Section 4: Divide and Conquer & Dynamic Programming

Haneul Shin

February 2021

Table of Contents

Divide and Conquer

Divide and Conquer

Approach

1. **Divide** the problem into independent subproblems.
2. **Conquer** the subproblems by solving them recursively.
3. Combine the solutions to the subproblems to solve the original problem.

Merge Sort $4, 5, 7, 2, 1, 3, 8, 6$

Karatsuba's Algorithm

A divide and conquer algorithm to multiply two n digit numbers efficiently.

Recall: naive algorithm takes quadratic time.

Multiplication by Divide & Conquer

Suppose we are given x and y.

We write

$$x = 10^{n/2}a + b, \quad y = 10^{n/2}c + d.$$

Then,

$$xy = ac10^{n} + (ad + bc)10^{n/2} + bd.$$

Multiplication by Divide & Conquer

Suppose we are given x and y.

We write

$$x = 10^{n/2}a + b, \quad y = 10^{n/2}c + d.$$

Then,

$$xy = ac10^{n} + (ad + bc)10^{n/2} + bd.$$

Time complexity: $T(n) = 4T(n/2) + O(n) \quad \rightarrow \quad T(n) = O(n^2)$

Karatsuba's Algorithm

$$(a+b)(c+d) = (ad+bc) + (ac+bd)$$

$$xy = ac10^n + (ad+bc)10^{n/2} + bd$$

Compute ac, bd, $(a+b)(c+d)$

Karatsuba's Algorithm

$$(a+b)(c+d) = (ad+bc) + (ac+bd)$$

$$xy = ac10^n + [(a+b)(c+d) - ac - bd]10^{n/2} + bd$$

Compute ac, bd, $(a+b)(c+d)$

Time complexity: $T(n) = 3T(n/2) + O(n) \quad \rightarrow \quad T(n) = O(n^{log_2 3})$

Karatsuba's Algorithm

$$1234 \cdot 3281$$

Karatsuba's Algorithm

$$1234 \cdot 3281$$

$$(12 \cdot 10^2 + 34)(32 \cdot 10^2 + 81)$$

Karatsuba's Algorithm

Compute ac, bd, $(a+b)(c+d)$

$$1234 \cdot 3281$$

$$(12 \cdot 10^2 + 34)\,(32 \cdot 10^2 + 81)$$

$$12 \cdot 32,\; 34 \cdot 81,\; (12+34) \cdot (32+81)$$

Karatsuba's Algorithm

Compute ac, bd, $(a+b)(c+d)$

$$1234 \cdot 3281$$

$$(12 \cdot 10^2 + 34)(32 \cdot 10^2 + 81)$$

$$12 \cdot 32, \; 34 \cdot 81, \; (12+34) \cdot (32+81)$$

$$384, \; 2754, \; 5198$$

Karatsuba's Algorithm

$$xy = ac10^n + [(a+b)(c+d) - ac - bd]10^{n/2} + bd$$

$$1234 \cdot 3281$$

$$(12 \cdot 10^2 + 34)(32 \cdot 10^2 + 81)$$

$$12 \cdot 32,\ 34 \cdot 81,\ (12+34)\cdot(32+81)$$

$$384,\ 2754,\ 5198$$

$$1234 \cdot 3281 = 384 \cdot 10^4 + (5198 - 384 - 2754) \cdot 10^2 + 2754$$

$$= 4{,}048{,}754$$

Dynamic Programming

Dynamic Programming

Approach

1. Divide a problem into *potentially overlapping* subproblems.
2. Conquer the subproblems by solving them recursively.
3. Combine the solutions to the subproblems to solve the original problem.

Comparison

Divide and Conquer

1. Divide the problem into *independent* subproblems.
2. Conquer the subproblems by solving them recursively.
3. Combine the solutions to the subproblems to solve the original problem.

Dynamic Programming

1. Divide a problem into *potentially overlapping* subproblems.
2. Conquer the subproblems by solving them recursively.
3. Combine the solutions to the subproblems to solve the original problem.

Fibonacci Numbers

$$F_0 = 0,\ F_1 = 1,\ F_n = F_{n-1} + F_{n-2} \text{ for } n \geq 2$$

Types of Dynamic Programming

1. Top-down DP (recursion with memoization)
2. Bottom-up DP

Longest Path Problem

Design an efficient algorithm to find the longest path in a directed acyclic graph whose edges have real-number weights. (Problem Set 2, Problem 5)

Longest Path Problem

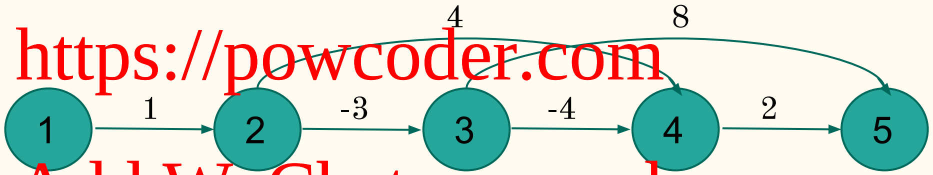
Design an efficient algorithm to find the longest path in a directed acyclic graph whose edges have real-number weights. (Problem Set 2, Problem 5)

Solution

1. Sort the vertices in topological order.
2. Create an array *dist* such that *dist*[v] is the length of the longest path starting at v. Initialize each distance to be 0.
3. Iterate over the vertices in reverse topological order. For each vertex v:
   a. Take the maximum possible sum of a child u's max path value and the edge weight of (v, u).
4. Return the maximum distance for any vertex.

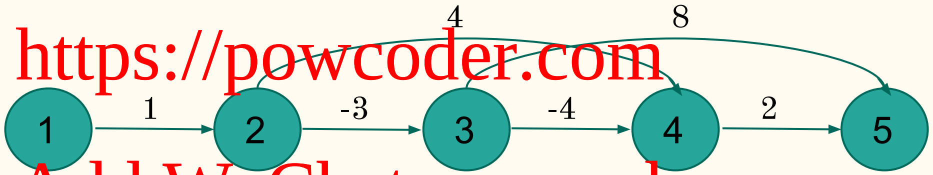
Longest Path Problem

Design an efficient algorithm to find the longest path in a directed acyclic graph whose edges have real-number weights. (Problem Set 2, Problem 5)



Longest Path Problem

Design an efficient algorithm to find the longest path in a directed acyclic graph whose edges have real-number weights. (Problem Set 2, Problem 5)

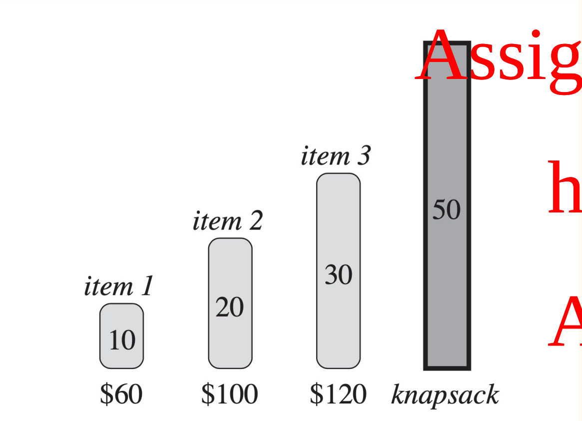


DP Relation: $\text{dist}[v] = \max\left(\max_{(v,\,u)\,\in E}\left(\text{dist}[u] + w(v,\,u)\right),\, 0\right)$

0/1 Knapsack Problem

Given n items with weights $\{w_1, w_2, ..., w_n\}$ and values $\{v_1, v_2, ..., v_n\}$ and a knapsack with capacity W, determine the maximum possible value attained by the knapsack.

0/1 Knapsack Problem



0/1 Knapsack Problem Solution

Let $m[i, j]$ be the maximum possible value using the first i items using weight $\leq j$

0/1 Knapsack Problem Solution

Let $m[i, j]$ be the maximum possible value using the first i items using weight $\leq j$

- $m[0, j] = 0$

- If $w[i] > j$, $m[i, j] = m[i - 1, j]$

- If $w[i] \leq j$, $m[i, j] := \max(m[i - 1, j], m[i - 1, j - w[i]] + v[i])$

0/1 Knapsack Problem Solution

for j from 0 to W:

    $m[0, j] := 0$

for i from 1 to n:

    for j from 0 to W:

        if $w[i] > j$:

            $m[i, j] := m[i - 1, j]$

        else:

            $m[i, j] := \max(m[i - 1, j], m[i - 1, j - w[i]] + v[i])$

Practice Problems

Problem 1

Given an array a with n integers, design a divide & conquer algorithm to find the sum of the maximum sum subarray. (This is a subarray with the maximum possible sum, which may be empty).

Example: For $a = [2, 3, -7, 1, 5, -3]$, the maximum sum subarray has sum 6.

Problem 2

Given an array a with n integers, find the length of a longest increasing subsequence of the array. (This is a maximum-length subsequence of the array such that each element is strictly larger than the previous element.)

Problem 3

Given a set of coin values $c = \{c_1, c_2, ..., c_k\}$ and a target sum of money n, determine the number of ways to produce the target sum where order matters. For instance, if c = $\{1, 2\}$ and $n = 3$ there are 3 distinct ways (1+2, 2+1, 1+1+1).