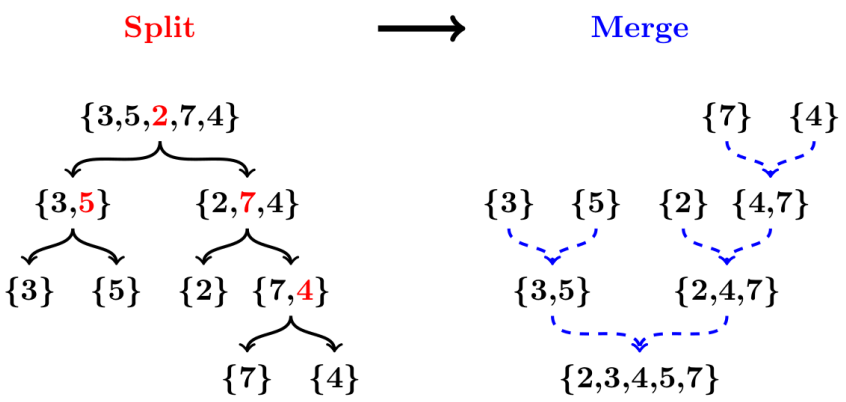
# Taller 2:
# Merge-sort

El algoritmo *merge-sort* toma una aproximación divide-and-conquer al problema de la organización de los elementos de un arreglo. Durante el *merge*,




dos arreglos ordenados contiguos deben ser combinados en un arreglo mas grande que también esté ordenado. Esto se puede lograr utilizando un arreglo auxiliar al cual se puedan copiar en orden.

```
1  MERGE(v, ini, fin, tmp)
2    mid = (fin + ini) / 2
3    p1 = ini
4    p2 = mid
5    pt = ini
6    while p1 < mid and p2 < fin
7      if v1[p1] < v2[p2]
8        tmp[pt] = v1[p1]
9        pt = pt + 1
10       p1 = p1 + 1
11     else
12       tmp[pt] = v2[p2]
```

```
13          pt = pt + 1
14          p2 = p2 + 1
15        while p1 < sz1
16          tmp[pt] = v1[p1]
17          pt = pt + 1
18          p1 = p1 + 1
19        while p2 < sz2
20          tmp[pt] = v2[p2]
21          pt = pt + 1
22          p2 = p2 + 1
23        for i = ini to fin
24          v[i] = tmp[i]
```

El proceso del merge sort puede realizarse entonces recursivamente de la siguiente manera

```
1   MERGESORT(v, ini, fin, tmp)
2     mid = (fin + ini) / 2
3     if(ini < fin)
4       MERGESORT(v, ini, mid, tmp)
5       MERGESORT(v, mid, fin, tmp)
6       MERGE(v, ini, fin, tmp)
```

1. Escriba un programa serial que organice un arreglo de enteros implementando el algoritmo merge-sort

2. Teniendo en cuenta la naturaleza recursiva del algoritmo, piense en una manera de para paralelizarlo usando las herramientas vistas de openMP

3. Implemente su versión paralela y asegúrese de su correctitud. Verifique el *speedup* obtenido para diferentes cantidades de procesadores