[redacted]

88/100

A

UNIVERSIDAD TORCUATO DI TELLA

Tecnología Digital V: Diseño de algoritmos
Licenciatura en Tecnología Digital
Primer Semestre, 2023

Primer parcial

1. (35 puntos) Dado un conjunto A de números enteros positivos y dos enteros ℓ y u, el RANGE-SUM PROBLEM consiste en determinar si existe un subconjunto $S \subseteq A$ tal que la suma de los elementos de S esté entre ℓ y u. Por ejemplo, si tenemos como entrada el conjunto $C = \{2, 3, 6, 10\}$ y además $\ell = 14$ y $u = 16$, entonces la respuesta es afirmativa, dado que el subconjunto $S = \{2, 3, 10\}$ suma 15, que está entre 14 y 16. Consideremos el siguiente algoritmo recursivo para este problema. En esta función, el primer parámetro es un arreglo de enteros que contiene el conjunto A, el segundo parámetro es la longitud del arreglo A, y los siguientes parámetros son los valores ℓ y u.

```
1  bool rangesum(int* A, int n, int ℓ, int u)
2  {
3      if (n == 0)
4      {
5          return ℓ <= 0 && u >= 0;
6      }
7      else
8      {
9          int ultimo = A[n-1];
10         return rangesum(C, n-1, ℓ-ultimo, u-ultimo)
11             || rangesum(C, n-1, ℓ, u);
12     }
13 }
```

8 *a*) (15 puntos) Explicar por qué el algoritmo es correcto. Pista: Observar que las llamadas recursivas de las líneas 10 y 11 consideran el caso en el que $A[n-1]$ pertenezca y el caso en el que $A[n-1]$ no pertenezca al subconjunto buscado, respectivamente.

20 *b*) (20 puntos) Si $u < 0$ entonces ya no tiene sentido realizar la recursión, porque no existe ningún subconjunto de A que tenga una suma negativa. ¿Cómo se puede modiificar el código para interrumpir la recursión en este caso?

2. (30 puntos) Un grafo es un árbol si no tiene ciclos y es conexo.

10 *a*) (10 puntos) Existe algún árbol T con tres o más vértices cuyo complemento sea isomorfo a T?

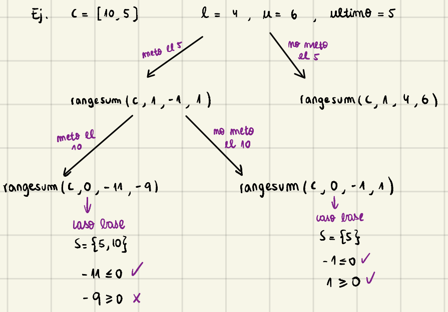
15 *b*) (20 puntos) Demostrar que si se elimina de un árbol un vértice de grado d, entonces el grafo resultante tiene exactamente d componentes conexas.

3. (35 puntos) Dado un digrafo $D = (N, A)$ con una función de distancia $w : A \to \mathbb{R}_{\geq 0}$, buscamos encontrar caminos mínimos que pasan por un determinado nodo $k \in N$.

20 *a*) (20 puntos) Implementar una función que, dado el digrafo pesado D y dados tres nodos $s, t, k \in N$, retorne la longitud de un camino mínimo de s a t que pase por k. Justificar por qué la función es correcta. Sugencia: Utilizar el Algoritmo de Dijkstra como algoritmo auxiliar.

15 *b*) (15 puntos) Asumiendo que se tiene la función del item anterior, implementar una función que calcule para todo par de vértices $s, t \in N$, la longitud del camino mínimo que pasa por k (k es parámetro de la función). Justificar por qué la función es correcta.

Ejercicio 1

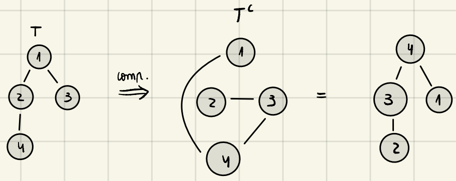


En primer lugar, es correcto que en base a un problema, lo va achicando hasta llegar al caso base. Aparte, considera todos los casos, si el "actual" pertenece al subconjunto o si no.

b)
```
else if (u ≥ 0) {
    choclo
} else {
    return;
}
```

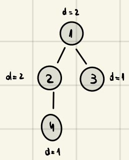
Ejercicio 2

a)



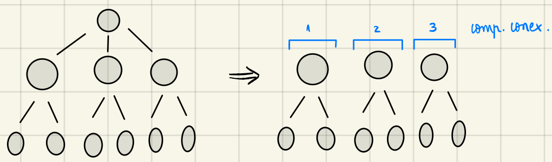
es isomorfo a T!

b)

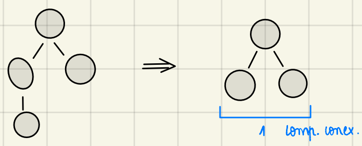


Si se elimina un nodo de grado d, pueden suceder 3 situaciones:

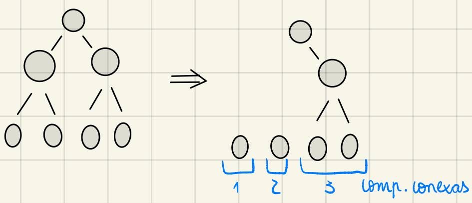
* Que se elimine una raíz: una raíz tiene grado d, lo mismo que decir que tiene d hijos. Si se elimina el padre de d hijos, quedarán d componentes que no están conectadas entre sí, pero que son conexas. Ej.



* Que se elimine una hoja: como TODA hoja tiene grado 1, siempre está asociada a su padre y ya. Si se borra no afecta al grafo, quedándose como la única componente conexa.



* Que se elimine uno del medio: si sucede este caso, estaría arruinando las conexiones entre su padre y su/s hijo/s. Esas conexiones que arruina es igual a su grado y a la cant. de comp. conexas que deja.



Ejercicio 3

a) Digrafo D = (N, A)

```
cam_min_K (D, s, t, K, res)
    D1 = Dijkstra (D, s, K)      (obligo que pase por K)
    D2 = Dijkstra (D, K, t)
    res = D1 + D2
    return res
```

b)

```
for (int i=0, i < |N|, i++)
    for (int j=0, j < |N|, j++)
        if (i ≠ j):
            cam_min_K (D, N[i], N[j], K, res)
return res
```