Solución Examen 2

Este examen es individual. Tiene una duración de 1 hora 20 min. Está prohibido usar el libro, apuntes, impresiones y cualquier tipo de dispositivo electrónico (celulares, agendas electrónicas, cámaras digitales, etc.).

Nombre:____ Código: ____________________

1. Dada una cadena de caracteres y un conjunto de palabras que representa el diccionario de palabras válidas en un lenguaje, determinar si se puede representar la cadena como una concatenación de palabras válidas en el diccionario. Se permite repetir palabras del diccionario para representar la cadena.

Ejemplos:

Dado el siguiente diccionario con 3 palabras:

ayer
hoy
para

La cadena paraayerparahoy se puede representar como la concatenacion de las palabras para, ayer, para y hoy

Por otro lado la cadena parayerparahoy no se puede representar como una concatenación de palabras en el diccionario.

a. [10%] Describir entradas y salidas del problema

E/S	Nombre	Tipo	Descripción
E	a	String	Cadena a representar
E	d	Array [0,N) of String	Palabras del diccionario
S	b	boolean	True si se puede armar s con las palabras en d, con la posibilidad de repetir palabras

b. [15%] Definir una ecuación de recurrencia para una función r(i,j) que determine si existe alguna forma de representar las primeras i letras de la cadena como una concatenación de palabras en en el diccionario y terminando en la palabra j del diccionario

Utilizando como caso base la cadena vacia, representada por i=0 y la cual siempre se puede representar, la recurrencia depende de si la palabra j es sufijo del prefijo de tamaño i de la cadena. En caso de no serlo, r(i,j) es falso. En caso de serlo, la respuesta depende de que exista alguna forma de representar el prefijo de tamaño i-len(d[j]) con la posibilidad de usar todas las palabras de d. Se define primero un predicado para determinar si una cadena c es sufijo del prefijo de tamaño i de una cadena b:

$$s(b,i,c) \equiv i \geq len(c) \wedge (\forall k \mid 0 \leq k < len(c): b[i-len(c)+k] = c[k])$$

Con esto, se define r con la siguiente ecuación de recurrencia:

$$r(i,j) \equiv \textbf{true} \text{ si } i=0$$
$$r(i,j) \equiv \textbf{false} \text{ si } i>0 \wedge \neg s(a,i,d[j])$$
$$r(i,j) \equiv s(a,i,d[j]) \wedge (\exists k \mid 0 \leq k<n: r(i-len(d[j],k))) \text{ si } i>0 \wedge s(a,i,d[j])$$

La respuesta al problema sería la disyunción de los posibles valores de r(len(a),k)

c. [10%] Dibujar el grafo de necesidades relacionado con esta ecuación. Explicar qué representan los vértices y qué representan los ejes del grafo. Determinar si el grafo tiene ciclos.

	la	hoy	mala	hora
	T	T	T	T
l	F	F	F	F
a	T	F	F	F
m	F	F	F	F
a	F	F	F	F
l	F	F	F	F
a	F	F	T	F

Los vértices corresponden a los posibles valores de r. Los ejes son los valores necesarios para calcular cada valor de r(i,j) de acuerdo con la ecuación de recurrencia. El grafo no tiene ciclos porque todas las dependencias tienen un valor menor que i

d. [20%] Desarrollar un algoritmo de programación dinámica para resolver el problema.

```
fun armar (a: str, d: array[0,N) of str) ret b: bool
var r: array [0,len(a)]x[0,N) of bool
var i,j,k: nat
var s: bool
i,b:=0,false
do i≤len(a) →
      j:=0
      do j< N →
            s:=substrP(a,i,d[j])
            if i = 0 → r[i,j] := true
            [] i > 0 ∧ ¬s → r[i,j] := false
            [] i > 0 ∧ s →
                  r[i,j],k := false,0
                  do k<N → r[i,j],k := r[i,j] V r[i-len(d[j]),k],k+1 od
            fi
            if i== len(a) → b := b V r[i,j]
            [] i< len(a) → skip
            fi
            j:=j+1
      od
      i:=i+1
od
ret b

fun substrP (b: str, l: nat, c: str ) ret r: bool
var k,n: nat
n,k,r:= len(c),0,l≥len(c)
do k < n ∧ r → r,k:=r ∧ b[l-n+k]=c[k],k+1  od
ret r
```

e. [10%] Determinar la complejidad temporal y espacial del algoritmo.

T(M,N) es O(M*N^2) donde M es el tamaño de la cadena a y N el tamaño del diccionario. El ciclo externo da M vueltas y tanto el ciclo controlado por j como el que calcula el valor de r[i,j] en el caso recursivo dan N vueltas.

S(M,N) es O(M*N) porque la matriz r tiene dimensiones (M+1)*N

2. Diseñar un algoritmo para resolver el siguiente problema: En biología de sistemas se dice que un gen A regula (positivamente) a otro gen B si es necesario que la proteína que se produce desde A esté expresada para que se pueda expresar el gen B. Se conoce como “regulador maestro” a un gen que, ya sea de manera directa o indirecta (a traves de una cadena de reguladores), sea necesario para la expresión de todos los genes de la red. Dada una red de regulación, se quiere determinar si existe algun regulador maestro.

a. [10%] Describir entradas y salidas del problema

E/S	Nombre	Tipo	Descripción
E	G	Array [0,N) of Set of nat	Grafo dirigido representado como lista de adyacencias. Se numeran los genes de 0 hasta N-1. Cada posición i del arreglo contiene un conjunto de los ids de los genes regulados por i
S	b	bool	True si existe algun regulador maestro

b. [25%] Desarrollar un algoritmo que resuelva el problema

Desde cada vértice fuente se hace un recorrido. Si el recorrido cubre todos los vértices, entonces el gen fuente es un regulador maestro

```
fun findMaster (G: array [0,N) of Set of nat) ret b: bool
var i: nat
i,b:=0,false;
do i<N ∧ ¬b → b,i := b V ismaster (G, i),i+1 od
ret b

fun ismaster ( G: array [0,N) of Set of nat, s: nat) ret b:bool
var q: queue of nat
var visited: array [0,N) of bool
var v,w,n,i:nat
q.enqueue(s)
i:=0
do i<N → visited[i],i:=false,i+1 od
n,visited[s]:=1,true
do q.size()>0 →
      v:= q.dequeue()
      for w in E[v] →
            if ¬visited[w] →
                  q.enqueue(w)
                  n,visited[w] :=n+1, true
            [] visited[w] → skip
            fi
      rof
od
b:= n=N
ret b
```