Mini-entrega 1: Subset-sum problem

Ignacio Pardo

2023-03-16

Enunciado

Dado un multiconjunto (es decir, un conjunto con repeticiones) de números enteros C = {a1, a2, . . . , an} y un número k, el SUBSET-SUM PROBLEM consiste en determinar si existe un subconjunto S ⊆ 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.

```
bool subsetsum(int* C, int n, int k)
{
    if (n == 0)
    {
        return k == 0;
    }
    else
    {
        int ultimo = C[n-1];
        return subsetsum(C, n-1, k) && subsetsum(C, n-1, k-ultimo);
    }
}
```

1. Este algoritmo tiene un bug. ¿Dónde está el problema?

Al llamar a la recursión, los resultados de las llamadas se juntan con un AND, cuando debería ser un OR, ya que si el resultado de una llamada es false, no implicaría que el resultado de la recursión sea false.

2. Escribir un caso de test que manifieste este bug.

```
int C[] = {3, 3, 7, 9};
int n = 4;
int k = 15;
```

Para el la rama que siempre llama `subsetsum(C, n-1, k)`:

```
- n: 4, k: 15, c: {3 3 7 9} -> subsetsum({3 3 7}, 3, 15)
- n: 3, k: 15, c: {3 3 7} -> subsetsum({3 3}, 2, 15)
- n: 2, k: 15, c: {3 3} -> subsetsum({3}, 1, 15)
- n: 1, k: 15, c: {3} -> subsetsum({}, 0, 15)
- n: 0, k: 15 == 0 -> false
```

Cuando en cambio, para la rama que siempre llama `subsetsum(C, n-1, k-ultimo)`:

```
- n: 4, k: 15, c: {3 3 7 9} -> subsetsum({3 3 7}, 3, 15)
- n: 3, k: 6, c: {3 3 7}    -> subsetsum({3 3}, 2, 6)
- n: 2, k: 6, c: {3 3}  -> subsetsum({3}, 1, 6)
- n: 1, k: 3, c: {3}    -> subsetsum({}, 0, 3)
- n: 0, k: 0 == 0 -> true
```

Se cumple que un llamado a la recursión es true, por lo que el resultado de la recursión debería ser true, pero el algoritmo lo devuelve como false ya que la union de todos los resultados resulta en false por el AND: `false && ... && subsetsum(C, n-1, k-ultimo) -> false`

3. Proponer una corrección para este algoritmo, y argumentar por qué con esta corrección el caso de test del punto anterior deja de fallar.

Reemplazando:

```
return subsetsum(C, n-1, k) && subsetsum(C, n-1, k-ultimo);
```

Por:

```
return subsetsum(C, n-1, k) || subsetsum(C, n-1, k-ultimo);
```

Solucionamos la union de los resultados de las llamadas a la recursión. Esto hace que si un llamado a la recursión es true, el resultado de la recursión sea true, pero en cambio, si todos los llamados a la recursión son false, el resultado de la recursión sea false.

```
#include <iostream>

using namespace std;

bool subsetsum(int* C, int n, int k)
{
    cout << "n: " << n << " k: " << k << endl;

    for (int i = 0; i < n; i++)
    {
        cout << C[i] << " ";
    }

    cout << endl;

    if (n == 0)
    {
        return k == 0;
    }
    else
    {
        int ultimo = C[n-1];
        // return subsetsum(C, n-1, k) && subsetsum(C, n-1, k-ultimo);
        return subsetsum(C, n-1, k) || subsetsum(C, n-1, k-ultimo);
    }
}

int main()
{
    int C[] = {3, 3, 7, 9};
    int n = 4;
    int k = 15;
    bool res = subsetsum(C, n, k);
    cout << res << endl;
}
```