3) Dados los siguientes vectores en 7 dimensiones:

V1 = [0, 1, 0, 1, 1, 0, 0]

V2 = [1, 0, 1, 0, 1, 0, 1]

V3 = [1, 1, 1, 0, 0, 1, 0]

V4 = [0, 1, 0, 0, 1, 0, 1]

V5 = [1, 1, 1, 1, 1, 1, 0]

V6 = [1, 1, 1, 1, 0, 0, 0]

V7 = [0, 0, 0, 0, 0, 0, 1]

V8 = [0, 0, 0, 1, 1, 1, 0]

Se utiliza un minhash para la distancia de hamming usando los bits 0, 1 y 5. Es decir que tendremos r = 3 y b = 1.

Decidimos usar para LSH una tabla de hashing de 6 registros con buckets de tamaño 4.

a. Realice el pre-procesamiento necesario para poder usar LSH con los vectores. Indique precisamente qué estructura tendrá la tabla, qué se almacena en la misma y de qué forma.

b. Queremos hallar el vector más cercano a Q = [1, 0, 1, 0, 0, 0, 1]. Explique de qué forma encontraría dicho vector a partir de la estructura construida en el punto a.

c. La tabla usada para este ejercicio tiene capacidad para 24 registros que es el triple de los registros que almacenamos (8). Indique de qué forma podría reducirse la cantidad de espacio usada. (25pts)

**RESOLUCION**

1. Para resolver el pre-procesamiento, primero hay que considerar que los hiperplanos con los cuales se resolverá la distancia de hamming son:

H1 = [0,0,0,0,0,0,1]

H2 = [0,0,0,0,0,1,0]

H3 = [0,1,0,0,0,0,0]

Por lo tanto los minhashes quedan definidos como:

MH(Vi,Hj) Bit j prendido en Hj == Bit j en Vi == es 0, != es 1

Y si aplico los minhashes para cada uno de los vectores:

MH(V1,H1) = 0 ; MH(V1,H2) = 1 ; MH(V1,H3) = 0

MH(V2,H1) = 1 ; MH(V2,H2) = 0 ; MH(V2,H3) = 0

MH(V3,H1) = 1 ; MH(V3,H2) = 1 ; MH(V3,H3) = 1

MH(V4,H1) = 0 ; MH(V4,H2) = 1 ; MH(V4,H3) = 0

MH(V5,H1) = 1 ; MH(V5,H2) = 1 ; MH(V5,H3) = 1

MH(V6,H1) = 1 ; MH(V6,H2) = 1 ; MH(V6,H3) = 0

MH(V7,H1) = 0 ; MH(V7,H2) = 0 ; MH(V7,H3) = 0

MH(V8,H1) = 0 ; MH(V8,H2) = 0 ; MH(V8,H3) = 1

Y para aplicar los minhashing en conjunto (AND minhashes por ser r=3), entonces defino una familia de hashing:

F(MH1,MH2,MH3) = (a.MH1 + b.MH2 + c.MH3) % 7 % 6

Entonces si aplicamos la familia para todos los vectores, tomando a=4, b=2, c=1 nos queda que:

F(V1) = (4\*0 + 2\*1 + 1\*0) % 7 % 6 = 2%7%6 = 2

F(V2) = (4\*1 + 2\*0 + 1\*0) % 7 % 6 = 4%7%6 = 4

F(V3) = (4\*1 + 2\*1 + 1\*1) % 7 % 6 = 7%7%6 = 0

F(V4) = (4\*0 + 2\*1 + 1\*0) % 7 % 6 = 2%7%6 = 2

F(V5) = (4\*1 + 2\*1 + 1\*1) % 7 % 6 = 7%7%6 = 0

F(V6) = (4\*1 + 2\*1 + 1\*0) % 7 % 6 = 6%7%6 = 0

F(V7) = (4\*0 + 2\*0 + 1\*0) % 7 % 6 = 0%7%6 = 0

F(V8) = (4\*0 + 2\*0 + 1\*1) % 7 % 6 = 1%7%6 = 1

Por lo que la table de hashing queda:

|  |
| --- |
| {V3, V5, V6, V7} |
| {V8} |
| {V1,V4} |
|  |
| {V2} |
|  |

1. Q=[1,0,1,0,0,0,1].

Para encontrar los elementos similares al query lo que se debe hacer es pasar por el Hashing de LSH al vector query. De esta manera, se le asignaría una posición en la tabla en base al resultado de los minhashing del mismo.

Debido a la definición de LSH, aquellos elementos que colisionen en una misma posición es porque son similares entre sí; entonces, aplicando hashing a Q:

MH1(Q) = 1 ; MH2(Q) = 0 ; MH3(Q) = 0

F(Q) = (4\*1 + 2\*0 + 1\*0) % 7 % 6 = 4

Entonces el Query es similar a los elementos que están en la posición 4 de la tabla de hash. Obteniendo los elementos de esa posición tenemos que el Query es similar al vector V2.

Si se quiere explicitar una diferencia máxima entre el Query y los candidatos a ser similares, entonces ahora con una cantidad de elementos reducidos podemos comparar 1 a 1 para llegar a conclusiones finales reales de cuales cumplen con nuestra similitud mínima para considerarlos similares.

1. Lo que debería realizarse es una optimización con respecto a la familia de Hashing, para que de esta manera, habiendo 6 posiciones, los elementos se almacenen 1 o 2 por bucket. De esta manera, podrían disminuirse la cantidad de registros por buckets a 2.