

Modelo del primer parcial

1. (30 puntos) Dado un multiconjunto (es decir, un conjunto con repeticiones) de números enteros $C = \{a_1, a_2, \dots, a_n\}$ y un número k , el SUBSET-SUM PROBLEM consiste en determinar si existe un subconjunto $S \subseteq C$ tal que la suma de los elementos en S sea k . Por ejemplo, si tenemos como entrada el conjunto $C = \{2, 2, 5, 10\}$ y $k = 9$, entonces la respuesta es el multiconjunto $S = \{2, 2, 5\}$, cuya suma es 9. En cambio, para $k = 11$ no hay solución al problema.

Consideremos el siguiente algoritmo de programación dinámica, que contiene dos bugs.

```

1 bool subsetsum_pd(int* C, int n, int k)
2 {
3     // Retorna una matriz de bool de nxk
4     // Los valores por default es False
5     m** = crear_matriz<bool>(n+1,k+1);
6
7     for (int i = 0; i <= n; i++)
8         m[i][0] = False;
9         True
10    for (int l = 0; l <= k; l++)
11        m[0][l] = False;
12
13    for (int i = 1; i <= n; i++) {
14        for (int l = 1; l <= k; l++) {
15            if (l < C[i]) No lo pongo
16                m[i][l] = m[i-1][l] manu;
17            else lo pongo
18                m[i][l] = m[i-1][l] || m[i-1][l - C[i]];
19        }
20    }
21
22    return m[n][k];
23 }
```

$C = \{2, 2, 3\}$ $k = 4$

$n = 3$

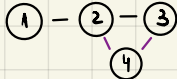
$\begin{smallmatrix} n \backslash k \\ \end{smallmatrix}$	0	1	2	3	4
0	T	F	F	F	F
1	T				
2	T				
3	T				


- a) Dar dos casos de test con que consideren $|C| \geq 2$ para evidenciar cada uno de los bugs.¹
 - b) Proponer una corrección para este algoritmo, y argumentar por qué con esta corrección los casos de test del punto anterior dejan de fallar.
2. (30 puntos) Un grafo es un árbol si no tiene ciclos y es conexo.
- a) Existe algún árbol T con 4 vértices, tres de grado 1 y uno de grado 2?
 - b) Demostrar que si se agrega una arista nueva a un árbol se crea un ciclo.
3. El algoritmo de Dijkstra resuelve el problema de camino mínimo en grafos dirigidos.
- a) Dado un digrafo $D = (N, A)$ con una función de distancia $w : A \rightarrow \mathbb{R}_{\geq 0}$, y sea $s \in N$. Proponer un algoritmo que retorne un subconjunto $S \subseteq N$ con todos los nodos tal que el costo de llegar desde s es lo sumo C . Justificar por qué el algoritmo es correcto.

¹No forma parte del modelo de parcial: se puede dar un caso de test donde la función actual retorne el valor correcto?

Ejercicio 2

a) Árbol con 4 vértices, 3 de grado 1 y uno de grado 2.

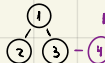
Parecería que no. Primero, si fuerzo que haya 3 nodos de grado 1, hasta ahí todo OK. Pero donde agregue el 4º nodo si lo fuerzo a tener grado 2, rompo con algún grado de algún nodo, si o si.  (uno de 3 posibilidades, es lo mismo).

Y si fuerzo que haya uno de grado 2?  1 tiene grado 2 y 2 y 3 grado 1. Por ahora todo OK.

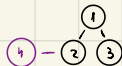
Agrego el 4º (grado 1 si o si).



No cumple



No cumple



No cumple

Una handshaking

$$\sum d(v) = 2 \times \text{cant. aristas}$$

$$3 + 2 = 2 \times 3$$

$$5 \neq 6 \quad \text{NO VALE!}$$

b) Si tengo l nodos, donde $l-1$ son hijos y uno es raíz y hay $l-1$ aristas, si agrego una arista más (o sea que hay l aristas) se agrega un camino para ese par de vértices que antes no tenía, entonces como x def tienen un único camino y ahora ya no hay un único camino, se forma un ciclo.



petita

- b) ¿Cómo se puede utilizar el algoritmo de Dijkstra para determinar si un grafo **no dirigido** es conexo?. Justificar y escribir el pseudocódigo del algoritmo propuesto.

Ej. 3

a) subconjunto (s, grafo, c)

di = dijkstra(s, grafo) → devuelve una lista de tuplas tipo [(nodo, dist), ..., (nodo, dist)]

for d in di:

if d[1] <= c:

res.append(d[0])

return res

b) Dijkstra funciona tanto con grafos dirigidos como con digrafos.

Podemos ejecutarlo, y el mismo devuelve las distancias desde s a todos los nodos. Si llego al punto en que una distancia es 0 es porque ya recorri toda la comp. conexa de s, ya que siempre agarré el vértice de menor dist. Entonces, de ahora en más todos los vértices sin visitar serán desconectados y el grafo será no conexo.

es_conexo(s, grafo)

res = true

di = dijkstra(s, grafo) → [(nodo, dist), ..., (nodo, dist)]

for d in di:

if d[1] == inf:

res = false

return res