.UBAfiuba
FACULTAD DE INGENIERÍA

Programación dinámica: Subset Sums y Knapsacks

Teoría de Algoritmos I (75.29 / 95.06)

Ing. Víctor Daniel Podberezski
vpodberezski@fi.uba.ar

Subset Sums

Sea

Un conjunto de “n” elementos $E=\{e_1,e_2,\ldots,e_n\}$

donde cada elemento e_i cuanta con un peso asociado w_i

Queremos

Seleccionar un subset de elementos de E con el mayor peso posible

que no supere un valor W de peso máximo

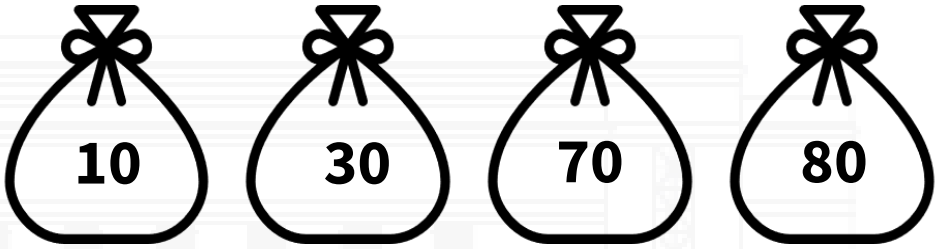
¿Existe Solución greedy óptima?

Criterios de selección

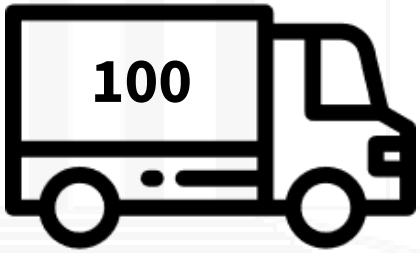
Primero más pequeños

Primero más grandes

...



No existe solución greedy óptima



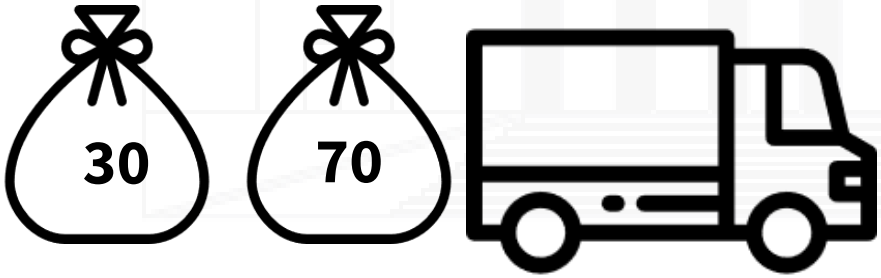
Solución por fuerza bruta

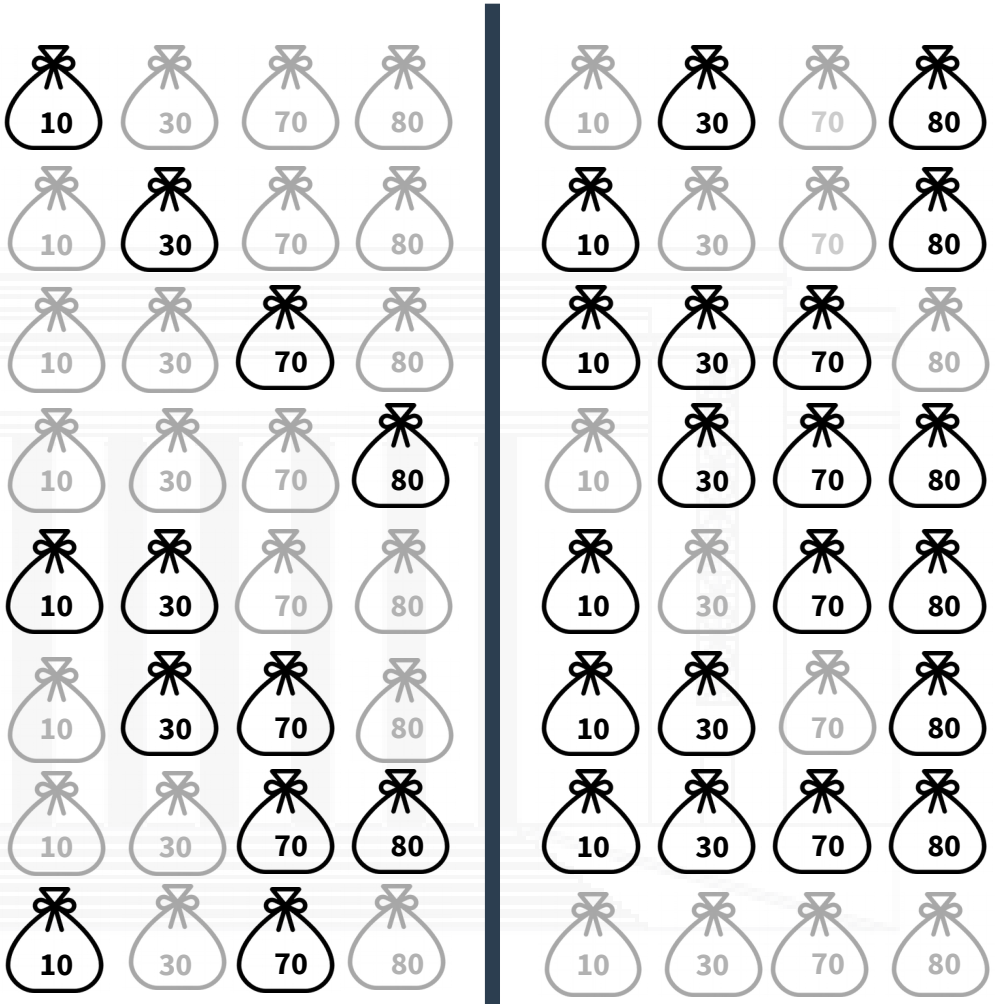
En una solución óptima

Un elemento e_i puede estar o no

Si tengo n elementos

Pueden existir 2^n combinaciones





Buscando mejor solución

Podremos usar programación dinámica?

Existe forma de vincular la selección de un elemento i con los elementos i-1 anteriores?

Es fácil ver que

Si $e_i \notin$ solución → MAX_PESO(e_i) = MAX_PESO(e_i-1)

Pero …

Si $e_i \in$ solución → MAX_PESO(e_i) = w_i + MAX_PESO(???)

Nos falta algo… una variable...

Una cuestión de peso...

Si $e_i \in$ solución

Consume w_i en del W peso disponible

Podemos plantear:

Si $e_i \notin$ solución → MAX_PESO(e_i,W) = MAX_PESO(e_i-1, W)

Si $e_i \in$ solución → MAX_PESO(e_i,W) = w_i + MAX_PESO(e_i-1, W-w_i)

La mejor solución hasta e_i

Máximo {$e_i \notin$ solución , $e_i \in$ solución }

Subproblemas y recurrencia

Llamaremos

MAX_PESO(i,p)

al problema de determinar el peso máxima que no supere p, utilizando los primeros i elementos del conjunto.

Queremos obtener

MAX_PESO(n,W)

Recurrencia

$$\begin{cases} \text{MAX\_PESO}(i,p)=0 & \text{, si } i=0 \text{ o } p=0 \\ \text{MAX\_PESO}(i,p)=max\{\begin{matrix} w_i+\text{MAX\_PESO}(i-1,p-w_i), \\ \text{MAX\_PESO}(i-1,p) \end{matrix}\} & \text{, si } i>0 \text{ y } p>0 \end{cases}$$

Solución iterativa

Complejidad

Temporal: n*W

Espacial: n*W

Es un algoritmo pseudo polinomial

Si el peso es muy grande nos obliga a realizar muchos cálculos

```
Desde i=0 a n
     OPT[i][0] = 0

Desde p=0 a W
     OPT[0][p] = 0

Desde i=1 a n   // elementos
     Desde p=1 a W  // pesos

          enOptimo = w[i] + OPT[i-1,p-w[i]]
          noEnOptimo = OPT[i-1,p]

          si enOptimo > noEnOptimo
               OPT[l][p] = enOptimo
          sino
               OPT[l][p] = noEnOptimo

Retornar OPT[n,W]
```

Knapsacks

Sea

Un conjunto de “n” elementos $E=\{e_1,e_2,\ldots,e_n\}$

donde cada elemento e_i cuanta con un peso asociado w_i

y una ganancia de v_i

Queremos

Seleccionar un subset de elementos de E con la mayor ganancia posible

que no supere un valor W de peso máximo

Análisis y Subproblema

Es una variante de Subset Sum

Podemos aprovechar el análisis anterior?

Cada elemento e_i que está en la solución

Consume w_i de espacio

Suma v_i de ganancia

Llamaremos

MAX_GANANCIA(i,p)

al problema de determinar la ganancia máxima que no supere p, utilizando los primeros i elementos del conjunto.

Recurrencia

Podemos plantear:

Si $e_i \notin$ solución → MAX_GANANCIA(e_i,W) = MAX_GANANCIA(e_i-1, W)

Si $e_i \in$ solución → MAX_GANANCIA(e_i,W) = v_i + MAX_GANANCIA(e_i-1, W-w_i)

Queremos obtener

MAX_GANANCIA(n,W)

Recurrencia

$$\text{MAX\_GANANCIA}(i,p) = 0 \quad \text{, si } i=0 \text{ o } p=0$$

$$\text{MAX\_GANANCIA}(i,p) = max\{v_i + \text{MAX\_GANANCIA}(i-1, p-w_i), \text{MAX\_GANANCIA}(i-1,p)\} \text{, si } i>0 \text{ y } p>0$$

Solución iterativa

Complejidad

Temporal: n*W

Espacial: n*W

Es un algoritmo pseudo polinomial

Si el peso es muy grande nos obliga a realizar muchos cálculos

```
Desde i=0 a n
     OPT[i][0] = 0

Desde p=0 a W
     OPT[0][p] = 0

Desde i=1 a n   // elementos
     Desde p=1 a W  // pesos

          enOptimo = v[i] + OPT[i-1,p-w[i]]
          noEnOptimo = OPT[i-1,p]

          si enOptimo > noEnOptimo
               OPT[l][p] = enOptimo
          sino
               OPT[l][p] = noEnOptimo

Retornar OPT[n,W]
```

Consideraciones

Tanto para Subset Sum como en knaspsak

Si solo se desea calcular el máximo se puede reducir la complejidad espacial (para el subproblema “i” solo se utiliza los resultados de “i-i”)

complejidad espacial: O(W)

Si se requiere reconstruir la selección realizada, se puede agregar un indicador binario (si / no) para el subproblema “i”,”p” sobre conviene elegir o no el elemento en la solución.

Luego se puede desde el subproblema n,W reconstruir para atrás las selecciones.

.UBAfiuba

FACULTAD DE INGENIERÍA

Presentación realizada en Abril de 2020