Curso C

Ordenación Vectores

Ordenación Vectores

- Método de la Burbuja
  - Los números a ordenar en el vector recorren la lista a borbotones hacia el inicio.
  - Reglas:
    1. Sólo se comprueban dos números a la vez, comenzando por los dos primeros
    2. Si el primer nº es más pequeño, dejarlo como esta. Caso contrario, intercambiar números
    3. Avanzar un numero y comparar con el que le sigue a continuación. (nuevo par a comparar)
    4. Repetir hasta que no se requiera ningún cambio tras una pasada completa al vector.

Ver Ejemplo y Código

Ordenación Vectores - Burbuja

```
#define numero_maximo  9

int ordenar_burbuja (int vector [ ] ){
  int indice;
  int seguir;
  int temp;
  int comp=0;
  do{
     seguir=0;
     for(indice=0; indice < numero_maximo; indice++){
          comp++;
         if((vector[indice] > vector[indice+1] && (indice != numero_máximo -1) ){
              temp=vector[indice];
              vector[indice]=vector[indice+1];
              vector[indice+1]=temp;
           }
       }
    }while(seguir);
    return (comp);
}
```

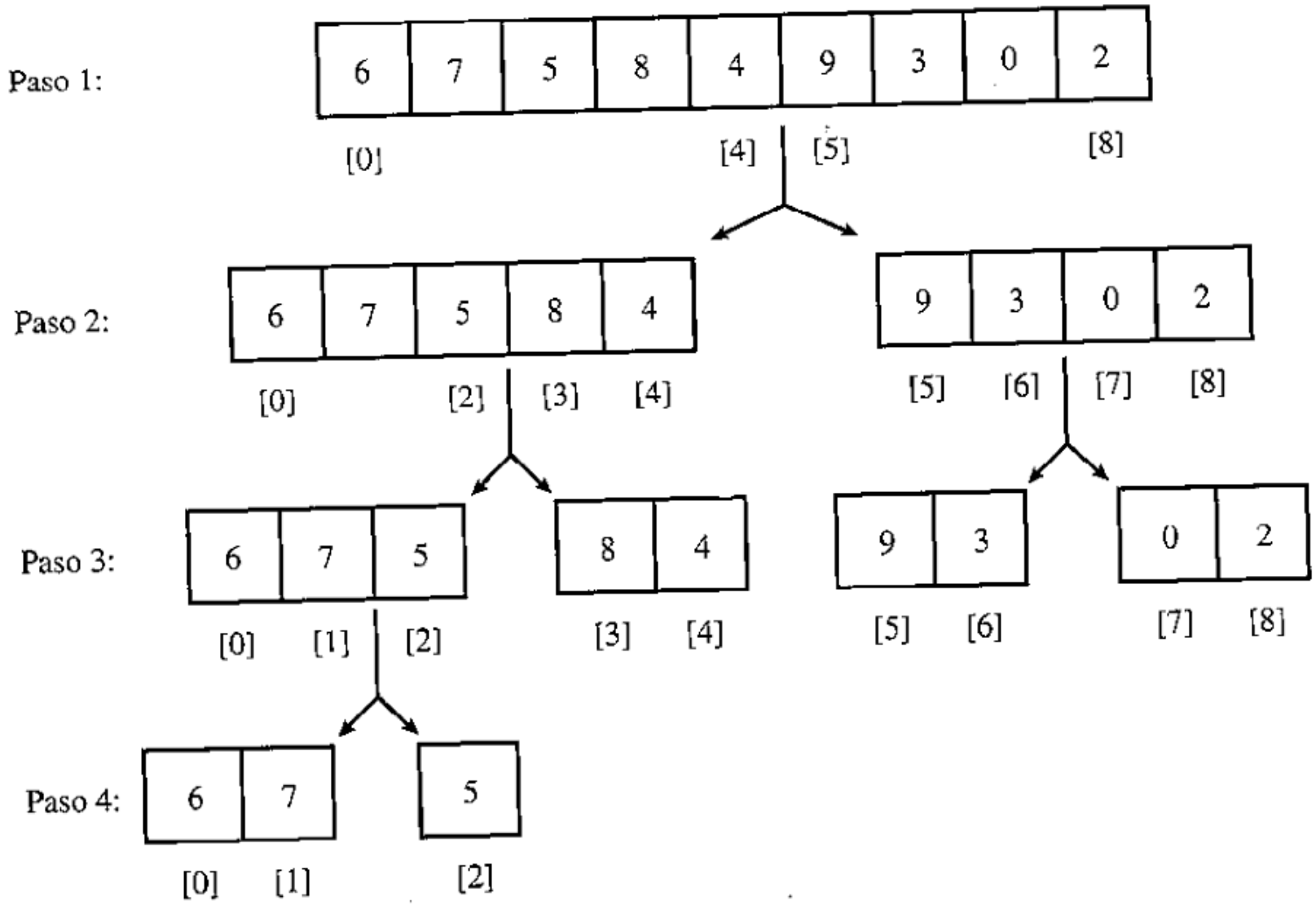
Secuencia	Nº Comparaciones
6 7 5 8 4 9 3 0 2	72

Ordenación Vectores

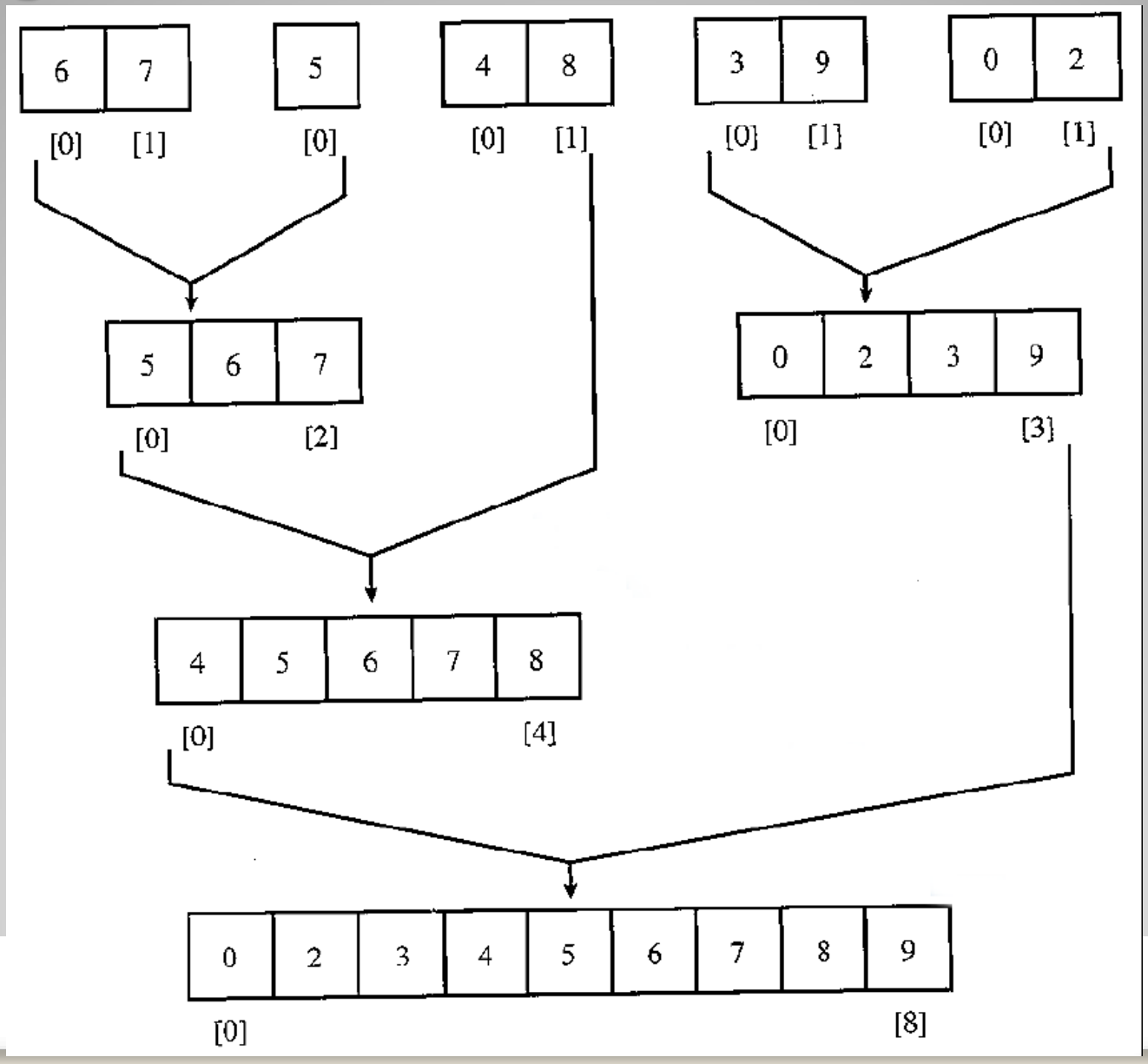
- Método por fusión (merge sort)
  - La lista entera se divide en listas más pequeñas que requieren menos comparaciones para ordenarlas.
  - Reglas:
    1. Determinar la mitad del vector.
       1. Dividimos vector en DOS subvectores
    2. Cada subvector se divide en 2 nuevas partes
       1. Hasta que subvectores tengan 1 ó 2 elementos
    3. Mezclar subvectores comparando sólo un elemento cada vez.

Ver Ejemplo y Código

Merge sort – paso 1



Merge sort - mezcla



```
void Ordenar(int ent [ ], int a, int b, int sal [ ]){
  int m;
  int sal1[20], sal2[20];
  if(a==b) //solo un elemento
    sal[0]=ent[a];
  else
//dos elementos
    if(1==(b-a)) {
      if(ent[a]<=ent[b]){
        sal[0]=ent[a];
        sal[1]=ent[b];
      }
      else{
        sal[0]=ent[b];
        sal[1]=ent[a];
      }
      comparaciones++;
    } else {
       /*Dividir vectores de 3 o mas*/
       m = a + (b-a)/2;
       /*Ordena Primera Mitad*/
       Ordenar(ent,a,m,sal1);
       /*Ordena segunda Mitad*/
       Ordenar(ent,m+1,b,sal2);
       /* Mezclar */
       Mezclar(sal1, sal2, (1+m-a), (b-m), sal);
    }
 }
```

```
void Mezclar(int ent1[], int ent2[], int n1, int n2, int sal[]){
  int i=0,j=0,k=0;
  while((i<n1) && (j<n2)){
  /* comprobar si primer elemento
      es el mas pequeño */
      if(ent1[i]<=ent2[j]){
          sal[k]=ent1[i];
           i++;
       }else{
         sal[k]=ent2[j];
         j++;
       }
       k++;
      comparaciones++;
  }  /*comprobar si hay elementos a la izda
       en el primer vector */
      if(i!=n1){
        do{
          sal[k]=ent1[i];
          i++;
          k++;
        }while(i<n1);
      }else{
        do{
          sal[k]=ent2[j];
          j++;
          k++;
        }while(j<n2);
      }}
```

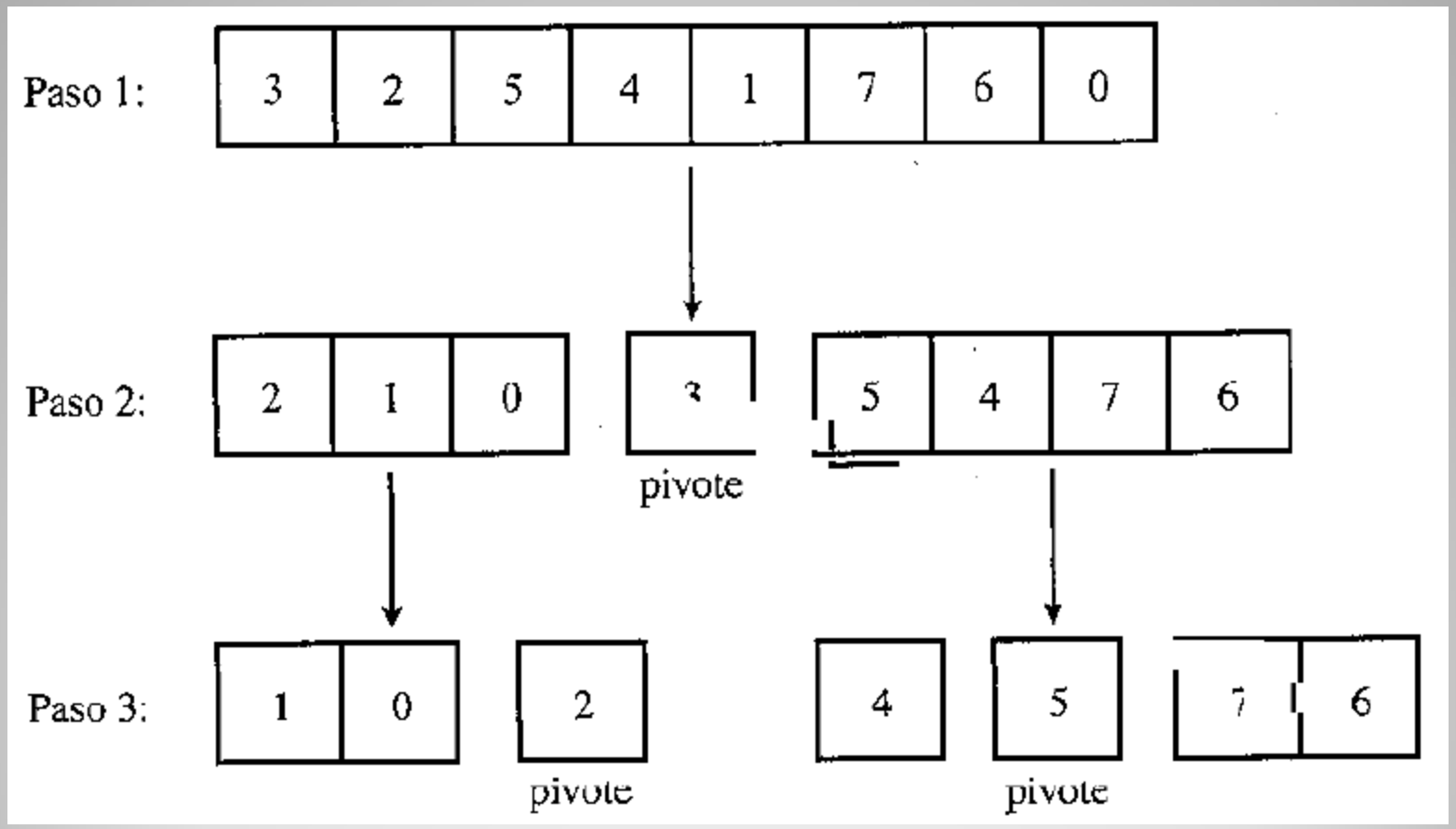
Comparaciones: 21

Ordenación Vectores

- Método rapido (quick sort)
  - La idea básica es usar un elemento como **pivote**, de manera que el vector se divida en dos partes: menores o iguales al **pivote** y mayores que el **pivote**
  - Reglas:
    1. Elegir pivote y dividir en dos vectores.
    2. Repetir proceso con subvectores
       1. Hasta contener 1 ó 2 elementos. (y ordenar)
    3. Combinar en vector resultado

Ver Ejemplo y Código

Quick sort


Paso 1:
3
2
5
4
1
7
6
0
Paso 2:
2
1
0
3
pivote
5
4
7
6
Paso 3:
1
0
2
pivote
4
5
pivote
7
6

```
void Quick(int ent [ ], int a, int b, int sal [ ]){
  int pivote,i=0,j=0,k=0,z=0;
  int ent1[20], ent2[20];
  int sal1[20], sal2[20];
  if(b!=-1) /*solo un elemento*/
     if(a==b)
        sal[0]=ent[a];
     else /*dos elementos*/
       if(1==(b-a)){
          if(ent[a]<=ent[b]){
                sal[0]=ent[a];
                sal[1]=ent[b];
          }else{
                sal[0]=ent[b];
                sal[1]=ent[a];
          }
          comparaciones++;
      }else{ /*tres o mas elementos*/
  pivote=ent[0];
  while(k<=b){
     if(pivote > ent[k]){
        ent1[i]=ent[k];
        i++;
      }else{ /* pivote es mas pequeño */
        ent2[j]=ent[k];
         j++;
      }
      k++;
      comparaciones++;
  }
  /* Ordenar particion mas pequeña */
  Quick(ent1,0,i-1,sal1);
  /* Ordenar particion mas grande */
  Quick(ent2,0,j-1,sal2);
  /* Escribir en salida el vector mas pequeño*/
  for(k=0;k<i;k++){
      sal[z]=sal1[k];
       z++;
  }
  /*Escribir pivote*/
  sal[z]=pivote;
  z++;
  /* escribir vector mas grande en salida */
  for(k=0;k<j;k++){
      sal[z]=sal2[k];
      z++;
  }
}
```

Comparaciones: 21

Qsort()

- C proporciona función qsort que implementa método quick sort para datos numéricos
  - qsort(direccion_vector, num_elementos, tamaño_elemento, funcion_comparacion)

```
#include <stdio.h>
#include <stdlib.h>
int comparaciones=0;
int cmp(const void*a, const void*b);
/*programa principal*/

int main(){
  int vec_ent[20]={6,7,5,8,4,9,3,0,2};
  int n=9;
  qsort(vec_ent,n,sizeof(vec_ent[0]),cmp);
}

int cmp(const void *a, const void *b){
   comparaciones++;
   return (*((int*)a)-*((int*)b));}
```

Comparaciones: 23