

RETO #3

Juanita Gil Arango-j.gila2@uniandes.edu.co - 202111556

Gabriela Carvajal g.carvajal@uniandes.edu.co - 202111058

Complejidad temporal:

Req 1:

<pre>125 # Requerimiento 1 126 def AvistCiudad(catalog, ciudad): 127 pareja = om.get(catalog['city'], ciudad) 128 valor = me.getValue(pareja) 129 listaOrdenada = sa.sort(valor, compareDates) 130 size = lt.size(listaOrdenada) 131 sizeAvist = lt.size(om.keySet(catalog['city'])) 132 return size, sizeAvist, listaOrdenada</pre>	O(n(log(n))), ya que es la complejidad temporal del sort
--	--

Req 2:

<pre>142 def duration(catalog, segmin, segmax): 143 tiempos= om.values(catalog['duration (seconds)'], segmin, segmax) 144 lst= lt.newList() 145 for avist in lt.iterator(tiempos): 146 for x in lt.iterator(avist): 147 lt.addLast(lst, x) 148 listaOrdenada = sa.sort(lst, compareDates) 149 first= lt.subList(listaOrdenada, 1, 3) 150 last=lt.subList(listaOrdenada, lt.size(listaOrdenada)-2, 3) 151 return (listaOrdenada, first, last)</pre>	O(n)^2, ya que, en el peor de los casos, se cumplen ambos for, el de afuera y el de adentro
--	---

Req 3:

<pre>153 #req 3 154 def AvistHora(catalog, inferior, superior): 155 inf = datetime.datetime.strptime(inferior,"%H:%M:%S") 156 inf2 = inf.time() 157 sup = datetime.datetime.strptime(superior,"%H:%M:%S") 158 sup2 = sup.time() 159 llaves_rango = om.keys(catalog['datetime hora'],inf2, sup2) 160 lista = lt.newList() 161 for hora in lt.iterator(llaves_rango): 162 entry = om.get(catalog['datetime hora'], hora) 163 valor = me.getValue(entry) 164 for linea in lt.iterator(valor): 165 lt.addLast(lista,linea) 166 listaOrdenada = sa.sort(lista, compareDates) 167 size = lt.size(listaOrdenada) 168 return size, listaOrdenada</pre>	O(n)^2, ya que, en el peor de los casos, se cumplen ambos for, el de afuera y el de adentro
--	---

Req 4:

<pre> 170 #Req 4 171 def avistRangoFechas(catalog, inferior, superior): 172 inf = datetime.datetime.strptime(inferior, "%Y-%m-%d") 173 inf2 = inf.date() 174 sup = datetime.datetime.strptime(superior, "%Y-%m-%d") 175 sup2 = sup.date() 176 llaves_rango = om.keys(catalog['datetime'], inf2, sup2) 177 lista = lt.newList() 178 for fecha in lt.iterator(llaves_rango): 179 entry = om.get(catalog['datetime'], fecha) 180 valor = me.getValue(entry) 181 for linea in lt.iterator(valor): 182 lt.addLast(lista, linea) 183 listaOrdenada = sa.sort(lista, compareDates) 184 size = lt.size(listaOrdenada) 185 return size, listaOrdenada </pre>		<p>$O(n)^2$, ya que, en el peor de los casos, se cumplen ambos for, el de afuera y el de adentro</p>
---	--	---

Req 5:

<pre> 186 #req 5 187 def avistZona(catalog, longmin, longmax, latmin, latmax): 188 zonas= lt.newList() 189 avist= om.values(catalog['longitude'], longmin, longmax) 190 for avistamientos in lt.iterator(avist): 191 for avistamiento in lt.iterator(avistamientos): 192 if float(avistamiento['latitude']) >= latmin and float(avistamiento['latitude']) <= latmax: 193 lt.addLast(zonas, avistamiento) 194 total= lt.size(zonas) 195 first= lt.subList(zonas, 1, 3) 196 last=lt.subList(zonas, total-2, 3) 197 return (total, first, last) 198 </pre>	<p>$O(n)^2$, ya que, en el peor de los casos, se cumplen ambos for, el de afuera y el de adentro</p>
---	---

Requerimientos individuales:

Req. 2: Gabriela Carvajal

Req. 3: Juanita Gil