Estructuras de Datos 2 - ST0247
Examen Parcial 2

Nombre:.....
Departamento de Informática y Sistemas
Universidad EAFIT

Octubre 26 de 2017

Criterios de calificación

- Selección múltiple con única respuesta
  - Respuesta correcta: 100 %
  - Respuesta incorrecta: 0 %
- Completar código
  - Respuesta correcta 100 %
  - Respuesta incorrecta o vacía 0 %

NOTAS IMPORTANTES:

- Responda en la hoja de PREGUNTAS
- Marque la hoja de PREGUNTAS

1. Prog. Dinámica 30 %

La distancia de Levenshtein es el número mínimo de operaciones requeridas para transformar una cadena de caracteres en otra. Se entiende por operación, una inserción, eliminación o la sustitución de un carácter. Es útil en programas que determinan cuán similares son dos cadenas de caracteres, como es el caso de los correctores de ortografía. Como un ejemplo, la distancia de Levenshtein entre "casa" y "calle" es de 3 porque se necesitan al menos tres ediciones elementales para cambiar uno en el otro:

1. casa → cala (sustitución de 's' por 'l')
2. cala → calla (inserción de 'l' entre 'l' y 'a')
3. calla → calle (sustitución de 'a' por 'e')

A continuación está una implementación del algoritmo en Java:

```
int minimum(int a, int b, int c) {
  return Math.min(Math.min(a, b), c);
}

int Levenshtein(String lhs, String rhs) {
  int[][] distance = new int[lhs.length() + 1]
                            [rhs.length() +1];
  for (int i = 0; i <= lhs.length(); i++)
    distance[i][0] = i;
  for (int j = 1; j <= rhs.length(); j++)
    distance[0][j] = j;
  for (int i = 1; i <= lhs.length(); i++)
    for (int j = 1; j <= rhs.length(); j++)
     distance[i][j] = minimum(
      distance[i - 1][j] + 1,
      distance[i][j - 1] + 1,
      distance[i - 1][j - 1] +
      ((lhs.charAt(i - 1) ==
        rhs.charAt(j - 1)) ? 0 : 1));
  return distance[lhs.length()][rhs.length()];
}
```

En Java, el operador incógnita (?) funciona de la siguiente forma: Si `algo` es verdadero, entonces retorna un `valor`, de lo contrario retorna otro `valor`, así: `algo? un valor : otro valor`.

(15 %) Complete la siguiente tabla, siguiendo el algoritmo de programación dinámica de la distancia de Levenshtein:

		c	a	l	l	e
c						
a						
s						
a						

(15 %) Complete la siguiente tabla, siguiendo el algoritmo de programación dinámica de la distancia de Levenshtein, para encontrar la distancia entre madre y mama:

		m	a	d	r	e
m						
a						
m						
a						

2. Prog. Dinámica 20 %

Considere un grafo dirigido representado por la siguiente matriz de adyacencia:

	0	1	2	3
0	0	1	15	6
1	2	0	7	3
2	9	6	0	12
3	10	4	8	0

3.1 (10 %) Complete, por favor, la siguiente tabla, usando el algoritmo de programación dinámica del agente viajero, también conocido como el **algoritmo de Held-Karp**. El algoritmo funciona de la siguiente forma. Primero, calcula todos los subconjuntos del conjunto de vértices sin incluir el primer vértice (es decir, el vértice 0). Después, crea una tabla en la que se coloca cuál es el costo del camino más corto para ir desde 0 hasta cada vértice pasando por los vértices del conjunto vacío. Después, se hace la misma pregunta pero esta vez pasando por los vértices de un conjunto de un elemento. Posteriormente, la misma pregunta pero pasando por los vértices de un conjunto de dos elementos. Y así sucesivamente, hasta poder responder la pregunta de cuál es el costo mínimo de un camino que inicia en el vértice 0 y termina en el vértice 0 y pasa por todos los vértices del conjunto de vértices quitando el vértice 0. Para cada paso, utiliza los valores calculados previamente en la tabla. Para el primer paso utiliza solamente la información que hay en la matriz de adyacencia que representa el grafo.

En la tabla, la primera línea significa, ¿cuál es el costo para ir desde 0 hasta 1 sin pasar por ningún otro vértice (es decir, $\emptyset$) y quién el antecesor visitado antes de ir al vértice 1.

	Costo	Antecesor
$[1, \emptyset]$	1	0
$[2, \emptyset]$		
$[3, \emptyset]$		
$[2, \{1\}]$		
$[3, \{1\}]$		
$[1, \{2\}]$		
$[3, \{2\}]$		
$[1, \{3\}]$		
$[2, \{3\}]$		
$[3, \{1, 2\}]$		
$[1, \{2, 3\}]$		
$[2, \{1, 3\}]$		
$[0, \{1, 2, 3\}]$		

3.2 (10 %) ¿Cuál es el camino con la solución y el costo (también conocido como el peso o la distancia) total del camino?

Camino: ___ → ___ → ___ → ___ → ___ Costo: ___

3. Algo. voraces 30 %

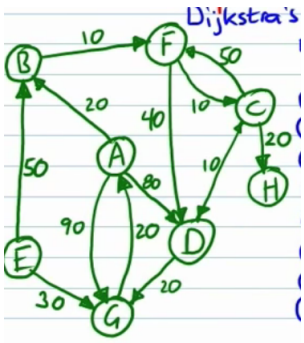
El algoritmo de Dijkstra sirve para encontrar el camino más corto de un vértice a todos los demás de un grafo. A continuación una implementación en Java.

```
int minVertex (int [] dist, boolean [] v) {
 int x = Integer.MAX_VALUE; //Infinity
 int y = -1;
 for (int i=0; i<dist.length; i++)
  if (!v[i] && dist[i]<x)
     y=i; x=dist[i];
 return y;
}

int [] dijsktra(Graph dg, int source) {
  int [] dist = new int [dg.size()];
  int [] pred = new int [dg.size()];
  boolean [] visited = new boolean [dg.size()];
  for (int i=0; i<dist.length; i++)
    dist[i] = Integer.MAX_VALUE;
  dist[source] = 0;
  for (int i=0; i<dist.length; i++) {
    int next = minVertex (dist, visited);
```

```
    visited[next] = true;
    ArrayList<Integer> n =
        dg.getSuccessors (next);
    for (int j=0; j<n.size(); j++) {
      int v = n.get(j);
      int d = dist[next] +
          dg.getWeight(next,v);
      if (dist[v] > d) {
        dist[v] = d;
        pred[v] = next;
      }}}
  return pred;
}
```

Considere el siguiente grafo:



4.1 (20 %) Complete, por favor, la siguiente tabla, usando el algoritmo de Dijkstra para encontrar el camino más corto del punto A a todos los demás. En la tabla, la palabra "Vi" significa "Visitados".

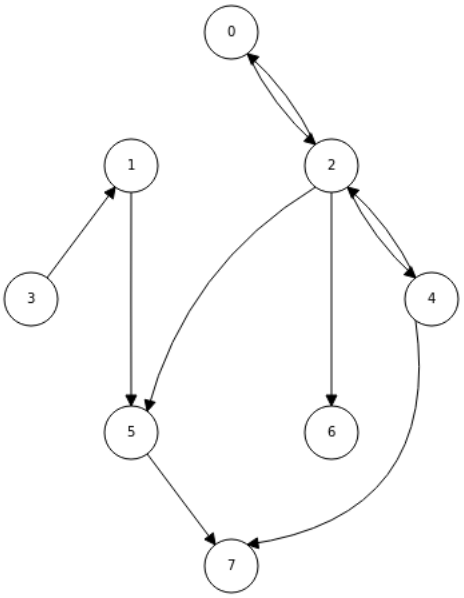
Paso	Vi	B	C	D	E	F	G	H
1	A	20,A	∞	80, A	∞	∞	90, A	∞
2	B	20,A	∞	80, A	∞	30,B	90,A	∞
3	F							
4								
5								
6								
7								
8								

4.2 (10 %) ¿Cuál es el camino más corto de A a G?

---

4. Recorridos de grafos 20 %

Para el grafo siguiente, complete la salida.



Complete el orden en que se recorren los nodos usando **búsqueda en amplitud** (en Inglés BFS) a partir de cada nodo. Si hay varias opciones de recorrer el grafo con BFS, elija siempre el vértice más pequeño.

$0 \rightarrow$
$1 \rightarrow 5 \rightarrow 7$
$2 \rightarrow$
$3 \rightarrow$
$4 \rightarrow$
$5 \rightarrow$
$6 \rightarrow$
$7 \rightarrow$