Tutorial 1 : Dynamic programming

Operations research, 3rd semester.

2024

Exercice 1 — *Starting exercise* Here is a recursive algorithm. Apply the dynamic programming method to this algorithm to produce a memoization version and an iterative version. What are the complexities of the 3 algorithms ?

function F(n)
  Si $n \leq 1$ **Alors**
    Renvoyer 1
  Sinon Si n is even **Alors**
    Renvoyer F(n + 1) * F(n - 2)
  Sinon
    Renvoyer F(n - 3) + F(n - 2)

Exercice 2 — *Pocket money*

A student wants to earn some money when she is not at the university. She can work at most T hours by week so that she has time left to study. She found n possible jobs and it is possible for her to divide her time into multiple jobs (for example, she can work two hours for the first job, ten for the third one, ...). The following table gives the reward $s(j, i)$ depending on how many hours j per week she decides to work for each job i, where $T = n = 4$.

Nb hours of work \ Job	Job I	Job II	Job III	Job IV
0	0	0	0	0
1	26	23	16	19
2	39	36	32	36
3	48	44	48	47
4	54	49	64	56

1. Using a naive recursive algorithm, compute how much money she can earn at most.
2. Deduce a memoization dynamic programming algorithm to solve the problem. Use this algorithm on the example.
3. Write the iterative version of the dynamic programming algorithm algorithm. Use this algorithm on the example.
4. What are the complexities of the three algorithms ?
5. How could we compute the number of hours the student has to work for each job to obtain the optimal reward ?

Exercice 3 — *The knapsack problem*

This is a classical problem of Operations Research.

Given :

— a bag of volume V,

— n type of items, numbered from 1 to n

— there are m_i items of type i

— each item of type i has a volume v_i

— each item of type i has a value u_i

we want to fill the bag maximizing the total value of the objects we put inside.

We are going to solve this problem using dynamic programming. Let x_i be the number of times the object i is put in the bag.

1. What is the value $U(x_1, x_2, \ldots, x_n)$ of the bag ?
2. Rewrite the volume constraint as $V(x_1, x_2, \ldots, x_n) \leq V$.
   Let $f(j, b)$ be the maximum value of a bag of volume b using only the first j types of objects, for $b \in [\![0, V]\!]$ and $i \in [\![1; n]\!]$, *i.e.* the maximum value of $U(x_1, \ldots, x_j)$ such that $V(x_1, \ldots, x_j) \leq b$.
3. How computing $f(j, b)$ for all j and b help us to solve the knapsack problem ?
4. What is the value of $f(1, b)$? (clue : it depends on the value of b)
5. Give a recursive formula for f.
6. Write an algorithm to compute f for all j and b. What is its complexity ? What would be the complexity of a brute force algorithm to solve the knapsack problem ?
7. Apply one of your dynamic programming algorithm to solve, in the following example, the knapsack problem such that $V = 10$

objet	1	2	3
v_i	3	5	2
m_i	2	2	4
u_i	10	15	8

Exercice 4 — *Square patch*

Let A be a $n \times n$ square binary matrix (only 1 and 0 entries). Describe an algorithm to find the largest square submatrix of A containing only 1 entries ; and the coordinates of its upper left corner. What is its complexity ?

Exercice 5 — *Climbing stairs*

1. You are in front of a stair with m steps. You can climb the steps one by one and/or two by two. How many ways of climbing this stair are there ?
   We now consider that you can climb α steps at a time, for $\alpha \in (\alpha_1, \alpha_2, \ldots, \alpha_p)$. For example, in the previous question, $p = 2$, $\alpha_1 = 1$ and $\alpha_2 = 2$. Wa assume that the numbers α_i are distinct and sorted. Let $N(m)$ be the number of ways of climbing an m steps stair.
2. Give a recursive formula for $N(m)$.
3. By using pseudo-code, describe a dynamic programming algorithm to solve this problem. What is the complexity of this algorithm ?
4. We assume $p = 3, \alpha_1 = 2, \alpha_2 = 3, \alpha_3 = 5$, give $N(m)$ for $0 \leq m \leq 12$.