CSE 101 Winter 2018, Professor Andrew B. Kahng
TA Bekhzod Soliev

Dynamic Programming. Introduction

Dynamic programming is a powerful mathematical technique that breaks a problem into subproblems of smaller dimensions, solves these subproblems only *once*, *memorizes* the result for each subproblem and then uses it to get an answer to subproblems of larger dimensions. Eventually, this leads to the solution of the initial problem. In order to get an optimal answer, each of the problems must be solved optimally.

There are two ways of how a problem can be solved with the help of dynamic programming.

Top-down approach. In this method, we break a problem into smaller subproblems, solve each subproblem and then combine the results to get an answer to the initial problem. The relation between the problem and its subproblems is represented by recurrent formulas.

Bottom-up approach. In this method, we find an answer to the current problem and then use its result to get answers to problems of larger dimensions. Our initial problem will be their subproblem.

In this homework, we will consider a top-down approach, although many problems can be solved in both ways.

Let's show how dynamic programming works with examples.

Example 1. Find the number of binary strings of the length n that do not have two consecutive zeros.
Solution. Let f_i be the number of binary strings of the length i that do not have two consecutive zeros. The dimension of the problem in this case will be defined by variable i. Let's assume that all smaller subproblems are found optimally, i.e. f_k is the number of binary strings of length k that do not have two consecutive zeros for all $k < i$. We will define the lower bound for k later.

Now let's try and find f_i by using (some of) the values for f_k for all $k < i$.

Let's consider a binary string of the length $i > 0$ that doesn't have two consecutive zeros. This string can end with either 1 or 0. If the string ends with 1, then the rest of it (first $i-1$ characters) should not contain two consecutive zeros. The number of such binary strings is f_{i-1}. If the string ends with 0, then the rest of the string shouldn't have two consecutive zeros and shouldn't end with 0. Because if it ends with 0, then we will have two consecutive zeros at positions $i-1$ and i. This means that the digit on position $i-1$ is one. The number of binary strings of length $i-1$ that end with one equals f_{i-1}. Thus, we get that $f_i = f_{i-1} + f_{i-2}$. As we can see, we need two base cases: $f_0 = 1, f_1 = 2$: there is one string of the length 0 that doesn't have any two consecutive zeros (an empty set) and there are two strings of the length 1 that do not have two consecutive zeros (one and zero). The answer to the question is f_n.

The complexity of this solution is $\mathcal{O}(n)$.

Example 2. You can perform the following operations on an input number x:

1. $x := x + 1$
2. $x := 2 * x$
3. $x := 3 * x$
4. $x := 5 * x$

In how many ways can you turn number 1 into a given integer positive number n?

Solution. Let f_i be the number of ways to turn number 1 into number i. If we are given number f_i, then we know that there will be at least f_i ways to turn number 1 into number $i+1$. Indeed, if we represent 1 to i and then add 1 to the result, we will get $i+1$. Similarly there are at least f_i ways to turn 1 into $2*i$, $3*i$ and $5*i$. Let the base case be $f_1 = 0$ (we can't turn 1 into 1 using the mentioned operations). Let $f_i = 0$ for all $1 < i \leq n$. Then, for all $1 \leq i \leq n-1$ (from 1 to n), we will make the following updates

$$f_{i+1} = f_{i+1} + f_i$$

$$f_{2*i} = f_{2*i} + f_i$$
$$f_{3*i} = f_{3*i} + f_i$$
$$f_{5*i} = f_{5*i} + f_i$$

(we only need to consider indexes that are no greater than n). The answer to the questions is f_n.

The complexity of the solution is $\mathcal{O}(n)$.

Example 3. You are given a knapsack of weight W and n items of weights $w_1, w_2, ..., w_n$. You need to fill the knapsack in such a way that the total weight of all items in the knapsack is maximized and doesn't exceed the capacity of the knapsack.

Solution. Let $f_{i,k}$ be the optimal weight of all items in the knapsack of weight k when we can use the first i items. Here, we have defined a problem of the dimensions (i, k). These two parameters help us define a problem (note that function f may also depend on variables that have no connection with the dimension of the problem).

Let's assume that we know optimal answers to all subproblems of smaller sizes, i.e. to all problems (i', k'), such that $i' \leq i$ and $k' \leq k$ and $(i, k) \neq (i', k')$. Now, we want to find the answer to the problem (i, k). Let's look at the item number i. There are two possible actions that we can take: we can either include item i in the knapsack, if there is enough capacity for it, or we can ignore it. If we ignore item i, this means that we have to fill the knapsack of the weight k optimally without using this item. Thus, we can only include items from 1 to $i - 1$. Therefore, the subproblem where we don't include item i will be $(i - 1, k)$. The answer to this subproblem is $f_{i-1,k}$. Now, let's consider the case where we try to put item i into the knapsack. It is only possible, if the capacity of the knapsack is not less than the weight of the item i, i.e. $k \geq w_i$. If this condition is true, then we put the item in the knapsack. As regards the rest of the space in the knapsack, we will try to fill it with items $1, 2, ..., i - 1$ in the most optimal way. This subproblem will have dimension $(i - 1, k - w_i)$, where $k - w_i$ is an available capacity left in the knapsack after we put item i in it. So, all smaller subproblems of problem (i, k) can be divided into two sets: when we include item i in the knapsack and when we don't. Using all these observations we can calculate the value $f_{i,k}$: The answer to the problem is $f_{n,W}$.

The complexity of this solution is $\mathcal{O}(nW)$.

Set 1

Problem 1

Statement Find the number of strings of length n that consist of digits $0, 1, 2$ and:

1. Do not have substring "00"
2. Do not have substring "01"
3. Do not have substrings "01" and "10"
4. Have exactly m zeros
5. Have exactly m zeros and no substrings "01" and "10"

Problem 2

Statement There is a robot on the line at coordinate 1. The robot can make get from coordinate i to one of the coordinates $i+1$, $i+2$, ... , $i+k$, where k is some positive integer number, in one step. Find in how many ways the robot can get from coordinate 1 to coordinate n.

Problem 3

Statement There is a robot on the line at coordinate 1. The robot can get from coordinate i to one of the coordinates $i+1$, $i+2$, ... , $i+k$, where k is some positive integer number, in one step. When the robot visits coordinate i, it costs the robot a_i amount of energy, where a_i is some real number. Find the path from coordinate 1 to coordinate n with a minimal cost and output this path.

Problem 4

Statement There is a robot on the line at coordinate 1. The robot can get from coordinate i to one of the coordinates $i+b_1$, $i+b_2$, ... , $i+b_k$, where k and all b_i are positive integer numbers, in one step. When the robot gets to coordinate i from coordinate j, it costs the robot $a_{i,j}$ amount of energy, where $a_{i,j}$ is some real number. Find the path from coordinate 1 to coordinate n with a minimal cost and output this path.

Problem 5

Statement There is a robot on the line at coordinate 1. The robot can get from coordinate i to one of the coordinates $i+1$, $i-1$, $i+2$, $i-2$, ... , $i+k$, $i-k$, where k is some positive integer number, in one step. The robot, however, can't get to coordinates less than 1 or greater than n. Find the number of ways in which the robot can get from coordinate 1 to coordinate n in no more than m steps.

Problem 6

Statement You are given a set of n integer numbers $a_1, a_2, ..., a_n$. Find a subset of these numbers that doesn't include any neighboring numbers and such that the sum of all numbers in this subset is as large as possible.

Problem 7

Statement You are given a board of dimensions $n \times m$. There is a robot on cell $(1, 1)$. The robot can get from cell (i, k) to cells $(i+1, k)$, $(i, k+1)$ and $(i+1, k+1)$ in one step, as long as it stays on the board. In how many ways can the robot get from cell $(1, 1)$ to cell (n, m).

Problem 8

Statement You are given a board of dimensions $n \times m$. There is a robot on cell $(1, 1)$. The robot can get from cell (i, j) to cells $(i + 1, j)$, $(i, j + 1)$ and $(i + 1, j + 1)$ in one step, as long as it stays on the board. When robot visits cell (i, j), it spends $a_{i,j}$ amount of energy, where all $a_{i,j}$ are real numbers. Output the path of the minimal cost for the robot to get from cell $(1, 1)$ to cell (n, m).

Problem 9

Statement You are given a board of dimensions $n \times m$. There is a robot on cell $(1, 1)$. The robot can get from cell (i, j) to any neighboring cell in one step, as long as it stays on the board. When the robot visits cell (i, j), it spends $a_{i,j}$ amount of energy, where all $a_{i,j}$ are real numbers. Output the path of the minimal cost from cell $(1, 1)$ to cell for the robot to get (n, m), if the robot can't make more than k steps (the length of the path can't be more than k)

Problem 10

Statement You are given a board of dimensions $n \times m$. There is a robot at cell $(1, 1)$. The robot can make steps of two types:

1. It can get from current cell (i, j) to cell $(i + 1, j)$
2. It can get from current cell (i, j) to any cell (i, x), where $1 \leq x \leq m$

After each step the robot should stay on the board. The robot can make step of the second type only once for each row. When the robot visits cell (i, j), it spends $a_{i,j}$ amount of energy, where all $a_{i,j}$ are real numbers. When the robot gets from column i to column j, it spends $c_{i,j}$ amount of energy, where all $c_{i,j}$ are real numbers.

1. Find the number of ways in which the robot can get from cell $(1, 1)$ to cell (n, m)
2. Output the path of the minimal cost for the robot to get from cell $(1, 1)$ to cell (n, m).

Set 2

Problem 1

Statement You are given a knapsack of positive integer capacity W. There are n items. Item i has weight w_i and cost p_i. All weights and costs are positive integer numbers. You are asked to fill the knapsack with items, such that their total cost is maximized, if:

1. You can take each item only once
2. You can take each item as many times as you want
3. You can take each item only b_i times, where b_i is some positive integer number for all $1 \leq i \leq n$
4. You can take a fractional part of each item (which could be bigger than one)

Problem 2

Statement You are given two knapsacks of positive integer capacities W and V. There are n items. Item i has weight w_i and cost p_i. All weights and costs are positive integer numbers. You are asked to fill the knapsacks with items, such that their total cost over two knapsacks is maximized. You can take each item only once.

Problem 3

Statement You are given a set of n positive integer numbers $a_1, a_2, ..., a_n$ and positive integers K and m. Find the number of subsets of these numbers, such that the sum of all numbers in each subsets equal K by modulo m.

Problem 4

Statement You are given a set of n positive integer numbers $a_1, a_2, ..., a_n$. Divide them into two sets such that the absolute difference of sums of both sets is minimized. The sum of a set is a sum of all numbers in a set.

Problem 5

Statement You are given a set of n numbers: $1, 2, ..., n$. In how many ways is it possible to assign them into k groups, where k is a positive integer number, such that each group has at least one number, the order of numbers inside of the groups doesn't matter while the order of the groups does matter.

Problem 6

Statement You are given a positive integer n. In how many ways can this integer be represented as a sum of positive integer numbers, if they must come in a non-decreasing order. For example, $5 = 4 + 1 = 3 + 2 = 2 + 2 + 1 = 1 + 1 + 1 + 1 + 1$

Problem 7

Statement You are given a list of n positive integer numbers $a_1, a_2, ..., a_n$ and a positive integer K. Your goal is to perform the following operations over this list:

1. Divide the list from the first number to the last one into K consecutive groups
2. For each group, find the sum of all numbers in it
3. Find the xor-product of all resulting sums (there are K of them)

Your goal is to divide numbers into K groups in such a way that the xor-product is maximized.

Set 3

Problem 1

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. Your goal is to find the number of increasing subsequences $a_{i_1}, a_{i_2}, ..., a_{i_l}$, where $1 \leq i_1 < i_2 < ... < i_l \leq n$ and $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_l}$

Problem 2

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. Your goal is to find the longest increasing subsequence $a_{i_1}, a_{i_2}, ..., a_{i_l}$, where $1 \leq i_1 < i_2 < ... < i_l \leq n$ and $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_l}$

Problem 3

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. Your goal is to find the number of increasing subsequences $a_{i_1}, a_{i_2}, ..., a_{i_l}$, where $1 \leq i_1 < i_2 < ... < i_l \leq n$ and $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_l}$

Problem 4

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. Your goal is to find the number of longest increasing subsequences $a_{i_1}, a_{i_2}, ..., a_{i_l}$, where $1 \leq i_1 < i_2 < ... < i_l \leq n$ and $a_{i_1} \leq a_{i_2} \leq ... \leq a_{i_l}$

Problem 5

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. Your goal is to find the longest subsequence $a_{i_1}, a_{i_2}, ..., a_{i_l}$, such that $1 \leq i_1 < i_2 < ... < i_l \leq n$ and $|a_{i_{k-1}} - a_{i_k}| \geq |a_{i_k} - a_{i_{k+1}}|$ for $k \geq 2$. If such a sequence doesn't exist, then output -1.

Problem 6

Statement You are given a sequence of n positive integer numbers $a_1, a_2, ..., a_n$. A mountain sequence is a sequence in which numbers first go in an non-decreasing order and then in a non-decreasing order. Your goal is to find the longest mountain subsequence.

Problem 7

Statement You are given two sequences of n and m positive integers. Find the longest common subsequence.

Problem 8

Statement You are given two sequences of n and m positive integers. Find the number of common subsequences.

Problem 9

Statement You are given three sequences of n, m and q positive integers. Find the longest common subsequence.

Problem 10

Statement You are given two sequences of n and m positive integers. Find the longest common increasing subsequence.

Set 4

Problem 1

Statement You are given two strings $s = s_1 s_2 ... s_n$ and $t = t_1 t_2 ... t_m$. You may perform three types of operations on a string: delete a symbol, insert a symbol or replace existing symbol with any other possible symbol. Assume that all symbols are lowercase English letters. What is the minimum number of operations you need to transform string s to string t?

Problem 2

Statement You are given two strings $s = s_1 s_2 ... s_n$ and $t = t_1 t_2 ... t_m$. Characters of strings are lowercase English letters. Find the longest common substring.

Problem 3

Statement You are given a string $s_1 s_2 ... s_n$. Find the largest substring which is a palindrome. String is a palindrome if it equals its reversed self.

Problem 4

Statement You are given a string $s_1 s_2 ... s_n$. Find the number of substrings which are palindromes. String is a palindrome if it equals its reversed self.

Problem 5

Statement You are given a string $s_1 s_2 ... s_n$, where s_i belongs to a set $\{'(',')','\{','\}','[',']'\}$. Remove the minimal number of brackets from the string to get a correct bracket sequence.

Problem 6

Statement You are given a string $s_1 s_2 ... s_n$, where s_i belongs to a set $\{'(',')','\{','\}','[',']'\}$. Add the minimal number of brackets to the string to get a correct bracket sequence.

Problem 7

Statement You are given a string $s_1 s_2 ... s_n$, where s_i belongs to a set $\{'(',')','\{','\}','[',']'\}$. Find the largest substring which is a correct bracket sequence.

Problem 8

Statement You are given a list of t matrices $A_1, A_2, ..., A_t$ of dimensions $(n_1, m_1), (n_2, m_2), ..., (n_t, m_t)$. At each step, you can multiply any two neighboring matrices and change them with their product. The multiplication of two matrices with dimensions (n, m) and (m, k) takes nmk operations. It is guaranteed that any two neighboring matrices can be multiplied in the initial list. Perform n-1 steps such that the total cost is minimized.

Problem 9

Statement You are given a list of n integer numbers $a_1, a_2, ..., a_n$. At each step, you can take any neighboring numbers a_i, a_{i+1} and replace them with the sum of their squares $a_i^2 + a_{i+1}^2$. This operation costs $a_i * a_{i+1}$. Perform $n - 1$ such operations by minimizing the total cost as much as possible.

Set 5

Problem 1

Statement You are given an undirected graph $G = (V, E)$ with n nodes and m edges. You are also given two distinct nodes s and t. Find the number of paths from node s to node t of the length k, where k is some positive integer.

Problem 2

Statement You are given a full undirected graph $G = (V, E)$ with n nodes and m edges. Each edge has some real positive weight. Find the shortest distance between all pairs of nodes.

Problem 3

Statement You are given a directed graph $G = (V, E)$ with n nodes and m edges. The graph may contain self-loops and repeated edges. Each edge has some real weight. You are also given a node s. Find shortest distance from node s to all other nodes. If the shortest distance doesn't exist, output "inf".

Problem 4

Statement You are given a directed graph $G = (V, E)$ with n nodes and m edges. The graph may contain self-loops and repeated edges. Each edge has some real weight. Find a negative cycle, if such exists, and output it. Otherwise output -1.

Problem 5

Statement You are given a directed acyclic graph $G = (V, E)$ with n nodes and m edges. Each edge has some real weight. You are also given a node s. Find the shortest distance between all pairs of nodes.

Problem 6

Statement You are given a tree with n nodes. Each node in a tree has some real weight. Find the set of nodes, such that none of them are adjacent and their total weight is maximized.

Problem 7

Statement You are given a tree with n nodes. Each edge in a tree has some real weight. Find the set of edges, such that none of them are adjacent and their total weight is maximized.

Problem 8

Statement You are given a tree with n nodes. Each node in a tree has some real weight. Find a subtree with a maximal sum. The sum of a subtree is a sum of weights of all nodes in the subtree.

Problem 9

Statement You are given a tree with n nodes. Find the sum of lengths of all paths in the tree.

Problem 10

Statement Find the number of all connected graphs of n nodes. Output the number by modulo $10^9 + 7$

Set 6

Problem 1

Statement You are given m sets $A_1, A_2, ..., A_m$. Each consists of integer numbers in the range $[1, n]$. Find the minimal number of sets, such that their union gives all integer numbers in the range $[1, n]$.

Problem 2

Statement You are given an undirected graph $G = (V, E)$ with n nodes and m edges. Find the number of Hamiltonian paths in it.

Problem 3

Statement You are given an undirected weighted graph $G = (V, E)$ with n nodes and m edges and integer weights. Find the shortest Hamiltonian path in this graph.

Problem 4

Statement You are given an undirected graph $G = (V, E)$ with n nodes and m edges and integer weights. Find the maximum matching in this graph.