9'5

Problema 43

GÓMEZ LÓPEZ, CARLOS (TAIS45)

ID envio	Usuario/a	Hora envío	Veredicto
62831	TAIS45	2022-11-16 10:36	AC
62772	TAIS45	2022-11-16 09:51	AC
62767	TAIS45	2022-11-16 09:47	AC
62764	TAIS45	2022-11-16 09:41	WA

Fichero Source.cpp

```
*
* Indicad el nombre completo y usuario del juez de quienes habéis hecho esta solución:
* Estudiante 1: Carlos Gómez López TAIS45
* Estudiante 2: Clara Sotillo Sotillo TAIS96
*

  El coste en tiempo de la función en el peor de los casos es O(N*P), siendo N el número de
   festivales distintos que ha
  seleccionado Sergio y P el presupuesto del que dispones Sergio.
  El coste en espacio en el peor de los casos es O(P), siendo P el presupuesto del que
  dispones Sergio.

  Primero lo que hemos hecho ha sido pensar en la recurrencia del ejercicio, obteniendo lo
  siguiente:

  conciertos(i,j) = es el número de conciertos de i festivales que podemos ver con el
  presupuesto j

                      { conciertos(i,0) = 0
  conciertos(i,j) =   { conciertos(0,j) = 0
                      { conciertos(i-1,j)
     si precio[i] < j
                      { max(conciertos(i-1,j), conciertos(i,j-precio[i]) + grupos[i])
     si precio[i] >= j

  Tenemos dos casos bases:

   - El primero es el caso en el que tiene festivales seleccionados, pero no tiene
  presupuesto para ver ninguno, en ese caso
     no necesitamos ver conciertos:

     conciertos(i,0) = 0

   - El segundo es el caso en el que tiene presupuesto para ver festivales, pero no tiene
  ningún festival seleccionado,
     entonces no necesitariamos ver conciertos:

     conciertos(0,j) = 0

  Dentro de los casos recursivos podemos distinguir otros dos:
```

- Si el precio de dicho festival es menor que el dinero del que dispone, entonces solo podremos buscar
  soluciones que no utilicen el festival i, y lo hagan mejor posible con el resto de festivales, del 1 al i-1:

  conciertos(i,j) = conciertos(i-1,j) , si precio[i] < j

- Si el precio de dicho festival es mayor o igual que el dinero del que dispone, probamos dos opciones:
   Intentar ver el máximo número de festivales con el presupuesto j, sin ver el festival i

ó

   Usarlo e intentar conseguir el resto (j - precio[i]) con los mismos festivales (del 1 al i), habiendo sumado el número
   de grupos que participan en el festival i (grupos[i]).

   Nos quedamos con la menjor solución, la mayor:

   conciertos(i,j) = max(conciertos(i-1,j), conciertos(i-1,j-precio[i]) + grupos[i]) si precio[i] >= j

Implementamos la función de manera ascendente rellenando la tabla de tamaño (N+1)(P+1), de arriba a abajo y de izquierda a derecha.
Es ascendente ya que recorres de los que tienen menor presupuesto a los que más tienen.

Una vez rellena podemos reconstruir la solución retrocediendo desde la última casilla.

Explicación de ejercicio y porqué el coste mencionado anteriormente:

Primero creamos un struct llamado tFestival el cual va a contener tanto el precio de este como la cantidad de grupos que lo conforman.
Estos festivales los guardamos en un vector el cual correspondría a lo que en la recurrencia hemos puesto como precio[i] y grupos[i].
Como inicio hemos hecho una matriz (la cual se muestra comentada), con dicha matriz de tamaño (N+1)*(P+1), hemos aplicado
la recurrencia anterior, para obtener la solución, el problema de utilizar una matriz en vez de un vector, significa que aunque
el coste en tiempo sea O(N*P), en espacio si utilizas una matriz es O(N*P) también mientras que si utilizasemos un vector
el coste en espacio sería O(P), por lo que es mejor.

Por este motivo anterior, hemos pensado en otra solución que nos proporciona mejor coste, y es utilizando un vector.
Utilizando la misma recurrencia anterior, se obtiene la solución, recorriendo las columnas de derecha a izquierda,
ya que solo necesitas los subproblemas de las filas anteriores.

```
typedef struct {
    int grupos;
    int precio;
}tFestival;


void gruposAVisitar(vector<tFestival> festivales, int N, int P) {

    /* MATRIZ */

   /* Matriz<int> visitar(N + 1, P + 1, 0);

    for (int i = 1; i <= N; i++) {
        for (int j = 1; j <= P; j++) {
            if (j >= festivales[i - 1].precio) visitar[i][j] = max(visitar[i - 1][j], visitar
    [i - 1][j - festivales[i - 1].precio] + festivales[i - 1].grupos);
            else visitar[i][j] = visitar[i - 1][j];
        }
    }

    cout << visitar[N][P] << "\n";*/

    /* VECTOR */

    vector<int> visitar(P + 1, 0);

    for (int i = 1; i <= N; i++) {
        for (int j = P; j >= festivales[i - 1].precio; j--) {
            visitar[j] = max(visitar[j], visitar[j - festivales[i - 1].precio] + festivales[i
     - 1].grupos);
        }
    }

    cout << visitar[P] << "\n";

}

bool resuelveCaso() {

    int P, N;
    cin >> P >> N;  // presupuesto y número de festivales

    if (!cin) return false;

    vector<tFestival> festivales(N);
    for (int i = 0; i < N; i++) cin >> festivales[i].grupos >> festivales[i].precio;

    gruposAVisitar(festivales, N, P);

    return true;
}
```