Compra Sin Vergüenza

Examen Mundial de Programación - Curso 2024

Día N+1

Advertencia

Si usted está leyendo este documento si haberlo descomprimido, ciérrelo inmediatamente y descomprima el contenido del archivo zip que recibió en el escritorio. Una vez hecho esto, reanude su trabajo desde allí. Trabajar antes de descomprimir le hará perder sus cambios, y no tendrá oportunidad de reclamar.

En la tienda online *Compra Sin Vergüenza* hay N productos diferentes, cada uno con un precio individual. Usted quiere adquirir cierta cantidad de algunos de estos productos, por ejemplo, 3 paquetes de perritos, 2 bolsas de leche en polvo, 1 kg de pollo (sin pechuga), y 35 paquetes de baterías AAA recargables para la linterna (por razones obvias).

Para beneplácito de los clientes, a los comerciales de *Compra Sin Vergüenza* se les ha ocurrido crear combos: paquetes de dos o más productos que tienen un precio distinto (a menudo menor pero no necesariamente) que la suma de todos los productos incluídos. Teniendo esto en cuenta, usted pudiera obtener los productos que necesita ya sea comprándolos directamente, o comprando combos que los contengan, o una combinación de ambos. En general, a usted no le importa tener un producto extra que no necesite si con eso logra disminuir el costo total de adquirir lo que sí necesita (motivo por el cuál tiene cientos de frazadas de piso en el closet).

El problema a resolver consiste entonces en optimizar el costo de la compra de forma que se garantice que al menos se obtiene una cantidad igual o mayor de cada producto que usted necesita.

El método a implementar tiene la siguiente signatura:

```
static class Exam
{
    static int Solve(
        IProduct[] products,
        ICombo[] combos,
        IProductQuantity[] desired)
    {
        throw new NotImplementedException();
        // tire su backtrack de toda la vida aquí
    }
}
```

Este método recibe tres argumentos:

- `products`: un arreglo de `IProduct` que contiene la información de cada producto disponible. Cada `IProduct` tiene dos propiedades: `Price` y `Name`.
- `combos`: un arreglo de `ICombo` que contiene la información de cada combo disponible. Cada `ICombo` tiene dos propiedades: `Price` y `Products`. La propiedad `Products` es un arreglo de `IProductQuantity` que contiene la información de cada producto incluido en el combo, junto con la cantidad de unidades de ese producto que se incluyen en el combo.
- `desired`: un arreglo de `IProductQuantity` que contiene la información de cada producto que usted necesita, junto con la cantidad de unidades de ese producto que necesita. Aquí solo aparecen los productos que usted necesita (con cantidad mayor que cero).

Las clases mencionadas tienen la siguiente definición:

```
public class IProduct
{
    public int Price { get; }
    public string Name { get; }
}

public class IProductQuantity
{
    public IProduct Product { get; }
    public int Quantity { get; }
}

public class ICombo
{
    public int Price { get; }
    public IProductQuantity[] Products { get; }
}
```

El método `Solve` debe retornar el costo mínimo de la compra que garantice que al menos se obtiene una cantidad igual o mayor de cada producto que usted necesita. Puede combinar tantas cantidades de cualquier combo como de productos individuales, por tanto la compra siempre es posible.

Ejemplo

Supongamos que la tienda tiene los siguientes productos:

Producto	Precio
Perritos	1000
Leche	2000
Pollo	3000
Baterías	400
Frazadas	800

Además tenemos los combos:

- 2 Perrito + 2 Leche + 1 Frazada por 5000 pesos.
- 1 Pollo + 1 Leche por 4000 pesos.
- 10 Baterias por 3000 pesos.

Y usted desea comprar 3 Perritos, 2 Leches, 1 Pollo y 35 Baterías.

En este caso, el costo de adquirir cada producto por separado sería:

- 3 Perritos: 3 * 1000 = 3000 pesos.
- 2 Leches: 2 * 2000 = 4000 pesos.
- 1 Pollo: 1 * 3000 = 3000 pesos.
- 35 Baterías: 35 * 400 = 14000 pesos.
- Total: 3000 + 4000 + 3000 + 14000 = 24000 pesos.

Sin embargo, usted puede hacer la siguiente compra:

- 1 Combo de 2 Perritos + 2 Leches + 1 Frazada: 5000 pesos.
- 4 Combo de 10 Baterías: 12000 pesos.
- 1 Pollo: 3000 pesos.
- 1 Perrito: 1000 pesos.
- Total: 5000 + 12000 + 3000 + 1000 = 21000 pesos.

Que es un costo menor que comprar cada producto por separado, y cubre todas las necesidades aunque tiene 5 paquetes de baterías adicionales (que tampoco nunca vienen mal, eh?) y otra frazada para la colección (seguro algo se le ocurre para el próximo Halloween).

Este ejemplo puede ser expresado en el siguiente código de C# (asumiendo que tiene las clases que implementan las interfaces).

```
IProduct perritos = new Product { Name = "Perritos", Price = 1000 };
IProduct leche = new Product { Name = "Leche", Price = 2000 };
IProduct pollo = new Product { Name = "Pollo", Price = 3000 };
IProduct baterias = new Product { Name = "Baterías", Price = 400 };
IProduct frazadas = new Product { Name = "Frazadas", Price = 800 };

ICombo combo1 = new Combo
{
    Price = 5000,
    Products = new[]
    {
        new ProductQuantity { Product = perritos, Quantity = 2 },
        new ProductQuantity { Product = leche, Quantity = 2 },
        new ProductQuantity { Product = frazadas, Quantity = 1 }
    }
};

ICombo combo2 = new Combo
{
    Price = 4000,
```

```
    Products = new[]
    {
        new ProductQuantity { Product = pollo, Quantity = 1 },
        new ProductQuantity { Product = leche, Quantity = 1 }
    }
};

ICombo combo3 = new Combo
{
    Price = 3000,
    Products = new[]
    {
        new ProductQuantity { Product = baterias, Quantity = 10 }
    }
};


IProduct[] products = { perritos, leche, pollo, baterias, frazadas };
ICombo[] combos = { combo1, combo2, combo3 };
IProductQuantity[] desired = {
    new ProductQuantity { Product = perritos, Quantity = 3 },
    new ProductQuantity { Product = leche, Quantity = 2 },
    new ProductQuantity { Product = pollo, Quantity = 1 },
    new ProductQuantity { Product = baterias, Quantity = 35 },
};

int result = Exam.Solve(products, combos, desired);

Debug.Assert(result <= 21000);
```

Notas

- No se puede comprar una fracción de un producto o combo.
- Asuma que hay una infinita cantidad de cada producto y combo.
- La relación de igualdad es **por referencia** entre instancias de `IProduct`.

> **!** Advertencia
>
> Este problema es una variante de un conocido problema NP-duro, por lo que no se conocen y probablemente no existan soluciones eficientes para todos los casos. Concéntrese en implementar una búsqueda exhaustiva que funcione correctamente.