

REVIEW: TOP DOWN DESIGN

Algorithm Design

Strategy: Top-Down Design

Technique: Divide-and-Conquer

1

ALGORITHM DESIGN – TOP DOWN DESIGN

- Top-Down Design (Top Down Decomposition)

1. Divide the problem into sub problems.
2. Find solutions for sub problems
3. Combine the sub solutions.

- Note:

Steps 1 and 3 are not independent!

End of Note.

TOP DOWN DESIGN – ATOMIC SUB-PROBLEMS

- Top-Down Design (Top Down Decomposition)
 1. Divide the problem into sub problems.
 2. Find solutions for sub problems
 3. Combine the sub solutions.
- How do we find solutions for sub problems?
 - Apply top-down design recursively
 - Q: When do we stop dividing?
 - A: When we reach “atomic” problems.
 - Atomic problems have known solutions
 - Exercise: Provide examples!

TOP DOWN DESIGN – NUMBER OF SUB-PROBLEMS

○ Question:

- When a problem is divided into sub-problems how does one decide on the number of sub-problems (in general)?

○ Answer:

- In general, working with fewer sub-problems is easier to manage
 - i.e. results in lesser overhead – at design time as well as at execution time – as the division is recursive!

○ Exception:


- For instance, if the underlying machine model includes a large number of processors:
 - then the number of sub-problems should - eventually, i.e. after some number of divisions - be large
 - so as to utilize the number of processors.

DESIGN TECHNIQUE: DIVIDE-AND-CONQUER

- **Divide-and-Conquer** is a special case of Top-Down Design:
 - The structure of one or more sub-problems is same as that of the (original) problem:

DIVIDE-AND-CONQUER - INDUCTION

- Divide-and-Conquer is a special case of Top-Down Design:

- The structure of one or more sub-problems is the same as that of the (original) problem:
 - i.e. the structure of the problem is inductive and
 - therefore the structure of the solution will be inductive / recursive
- Induction referred to in here need not be numeric:
it is usually **structural induction***
- 

DIVIDE-AND-CONQUER - INDUCTION

- Divide-and-Conquer is a special case of Top-Down Design:

- The structure of one or more sub-problems is the same as that of the (original) problem:
 - i.e. the structure of the problem is inductive and
 - therefore the structure of the solution will be inductive / recursive
- Of course for the recursion to terminate (or *the induction to have a base case*)
 - the sub-problem(s) of the same structure as the original problem
 - must have size(s) that is/are less than the size of the original problem.