

P-Aufgaben

Altklausuren

Hinweise zur HA



Folien: [go.tum.de/904005](https://go.tum.de/904005)

# P13.02

Polymorphie-Quiz, [Hinweise hier](#)



**Mentimeter**

[menti.com](https://menti.com)

Session Code: 61 35 36



P-Aufgaben

Altklausuren

Hinweise zur HA

# P13.01

P-Aufgaben

Altklausuren

Hinweise zur HA

Threadpools, eigenen Threadpool implementieren

- Konstruktor mit max. Zahl von Threads  
> **public** Threadpool(**int** amountThreads)
- Methode submit, um Aufgaben hinzuzufügen  
> **public** Future submit(Task task)
- Methode shutdown zum beenden des Threadpools  
> **public void** shutdown()

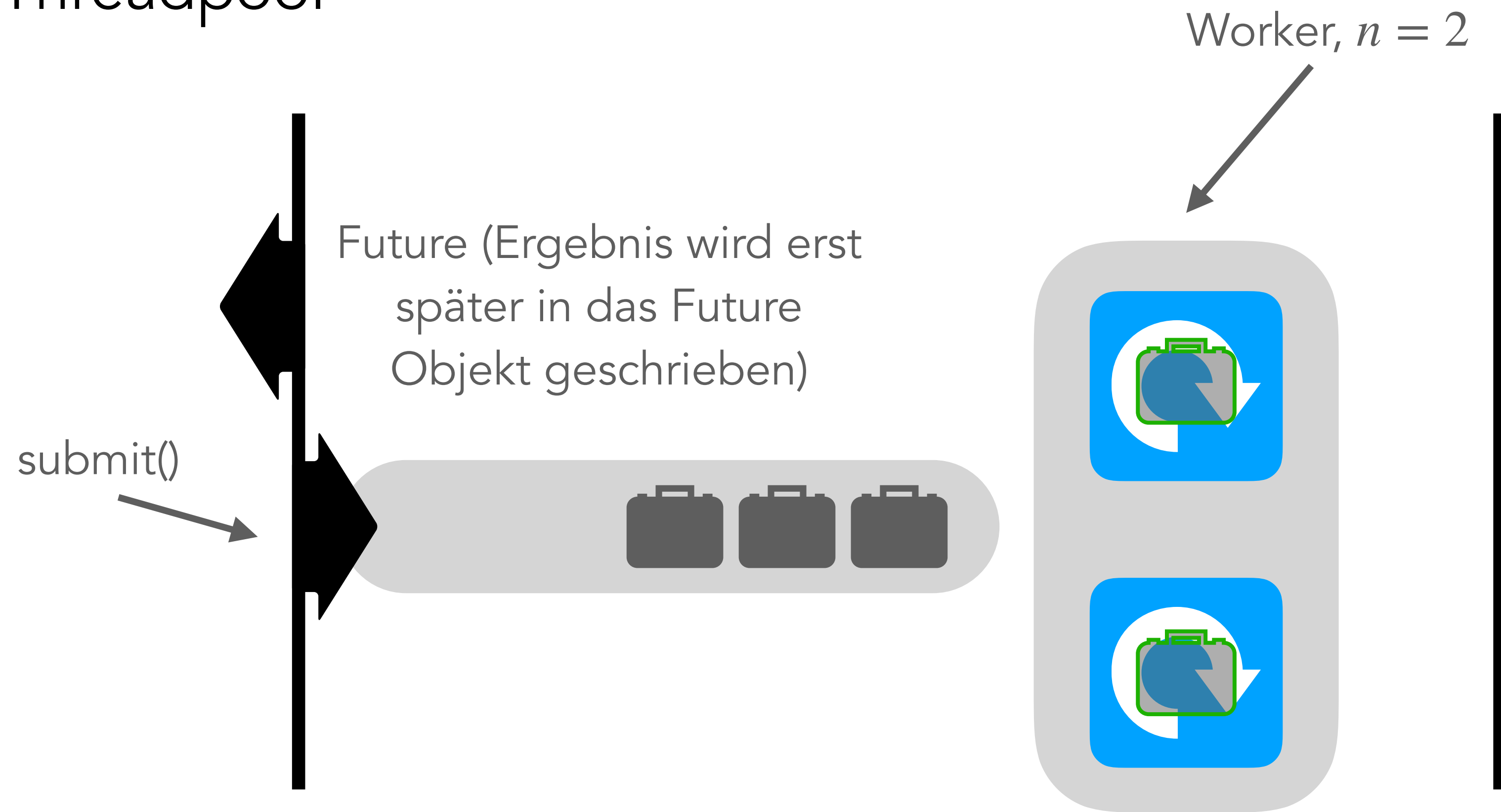
# P13.01

P-Aufgaben

Altklausuren

Hinweise zur HA

## ThreadPool



## P13.01

P-Aufgaben

Altklausuren

Hinweise zur HA

## ThreadPool

<b>Task</b>
- runnable :Runnable
- future :Future

<b>Future</b>
- finished : <b>boolean</b>
+ get() : <b>void</b>
+ finish() : <b>void</b>

<b>ThreadPool</b>
- queue :LinkedBlockingQueue
- running : <b>boolean</b>
- workers :Thread[]
+ ThreadPool(number <b>int</b> )
+ submit(Task t) :Future
+ shutdownNow() : <b>void</b>

Setzt finished auf true  
und benachrichtigt  
wartende Threads

Wartet (Thread), bis  
Ergebnis fertig ist;  
(finished = true)

Im Konstruktor muss bereits der  
ThreadPool gestartet werden; dafür  
bieten sich 'worker' (TODO) Runnables  
auf den 'worker' -Threads an

# Altklausuren

P-Aufgaben

Altklausuren

Hinweise zur HA

Auch auf der Website des Tutoriums verlinkt.

Jahr	Download
WS2017E	<a href="https://www.moodle.tum.de/pluginfile.php/1901807/mod_folder/content/0/WS17.pdf?forcedownload=1">https://www.moodle.tum.de/pluginfile.php/1901807/mod_folder/content/0/WS17.pdf?forcedownload=1</a>
WS2018E	<a href="https://www.moodle.tum.de/pluginfile.php/1901807/mod_folder/content/0/WS18.pdf?forcedownload=1">https://www.moodle.tum.de/pluginfile.php/1901807/mod_folder/content/0/WS18.pdf?forcedownload=1</a>
WS2017R	<a href="https://www.moodle.tum.de/pluginfile.php/1747577/mod_folder/content/0/info1_wdh.pdf?forcedownload=1">https://www.moodle.tum.de/pluginfile.php/1747577/mod_folder/content/0/info1_wdh.pdf?forcedownload=1</a>
WS2012R	<a href="http://www14.in.tum.de/lehre/2016WS/info1/info1_klausur.pdf">http://www14.in.tum.de/lehre/2016WS/info1/info1_klausur.pdf</a>
WS2016R	<a href="http://www14.in.tum.de/lehre/2016WS/info1/examWdh.pdf">http://www14.in.tum.de/lehre/2016WS/info1/examWdh.pdf</a>
WS2016E	<a href="http://www14.in.tum.de/lehre/2016WS/info1/exam.pdf">http://www14.in.tum.de/lehre/2016WS/info1/exam.pdf</a>

Offiziell

Was man  
noch so  
im Internet  
findet...

Die Altklausuren dienen nur der Orientierung. Eine geänderte ÜL hat meist auch Änderungen im Klausurlayout zur Folge. Mit den immer wiederkehrenden Aufgaben kann aber gerechnet werden. R: Retake, E: Endterm

# Hinweise zur HA

Die W13H01 gibt *viele Punkte und ist nicht schwer*. Viele Konzepte haben wir bereits gesehen.

- CSV Dateien Einlesen & HTML Dateien erstellen: P10.03
- Socket Kommunikation: P11.03
  - > jetzt HTTP-Requests (auch Strings) und keine Textnachrichten
- Sicherer paralleler Zugriff auf Daten: P12.02

Die oben genannten Konzepte werden angewandt - was sich ändert sind die Daten.

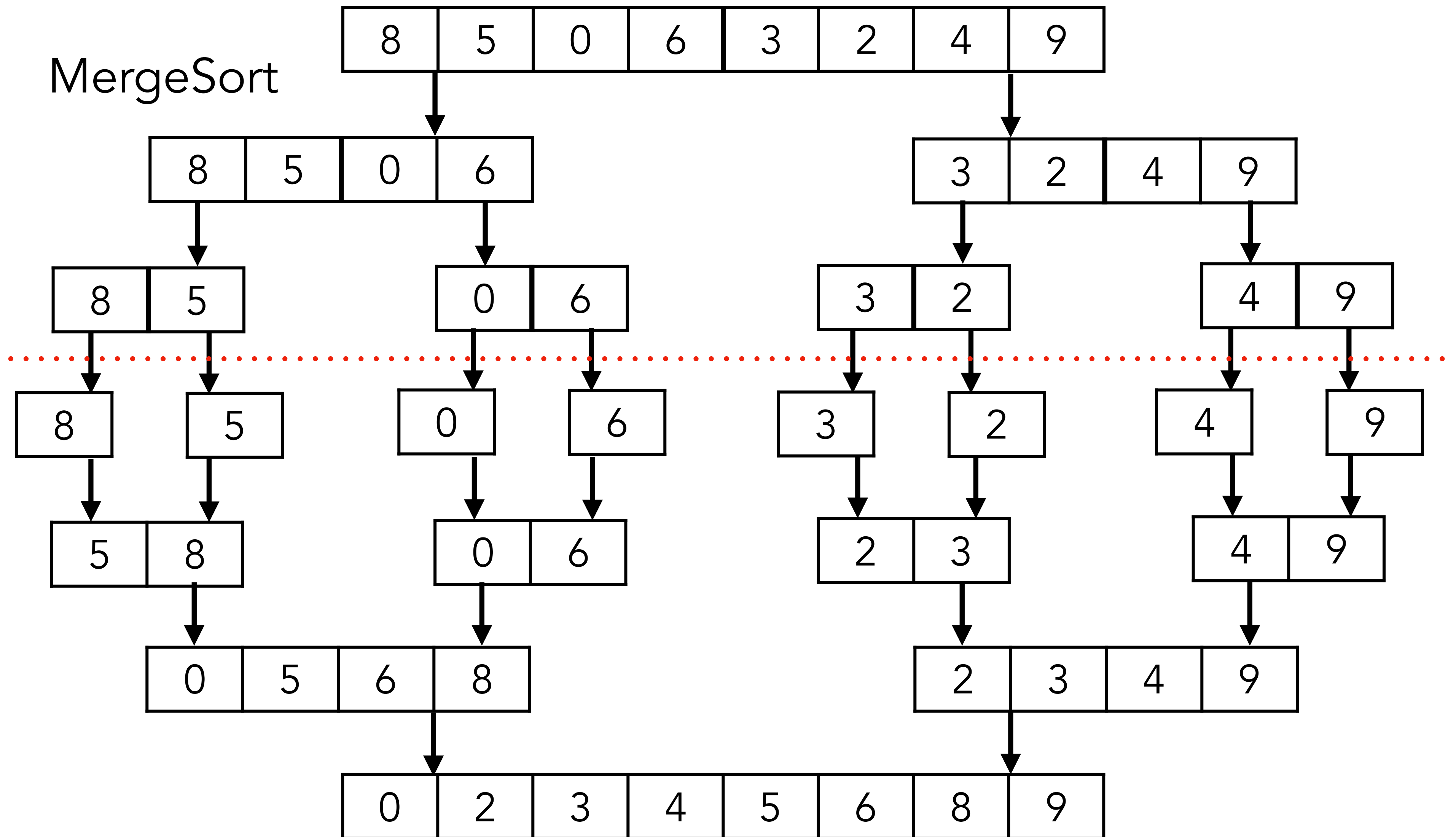
# Hinweise zur HA

P-Aufgaben

Altklausuren

Hinweise zur HA

MergeSort



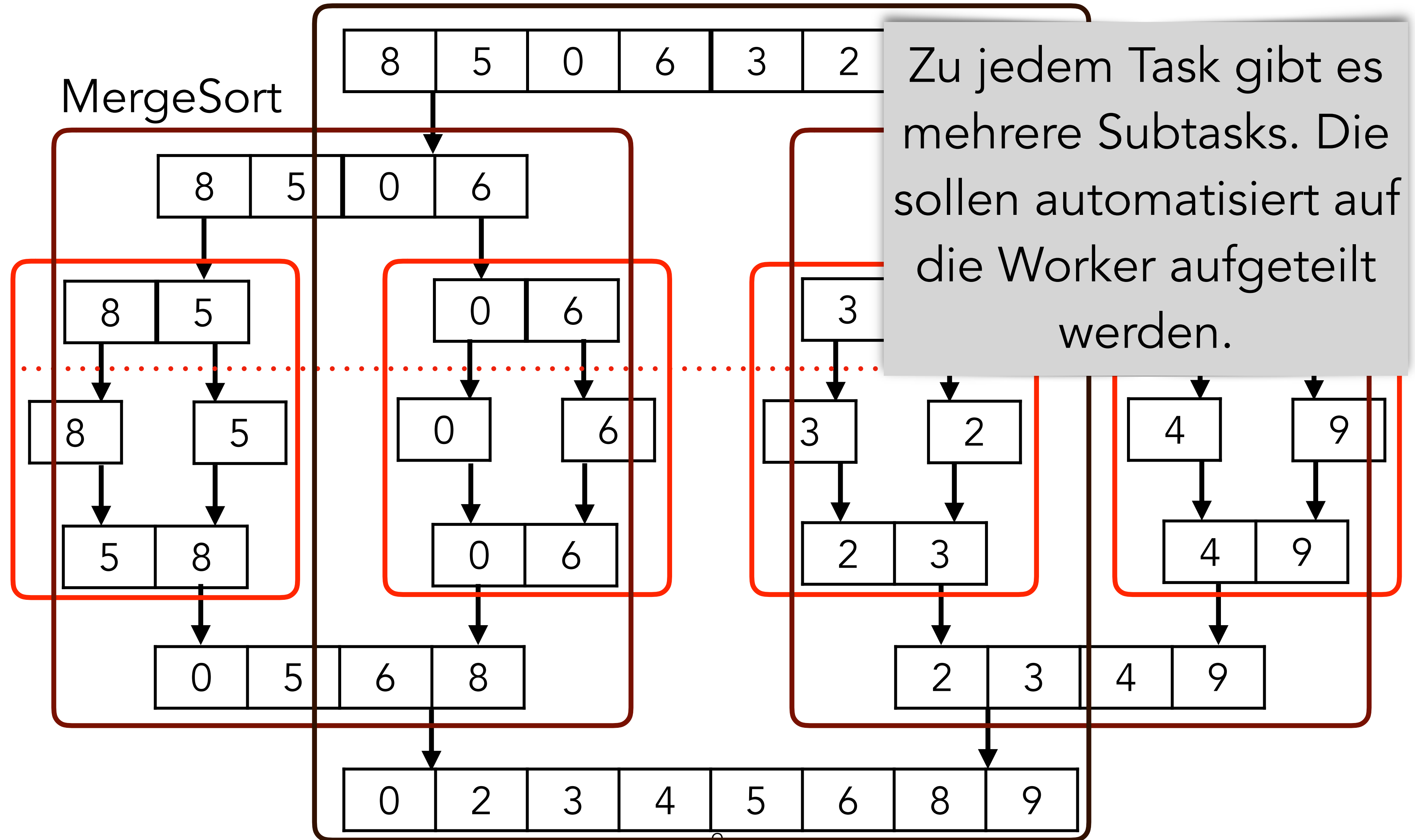


# Hinweise zur HA

P-Aufgaben

Altklausuren

Hinweise zur HA

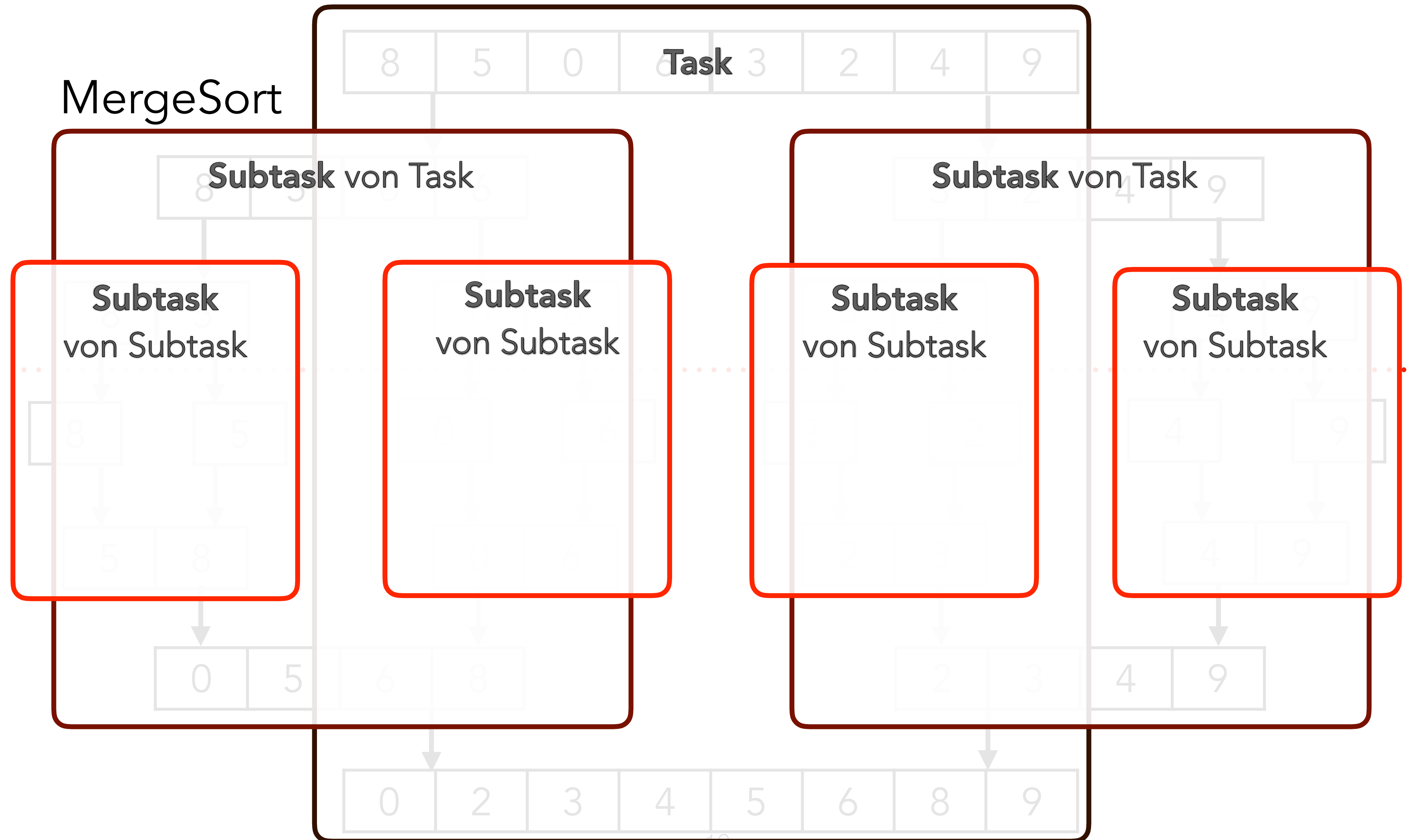


# Hinweise zur HA

P-Aufgaben

Altklausuren

Hinweise zur HA



# Hinweise zur HA

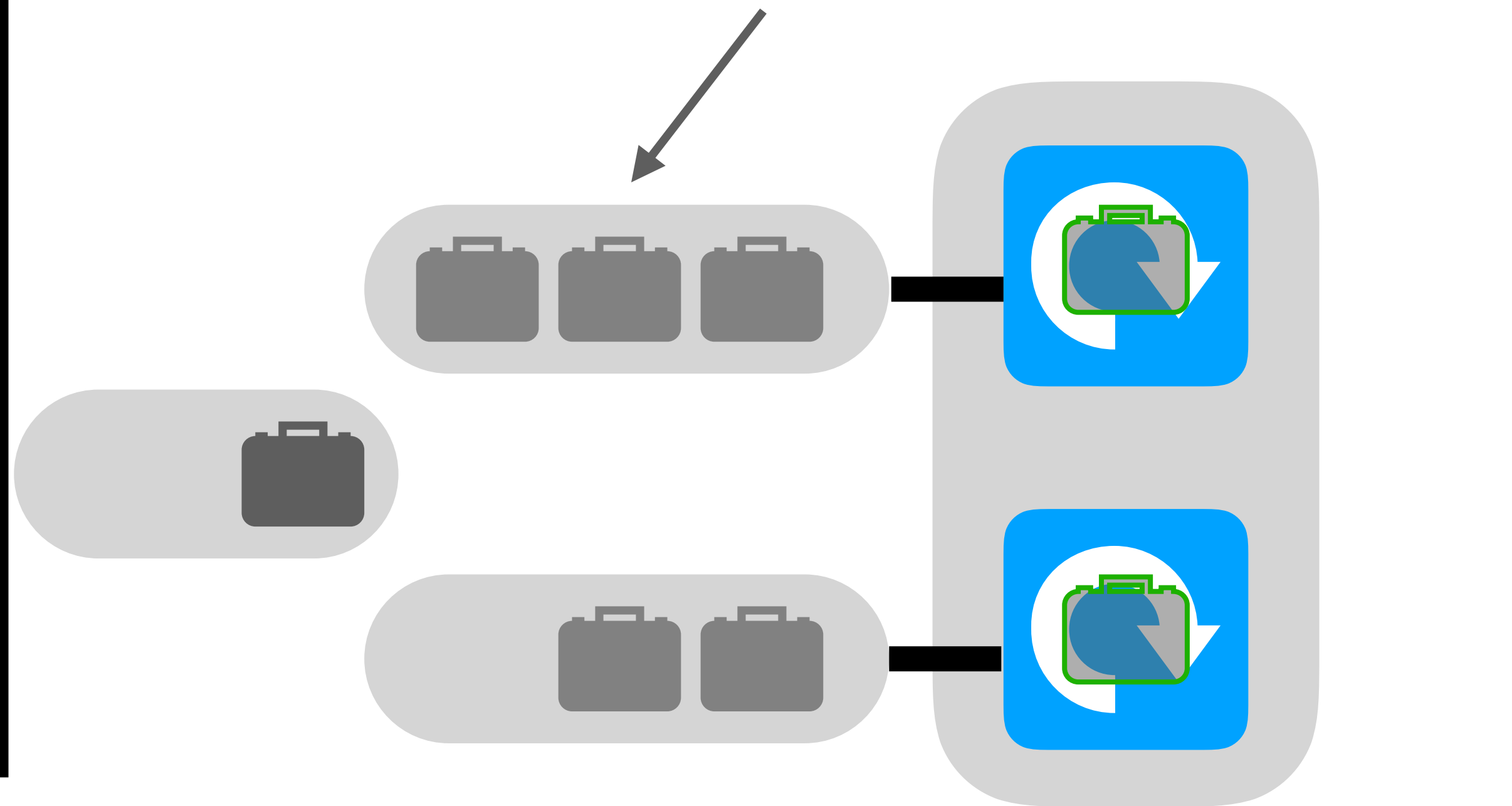
P-Aufgaben

Altklausuren

Hinweise zur HA

## ForkJoinPool

Jeder Worker hat eine zusätzliche eigene Queue, in die er Subtasks einreicht.



# Hinweise zur HA

P-Aufgaben

Altklausuren

Hinweise zur HA

ForkJoinPool, Codebeispiel

```
public class MyTask extends
RecursiveTask<Integer> {
    private final int c;
    public MyTask(int c) { this.c = c; }
    @Override
    protected Integer compute() {
        if (this.current <= 1) return 1;
        else {
            MyTask mt = new MyTask(this.c - 1);
            mt.fork();
            return mt.join() * this.c;
        }
    }
}
```