

Actividad 09  
Análisis y Diseño de Algoritmos  
Dr. Carlos Villaseñor  
Alumno: Eduardo Alberto Rodríguez García

---

1. Usa programación dinámica para resolver el problema de encontrar el máximo conjunto independiente con mayor peso. Usa el archivo MWIS.txt para obtener los datos, cada dato es el peso de un vértice en un grafo camino. Contesta lo siguiente: ¿Cuáles de los vértices 1, 2, 3, 4, 17, 117, 517, y 997 forman parte de la solución óptima?

```
def mwis(w):
    # Obtiene el valor optimo
    n = len(w)
    a = [0, w[0]]
    for i in range(2, n+1):
        a.append(max(
            a[i-1],
            a[i-2] + w[i-1]
        ))

    # Reconstruir la solucion
    i = n
    s = set()
    while i >= 1:
        if a[i] == a[i-1]:
            i -= 1
        else:
            s.add(i-1)
            i -= 2
    print(s)

w = []

with open("MWIS.txt", "r") as f:
    for line in f:
        line = line.strip()
        w.append(int(line))

mwis(w)
```

```
C:\Windows\System32\cmd.exe
6846, 2702392055, 2702392055, 2706280150, 2706280150, 2709490637, 2709789458, 2715175865, 2715175865, 2717143538, 271714
3538, 2720010273, 2727004053, 2727004053, 2730881236, 2733106401, 2733679513, 2738805939, 2738805939, 2740434267, 274383
8940, 2744480685, 2749020155, 2752592338, 2756505969, 2758077967, 2762523863, 2762523863, 2767247175, 2767803626, 277179
5294, 2771795294, 2774367135, 2774367135, 2784301511, 2784301511, 2786985342, 2787056641, 2792457505, 2792457505, 279909
7291, 2800762321, 2803834133, 2803834133, 2811864043, 2811864043, 2818005331, 2818005331, 2819263571, 2819263571, 282541
3686, 2825413686, 2833197139, 2834760264, 2842782793, 2842782793, 2845270590, 2851447647, 2851447647, 2860196624, 286019
6624, 2862470404, 2868288827, 2870891182, 2876271006, 2879388150, 2879468935, 2885818067, 2885818067, 2891121733, 289112
1733, 2899642620, 2899642620, 2903930589, 2909272641, 2913921317, 2917728124, 2919354633, 2922501159, 2927787609, 292778
7609, 2930219257, 2936024870, 2939848077, 2939848077, 2947394128, 2948442421, 2950698717, 2955353732]
{0, 2, 4, 7, 9, 12, 14, 17, 19, 21, 23, 25, 27, 30, 32, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65,
68, 71, 74, 76, 78, 80, 82, 84, 87, 89, 91, 93, 95, 97, 99, 102, 105, 107, 109, 111, 114, 116, 119, 121, 123, 125, 127,
130, 132, 135, 138, 140, 142, 144, 146, 148, 150, 152, 154, 156, 159, 161, 163, 165, 167, 169, 172, 174, 176, 178, 180,
182, 184, 186, 189, 192, 194, 196, 198, 200, 202, 204, 206, 208, 210, 213, 215, 217, 220, 222, 225, 227, 229, 231, 233,
235, 237, 239, 242, 244, 246, 248, 251, 253, 255, 257, 260, 262, 264, 266, 268, 270, 272, 274, 276, 278, 280, 282, 284,
286, 288, 291, 293, 295, 297, 299, 301, 303, 305, 307, 309, 311, 313, 315, 317, 320, 322, 324, 326, 328, 330, 332, 334,
336, 338, 340, 342, 344, 346, 348, 350, 352, 354, 356, 358, 360, 362, 364, 366, 368, 370, 372, 374, 376, 378, 380, 382,
384, 386, 388, 390, 392, 394, 396, 398, 401, 403, 405, 407, 409, 412, 414, 416, 419, 421, 424, 426, 428, 430, 432, 434,
436, 438, 440, 442, 444, 446, 448, 450, 453, 455, 457, 459, 461, 463, 465, 467, 469, 471, 473, 475, 477, 480, 482, 484,
487, 489, 491, 493, 495, 497, 499, 502, 504, 507, 509, 511, 513, 516, 518, 520, 522, 524, 526, 528, 530, 533, 536, 538,
540, 542, 544, 547, 549, 551, 553, 555, 557, 559, 562, 565, 568, 570, 572, 574, 576, 578, 580, 583, 585, 587, 589, 591,
593, 595, 597, 599, 601, 603, 605, 607, 609, 612, 614, 616, 618, 620, 622, 624, 627, 629, 631, 633, 636, 638, 640, 642,
644, 646, 648, 651, 653, 655, 657, 659, 661, 663, 665, 668, 670, 672, 675, 678, 681, 683, 685, 687, 689, 691, 694, 696,
698, 701, 703, 705, 707, 709, 712, 714, 716, 718, 720, 722, 725, 727, 729, 732, 734, 736, 738, 740, 742, 744, 747, 749,
751, 754, 756, 759, 762, 765, 767, 769, 771, 773, 775, 777, 780, 783, 785, 787, 789, 792, 794, 796, 798, 800, 802, 805,
808, 811, 813, 815, 818, 820, 822, 824, 826, 828, 831, 833, 836, 838, 840, 842, 845, 848, 850, 852, 854, 856, 858, 860,
863, 865, 867, 870, 872, 875, 877, 879, 881, 883, 885, 887, 889, 891, 893, 895, 897, 899, 902, 904, 906, 908, 910, 912,
914, 916, 918, 920, 923, 926, 928, 931, 933, 935, 937, 939, 941, 943, 945, 947, 949, 951, 953, 955, 957, 959, 961, 963,
965, 968, 970, 972, 974, 976, 978, 980, 982, 984, 986, 988, 990, 992, 994, 997, 999}
C:\Users\Hemad\OneDrive - AgileThought\Documents\Projects\UAG Algorithms\activity9>
```

2. Usa programación dinámica para resolver el siguiente problema de la mochila. Suponga una capacidad máxima de 140 unidades. ¿Cuál es el valor óptimo de la mochila?, ¿Cuáles son los objetos que debemos tomar?

|       | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 10 |
|-------|----|----|----|----|----|----|----|----|----|----|
| Valor | 79 | 32 | 47 | 18 | 26 | 85 | 33 | 40 | 45 | 59 |
| Peso  | 85 | 26 | 48 | 21 | 22 | 95 | 43 | 45 | 55 | 52 |

```
import numpy as np

object_dict = {
    0:{"value":79, "weight":85},
    1:{"value":32, "weight":26},
    2:{"value":47, "weight":48},
    3:{"value":18, "weight":21},
    4:{"value":26, "weight":22},
    5:{"value":85, "weight":95},
    6:{"value":33, "weight":43},
    7:{"value":40, "weight":45},
    8:{"value":45, "weight":55},
    9:{"value":59, "weight":52}
}

number_of_values = len(object_dict)
item_values_list = range(number_of_values + 1)
```

```

max_weight = 140
weights_list = range(max_weight + 1)
cells = [[0 for w in weights_list] for i in item_values_list]

for i in item_values_list:
    for w in weights_list:
        if i == 0 or w == 0:
            cells[i][w] = 0
        elif object_dict[i-1]["weight"] <= w:
            current_item_value = object_dict[i-1]["value"]
            current_item_weight = object_dict[i-1]["weight"]
            cells[i][w] = max(cells[i-1][w], current_item_value + cells[i-1][w-current_item_weight])
        else:
            cells[i][w] = cells[i-1][w]

most_value = cells[number_of_values][max_weight]

print(np.matrix(cells))
print('Most optimal value stolen:', most_value)

remaining_weight = max_weight
for i in range(number_of_values, 0, -1):
    if most_value <= 0:
        break
    if most_value == cells[i - 1][remaining_weight]:
        continue
    else:
        print('Item included: ', list(object_dict.keys())[i - 1])
        most_value = most_value - object_dict[i - 1]["value"]
        remaining_weight = remaining_weight - object_dict[i - 1]["weight"]

```

```

[[ 0  0  0 ...  0  0  0]
 [ 0  0  0 ... 79 79 79]
 [ 0  0  0 ... 111 111 111]
 ...
 [ 0  0  0 ... 137 138 138]
 [ 0  0  0 ... 137 138 138]
 [ 0  0  0 ... 138 138 143]]
Most optimal value stolen: 143
Item included: 9
Item included: 7
Item included: 4
Item included: 3

```