HDownloader

Un grupo de fanáticos de series, música y apuntes de UTN con muy poca paciencia a la hora de bajar sus preciados archivos de la red, ha decidido construir HDownloader, su propio gestor de descargas(*) empleando el paradigma funcional en alguno de sus módulos.

El sistema gestiona **descargas** de conjuntos de archivos, que pueden pueden provenir de diversos servidores (como por ejemplo, grupos de la facultad o youtube), las cuales están priorizadas por el usuario en una escala de 1 a 5, y tienen un nombre para identificarlas. Se las representa con 4-tuplas de las forma: (Nombre, [Archivo], Servidor, Prioridad). Ejemplo:

descarga1 = (“Sistemas Operativos - Silberschatz”, [silberschatzParte1, silberschatzParte2], megaupload, 5)
descarga2 = (“Parcial de funcional Dentista”, [parcialDentista], pdep, 2)

Los **archivos** representan un recurso que puede ser descargado, de los que se conoce su URL, su tamaño en MBytes, y su taza de compresión (si no está comprimido es 0). Se los representa como ternas (URL, Tamaño, Compresión). Ejemplo:

silberschatzParte1 = (“www.megaupload.com/?d=CRK5NQB0 ”, 10, 0)

Por último, de los **servidores** se conoce su dominio (un String), y su política de asignación de ancho de banda, la cual nos permite saber la velocidad (en KBytes/s) a la que se puede descargar un archivo, dado el ancho de banda del que dispone HDownloader y el archivo en cuestión. Se lo representa como un par (Dominio, PoliticaAsignacionAnchoBanda).

Algunas política de otorgamiento de ancho de banda son (**):

- Descargar empleando todo el ancho de banda disponible. Por ejemplo, si HDownlaoder dispone de 50KB/s, descargará a 50KB/s
- Descargar empleando como mucho a una velocidad límite fijada por el servidor. Por ejemplo, si el gestor dispone de 50KB/s y el servidor fija un límite de 20KB/s, descargará a 20KB/s.
- Descargar con todo el ancho de banda disponible, dividido por 1 + la cantidad de descargas recientes que se han hecho contra el mismo. Por ejemplo, si un servidor registra 2 descargas recientes, y el ancho de banda disponible es de 100KB/s, entonces descargará a 33,33KB/s
- Descargar multiplicando el ancho de banda disponible por un coeficiente entre 0 y 1, si el archivo se encuentra en una lista negra del servidor, de la forma [(URL, Coeficiente)]. En caso contrario, descargar con todo el ancho de banda disponible
- Descargar aplicando sucesivamente un conjunto de políticas: algunos servidores implementan políticas de otorgamiento de ancho de banda que combinan algunas de las anteriores, partiendo del ancho de banda disponible para el gestor y pasando el resultado a la siguiente política. Por ejemplo, si se dispone de 50KB/s, y se tiene un servidor con una política de limite de ancho de banda a 10KB/s, y otra política de

cantidad descargas y registró 2, la velocidad de descarga será 3,33KB/s

Ej:

- youtube: descarga los archivos con todo el ancho de banda disponible. Su dominio es “www.youtube.com”
- pdep: descarga los archivos, a 200KBytes/s máximo. Su dominio es “www.pdep.com.ar”
- rapidshare: descarga a 240KBytes/s, y contabiliza las conexiones recientes (actualmente hay 2 conexiones recientes registradas) Su dominio es “www.rapidshare.com”.
- megaupload: lleva una lista negra de archivos; actualmente la lista contiene a la URL www.megaupload.com/?d=AJD5PMB1, que tiene un coeficiente de descarga de 0.2, y a la URL www.megaupload.com/?d=GAA0ACC9, con un coeficiente de 0.1

Se pide resolver los siguientes puntos sin emplear recursividad salvo que se indique lo contrario:

1. Desarrollar y usar las funciones urlArchivo/1, prioridadDescarga/1, compresionArchivo/1, etc, que permitan acceder a los componentes de los archivos y descargas
2. Saber el espacio en disco que necesitará una descarga. Este es la sumatoria de los tamaños en disco de los archivos que la componen, calculada como: tamaño * (1+ taza de compresión)
3. Desarrollar y utilizar una función contiene/2 que permita saber si una lista está contenida dentro de otra. Se puede utilizar recursividad.
4. Dado el dominio de un servidor, saber si un archivo está alojado en servidor, lo cual ocurre cuando la url del archivo contiene el dominio del servidor
5. Conocer la velocidad de descarga de un servidor, dado el servidor, el archivo y el ancho de banda disponible. Mostrar consultas para los cuatro servidores de ejemplo, e implementar cada uno de ellos.
6. Saber si un archivo está disponible en un servidor, es decir, si está alojado en el mismo, y la velocidad de descarga es mayor a 0
7. Saber si una descarga está disponible: lo está cuando todos sus archivos lo están.
8. Dado el ancho de banda de la conexión y una lista de descargas, obtener el tiempo total en minutos que tomará descargar los archivos disponibles. Asumiendo que los archivos se descargan de a uno por vez, el tiempo de descarga de un archivo es (1024 * tamaño del archivo) / (60 * velocidad de descarga del servidor desde donde se lo descarga).
9. Dado una lista de descargas a realizar, desarrollar una función que encuentre el nombre y la cantidad de archivos grandes (de más de 500MBytes) de la descarga por la que el gestor debería comenzar, según un criterio. Dar ejemplos de las siguientes consultas:
    a. Por prioridad: Se deberá empezar por la de mayor prioridad
    b. Por espacio en disco: se deberá comenzar por la descarga que menos espacio en disco requiera
10. Mencionar aquellos puntos (si los hubiera) donde se utilizaron

a. Funciones de orden superior, y en particular composición.
b. Aplicación parcial
c. Expresiones lambda
d. Listas por comprensión

(*) Programa que se encarga de, dado un conjunto de URLs, descargar los archivos de forma eficiente, considerando las prioridades del usuario, el uso de red, y la disponibilidad de los archivos.
(**) No son las únicas, podría haber más y la solución debe soportarlas

```
-- Servidores de ejemplo

youtube, megaupload, rapidshare :: Servidor

youtube = ( "www.youtube.com", todoElAnchoBanda)
megaupload = ("www.megaupload.com", maximoAnchoBanda 200.0)
rapidshare = ("www.rapidshare.com", politicasCombinadas [ (maximoAnchoBanda 240.0),
contabilizacionDescargas 2])


-- Solucion

type Descarga = (String, [Archivo], Servidor, Int)
type Archivo = (String, Double, Double)
type Servidor =  (String, Double -> Archivo -> Double)

-- 1
nombreDescarga (nombre,_,_,_) = nombre

archivosDescarga :: Descarga -> [Archivo]
archivosDescarga (_,archivos,_,_) = archivos

servidorDescarga :: Descarga ->  Servidor
servidorDescarga (_,_,servidor,_) = servidor

compresionArchivo :: Archivo -> Double
compresionArchivo (_,_,compresion)  = compresion

tamañoArchivo :: Archivo -> Double
tamañoArchivo (_,tamaño,_)  = tamaño

urlArchivo :: Archivo -> String
urlArchivo (url,_,_)  = url

dominioServidor :: Servidor -> String
dominioServidor (dominio,_) = dominio

-- 2
espacioDiscoDescarga :: Descarga -> Double
espacioDiscoDescarga descarga = foldr ((+).espacioDiscoArchivo) 0 (archivosDescarga
descarga)

espacioDiscoArchivo :: Archivo -> Double
```

```
espacioDiscoArchivo archivo = (1 + compresionArchivo archivo) * (tamañoArchivo archivo)
-- espacioEnDiscoNecesario descarga= sum(map tamanioArchivo (archivos descarga))
-- tamanioArchivo archivo= (1+compresionArchivo archivo)*(tamanio archivo)

-- 3

contiene lista1= all (flip elem lista1)

-- 4

alojadoEnServidorArchivo :: Servidor -> Archivo -> Bool
alojadoEnServidorArchivo servidor archivo = contains (dominioServidor servidor) (urlArchivo
archivo)

-- 5

--Politicas de asignacion de bw

todoElAnchoBanda = const.id
maximoAnchoBanda maximo = const.min maximo
listaNegra urlsBloqueadas anchoBanda (url,_,_)
   | null result = anchoBanda
         | otherwise = (snd.head) result
         where result = filter ((==url).fst) urlsBloqueadas

contabilizacionDescargas cantidadDescargasPrevias anchoBanda _ = anchoBanda / (1+
(fromIntegral cantidadDescargasPrevias))


politicasCombinadas politicas anchoBanda archivo
    = foldl (\anchoBandaDisponible politica -> politica anchoBandaDisponible archivo)
anchoBanda politicas


-- Velocidad de descarga desde un servidor para un archivo
velocidadEnServidorArchivo ::  Servidor -> Double -> Archivo -> Double
velocidadEnServidorArchivo (_,politicaAnchoBanda) anchoBanda archivo = politicaAnchoBanda
anchoBanda archivo


-- 6

disponibleEnServidorArchivo :: Servidor ->  Double -> Archivo  -> Bool
disponibleEnServidorArchivo servidor anchoBanda archivo
```

```
   = alojadoEnServidorArchivo servidor archivo && velocidadEnServidorArchivo servidor
anchoBanda archivo > 0

-- 7

disponibleDescarga :: Double -> Descarga  -> Bool
disponibleDescarga anchoBanda descarga
   = all (disponibleEnServidorArchivo (servidorDescarga descarga) anchoBanda)
(archivosDescarga descarga)

-- 8
tiempoDescargasDisponibles :: Double -> [Descarga] -> Double
tiempoDescargasDisponibles anchoBanda descargas
   = sum [ tiempoDescargarDescarga anchoBanda descarga  | descarga <- descargas,
disponibleDescarga anchoBanda descarga]


tiempoTotalDescargaArchivosDisponibles anchoDeBanda listaDescargas=
sum (map (tiempoDescarga anchoDeBanda) (filter (estaDisponible anchoDeBanda (servidor
descarga) listaDescargas))


tiempoDescargarDescarga :: Double -> Descarga -> Double
tiempoDescargarDescarga anchoBanda descarga
  = (sum.map tiempoDescargarArchivo') (archivosDescarga descarga)
  where
  tiempoDescargarArchivo'
        = tiempoDescargarArchivo (servidorDescarga descarga) anchoBanda

tiempoDescargarArchivo :: Servidor -> Double -> Archivo -> Double
tiempoDescargarArchivo servidor anchoBanda archivo = (1024 * tamañoArchivo archivo) / (60 *
velocidadDescarga archivo)
   where velocidadDescarga = velocidadEnServidorArchivo servidor anchoBanda


-- 9


empezarPorDescarga criterio descargas = (\descarga -> (nombreDescarga descarga,
cantidadArchivosPesados descarga)) (foldl1 criterio' descargas)
  where criterio' d1 d2  | criterio d1 >= criterio d2 = d1
                         | otherwise = d2
```

```
cantidadArchivosPesados descarga = (length.filter esPesadoArchivo) (archivosDescarga
descarga)

esPesadoArchivo = (>500).espacioDiscoArchivo
```