Análisis de algoritmos
Tarea 6
Minigoogle

Programar el algoritmo de Hurspool para búsqueda de palabras clave para los siguientes casos:

1) Entrada: palabra de búsqueda y archivo de texto (.txt) para realizar la búsqueda.
   Salida: No se encuentra o bien se encuentra y la posición donde inicia la palabra.
2) Entrada: palabra de búsqueda y múltiples archivos de texto (en una carpeta).
   Salida: La lista de archivos ordenados por relevancia (de mayor a menor).
   Por ejemplo:
   Cuantas veces se encuentra la palabra en cada archivo.
   Como segundo criterio si hay 2 documentos con el mismo numero de palabras clave, sera mas relevante el que lo tenga en mayor proporción respecto a su totalidad de palabras.
3) Entrada: Múltiples palabras de búsqueda y múltiples archivos.
   Salida: Los archivos ordenados por relevancia.

Para implementar este programa en la carpeta donde se buscara tenemos los siguientes archivos:

"Analisis de Algoritmos.txt",
"Artificial Intelligence A Modern Approach.txt",
"Introduction to algorithms.txt",
"Introduction to the Design and Analysis of Algorithms.txt",
"Pattern Recognition and Machine Learning .txt",
"pia-slides-01.txt",
"pia-slides-02.txt",
"pia-slides-03.txt",
"pia-slides-04.txt",
"pia-slides-05.txt",
"Probability and Computing Randomization and Probabilistic Techniques in Algorithms and Data Analysis.txt",
"PROLOG.txt",
"Remote Sensing Digital Image Analysis.txt"

El código esta desarrollado de manera que el programa es capaz de reconocer si la búsqueda es de una o mas palabras, con lo cual al realizar la búsqueda en la carpeta no es necesario especificarle el tipo de búsqueda a realizar.

El código se implementa en python:

```
import operator
import numpy as np
```

```
libros=["Analisis de Algoritmos.txt",
        "Artificial Intelligence A Modern Approach.txt",
        "Introduction to algorithms.txt",
        "Introduction to the Design and Analysis of Algorithms.txt",
        "Pattern Recognition and Machine Learning .txt",
        "pia-slides-01.txt",
        "pia-slides-02.txt",
        "pia-slides-03.txt",
        "pia-slides-04.txt",
        "pia-slides-05.txt",
        "Probability and Computing Randomization and Probabilistic Techniques in Algorithms
        and Data Analysis.txt",
        "PROLOG.txt",
        "Remote Sensing Digital Image Analysis.txt"]

contlibros=[0,0,0,0,0,0,0,0,0,0,0,0,0]

def construyeg(b): # construye un diccionario para representar la función g
        d={}
        for i in range(0,len(b)-1): # ignoramos el último caracter
                d[b[i]]=i
        return d

def bmh(a,b): # busca b dentro de a, retorna None (fracaso) o posición del calce
        g=construyeg(b) # Uso: g(c)=g.get(c,-1) para rellenar con -1 los valores faltantes
        n=len(a)
        m=len(b)
        k=m-1
        j=m-1
        while k<n:
                if j<0:
                        return k-m+1
                if a[k-(m-1-j)]==b[j]:
                        j=j-1
                else:
                k=k+(m-1-g.get(a[k],-1))
                j=m-1
        return None


#BUSQUEDA DE PALABRA EN ARCHIVO ESPECIFICO
palabra=input('Ingrese la palabra que desea buscar:')
```

```
texto=input("Ingrese el nombre del archivo de texto donde se buscara:")
cont=0
nlinea=0
with open(texto) as archivo:
        for linea in archivo:
                nlinea=nlinea+1
                busqueda=bmh(linea,palabra)
                if busqueda != None:
                        print("La palabra se encuentra en la linea:",nlinea)
                        print("La palabra se encuentra en el caracter:",busqueda)
                        cont=cont+1

print("El numero de veces que se encontro la palabra es:")
print(cont)

#BUSQUEDA DE PALABRA EN TODA LA CARPETA Y ORDENAMIENTO
palabra=input('Ingrese lo que desea buscar:')
palabras=palabra.split()
largo=len(palabras)

#si es una sola palabra
if largo==1:
        for i in range(13):
                cont=0
                nlinea=0
                libro=libros[i]
                with open(libro) as archivo:
                        for linea in archivo:
                                nlinea=nlinea+1
                                busqueda=bmh(linea,palabra)
                                if busqueda != None:
                                        cont=cont+1

                contlibros[i]=cont #se puede dividir entre nlinea para obtener la relacion de
                palabra/renglones

        dic=dict(zip(libros,contlibros))

        ord=(sorted(dic.items(),key=operator.itemgetter(1),reverse=True))
        for i in ord:
                print(i)

#si tiene mas de una palabra
```

```
if largo > 1:
        for i in range(13):

                cont=0
                nlinea=0
                libro=libros[i]
                with open(libro) as archivo:
                        for linea in archivo:
                                nlinea=nlinea+1
                                busqueda=bmh(linea,palabra)
                        if busqueda != None:
                                cont=cont+1
                        for j in range(largo):
                                palabrasola=palabras[j]
                                busqueda=bmh(linea,palabrasola)
                                if busqueda != None:
                                        cont=cont+1
                contlibros[i]=cont #se puede dividir entre nlinea para obtener la relacion de
                palabra/renglones

        dic=dict(zip(libros,contlibros))

        ord=(sorted(dic.items(),key=operator.itemgetter(1),reverse=True))
        for i in ord:
                print(i)
```

Para realizar las pruebas del código en el primer caso se busca la palabra “Dijkstra” en el libro Análisis de Algoritmos de Dr. Homero Ríos Figueroa y Dr. Fernando Montes González, obteniendo los siguientes resultados:

La palabra se encuentra en la linea: 321
La palabra se encuentra en el carácter: 2
La palabra se encuentra en la linea: 1683
La palabra se encuentra en el carácter: 71
La palabra se encuentra en la linea: 1698
La palabra se encuentra en el carácter: 47
La palabra se encuentra en la linea: 1763
La palabra se encuentra en el carácter: 39
La palabra se encuentra en la linea: 2048
La palabra se encuentra en el carácter: 52
La palabra se encuentra en la linea: 2775

La palabra se encuentra en el carácter: 28
El numero de veces que se encontró la palabra es:
6

Con lo cual podemos obtener el número total de veces que aparece la palabra en el texto, así como el lugar del archivo donde se encuentra dicha palabra.

Posteriormente se busca la palabra “artificial” en toda la carpeta, obteniendo los siguiente resultados, donde el número a la derecha es la cantidad de veces que aparece la palabra buscada:

('Artificial Intelligence A Modern Approach.txt', 67)
('Remote Sensing Digital Image Analysis.txt', 7)
('Analisis de Algoritmos.txt', 5)
('PROLOG.txt', 3)
('pia-slides-01.txt', 2)
('Introduction to algorithms.txt', 0)
('Introduction to the Design and Analysis of Algorithms.txt', 0)
('Pattern Recognition and Machine Learning .txt', 0)
('pia-slides-02.txt', 0)
('pia-slides-03.txt', 0)
('pia-slides-04.txt', 0)
('pia-slides-05.txt', 0)
('Probability and Computing Randomization and Probabilistic Techniques in Algorithms and Data Analysis.txt', 0)

Finalmente se busca la palabra “artificial intelligence” en toda la carpeta, obteniendo los siguientes resultados:

('Artificial Intelligence A Modern Approach.txt', 212)
('PROLOG.txt', 10)
('Remote Sensing Digital Image Analysis.txt', 8)
('Analisis de Algoritmos.txt', 5)
('Introduction to the Design and Analysis of Algorithms.txt', 4)
('pia-slides-01.txt', 4)
('Introduction to algorithms.txt', 1)
('Pattern Recognition and Machine Learning .txt', 0)
('pia-slides-02.txt', 0)
('pia-slides-03.txt', 0)
('pia-slides-04.txt', 0)

('pia-slides-05.txt', 0)
('Probability and Computing Randomization and Probabilistic Techniques in Algorithms and Data Analysis.txt', 0)

Conclusión:

Con lo cual se puede observar que el programa es capaz de presentar la lista de los archivos de texto ordenados del que mas veces tiene la palabra al que la tiene menos veces y por ultimo los que no tienen la palabra ni una vez. Ademas de ver que el algoritmo de hurspool es muy útil para realizar búsquedas de patrones en archivos de texto, ademas de tener una ejecución relativamente rapida.