

# **Preparación Específica para Concursos ACM-ICPC.**

---

## **Tema #5 : Programación Dinámica.**

**Conferencia #10: Programación dinámica II. Algoritmos mochila (0-1), mochila (0-1) con multiples elementos y máxima subsecuencia incremental (LIS)**

# Objetivos

---

- ❑ Caracterizar los algoritmos mochila (0,1), mochila (0,1) con múltiples elementos, máxima subsecuencia incremental (LIS).

# Objetivos

---

- ❑ Identificar los problemas que se pueden solucionar aplicando los algoritmos mochila (0,1), mochila (0,1) con múltiples elementos, máxima subsecuencia incremental (LIS).

# Contenidos

---

- ❑ Algoritmo mochila (0,1)
- ❑ Algoritmo mochila (0,1) con múltiples elementos.
- ❑ Algoritmo máxima subsecuencia incremental (LIS).

# Bibliografía

---

- *Manual de preparación para concursantes ACM-ICPC de la Universidad de Matanzas.*

# Algoritmo mochila (0,1).

---

- Dados  $n$  elementos  $e_1, e_2, \dots, e_n$  con pesos  $p_1, p_2, \dots, p_n$  y beneficios  $b_1, b_2, \dots, b_n$ , y dada una mochila capaz de albergar hasta un máximo de peso  $M$  (capacidad de la mochila).

# Algoritmo mochila (0,1).

---

- Queremos encontrar las proporciones de los  $n$  elementos  $x_1, x_2, \dots, x_n$  ( $0 \leq x_i \leq 1$ ) que tenemos que introducir en la mochila de forma que la suma de los beneficios de los elementos escogidos sea máxima.

# Algoritmo mochila (0,1).

---

$$V(i, p) = \begin{cases} 0 & \text{si } i = 0 \text{ y } p \geq 0 \\ -\infty & \text{si } p < 0 \\ \text{Max}\{V(i-1, p), V(i-1, p - p_i) + b_i\} & \text{en otro caso.} \end{cases}$$



# Algoritmo mochila (0,1).

---

- La complejidad del algoritmo viene determinada por la construcción de una tabla de dimensiones  $n \times M$  y por tanto su tiempo de ejecución es de orden de complejidad  $O(nM)$ .

# Algoritmo mochila (0,1) multiples elementos.

---

- Este problema se basa en el de la Mochila (0,1) pero en vez de existir  $n$  objetos distintos, de lo que disponemos es de  $n$  tipos de objetos distintos. Con esto, de un objeto cualquiera podemos escoger tantas unidades como deseemos.

# Algoritmo mochila (0,1) multiples elementos.

---

- Este problema se puede formular también como una modificación al problema de la Mochila (0,1), en donde sustituimos el requerimiento de que  $x_i=0$  ó  $x_i=1$ , por el que  $x_i$  sean números naturales.

# Algoritmo mochila (0,1) multiples elementos.

---

$$V(i, p) = \begin{cases} 0 & \text{si } p = 0 \\ (p \div p_i) b_i & \text{si } i = 1 \\ V(i-1, p) & \text{si } p < p_i \\ \text{Max}\{V(i-1, p), V(i, p - p_i) + b_i\} & \text{en otro caso.} \end{cases}$$

# Algoritmo mochila (0,1) multiples elementos.

---

- Utilizaremos un arreglo lineal con una capacidad de  $n \times M$  para almacenar los valores de  $V$  que vayamos obteniendo y así no repetir cálculos.

# Algoritmo mochila (0,1) multiples elementos.

---

- La complejidad del algoritmo es la que corresponde a la construcción del arreglo, es decir  $O(n \cdot M)$ .

# Algoritmo máxima subsecuencia incremental (LIS).

---

- El problema de la máxima subsecuencia incremental conocido también como LIS por su nombre en ingles *longest increasing subsequences*.

# Algoritmo máxima subsecuencia incremental (LIS).

---

- Plantea hallar dentro de una colección o secuencias de elementos de la forma  $a_1, a_2, a_3, \dots, a_n$  cualquier subconjunto o subsecuencia de los elementos dados en orden, de la forma  $a_{i_1}, a_{i_2}, a_{i_3}, \dots, a_{i_k}$  donde  $1 < i_1 < i_2 < i_3 < \dots < i_k < n$  donde cada elemento es estrictamente mayor que su anterior y estrictamente menor que su sucesor.



# Algoritmo máxima subsecuencia incremental (LIS).

---



# Algoritmo máxima subsecuencia incremental (LIS).

---

- Vamos a representar la colección de elementos como nodos de un grafo donde entre los elementos  $a_i$  y  $a_j$  a de existir una arista siempre y cuando se cumpla las siguientes restricciones  $a_i < a_j$  y  $i < j$ .

# Algoritmo máxima subsecuencia incremental (LIS).

---

- Si se analiza el grafo que se construye es un grafo dirigido acíclico (DAG). Por lo que la solución del problema inicial radica en encontrar el camino mas largo el grafo construido.

# Algoritmo máxima subsecuencia incremental (LIS).

---

for  $j=1,2,\dots,n$ :

$L(j)=1+\max(L(i):(i,j) \text{ pertenece } E)$

return  $\max_j L(j)$

# Algoritmo máxima subsecuencia incremental (LIS).

---

- Donde  $L(j)$  es la longitud del camino más largo que finaliza en el nodo  $j$ . Este caso se le suma uno porque la distancia del camino no la define la cantidad de aristas sino la cantidad de nodos que lo componen.

# Conclusiones.

---

- ❑ Los algoritmos de las Mochilas en sus variantes nos permiten obtener el mejor beneficio sin sobrepasar un límite de costo.
- ❑ Para ambas variantes del algoritmo de la Mochila la complejidad es la misma.

# Conclusiones.

---

- La implementación de LIS a utilizar en la solución de un problema radica en la cantidad de elementos inicial de la colección.

# Estudio Independiente

---

- Profundizar en los temas abordados con la lectura del capítulo Programación Dinámica del manual mencionado en la bibliografía del curso.



# Estudio Independiente

---

- ❑ Solucionar de Juez Caribeño Online COJ los siguientes problemas.
  - ❑ 1658 - Longest Increasing Subsequence (LIS)
  - ❑ 2487 - Charm Bracelet

# **Preparación Específica para Concursos ACM-ICPC.**

---

## **Tema #5 : Programación Dinámica.**

**Conferencia #10: Programación dinámica II. Algoritmos mochila (0-1), mochila (0-1) con multiples elementos y máxima subsecuencia incremental (LIS)**