Algoritmos y Estructuras de Datos
-DEMO- Examen Final -PRACTICA- -NOT THE REAL THING-

Conversion de Notas:
60-63 = 4
64-69 = 5
70-76 = 6
77-83 = 7
84-89 = 8
90-96 = 9
97+ = 10

Nombre:
Legajo:

Instrucciones: Los ejercicios deberían poder completarse en el enunciado mismo, en caso de quedarse sin lugar puede usar alguna hoja auxiliar.

Rúbrica:

Ej	1	2	3	4	5	6	7	8	9	10	Total
Pts	10	10	10	10	10	10	10	10	10	10	100

[1] Indicar V o F en cada una de las siguientes afirmaciones sobre funciones de hashing:

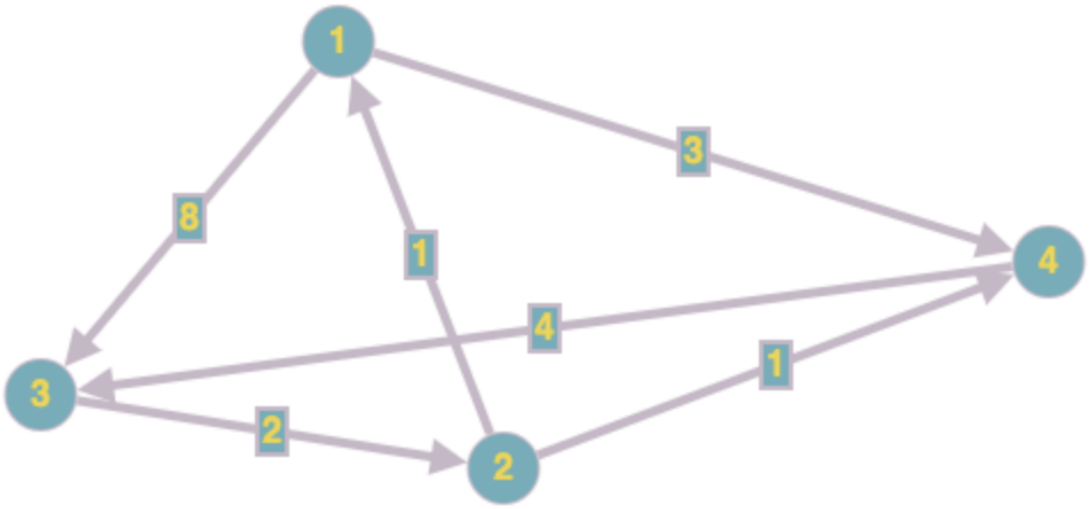
a) Si la función de hashing tiene pocas colisiones entonces la podemos usar como función de hashing criptográfica.
b) SHA256 produce menos colisiones que FNV.
c) La composición de dos funciones de hashing es una función de hashing.
d) Si h es una función de hashing entonces a*h(x) + b mod m es una función de hashing.

[2] Realizar las siguientes operaciones sobre la tabla de hashing usando como método de resolución de colisiones direccionamiento abierto y lineal simple.

Pos	Flag	Clave
0		
1		
2		
3		

insert(“azul”), insert(“verde”),insert(“amarillo”), insert(“blanco”),
delete(“azul”),insert(“naranja”),insert(“rojo”)
h(“azul”)=2, h(“verde”)=3, h(“amarillo”)=3, h(“blanco”)=2, h(“rojo”)=1, h(“naranja”)=1, h(“violeta”)=0
Finalmente indique la cantidad de pasos necesarios para buscar “blanco” y “violeta” en la tabla.

[3] Dado el siguiente grafo:



Calcular todos los caminos mínimos desde el nodo 1 usando el algoritmo de Dijkstra.

[4] Dado un grafo no-dirigido queremos programar la funcion path_exists(a,b, max_dist) que indique si existe un camino entre a y b de distancia max_dist o menor. Programar la función y analizar el orden de la misma. Tiene que ser lo más eficiente posible.

[5] Un robot puede recibir instrucciones para moverse hacia adelante 1,2 o 3 metros. Queremos que programe una función que calcule dada una distancia,de cuántas formas posibles puede el robot alcanzarla. Por ejemplo si la distancia es 3 hay 4 formas posibles: 1,1,1 / 1,2 / 2,1 / 3

[6] Dado el siguiente vector:

30	21	8	45	62	73	9	6	2	11	20	19

Usando bloques con b=3 y tablas dispersas tanto para el superbloque como para los bloques individuales le pedimos que muestre la tabla del superbloque y las tablas individuales de cada bloque, estas últimas tiene que construirlas usando el método de los cuatro rusos. Una vez construidas las estructuras indique cómo resuelve las consultas RMQ(1,7). RMQ(4,11)

[7] Dibujar el árbol cartesiano para el siguiente vector: 30,21,11,4,72,61,8,29

[8] Tenemos un count-min filter que tiene la siguiente estructura:

0	3	0	2	0	1	1
1	0	0	3	1	1	1
2	0	0	2	2	0	1

a) Estime la cantidad de apariciones de “amarillo” con h1=3, h2=3, h3=0
   _____
b) Indique cuántos elementos en total hemos insertado hasta ahora en el filtro.
   _____
c) ¿Cuál es la cardinalidad máxima posible que podemos devolver?
   _____
d) ¿Cuál es la cardinalidad mínima que podemos devolver?
   _____
e) ¿Qué valores debería darnos la función de hashing de un elemento nuevo para que su estimación sea exacta, es decir 1?
   _____

[9] Dibuje un grafo tal que:

- El grafo debe ser dirigido.
- Debe tener al menos 4 vertices y 6 aristas
- Las aristas deben tener pesos, algunas deben tener peso negativo.
- No puede haber ciclos negativos.
- Tiene que tener al menos un ciclo.

Finalmente aplique Floyd Warshall sobre el grafo mostrando el resultado final de la matriz.

[10] Dada una matriz de nxn (matriz cuadrada) queremos resolver el problema del elemento mínimo en una cierta submatriz: RMQ(i1,j1,i2,j2) que nos tiene que devolver el elemento mínimo de dicha submatriz, las submatrices pueden no ser cuadradas. Le pedimos diseñar una solución que permita resolver las consultas en O(n) (ojo no es O(nxn)), para ello va a necesitar un cierto pre-procesamiento que va a tener que explicar e indicar el orden del mismo.