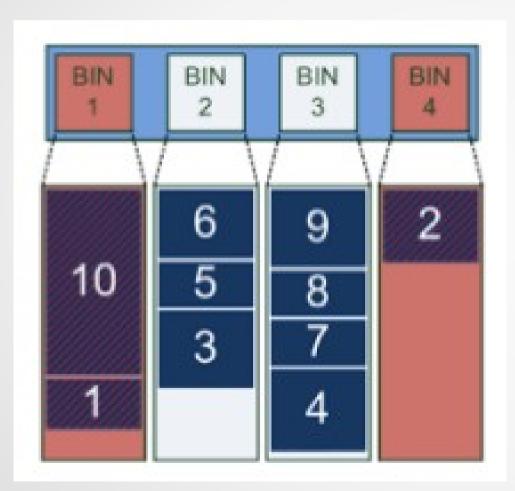
Bin Packing

Pablo Álvarez Cabreba Johanna Capote Robayna Cristina de la Torre Villaverde Guillermo Galindo Ortuño Adolfo Soto Werner

El problema



Datos:

- -Número de objetos a embalar
- -Tamaño de cada objeto
- -Volumen de cada embalaje

¿Cómo insertamos los objetos para minimizar el número de embalajes necesarios?

Diseño de la solución

```
S=Conjunto de contenedores de tamaño V.
C=Conjunto de candidatos
S=1
Mientras C != 0 hacer:
 x=mayor elemento de C
 C=C\setminus\{x\}
 Si {x} cabe en Si
   Si=Si U {x}
 Si no
    S aumenta su tamaño en 1
   Sn=\{x\}
Fin-mientras
                                n=índice del último vector de S
Devuelve S
                                i=índice del vector de S
```

Componentes Greedy

- Lista de candidatos: Los N objetos a empaquetar.
- Lista de candidatos utilizados: Los elementos que ya se han empaquetado.
- Función solución: Se han empaquetado los N objetos.
- Función de selección: Se selecciona el objeto de mayor volumen.
- Criterio de factibilidad: El volumen de los objetos contenidos en una caja es menor o igual que el volumen de la caja.
- Función objetivo: Minimizar el número de cajas necesarias.

Volumen de cada embalaje = 7 Número de objetos a insertar = 10 Vector de volúmenes de los objetos = {1,2,2,2,1,2,1,1,2,5}

1 2 2 1 2 1 1 2 5

1 2 2 1 2 1 1 2 5

S1

X

X

X

X

X

1 2 2 1 2 1 1 2 5

S1

X

X

X

X

X

X

X

1 2 2 1 2 1 1 2 5

XXXXXXXXX

1 2 2 1 2 1 1 2 5

 S1
 S2

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

1 2 2 1 2 1 1 2 5

 S1
 S2

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

1 2 2 1 2 1 1 2 5

S2 **S**3 **S**1 X X X X X X X X X X X X X

1 2 2 1 2 1 1 2 5

S3 **S**1 X X X X X X X X X X X X X

X

1 2 2 1 2 1 1 2 5

S3 **S**1 X X X X X X X X X X X X X X X

1 2 2 1 2 1 1 2 5

S3 **S**1 X X X X X X X X X X X X X X X

1 2 2 1 2 1 1 2 5

S3 **S**1 X X X X X X X X X X X X X X X X

Ejemplo: Solución final

1 2 2 1 2 1 1 2 5

 S1
 S2

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

 X
 X

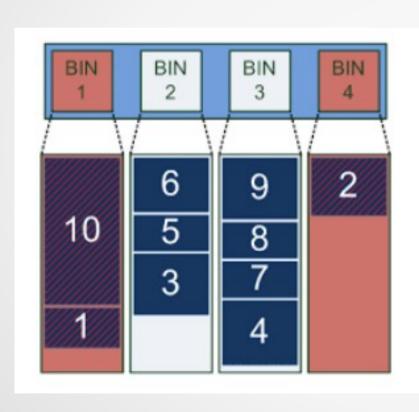
 X
 X

 X
 X

 X
 X

X X X X

Ejemplo: otra posible solución



S1	S2	S3	S4
		X	
X		X	
X	X	X	
X	X	X	
X	X	X	
X	X	X	X
X	X	X	X

Comparativa de soluciones

S1	S2	S3
X	X	
X	X	
X	X	X
X	X	X
X	X	X
X	X	X
X	X	X

Solución con el algoritmo implementado:

- -3 embalajes
- -2 espacios sin utilizar

Solución sin el algoritmo implementado:

- -4 embalajes
- -8 espacios sin utilizar

S1	S2	S3	S4
		X	
X		X	
X	X	X	
X	X	X	
X	X	X	
X	X	X	X
X	X	X	X

Eficiencia

```
8 Solucion AlgorimoGreedyBinPacking(Problema p){
    Solucion S:
    vector<pair<int,double> > candidatos(p.getNumObjetos());
10
11
    //Inicializamos lista de candidatos
12
    for (int i = 0; i < p.getNumObjetos(); i++) {</pre>
13
      candidatos[i].first = i;
14
      candidatos[i].second = p.getVolumen(i);
15
16
   // Añadimos la primera caja
17
    S.addCaja();
18
```

Eficiencia: O(n²)

```
20
    //Mientras queden objetos sin incluir en cajas
    while (!candidatos.empty()) {
21
22
      //Funcion de seleccion (elemento con el máximo volumen)
23
      vector<pair<int. double> >::iterator it, sol;
24
      pair<int, double> max(-1,0);
25
      for (it = candidatos.begin(); it != candidatos.end(); it++) {
26
27
        if (max.second < it->second) {
28
          max = *it:
29
          sol = it;
30
        }
31
32
33
      //Eliminamos el objeto elegido de la lista de candidatos
      candidatos.erase(sol);
34
35
36
       //Incluimos el objeto elegido en la solucion
37
       bool add = false:
      for (int i = 0; i < S.getNumCajas() && !add; i++) {</pre>
38
39
        //Si el objeto cabe en una caja existente, lo incluimos y salimos del bucle
        if (S.getVolumen(i) + max.second <= p.getVolumen()){</pre>
40
41
           S.addObjeto(i,max);
           add = true:
42
43
44
      //Si no, creamos una caja nueva y añadimos el objeto
45
      if(!add){
46
        S.addCaja();
47
        S.addObjeto(S.getNumCajas()-1, max);
48
49
50
```

O(n²)