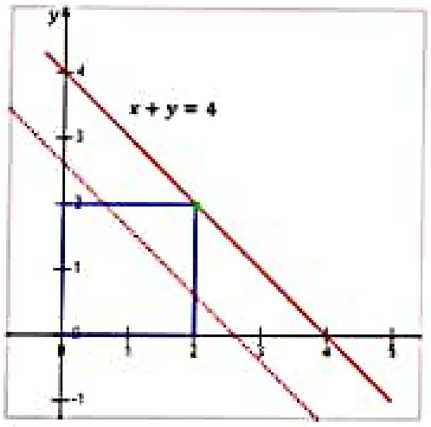
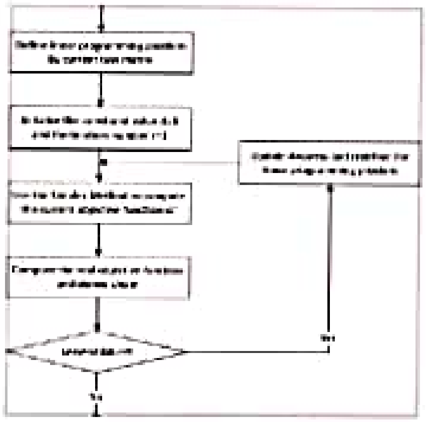
Abstract and Figures

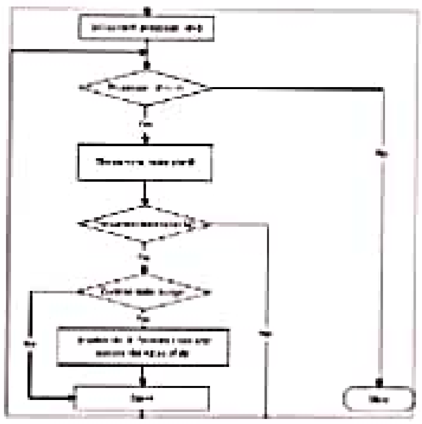
A real-time task scheduling system model was analyzed under a heterogeneous multiprocessor platform with task duplication. This analysis focused on the designs and performances of linear and dynamic programming algorithms for real-time task scheduling under a heterogeneous platform with task duplication. Moreover, experimental analyses were performed to evaluate the performances of different algorithms under different conditions. The advantages of the two proposed algorithms were compared under the same situations to discover which one achieves a higher task scheduling efficiency for a heterogeneous real-time system.



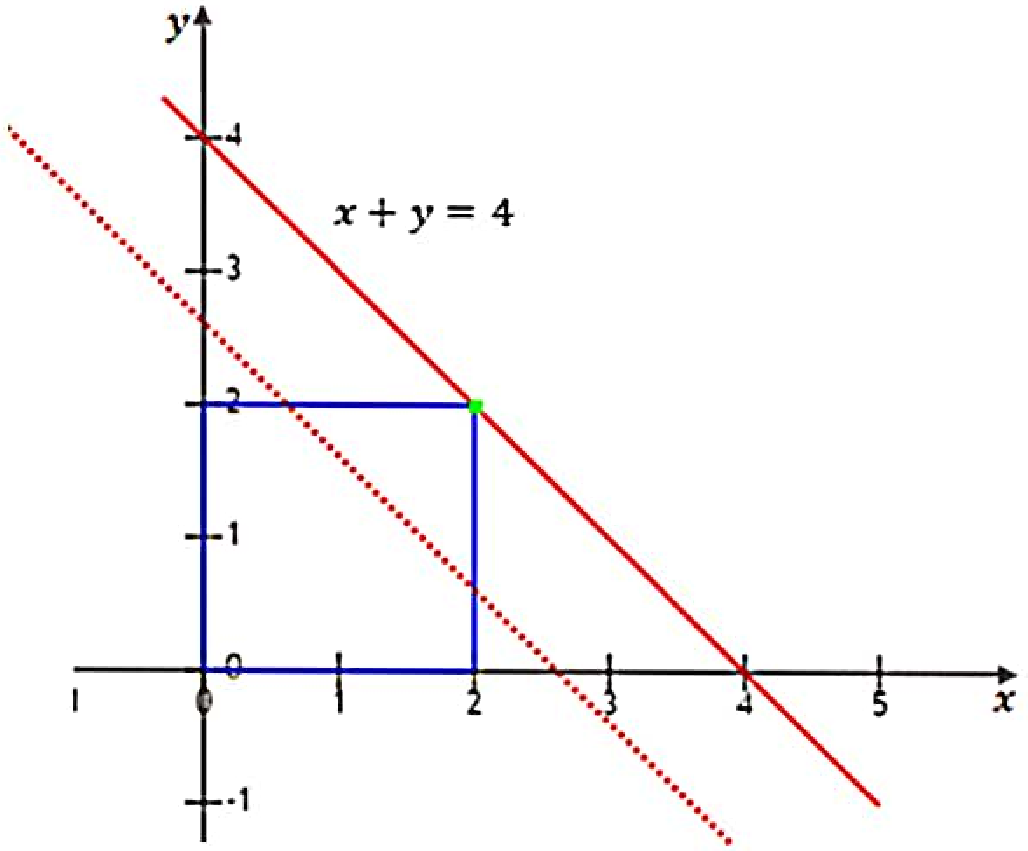
Solution to a simple linear...



Flowchart of iterative linear...

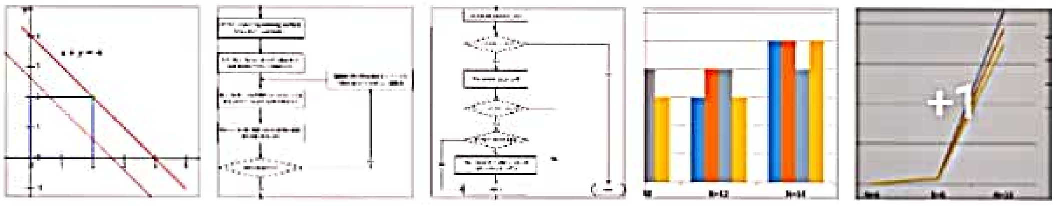


Flowchart of dynamic...



Solution to a simple linear programming problem

Source publication



Linear and dynamic programming algorithms for real-time task scheduling with task duplication

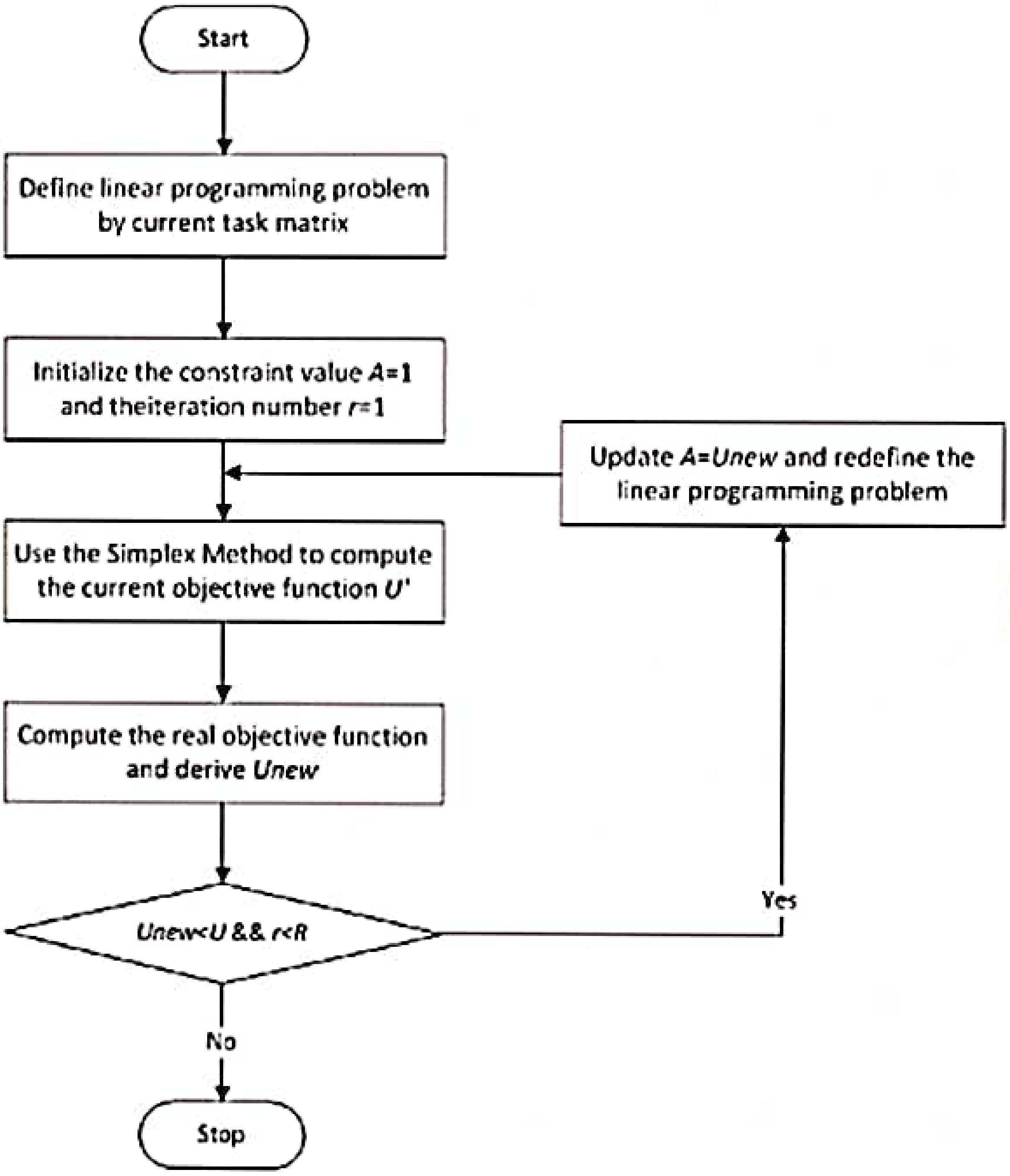
Article | Full-text available

Feb 2019

Weizhe Zhang · Yao Hu · Hui He · [...] · Allen Huaxin Chen

A real-time task scheduling system model was analyzed under a heterogeneous multiprocessor platform with task duplication. This analysis focused on the designs and performances of linear and dynamic programming algorithms for

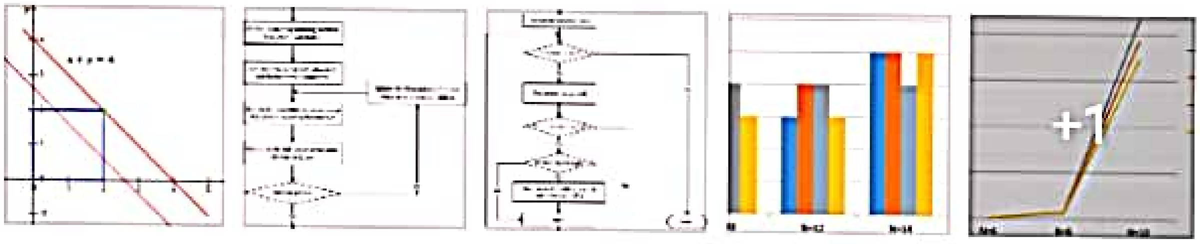
Advertisement



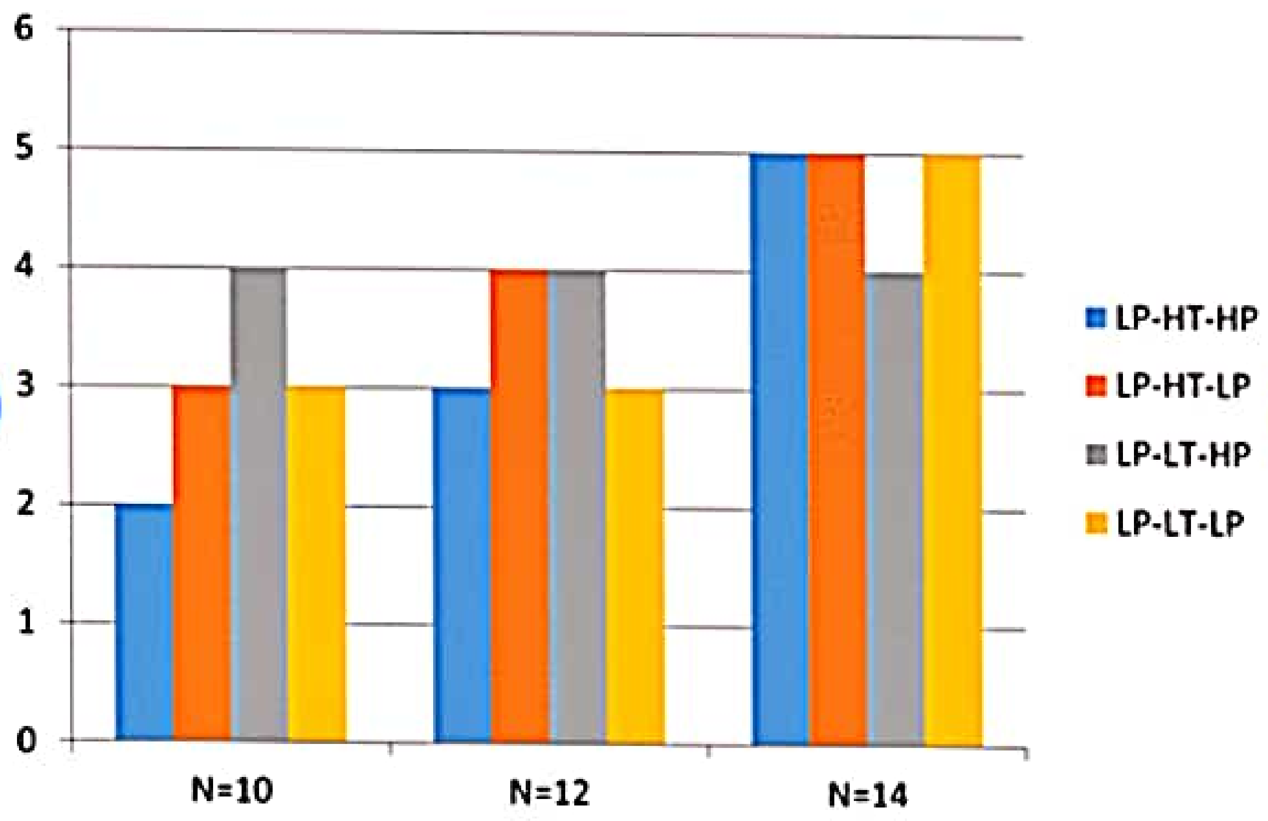
Start
Define linear programming problem by current task matrix
Initialize the constraint value A=1 and theiteration number r=1
Update A=Unew and redefine the linear programming problem
Use the Simplex Method to compute the current objective function U'
Compute the real objective function and derive Unew
Unew<U && r<R
Yes
No
Stop

Flowchart of iterative linear programming algorithm

Source publication

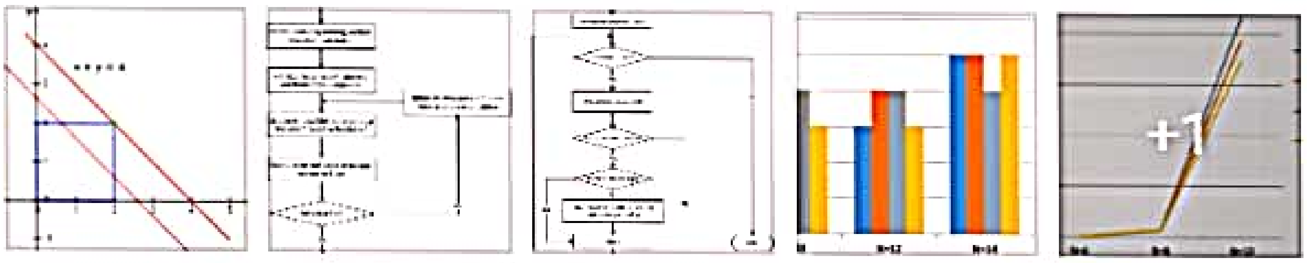


Linear and dynamic programming algorithms for real-time task scheduling with task duplication



Run time comparison of the linear programming algorithm when K=3\documentclass[12pt]{minimal} \usepackage{amsmath} \usepackage{wasysym} \usepackage{amsfonts} \usepackage{amssymb} \usepackage{amsbsy} \usepackage{mathrsfs} \usepackage{upgreek} \setlength{\oddsidemargin}{-69pt} \begin{document}$$K=3$$\end{document}

Source publication



Linear and dynamic programming algorithms for real-time task scheduling with task duplication

Dynamic Programming

Dynamic programming is a technique that breaks the problems into sub-problems, and saves the result for future purposes so that we do not need to compute the result again. The subproblems are optimized to optimize the overall solution is known as optimal substructure property. The main use of dynamic programming is to solve optimization problems. Here, optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem. The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.

The definition of dynamic programming says that it is a technique for solving a complex problem by first breaking into a collection of simpler subproblems, solving each subproblem just once, and then storing their solutions to avoid repetitive

Consider an example of the Fibonacci series. The following series is the Fibonacci series:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ,...

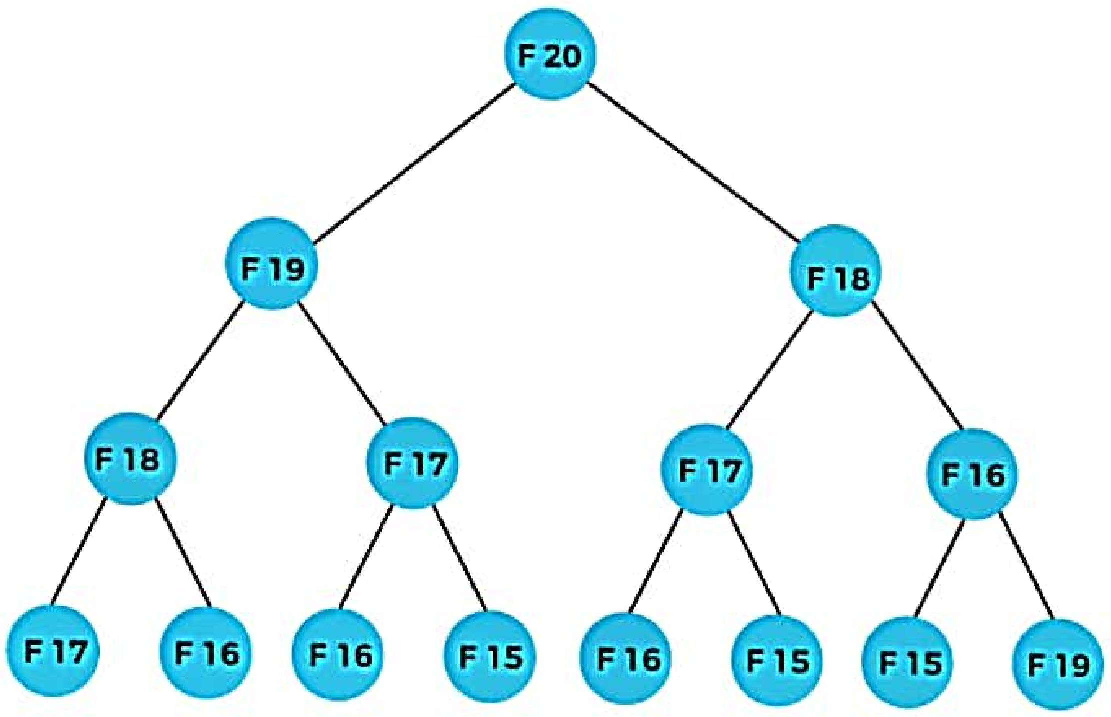
The numbers in the above series are not randomly calculated. Mathematically, we could write each of the terms using the below formula:

F(n) = F(n-1) + F(n-2),

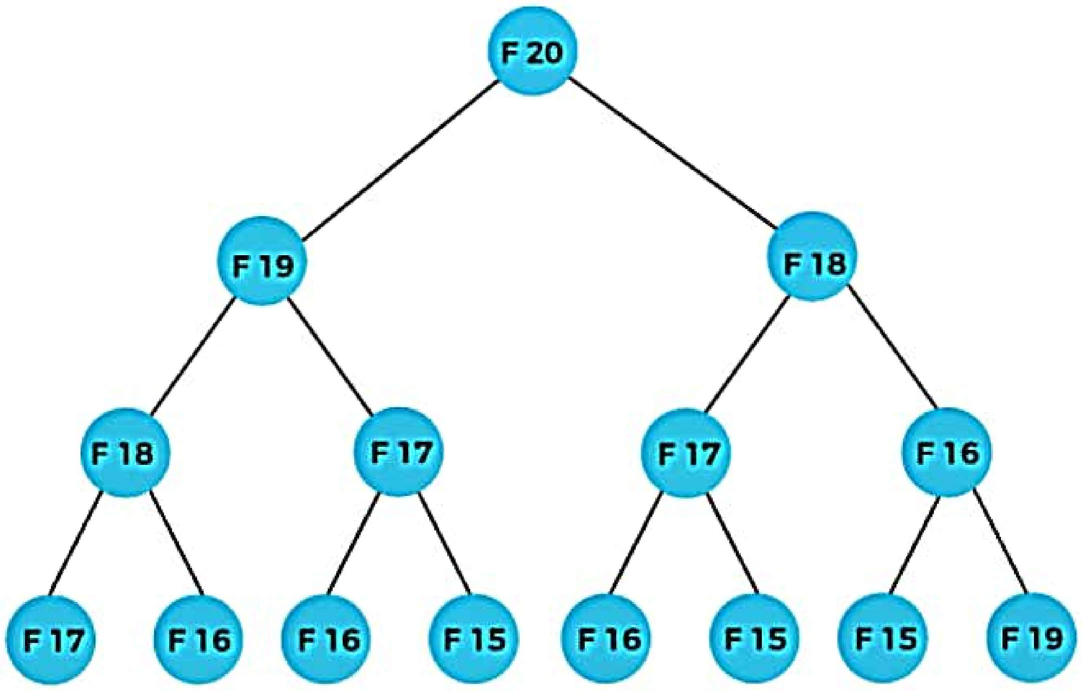
With the base values F(0) = 0, and F(1) = 1. To calculate the other numbers, we follow the above relationship. For example, F(2) is the sum **f(0)** and **f(1),** which is equal to 1.

How can we calculate F(20)?

The F(20) term will be calculated using the nth formula of the Fibonacci series. The below figure shows that how F(20) is calculated.



As we can observe in the above figure that F(20) is calculated as the sum of F(19) and F(18). In the dynamic programming approach, we try to divide the problem into the similar subproblems. We are following this approach in the above case where F(20) into the similar subproblems, i.e., F(19) and F(18). If we recap the definition of dynamic programming that it says the similar subproblem should not be computed more than once. Still, in the above case, the subproblem is calculated twice. In the above example, F(18) is calculated two times; similarly, F(17) is also calculated twice. However, this technique is quite useful as it solves the similar subproblems, but we need to be cautious while storing the results because we are



As we can observe in the above figure that F(20) is calculated as the sum of F(19) and F(18). In the dynamic programming approach, we try to divide the problem into the similar subproblems. We are following this approach in the above case where F(20) into the similar subproblems, i.e., F(19) and F(18). If we recap the definition of dynamic programming that it says the similar subproblem should not be computed more than once. Still, in the above case, the subproblem is calculated twice. In the above example, F(18) is calculated two times; similarly, F(17) is also calculated twice. However, this technique is quite useful as it solves the similar subproblems, but we need to be cautious while storing the results because we are