Laboratorio de Computación 1 (2023)

Tarea 1 de programación

Descripción del problema:

En teoría de la información, la distancia de Hamming es la distancia entre dos palabras válidas de un código (o un conjunto predefinido de palabras válidas). La distancia entre dos palabras p1 y p2 en código binario (compuesta por ceros y unos) equivale a la cantidad de posiciones de p1 (bits) que deben cambiarse para transformar a p1 en p2. Cuanta más distancia exista entre las palabras de un código, es menos probable que una palabra válida, al intercambiar ceros por unos debidos a errores de transmisión, se transforme en otra palabra válida, por lo que la distancia de Hamming es utilizada para detectar y corregir estos errores. Aplicada a códigos formados por dígitos decimales, la distancia entre dos palabras es la cantidad de posiciones que difieren. Por ejemplo la distancia de Hamming entre p1=[1,2,3,4,3,2,3,4] y p2=[9,2,3,4,0,2,3,1] es 3, ya que hay tres posiciones que difieren (la 1, la 5 y la 8).

Desarrollar en Octave las siguientes funciones:

dist = distHamming(v1, v2), que recibe dos vectores de dígitos y devuelve la distancia de Hamming entre ambos vectores. Si los vectores son de distinto largo o alguno de los vectores es vacío, la función debe devolver -1.

[pos, dist] = encontrarMasParecido(v1, v2), que recibe dos vectores de dígitos v1 y v2, y debe encontrar la sección de v1 de largo length(v2), que tenga la menor distancia de Hamming a v2. Esta función retorna:

(pos) es la posición de comienzo en v1 de la sección encontrada.

(dist) es la distancia de Hamming entre la sección encontrada y v2.

Si v2 es más largo que v1 o alguno de los vectores es vacío, la función debe devolver -1 en ambos parámetros.

Si en v1 hay más de una secuencia con distancia mínima a v2 debe devolverse la posición en v1 más pequeña.

e = estaEnMatriz(M, v), que recibe una matriz de dígitos M, y un vector de dígitos v, y retorna e=1 si la secuencia v está contenida en alguna fila de la matriz y e=0 en caso contrario.

Ejemplos:

dist = distHamming([2,3,4,2],[2,3,4,2]) devuelve dist=0

dist = distHamming([2,3,4,2],[1,3,9,2]) devuelve dist=2

[pos,dist] = encontrarMasParecido([1,2,3,4,3,4,2,5,6,7,3,2],[3,4]) devuelve pos=3 y dist=0

[pos,dist] = encontrarMasParecido([1,2,3,4,3,4,2,5,6,7,3,2],[3,4,9,4]) devuelve pos=3 y dist=1

e = estaEnMatriz([2,3,4,5;6,7,3,2;1,3,2,1;2,3,1,2;5,6,7,0], [7,3]) devuelve e=1

e = estaEnMatriz([2,3,4,5;6,7,3,2;1,3,2,1;2,3,1,2;5,6,7,0], [7,7]) devuelve e=0

Se pide:

Entregar los 3 archivos .m proporcionados por separado **(no en un .zip o archivo comprimido)** modificando únicamente el contenido de las funciones.

Tener en cuenta:

No está permitido utilizar facilidades de Octave que permitan resolver los problemas de forma trivial (funciones max, min, sort, etc.)

Los valores válidos de entrada son vectores de cualquier largo (o matrices de cualquier tamaño en el caso de la función estaEnMatriz) cuyas entradas son dígitos entre 0 y 9. No se evaluará el comportamiento de la entrega ante entradas inválidas (como escalares, vectores con números de más de una cifra, números negativos, letras, etc.).

Es imprescindible para el proceso de corrección que las funciones respeten los nombres, la cantidad y el orden de los parámetros de entrada y salida establecidos en la letra. Se proporcionan 3 archivos donde **solo debe completarse el contenido de cada función (sin modificar el nombre o parámetros de entrada y salida)**. No se permite definir otras funciones además de las 3 mencionadas. Está permitido invocar cualquiera de las 3 funciones dentro de otra.

La entrega debe producir la salida correcta al menos para los casos que figuran en la letra, pero podrá ser evaluada con casos de prueba adicionales en la etapa de corrección.

Para la corrección, las tareas se ejecutarán con la versión 7.1.0 de Octave. La ejecución se realizará desde la línea de comandos (sin interfaz gráfica).

En esta tarea, como en todos los problemas de este curso, se valorará, además de la lógica correcta, la utilización de un buen estilo de programación de acuerdo a los criterios impartidos en el curso. De esta manera, se hará énfasis en buenas prácticas de programación que lleven a un código legible y bien documentado, tales como:

- indentación adecuada
- utilización correcta y apropiada de las estructuras de control
- código claro y legible
- algoritmos razonablemente eficientes
- utilización de comentarios que documenten y complementen el código
- nombres mnemotécnicos para variables, constantes, etc.