DIVIDE AND CONQUER ALGORITHM

http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-046j-introduction-to-algorithms-sma-5503-fall-2005/video-lectures/lecture-3-divide-and-conquer-strassen-fibonacci-polynomial-multiplication/

"Really? — *my* people always say *multiply* and conquer."

Decide whether the statements are true or false.

1. The subject of the lesson is master theorem.
2. Students should go to recitation on Friday.
3. Homework should be submitted on Saturday.
4. Monday is a holiday.
5. Divide and Conquer is the same as divide and rule.
6. The question of family feud will be on the quiz.
7. Divide and Conquer was practiced by the Americans.
8. Cormen, Leiserson, Rivest and Stein are the authors of a book on algorithms.
9. There are three steps in Divide and Conquer.
10. The value of N is larger than it was in the original problem.
11. Merge algorithm fits into this system.
12. When you recursively solve each part of the array, it is Divide.

Listen again and fill in the missing words.

Good morning everyone. Today we are going to do some algorithms, ___________ algorithms, and we are going to use a lot of the, well, some of the simpler mathematics that we developed last class like the master theorem for solving ___________. We are going to use this a lot. Because we are going to talk about recursive algorithms today. And so we will find their running time using the master theorem. This is just the same as it was last time, I hope, unless I made a mistake. A couple of reminders. You should all go to recitation on Friday. That is required. If you want to, you can go to homework lab on Sunday. That may be a ____________ for you to actually work on your problem set a few hours early.

Well, actually, it's _______ on Wednesday so you have lots of time. And there is no class on Monday. It is the holiday known as Student Holiday, so don't come. Today we are going to cover something called Divide and Conquer. Also known as divide and rule or divide et impera for those of you who know Latin, which is a ______ and tested way of conquering a land by dividing it into sections of some kind. It could be different political factions, different whatever. And then somehow making them no longer ______ each other. Like starting a family feud is always a good method. You should remember this on your quiz. I'm kidding.

And if you can separate this big power structure into little power structures such that you _________ each little power structure then you can conquer all of them individually, as long as you make sure they don't get back together. That is divide and conquer as practiced, say, by the British. But today we are going to do divide and conquer as practiced in Cormen, Leiserson, Rivest and Stein or every other algorithm textbook. This is a very basic and very powerful algorithm design technique. So, this is our first real algorithm design experience.

We are still sort of mostly in the analysis mode, but we are going to do some _______ design. We're going to cover maybe only three or four major design techniques. This is one of them, so it is really important. And it will lead to all sorts of recurrences, so we will get to use everything from last class and see why it is useful. As you might expect, the first step in divide-and-conquer is divide and the second step is conquer.

But you may not have guessed that there are three steps. And I am leaving some blank space here, so you should, too. Divide-and-conquer is an algorithmic technique. You are given some big problem you want to solve, you don't really know how to solve it in an _________ way, so you are going to split it up into ___________. That is the divide. You are going to divide that problem, or more precisely the instance of that problem, the particular instance of that problem you have into subproblems. And those subproblems should be smaller in some sense. And smaller means normally that the value of N is smaller than it was in the original problem. So, you sort of made some progress. Now you have one, or more likely you have several subproblems you need to solve. Each of them is smaller. So, you recursively solve each subproblem.

That is the conquer step. You conquer each subproblem recursively. And then somehow you combine those solutions into a solution for the whole problem. So, this is the general divide-and-conquer ___________. And lots of algorithms fit it. You have already seen one algorithm that fits this paradigm, if you can remember. Merge sort, good. Wow, you are all awake. I'm ____________. So, we saw merge sort. And, if I am clever, I could fit it in this space. Sure. Let's be clever. A quick review on merge sort. Phrased in this 1, 2, 3 kind of method. The first step was to divide your ________ into two halves. This really doesn't mean anything because you just sort of think, oh, I will pretend my array is divided into two halves.

There is no work here. This is zero time. You just look at your array. Here is your array. I guess maybe you compute n/2 and take the floor. That takes __________ time. And you say OK, I am pretending my array is now divided into the left part and the right part. And then the interesting part is that you recursively solve each one. That's the conquer. We recursively sort each subarray. And then the third step is to ___________ those solutions.

Algorithms

Adapted from Wikipedia

Pre-reading

1) **What is an algorithm?**
2) **What are algorithms used for?**
3) **How old is the concept of an algorithm?**
4) **Which algorithms do you know?**
5) **What is the difference between determinism and randomness?**

In mathematics, computer science, and related subjects, an **algorithm** (derived from the name of mathematician al-Khwārizmī and transformed to match the Greek "arithmos"-number) is an effective method for solving a problem expressed as a finite sequence of steps. Algorithms are used for calculation, data processing, and many other fields.

Each algorithm is a list of well-defined instructions for completing a task. Starting from an initial state, the instructions describe a computation that proceeds through a well-defined series of successive states, eventually terminating in a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate randomness.

A partial formalization of the concept began with attempts to solve the Entscheidungsproblem (the "decision problem") posed by David Hilbert in 1928. Subsequent formalizations were framed as attempts to define "effective calculability" or "effective method"; those formalizations included the Gödel–Herbrand–Kleene recursive functions of 1930, 1934 and 1935, Alonzo Church's lambda calculus of 1936, Emil Post's "Formulation 1" of 1936, and Alan Turing's Turing machines of 1936–7 and 1939.

I. Match the words and their definitions.

pivot **assume** **pidgin code** **pseudocode** **GCD**

a) ....is a mixture of several programming languages in the same program

b) .... is a mixture of a programming language with natural language descriptions.

c) ....a fixed point supporting something which turns.

d) ....is the largest positive integer that divides the numbers without a remainder.

e) .... means to suppose that something it true.

II. Match paradigms and their descriptions.

a) Dynamic programming
b) Brute-force
c) Reduction
d) Linear programming
e) Divide and conquer
f) Search and enumeration
g) The greedy method

- 1) or **exhaustive search**. This is the naïve method of trying every possible solution to see which is best.
-
- 2) repeatedly reduces an instance of a problem to one or more smaller instances of the same problem (usually recursively) until the instances are small enough to solve easily. One such example merge sorting. Sorting can be done on each segment of data after dividing data into segments and sorting of entire data can be obtained in the conquer phase by merging the segments.
-
- 3) When a problem shows optimal substructure, meaning the optimal solution to a problem can be constructed from optimal solutions to subproblems, and overlapping subproblems, meaning the same subproblems are used to solve many different problem instances, this quicker approach avoids recomputing solutions that have already been computed.
-
- 4) it is similar to a dynamic programming algorithm, but the difference is that solutions to the subproblems do not have to be known at each stage; instead a "greedy" choice can be made of what looks best for the moment. It extends the solution with the best possible decision (not all feasible decisions) at an algorithmic stage based on the current local optimum and the best decision (not all possible decisions) made in a previous stage. It is not exhaustive, and does not give accurate answer to many problems. But when it works, it will be the fastest method.
-
- 5) When solving a problem using this method, specific inequalities involving the inputs are found and then an attempt is made to maximize (or minimize) some linear function of the inputs.
-
- 6) This technique involves solving a difficult problem by transforming it into a better known problem. The goal is to find an algorithm whose complexity is not dominated by the resulting reduced algorithm's.
-
- 7) Many problems can be modeled as problems on graphs. A graph exploration algorithm specifies rules for moving around a graph and is useful for such problems. This category also includes search algorithms, branch and bound enumeration and backtracking.

III. Which algorithms would you use in these situations?

a) Find the shortest path to a goal from a vertex in a weighted graph using the shortest path to the goal from all adjacent vertices. ....

b) Problems such as the maximum flow for directed graphs. ....

c) Finding the median in an unsorted list involves first sorting the list and then pulling out the middle element in the sorted list. ....

d) Problems such as playing chess. ....