Dagens plan

- kort repetisjon av amortisering
- paradigmer for design av algoritmer

Fundamental techniques: DIVIDE AND CONQUER

Divide and conquer is a general metodology for using recurence to design efficient algorithms. It is based on dividing a particular problem into one or more subproblems of the smaller size which are then recursively solved, and then the solutions are “merged” into the solution of the original problem. Exemples:

- binary search
- quick sort
- merge sort

Dynamic programming

Dynamic programming is another technique for designing data structures and algorithms, a bit more difficult to understand than divide-and-conquer. It is a technique to try when it seems that the problem on hand is exponential time, optimization problem. Dynamic programming yields a polynomial time algorithm, usually very easy to code. The problem must have some structure that we can exploit to obtain this simple solution.

- simple subproblems: there has to be a way of breaking the problem into subproblems and defining them with few indices
- subproblem optimization: an optimal solution to global problem must be a composition of optimal subproblem solutions.
- subproblem overlap: optimal solutions to unrelated problems can contain subproblems in common

Eksamples:

- Floyd-Warshals transitive closure algorithm
- matrix chain product
- 0-1 Knapsack problem
- the longest common subsequence problem

Greedy method

Exemples:

- Dijkstra
- Prim
- Kruskal
- Huffman coding

As dynamic programming, the greedy method is applied to optimization problems. In order to solve the problem one proceeds with the sequence of choices, localy optimizing at every step. The method does not always work, buut it works for problems that possess the “greedy property” which is that the global optimum can be found by a series of local optimizations.