

In the lab-classes this week we experiment with *dynamic programming algorithms*.

Update the environment: In your existing directory CS-242-Algorithms:

```
git checkout master
git pull git://github.com/OKullmann/CS-242-Algorithms.git master
```

Or if this directory (repository) doesn't exist (anymore), resp. you want to create a new repository, create a new clone by

```
git clone git://github.com/OKullmann/CS-242-Algorithms.git
```

In case you created a new clone, by (just) `git pull` you will pull from OKullmann's Github repository in the future. Now change to the subdirectory for week 8:

```
cd CS-242-Algorithms/200910/Week08/
```

Basic setup: We

1. read the code
2. run the code on small examples, to understand the algorithms.

Compilation is done by

```
Week08> make
```

Just one executable is produced:

```
ChangeMaking
```

Comments on the code This week very basic C++ code has been produced. The files are as follows:

1. `ChangeMaking.hpp` provides function `making_change` for computing the array `c` from the coin-values given by array `d` and for the value `N`, while function `pay_out` extracts the solution from a given `c`.
2. `ChangeMaking.cpp` is a simple application for running `making_change` and `pay_out`.

Preparations

1. Determine for the following cases, on paper, an optimal solution just by "guessing":
 - (a) $N = 33, d_1 = 4, d_2 = 7, d_3 = 9$.
 - (b) $N = 122, d_1 = 1, d_2 = 6, d_3 = 9$.
 - (c) $N = 231, d_1 = 1, d_2 = 6, d_3 = 11, d_4 = 34$.
2. Give an example with three coins d_1, d_2, d_3 , where it is guaranteed that the greedy algorithms will always work.

Show your **answers to the postgrads**.

Understanding the Making-Change algorithm Again, on paper, for $N = 10$ and $d_1 = 1$, $d_2 = 3$, $d_3 = 4$ simulate the change-making algorithm, calculating the c-table, and extracting the solution. Show your **simulation to the postgrads**.

Running the application The application is run for example like so:

```
> ./ChangeMaking 35264 45 23 26 21
Optimal number of coins needed: 786
Solution: 781*45 + 2*23 + 2*26 + 1*21 = 35264
```

1. Check all previous computations.
2. Run the example from the lecture, and check the c-table.
3. Run $N = 11$, $d_1 = 3$, $d_2 = 4$, $d_3 = 5$, and extract from the c-table *two* solutions.

Show your **computations to the postgrads**.

Final question Can we execute $N = 10^9$, and $d_1 = 1, \dots, d_{10} = 10$?! *Think first*, before you run it!