

Algorithm design strategies: decrease and conquer

1. Design strategies
2. Decrease and conquer

1. Design strategies

Some algorithm design techniques:

- decrease and conquer (incremental)
- divide and conquer
- transform and conquer
- dynamic programming (solve all subproblems)
- greedy
- randomize

These are design categories, not algorithms themselves.

2. Decrease and conquer strategy

How does it work?

- at start: all input data is unprocessed
- at each iteration part of input data is processed (completely)
- after each iteration, the amount of processed data increases
- algorithm stops when all data is processed

Example

Linear search through array for particular element

Search for 24 in array

17	-6	23	24	-3	24
----	----	----	----	----	----

↑
= 24?

□

Example

Find largest number in numerical array

17	-6	23	24	-3	24
----	----	----	----	----	----

largest = 24
↑
> largest?

□

Example

Insertion sort: a method for sorting elements in a list/array

Sort numbers in array from smallest to largest

start

17	-6	24	-3	24
----	----	----	----	----

goal

-3	-6	17	24	24
----	----	----	----	----

□

Strengths

- clear, intuitive
- easy to implement (small amount of code)
- works well for small amounts of data

Weakness

- often better (=more efficient) alternatives for larger amounts of data

Tämä teos on lisensoitu Creative Commons Nimeä-EiKaupallinen-EiMuutoksia 4.0 Kansainvälinen -lisenssillä. Tarkastele lisenssiä osoitteessa <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

tekijä: Frank Cameron

This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

made by Frank Cameron



