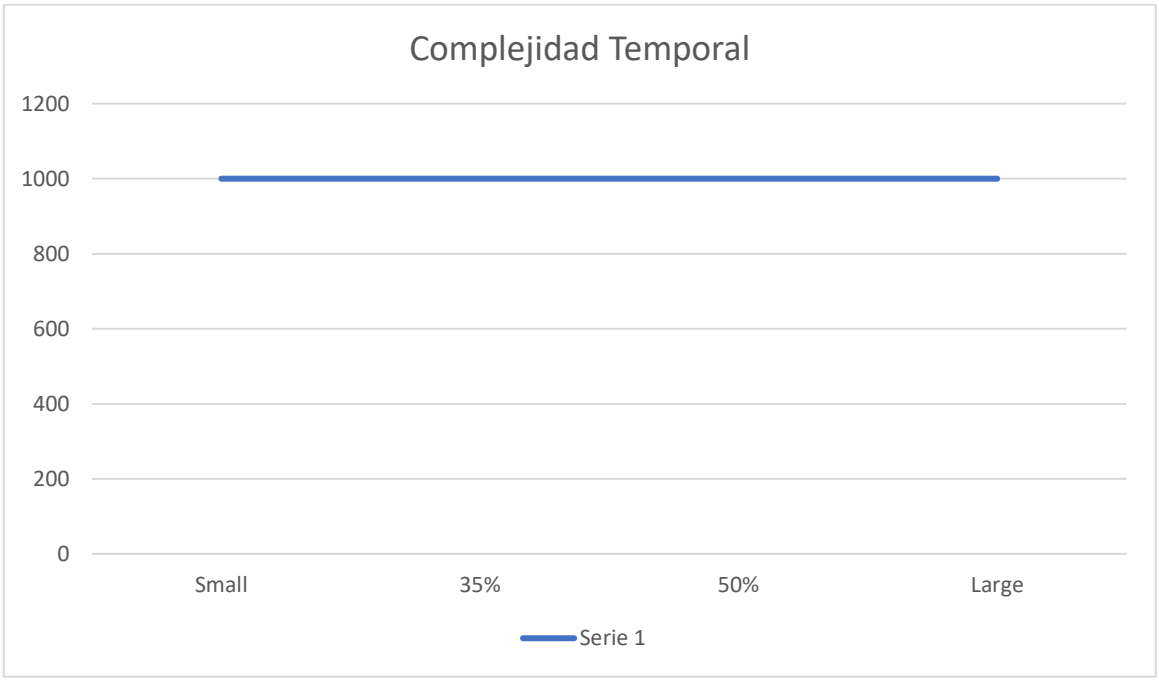
Santiago Rodriguez Bernal – 202011182

Análisis Codigo

Req 1.

```
def req1(catalogo, annoInicial, annoFinal):
    instanceCatalogo = catalogo
    instanceCatalogo["autores"]["elements"].sort(key=lambda elem: (float)(elem["BeginDate"]), reverse = True)
    resultado = []
    for i in instanceCatalogo["autores"]["elements"]:
        if (float)(i["BeginDate"])>(float)(annoFinal):
            continue
        if (float)(i["BeginDate"]) < (float)(annoInicial):
            break
        resultado.append(i)
    resultado.reverse()
    return resultado
```

La complejidad de este algoritmo es de 2N en big O(n), pues solo tiene que hacer dos cambias importantes primero sortea la lista de manera inversa y luego crea un loop donde va agregando los resultados que obtiene.



Se demora un segundo es prácticamente instantánea la respuesta en cualquier archivo.

Req 2

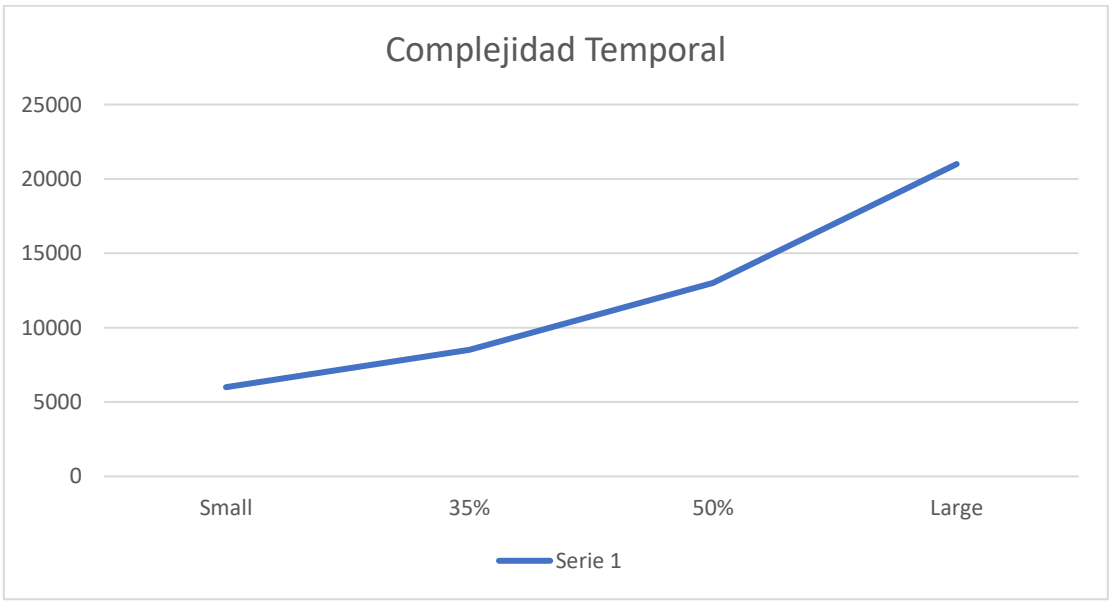
```
def req2(catalogo, annoInicial, annoFinal, sortFunction):
    """
    for i in range(lt.size(instanceCatalogo["obras"])):
        if lt.getElement(instanceCatalogo["obras"], i) == '':
            lt.deleteElement(instanceCatalogo["obras"], i)
    """
    instanceCatalogo = catalogo
    months = [0, 31, 59, 90, 120, 151, 181, 212, 243, 273, 304, 334]

    def sortingFunc(anno1, anno2):
        anno1use = anno1["DateAcquired"].split("-") if anno1["DateAcquired"].split("-")!=[''] else ["0" for _ in range(3)] #[2020, 10, 02]
        anno2use = anno2["DateAcquired"].split("-") if anno2["DateAcquired"].split("-")!=[''] else ["0" for _ in range(3)]
        firstAnno = (float)(anno1use[0]) + ((months[(int)(anno1use[1])-1] + (float)(anno1use[2]))/365) #2020.344
        secondAnno = (float)(anno2use[0]) + ((months[(int)(anno2use[1])-1] + (float)(anno2use[2]))/365)
        if((float)(firstAnno)>(float)(secondAnno)):
            return 1
        return 0
    sortingAlgorigthms[(int)(sortFunction)](lst = instanceCatalogo["obras"], cmpfunction = sortingFunc) # ShSort.sort(lst = instanceCatalogo["obr
    annoInicialUse = annoInicial.split("-") if annoInicial.split("-")!=[''] else ["0" for _ in range(3)] #[1920, 02, 20]
    firstAnno = (float)(annoInicialUse[0]) + ((months[(int)(annoInicialUse[1])-1] + (float)(annoInicialUse[2]))/365)#1920.216
    annoFinalUse = annoFinal.split("-") if annoFinal.split("-")!=[''] else ["0" for _ in range(3)] #[1985, 02, 20]
    lastAnno = (float)(annoFinalUse[0]) + ((months[(int)(annoFinalUse[1])-1] + (float)(annoFinalUse[2]))/365)#1985.216
    resultado = []
    for i in instanceCatalogo["obras"]["elements"]:
        dateAcquiredUse = i["DateAcquired"].split("-") if i["DateAcquired"].split("-")!=[''] else ["0" for _ in range(3)]#[1920, 02, 20]
        dateNICE = (float)(dateAcquiredUse[0]) + ((months[(int)(dateAcquiredUse[1])-1] + (float)(dateAcquiredUse[2]))/365)#1920.216
        if (float)(dateNICE)>(float)(lastAnno):
            continue
        if (float)(dateNICE) < (float)(firstAnno):
            break
        resultado.append(i)
    resultado.reverse()
    #for i in resultado:
    #    print(i["DateAcquired"], dateNICE)
    return resultado
```

Las primeras 7 líneas de código tiene una complejidad de O(1) pues simplemente esta transformando la fecha a años para que sea mas fácil manejar los datos.

Luego en el sort hay una complejidad O(n log n), pues se usa el algoritmo de Shell Sort.

En ultima instancia hay una complejidad de n pues hay un for donde se analiza la fecha de cada elemento



Req 3

```python
def req3(catalogo, artista):
    instanceCatalogo = catalogo
    artistaInfo = next(elem for elem in instanceCatalogo["autores"]["elements"] if elem["DisplayName"] == artista)
    obrasDelArtista = [elem for elem in instanceCatalogo["obras"]["elements"] if (int)(artistaInfo["ConstituentID"]) in ast.literal_eval(elem["Co
    tecnicas = [elem["Medium"] for elem in obrasDelArtista]
    seen = set()
    tecnicasUnicas = []
    for item in tecnicas:
        if item not in seen:
            seen.add(item)
            tecnicasUnicas.append(item)
    tecnicasFrecuencia = {i : 0 for i in tecnicasUnicas}
    for elem in obrasDelArtista:
        tecnicasFrecuencia[elem["Medium"]] += 1
    tecnicasFrecuencia=  {k: v for k, v in sorted(tecnicasFrecuencia.items(), key=lambda item: item[1], reverse=True)}
    resultado = obrasDelArtista
    return resultado, tecnicasFrecuencia
```

La complejidad de este algoritmo es muy baja, el primer ciclo de casi N que podemos encontrar esta en artista info, pero apenas encuentra el artista para entonces no alcanza a llegar a ser N,luego recorre la lista buscando el constitutenID de la obra y comparándola con el ID de los autores. Por ende tampoco llega a ser N, y en los últimos 2 ciclos, no se esta haciendo de todos los datos, sino de las técnicas entonces su complejidad total es N, pues no llega a ser mas de N.