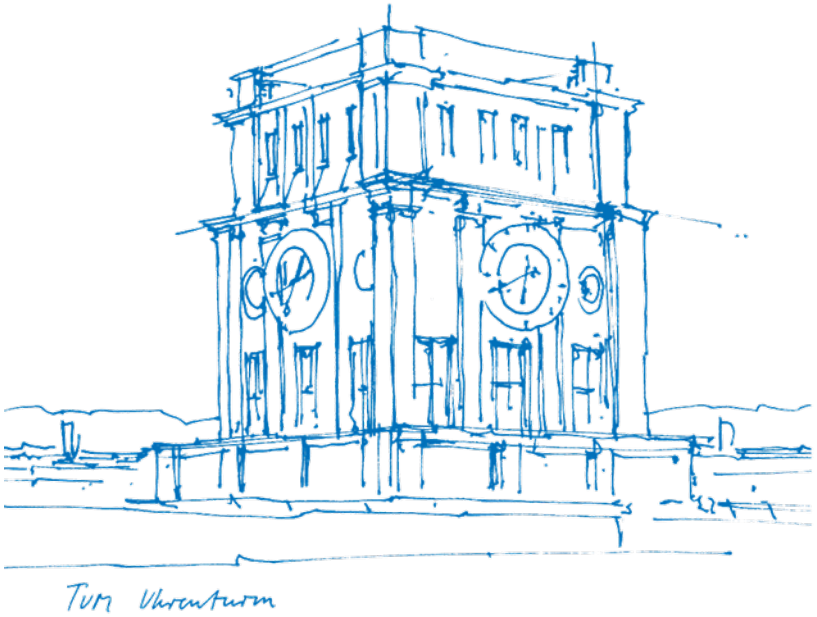
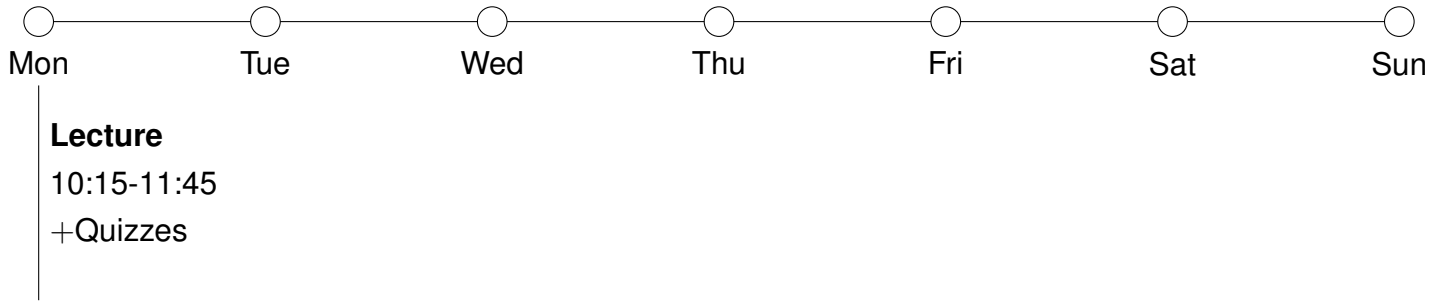
Tutorial 01 [1] - Sequential Programming

Bengisu Elis, M.Sc.
Chair for Computer Architecture and Parallel Systems (CAPS)
Technical University Munich
May 2, 2022

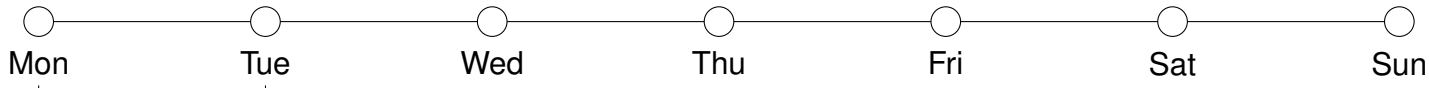


[1] Courtesy of Vincent Bode

Organization



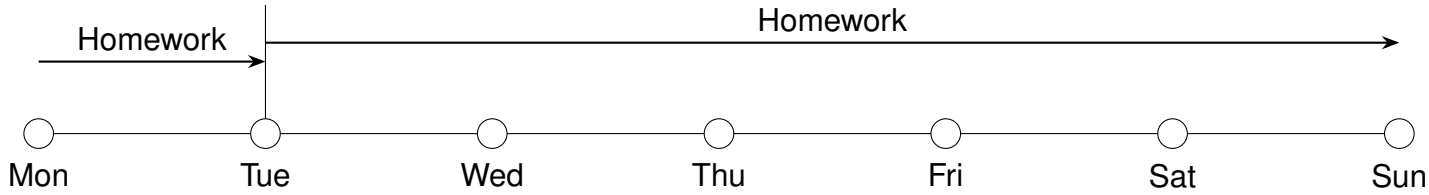
Organization



Lecture
10:15-11:45
+Quizzes

Exercises
12:00-14:00
+Theory
+In Class Exercises
+Quiz
+Homework Intro/Solution
+Q&A
+**Or** shorter demo weeks with only interesting code demos

Organization



Lecture
10:15-11:45
+Quizzes

Exercises
12:00-14:00
+Theory
+In Class Exercises
+Quiz
+Homework Intro/Solution
+Q&A
+**Or** shorter demo weeks with only interesting code demos

Organization

Lecture + Exercise Content

www.moodle.tum.de/course/view.php?id=63406

Programming Assignments

parprog.caps.in.tum.de

https://gitlab.lrz.de/lrr-tum/teaching/parprog/ss2022/published-assignments.git

Questions and Chat

chat.tum.de/channel/ParProg22

The Team

Prof. Martin Schulz
Lecturer

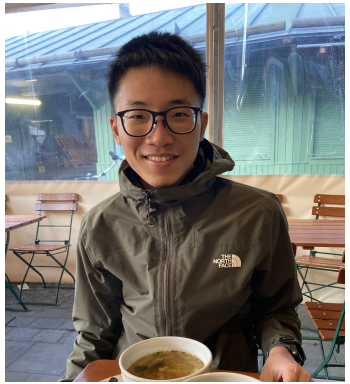
Dennis-Florian Herr
Teaching Assistant

Bengisu Elis
Teaching Assistant

The Team: Tutors

Nathan Mateo Marin Jimenez

Yakub Koray Budanaz



Hua-Ming Huang

Fauzan Ramadhian

Assignments

In class exercises

- Entirely optional
- Relevant to homework
- Can also be solved after session

Assignments

In class exercises

- Entirely optional
- Relevant to homework
- Can also be solved after session

Homework

- In teams with up to 3 members
- Each assignment is doable by 1 person
- Approx. 1/week with 1 week of time to solve
- Grade Bonus
  - 8 homeworks in total. (maybe subject to change)
  - Bonus earned if 75% of all homeworks successful (6 out of 8)
  - Bonus = grade enhancement of exam result by 0.3.
  - Only applies for a passed exam (original grade ≤ 4.0)
- Online submission with almost instant feedback
- We check: correctness, speedup, and plagiarism
- Solutions discussed in next exercise session

Theory

Amdahl's Law & Parallel Efficiency

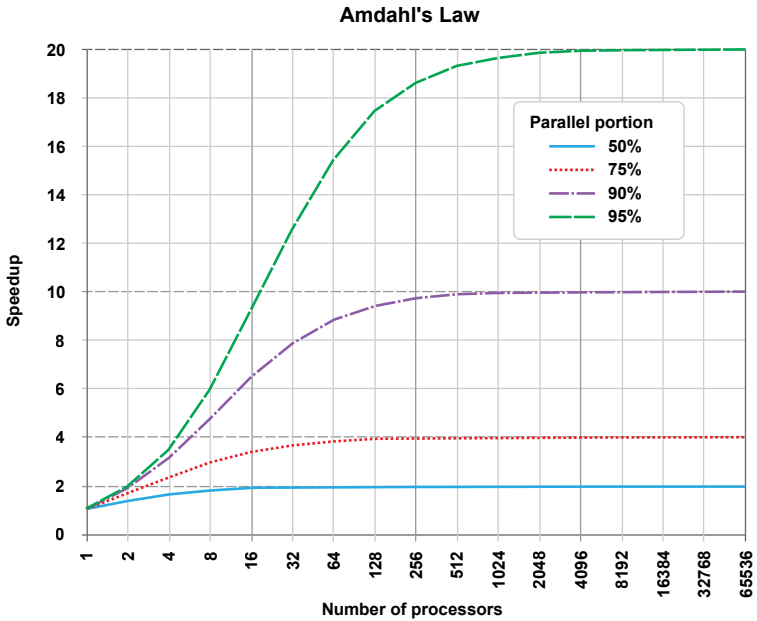


Figure 1: Well life is not perfect ! (Source). But rarely it can be optimized to be perfect.

$$Speedup = \frac{1}{(1-p) + p/N} \quad (1)$$

p = parallelizable part of the program
N = number of processors

Race Conditions

Definition

There is a data dependence from a source statement to a sink statement (sink depends on source), if and only if

1. both statements access the same memory location and at least one of them writes to it.
2. there is a feasible run-time execution path from the source to the sink.

Moodle Exercise

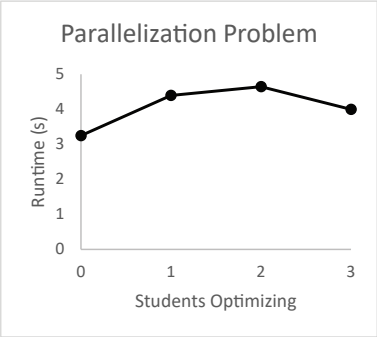
Moodle Quiz

https://www.moodle.tum.de/mod/quiz/view.php?id=2143814

15 minutes reserved for Quiz. Post questions to rocket chat or ask in-class.

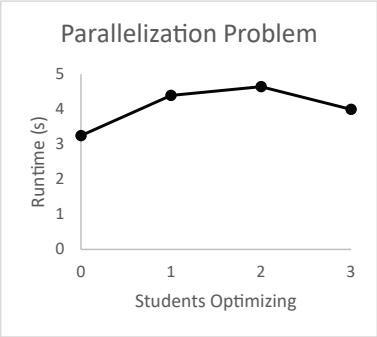
Q1-Q2

Original Runtime	Students Optimizing	Runtime Reduction	Optimized Runtime	Students Parallelizing	Parallel Code	Parallel Runtime	Sequential Code	Sequential Runtime	Total Runtime
10s	0	0%	10s	3	90%	2.25s	10%	1.00s	3.25s
10s	1	20%	8s	2	60%	1.20s	40%	3.20s	4.40s
10s	2	40%	6s	1	30%	0.45s	70%	4.20s	4.65s
10s	3	60%	4s	0	0%	0.00s	100%	4.00s	4.00s



Q1-Q2

Original Runtime	Students Optimizing	Runtime Reduction	Optimized Runtime	Students Parallelizing	Parallel Code	Parallel Runtime	Sequential Code	Sequential Runtime	Total Runtime
10s	0	0%	10s	3	90%	2.25s	10%	1.00s	3.25s
10s	1	20%	8s	2	60%	1.20s	40%	3.20s	4.40s
10s	2	40%	6s	1	30%	0.45s	70%	4.20s	4.65s
10s	3	60%	4s	0	0%	0.00s	100%	4.00s	4.00s



t = original runtime ; z= students parallelizing ; x= students optimizing

new runtime:

$$[t(1-0.3z)+\frac{t\cdot 0.3z}{4}]\cdot(1-0.2x) \qquad (2)$$

$$x+z=3 \text{ for } z\geq 0 \text{ and } x\geq 0$$

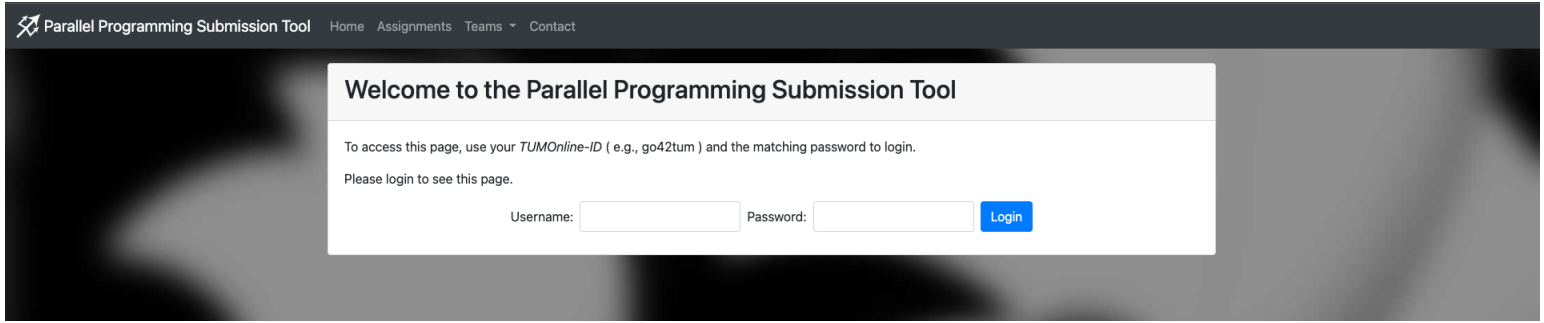
In Class Exercise 1

Where to find the exercise ?

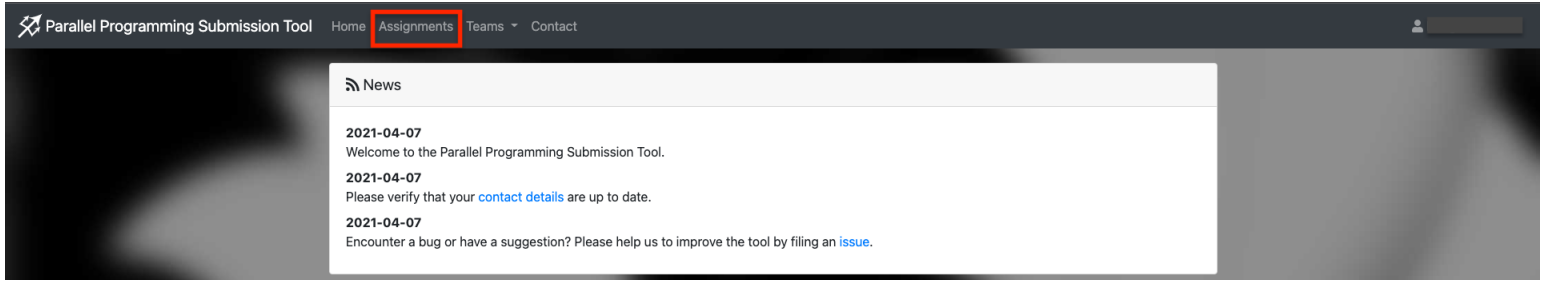
- Go to the following repository to get the exercise:
  https://gitlab.lrz.de/lrr-tum/teaching/parprog/ss2022/published-assignments
- Use git to clone the exercise to your local machine:
  cd your_folder
  git clone https://gitlab.lrz.de/lrr-tum/teaching/parprog/ss2022/published-assignments.git
- You can pull from this repository every time a new exercise is issued
- Go to folder Week_1 for this week's task, you can also find a README.md there

How to submit the solution ?

- Go to : https://parprog.caps.in.tum.de/Submission/login?next=/Submission/home
- Log-in with TUM Online ID



- Click "Assignments" section in the navigation bar:



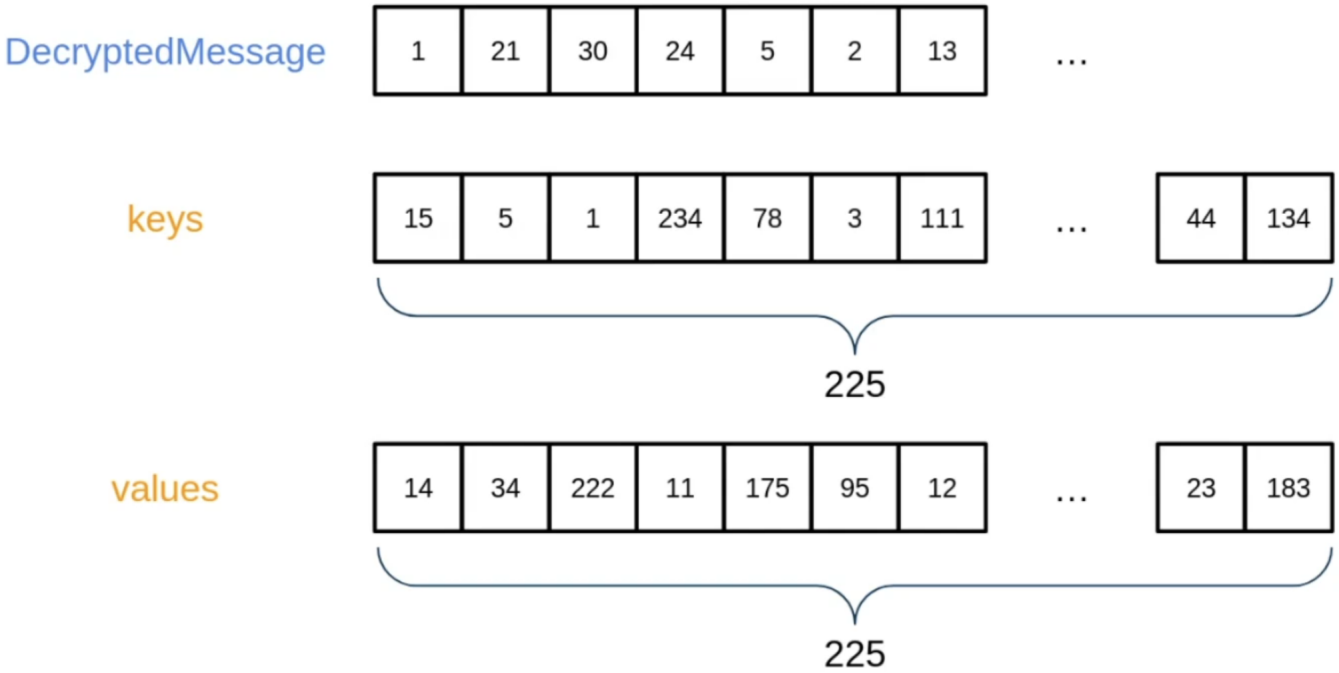
- Upload your code either by copying the code into the textbox or uploading the "student_submission.cpp" file
- After submission you will see your speed up on the same page.

In Class Exercise 1

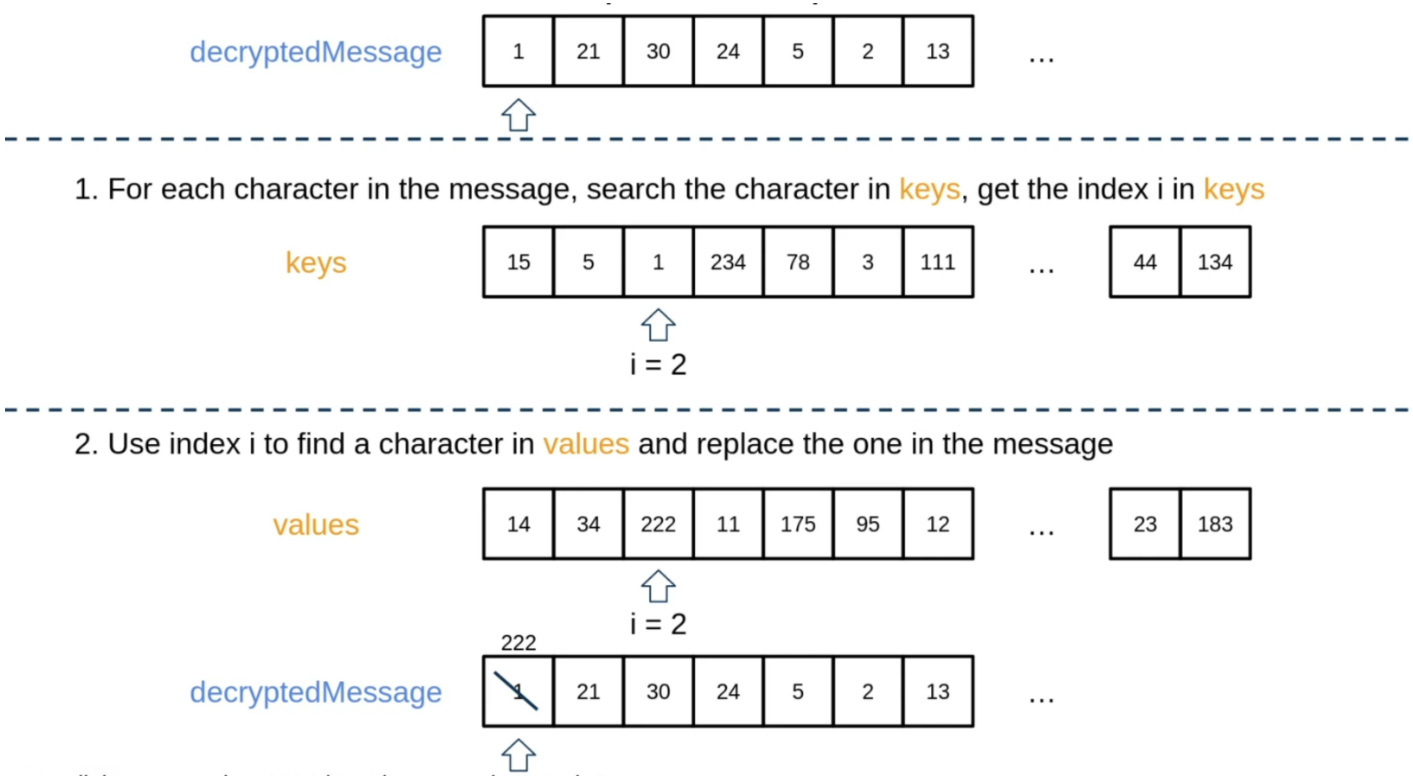
Video Resource

In Class Exercise 1

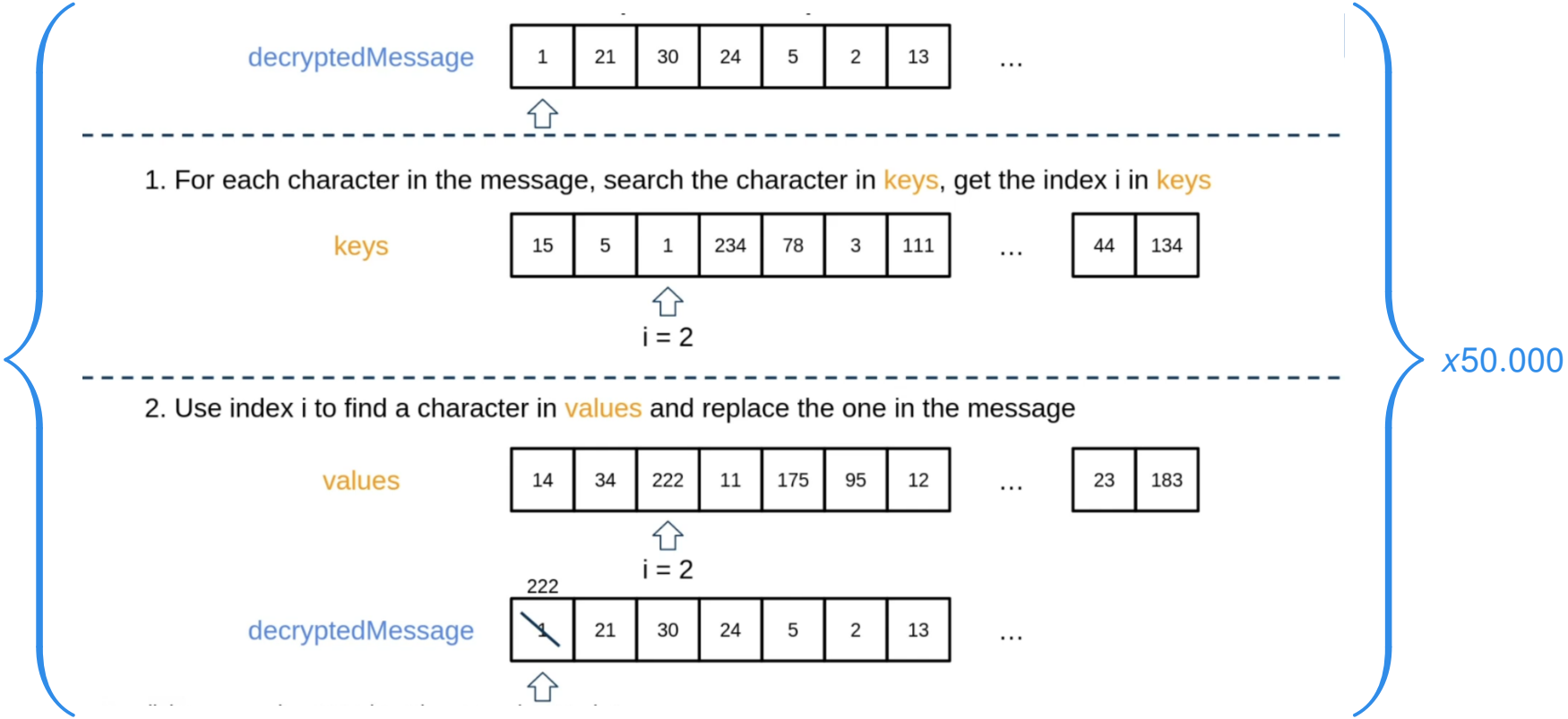
- **Task 1: Optimize the decryption process**
  - decrypt a message DecryptedMessage using dictionary where the keys are stored in keys and the values are stored in values .



In Class Exercise 1



In Class Exercise 1



In Class Exercise 1

10 mins reserved for In Class Exercise 1

Listing 1: Initial Code

```
int index = -1;
for (unsigned int i = 0; i < STRING_LEN; ++i){
    for (int k = 0; k < UNIQUE_CHARACTERS; ++k){
        if (decryptedMessage[i] == keys[k]){
            index = k;
        }
    }
    decryptedMessage[i] = values[index];
}
```

Listing 2: Optimized Code

```
int dict[UNIQUE_CHARACTERS];
for (unsigned int i = 0; i < UNIQUE_CHARACTERS; ++i){
    uint8_t key = keys[i];
    dict[key] = values[i];
}

// use the map instead of searching in originalCharacter
for (unsigned int i = 0; i < STRING_LEN; ++i){
    decryptedMessage[i] = dict[decryptedMessage[i]];
}
```

Homework Assignment 1

Teambuilding

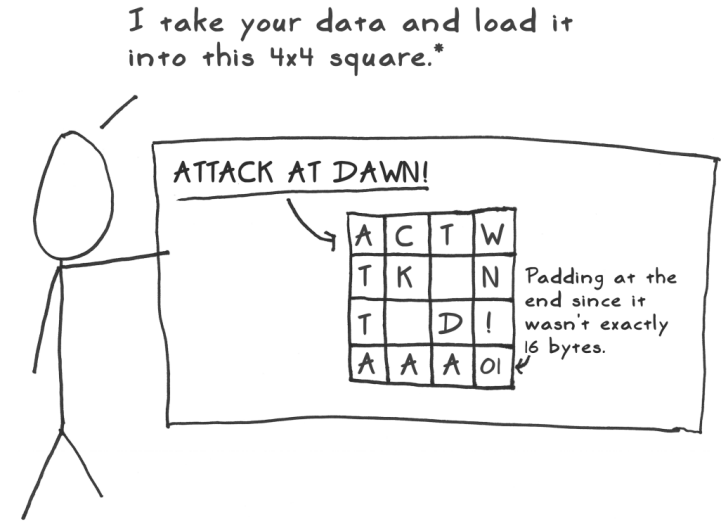
Go find a team!

Up to 3 students per team.

https://www.moodle.tum.de/mod/forum/view.php?id=2149207

Explanation: VV-AES

Step 1:

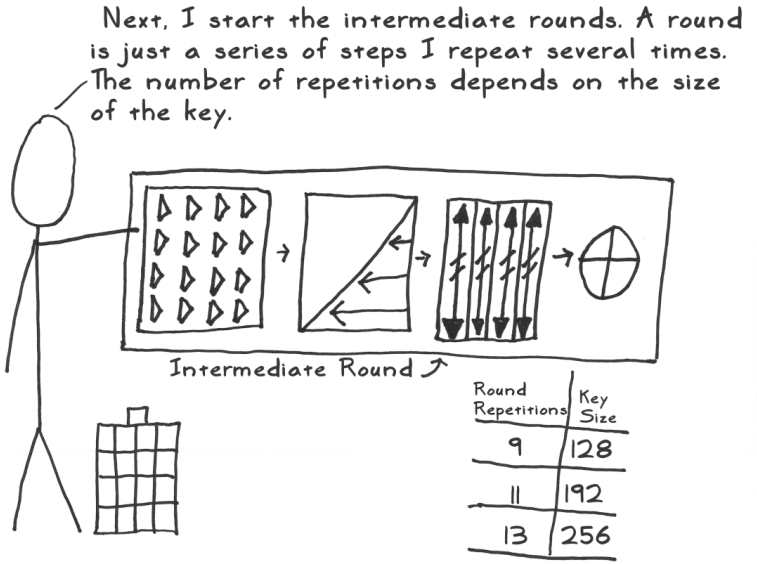


2

[2] Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Explanation: VV-AES

Step 2:

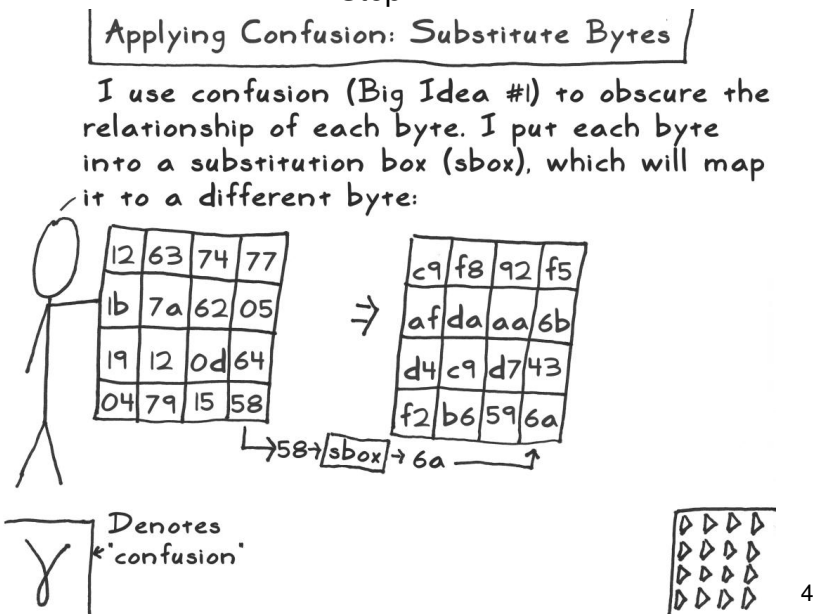


3

[3] Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Explanation: VV-AES

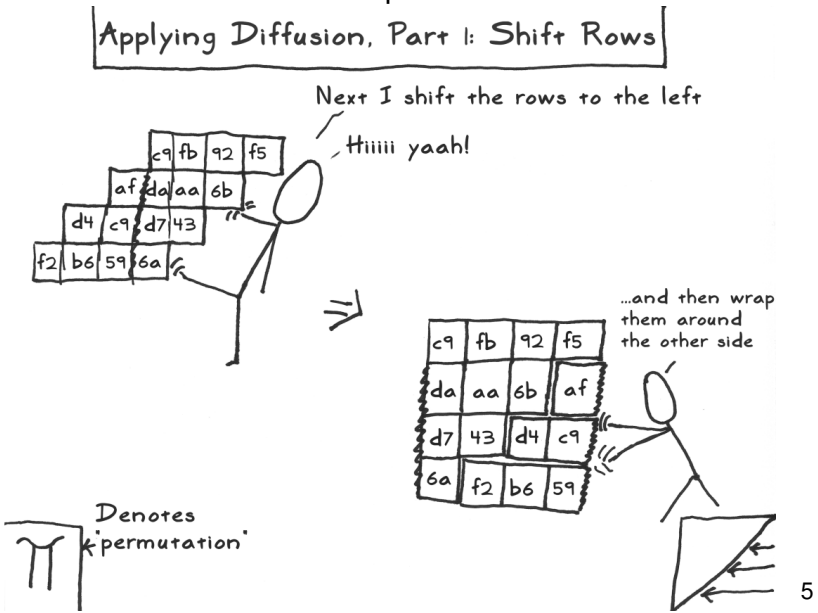
Step 2.1:



[4] Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Explanation: VV-AES

Step 2.2:

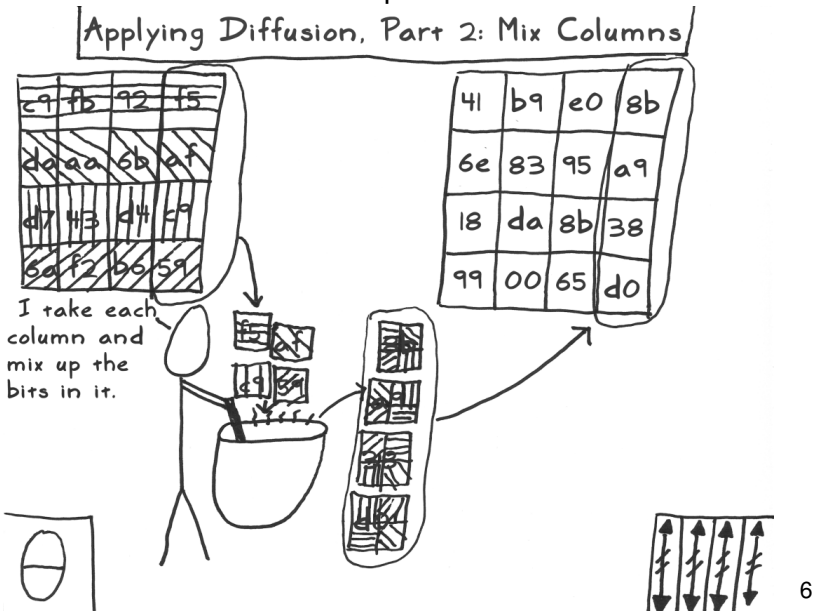


[5]

[5]Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Explanation: VV-AES

Step 2.3:

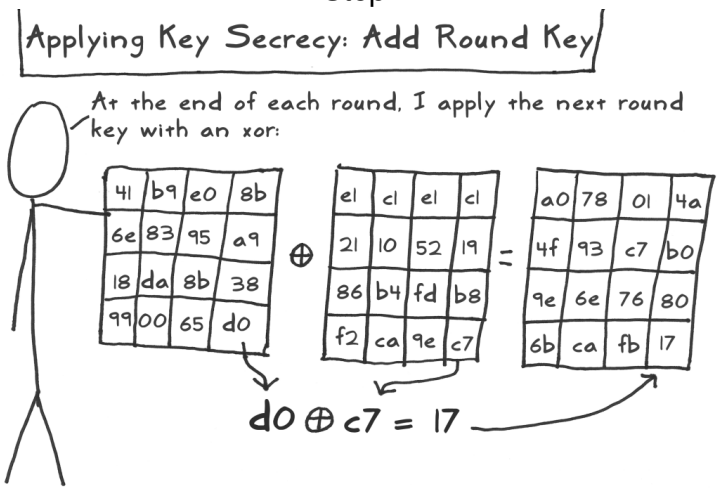


6

[6] Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Explanation: VV-AES

Step 2.4:



7

[7] Courtesy of Jeff Moser's A Stick Figure Guide to the Advanced Encryption Standard (AES)

Assignment: VV-AES

Constraints

- Only a single Thread.
- Locked in a container.
- Eventually killed when it takes too long to do its job.
- Reads input from a file called *stdin*.
- Writes output to a file called *stdout*.
- Can file complaints as they like to *stderr*.
- Can only remember $\sim 10^9$ pieces of information at a time without writing them down.
- Lacks access to the network.
- Any paper trail is burned after every batch of encrypting.

Assignment: VV-AES

Your job: Optimize the Single threaded Workflow

- Speedup of 8x in relation to the provided implementation.
- Note: VV-AES is a slightly simplified version of AES.
  - More information on real AES http://www.moserware.com/2009/09/stick-figure-guide-to-advanced.html.

Hints

Typical performance problems:

- Frequent memory allocation/deallocation
- Unnecessary copying or conversion of data
- Lack of cache awareness
- **Bad algorithms**
- **Reinventing the wheel**

Building and Running the Program

Developing

- Copy “sequential_implementation.cpp” to “student_submission.cpp”
- Write your code in “student_submission.cpp”

Build the program

- Makefile:
  `$> make`

Usage of the program

- Sequential:
  `$> echo $RANDOM | ./sequential_implementation`
- Your solution:
  `$> echo $RANDOM | ./student_submission`
- These programs expect a number to seed random problem generation on `stdin`. Replace the random number with a specific number when you want to re-run a specific problem.

No Linux? No Problem.

You have several options for getting access to a linux environment.

- Install linux in a virtual machine (e.g. VirtualBox)
  - Don't forget to assign multiple cores to the virtual machine
- Use WSL (Windows Subsystem for Linux) on Windows 10
- Use Rechnerhalle
  - Accessible at the workstations or remotely.
  - Remote ssh access: ssh <rbg-id>@lxhalle.in.tum.de
  - Your RBG-Id is the part in front of your @in.tum.de or @ma.tum.de email address.
- Ask the tutors; they will be more than happy to help you.

Recap & Questions

Covered today:

- Amdahl's Law
- Speedup & Parallel Efficiency
- Race Conditions

Questions

This slide is intentionally left blank.