Auxiliar 11 - Ordenamiento

Profesores: Nelson Baloian, Patricio Poblete
Benjamín Bustos
Sebastian Ferrada
Auxiliares: Vicente Olivares, Ricardo Valdivia
Sebastián Acuña
Valentina Aravena

P1. Quicksort de Yaroslavskiy

Implemente el algoritmo de ordenación propuesto por Yaroslavskiy. Su diferencia con el Quicksort usual es que utiliza dos pivotes en vez de uno. Compare los resultados con el quicksort original.

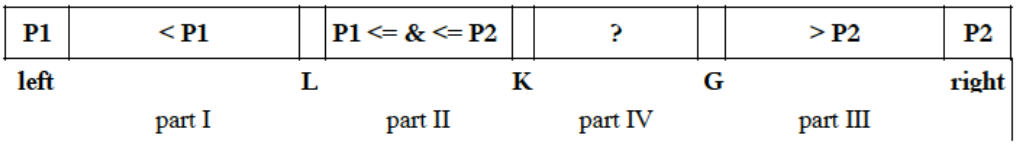

Figure 1: Invariante de Quicksort, versión de Yaroslavskiy.

P2. MergeSort Bottom-Up

En clases usted vió el algoritmo *mergesort* que sirve para ordenar un arreglo de largo n en tiempo $O(nlog(n))$, en sus dos variantes: *top down* y *bottom up*, la primera hace referencia a un arreglo que recursivamente va llamando al algoritmo para ordenarse, siguiendo un proceso de dividir para reinar, en cambio, la segunda divide su trabajo en 3 partes para ordenar el algoritmo:

(a) Identifica en el arreglo a ordenar las corridas (*runs*) de elementos ordenados consecutivos, identificando cada una por separado.

(b) Recorre el arreglo aplicando el algoritmo *merge* visto en clase a cada par de corridas identificadas anteriormente, formando nuevas corridas de elementos ordenados, pero con aun más elementos.

(c) Realiza iterativamente el paso anterior con las nuevas corridas formadas, hasta quedar con una sola corrida del tamaño del arreglo.

Programe esta versión de *Mergesort*, para esto, cree una función *make_runs* que se encargue de crear las corridas y almacenarlas en una lista y luego una función *make_merge* que tome como entrada los *runs*, les haga *merge* iterativamente y entregue un solo *run* que corresponda al arreglo ordenado. Estas dos funciones se usarán para el siguiente bloque de código que retorna finalmente el arreglo ordenado.

```
#El nuevo algoritmo de mergesort
def mergesort_bottom_up(arr):
  #Creamos los runs
  runs = make_runs(arr)
  #Hacemos los merges con los runs que creamos
  result = make_merge(runs)
  #Retornamos el arreglo ordenado
  return result
```

Observación: su función *make_merge* debe usar la función definida en el apunte:

```
def merge(a,b):
    i=0
    j=0
    while i<len(a) or j<len(b):
        if j>=len(b) or (i<len(a) and a[i]<=b[j]):
            yield a[i]
            i=i+1
        else:
            yield b[j]
            j=j+1
```