Soluciones entrenos OIE
Detecting Molecules

La compañía para la que Petr trabaja ha construído una máquina detectora de moléculas. Cada molécula tiene un peso entero positivo. La máquina tiene un rangode detección $[l, u]$, siendo l y u enteros positivos. La máquina puede detectar correctamente un cierto conjunto de moléculas si y solo si dicho conjunto contiene un subconjunto de moléculas tal que su peso total pertenezca al rango de detección de la máquina.

Formalmente, sean n moléculas con pesos enteros positivos $w_0, \ldots, w_{n-1}$. La detección es exitosa si existe un conjunto de índices distintos $I = \{i_1, \ldots, i_m\}$ tal que $l \leq w_{i_1} + \ldots + w_{i_m} \leq u$.

Debido a las particularidades de la máquina, se sabe que la diferencia entre l y u siempre es mayor o igual que la diferencia de peso entre la molécula más pesada y la más liviana.

Formalmente, $u - l \geq w_{\max} - w_{\min}$, donde $w_{\max} = \max(w_0, \ldots, w_{n-1})$ y $w_{\min} = \min(w_0, \ldots, w_{n-1})$.

Su tarea es escribir un programa que, o bien encuentre algún subconjunto de moléculas con peso total en el rango de detección, o bien determine que no existe tal subconjunto.

Subtareas

1. (9 puntos): $1 \leq n \leq 100$, $1 \leq w_i \leq 100$, $1 \leq u, l \leq 1000$, todos los w_i son iguales.
2. (10 puntos): $1 \leq n \leq 100$, $1 \leq w_i, u, l \leq 1000$, y $\max(w_0, \ldots, w_{n-1}) - \min(w_0, \ldots, w_{n-1}) \leq 1$.
3. (12 puntos): $1 \leq n \leq 100$ y $1 \leq w_i, u, l \leq 1000$.
4. (15 puntos): $1 \leq n \leq 10000$ y $1 \leq w_i, u, l \leq 10000$.
5. (23 puntos): $1 \leq n \leq 10000$ y $1 \leq w_i, u, l \leq 500000$.
6. (31 puntos): $1 \leq n \leq 200000$ y $1 \leq w_i, u, l < 2^{31}$.

Solución

Como $n \leq 2 \cdot 10^5$, tenemos tiempo para ordenar los valores que nos dan. A partir de ahora consideramos que $w_0, \ldots, w_{n-1}$ está ordenada.

En primer lugar vamos a intentar analizar la condición que nos dan $u - l \geq w_{\max} - w_{\min}$. Esta condición es un poco extraña: eso nos hace pensar que la clave para resolver el problema esté aquí. (De hecho, el problema sin esta condición, es decir, encontrar un subconjunto de unos enteros dados que tenga una suma determinada, es un problema computacionalmente muy difícil). Esta condición dice que la longitud del intervalo de pesos aceptables es mayor que la diferencia de pesos entre la molécula más pesada y la más ligera; esto implica que la longitud del intervalo tiene que ser mayor que la diferencia de pesos entre dos moléculas cualquiera. Transladándolo al contexto de escoger subconjuntos de moléculas, esto significa que si cambiamos una molécula del subconjunto por otra, la suma cambiará en una cantidad menor a la longitud del intervalo. Esto significa que, por ejemplo, si empezamos con un subconjunto de moléculas

cuyo peso sea menor que l y cambiamos una molécula por otra más pesada, entonces la suma aumentará pero no llegaremos a pasarnos de u, es decir, o bien seguiremos por debajo de l o bien caeremos dentro del intervalo. Esto es interesante porque sugiere que la solución puede ser algo como empezar por un subconjunto e ir cambiando moléculas una por una.

Esto nos lleva a pensar qué valores posibles hay para m, el número de moléculas en el consistirá nuestro subconjunto. Este valor tiene que satisfacer que $w_0 + w_1 + \ldots + w_{m-1} \leq u$, ya que el subconjunto que cojamos siempre tendrá un peso mayor al peso de las m moléculas menos pesadas, y por tanto ese peso tendrá que ser $\leq u$ para no pasarnos. De forma semejante, $w_{n-m} + w_{n-m+1} + \ldots w_{n-1} \geq l$. Si no hay ningún valor de m que satisfazca estas dos condiciones, entonces no puede existir ningún subconjunto.

Y si hay un valor que satisfazca, entonces sí podremos encontrarlo con el razonamiento que hemos hecho antes. Empezamos por coger las m primeras moléculas, cuyo peso será menor que u. Si el peso es mayor que l, ya lo tenemos; si no, vamos cambiando la molécula menos pesada por la siguiente molécula. Al final llegaremos a tener las últimas m moléculas, que suman más de l; como hemos ido añadiendo moléculas una por una, en algún momento hemos tenido que caer en el intervalo $[l, u]$, ya que como hemos dicho antes en un sólo movimiento no podemos pasar de peso menor que l a peso mayor que u.

Código

C++ (1)

Aquí está la solución implementada tal como está explicada en la resolución del problema.

```cpp
#include "molecules.h"
#include<algorithm>
#include<vector>

using namespace std;

typedef long long ll;
typedef pair<int, int> ii;
typedef vector<ii> vii;

int find_subset(int l, int u, int* w, int n, int* result) {

        //Almacenamos pares {valor, indice} y luego ordenamos el vector
        //Este truco nos permite recuperar los indices originales
        //del vector w para cuando necesitemos devolver el subconjunto
        vii v(n);
        for(int i=0; i < n; ++i) {
                v[i] = {w[i], i};
        }
        sort(v.begin(), v.end());

        //Calculamos el valor de m
        int m = 0;
        ll suma_min = 0;
        ll suma_max = 0;
        while(m < n && suma_max < l) {
                //vamos acumulando en suma_min la suma de las m moleculas
                //mas ligeras y en suma_max la de las m mas pesadas
                //paramos cuando la suma_max >= l (o ya hayamos cogido todas)
                suma_min += v[m].first;
                suma_max += v[n-1-m].first;
```

```
                m++;
        }

        //Si no se satisfacen las condiciones, retornamos que no hay subconjunto
        if(suma_max < l || suma_min > u) {
                return 0;
        }

        //Si se satisfacen, sabemos que se podra. Buscamos el indice idx
        //donde la suma de las m siguientes moleculas pasa a ser >= l
        ll suma = suma_min;
        int idx = -1;
        if(suma >= l) {
                idx = 0;
        }
        for(int i=0; idx == -1 && i+m < n; ++i) {
                suma -= v[i].first;
                suma += v[i+m].first;
                if(suma >= l) {
                        idx = i+1;
                }
        }

        //Ponemos el resultado, recuperando los indices originales
        //que estaban en las segundas posiciones de los pares
        for(int i=0; i < m; ++i) {
                result[i] = v[i+idx].second;
        }

    return m;
}
```

C++ (2)

Aquí hay una implementación ligeramente diferente y más compacta. Se puede llegar a ella modificando un poco el razonamiento anterior.

```
#include "molecules.h"
#include<algorithm>
#include<vector>

using namespace std;


typedef pair<int, int> ii;
typedef vector<ii> vii;

int find_subset(int l, int u, int* w, int n, int* result) {

        vii v(n);
        for(int i=0; i < n; ++i) {
                v[i] = {w[i], i};
        }
        sort(v.begin(), v.end());

```

```
        int fi = 0;
        long long cs = 0;
        for(int i=0; i < n; ++i) {
                cs += v[i].first;
                while(cs > u) {
                        cs -= v[fi].first;
                        fi++;
                }
                if(cs >= l) {
                        for(int j=fi; j <= i; ++j) {
                           result[j-fi] = (v[j].second);
                        }
                        return i-fi+1;
                }

        }

    return 0;
}
```