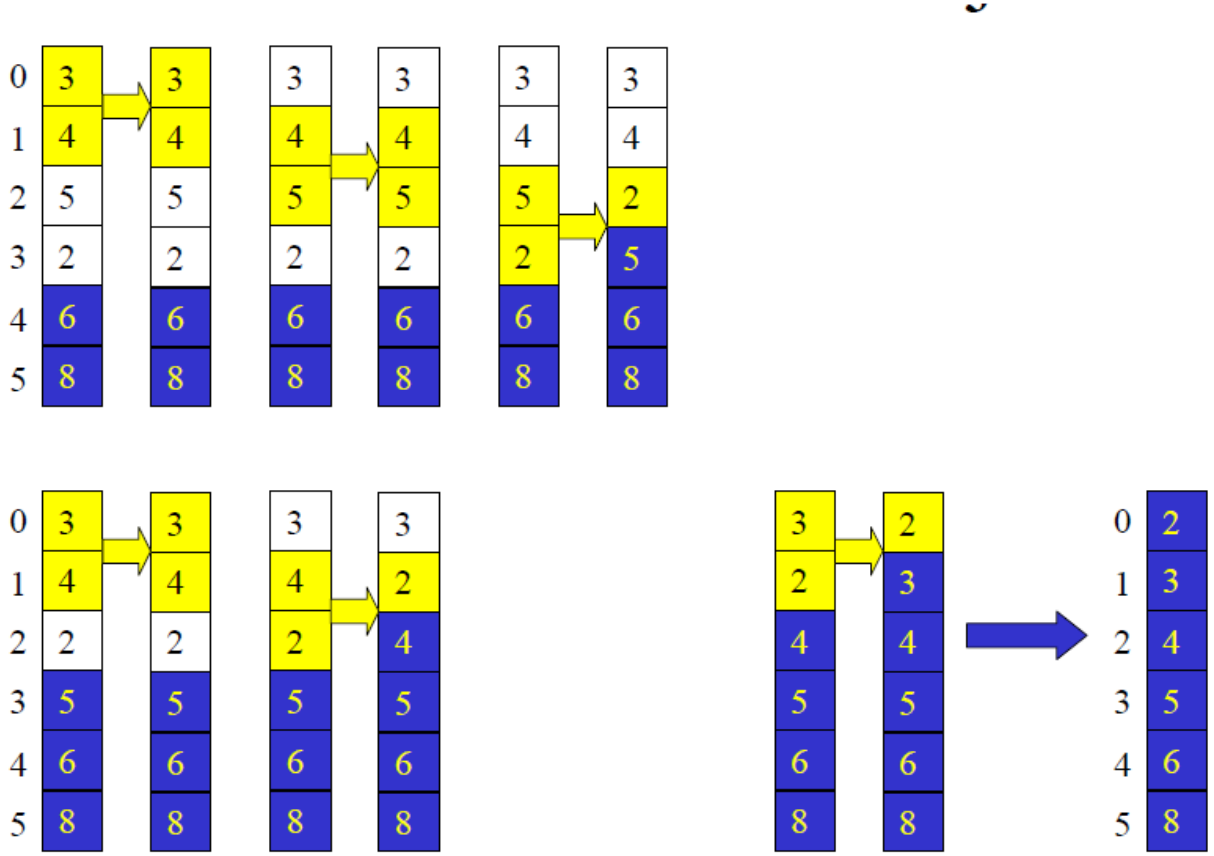
  Idea: vamos comparando elementos adyacentes y empujamos los valores más livianos hacia arriba (los más pesados van quedando abajo). Idea de la burbuja que asciende, por lo liviana que es.



Ordenamiento Burbuja



Ordenamiento Burbuja



Ordenamiento Burbuja

```
#include <stdio.h>
#define N 6
void intercambia(int *f,int *g) {
    int tmp;

    tmp = *f;
    *f = *g;
    *g = tmp;
}

int main() {
    int i,j, v[N]={3,4,5,2,6,8};

    for (i=N-1;i>1;i--)
        for (j=0;j<i;j++)
            if (v[j]>v[j+1])
        intercambia(&v[j],&v[j+1]);
    for (i=0;i<N;i++)
        printf("%d\n",v[i]);
    return 0;
}
```

```
for (i=N-1;i>0;i--)
   for (j=0;j<i;j++)
     if (V[j]>V[j+1])
         Intercambia(&A[j],&A[j+1]);
```

Búsqueda en Arrays Bidimensionales

Arreglos Bidimensionales

- Una matriz bidimensional tiene la siguiente forma:

$$A = \begin{bmatrix} a_{11} & a_{12} \ldots\ldots\ldots & a_{1n} \\ a_{21} & a_{22} \ldots\ldots\ldots & a_{2n} \\ \ldots\ldots\ldots & & \\ a_{m1} & a_{m2} \ldots\ldots\ldots & a_{mn} \end{bmatrix}_{mxn}$$

- – Para acceder al dato aij se hace de la siguiente manera: c=A[i][j];

Búsqueda

- Para buscar un elemento en un arreglo de dos dimensiones (el menor o el mayor), podemos suponer que uno de ellos es el menor (mayor), o mejor suponer un valor muy alto (o muy bajo), para luego contrastarlo uno a uno cada elemento, es decir una búsqueda secuencial.

Ejemplo de Búsqueda

```
#include <stdio.h>
#define N 3

int main() {
    int i,j,max,min, a[N][N];
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            a[i][j] = rand();
    max=-1000;
    min= 1000;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++) {
            if (a[i][j]>max)
                max = a[i][j];
            if (a[i][j]<min)
                min = a[i][j];
        }
    printf("el maximo es %d y el minimo es %d\n",max,min);
    return 0;
}
```