PROBLEMA REAL

Resoluble con un ordenador?

MODELIZACIÓN

ALGORITMO

EFICIENCIA

TÉCNICAS DE DISEÑO

ABSTRACCIÓN

DATOS

T.D.A.

CONTROL

Especificación

Implementación

Diseño

LENGUAJE DE PROGRAMACIÓN

METODOLOGÍAS DE LA PROGRAMACIÓN

PROGRAMA

VALIDACIÓN

PORTABILIDAD

(2)

- Problema — Operaciones más frecuentes sobre los datos

  - Escoger la E.D más eficientes para el problema.

- Ejemplo de Operaciones que aparecen en la mayoría de problemas.

  - Búsqueda o Consulta de un elemento
    - SECUENCIAL
    - BINARIA

  - INSERCCIÓN
  - BORRADO
  - Ordenación
    - (1)
      - INSERCCIÓN
      - BURBUJA
      - SELECCIÓN
    - (2)
      - MERGESORT
      - QUICKSORT

- EJEMPLO: Suponer que tenemos N enteros almacenados en un vector:

```
int * v;
int N;
```

La operación más frecuente es la ordenación. Los algoritmos que dispongamos son:

- SELECCIÓN
- .
- .
- Inserccion
- Burbuja
- MergeSort

¿Cuál escogeríamos?

SELECCIÓN

```
void Intercambiar (int &a, int &b)
        int aux=a;
        a=b;
        b=aux;

}
void Ordenacion_Seleccion (int *v, int n)
{
   for (int i=0; i<n-1 ; i++)
    for (int j= i+1 ; j<n ; j++)
       if (v[j] < v[i])
          Intercambiar(v[i], v[j]);

}
```

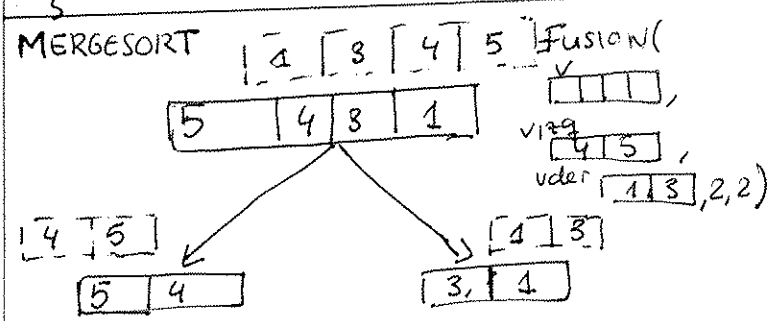
INSERCCIÓN

```
void Ordenacion_Insercion(int *v, int n)
{ for (int i=1; i<n ; i ++) {
        int pos = i;
      for (int j= i-1 ; j>=0 ; j--){
         if (v[j] > v[pos])
            pos = j;

      }
      Intercambiar (v[i], v[pos])

}}
```

BURBUJA

```
Burbuja
void Ordenacion_Seleccion (int *v, int n)
{
    for (int i=1 ; i<n ; i++)
      for (int j=0 ; j<n-i ; j++)
        if (v[j] > v[j+1])
            Intercambiar(v[j], v[j+1])
}
```

MERGESORT



(3)

MERGESORT (continuación)

```
void Fusion(int *vout, int*vi,
            int *vd, int ni, int nd ){

  int pi=0, pd=0, p=0;
  while (pi<ni && pd<nd){
        if (vi[pi] < vd[pd]){
            vout[p]=vi[pi];
            pi++;
        }
        else{
            vout[p]=vd[pd];
            pd++;
        }
        p++;
  }
  //si queda algo en vi
  while (pi<ni){
        vout[p]=vi[pi];
        p++; pi++;
  }
  //si queda algo en vd
  while (pd<nd){
        vout[p]=vd[pd];
        p++; pd++;
  }
}
```

```
void Orden-MergeSort ( int * v, int n) {
  if (n == 2){
     if (v[0] > v[1])
        Intercambiar(v[0],v[1]);
  }
  else
   if (n>2){
     int ni=n/2, nd=n-n/2;
     int *vi=new int[ni];
     int *vd=new int [nd];
     for(int i=0; i<ni; i++)
          vi[i]=v[i];
     for (int i=0; i<nd; i++)
          vd[i]=v[i+n/2];

     //Ordenamos la izquierda
     Orden_MergeSort(vi, ni);
     //Ordenamos la derecha
     Orden_MergeSort(vd,nd);
     // Fusionamos en v vi y vd
     Fusion(v, vi, vd, ni, nd);
     delete[] vi; delete[] vd;
   }
}
```

¿Cual escogeríamos?

- Aproximación 1:
  1) Escoger un ~~vector~~ conjunto de datos
     p.e 4 - 3 - 5 - 2 - 1
  2) Contar cada sentencia que se ejecuta como 1
  3) Escoger aquel que tenga un menor numero.

¿Qué problemas?