Übungen Betriebssysteme (BS)

U4 – Speicherverwaltung

https://moodle.tu-dortmund.de/course/view.php?id=34604

Peter Ulbrich

peter.ulbrich@tu-dortmund.de

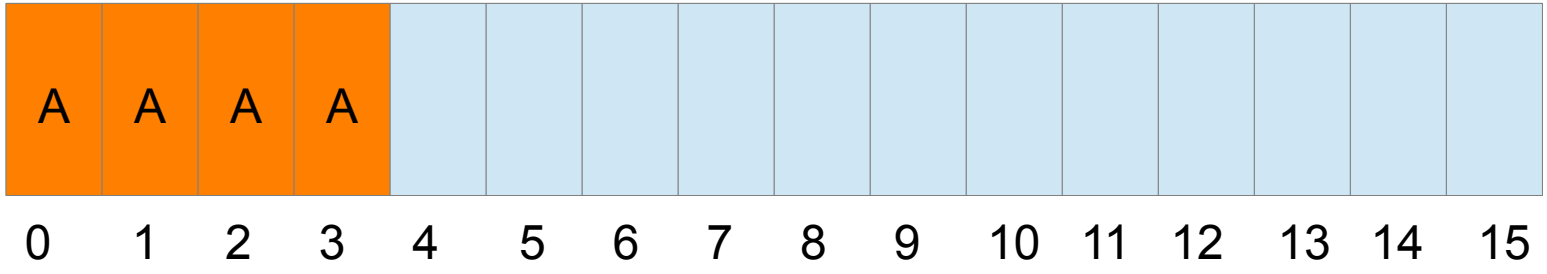
https://sys.cs.tu-dortmund.de/EN/People/ulbrich/

technische universität
dortmund

arbeitsgruppe
systemsoftware

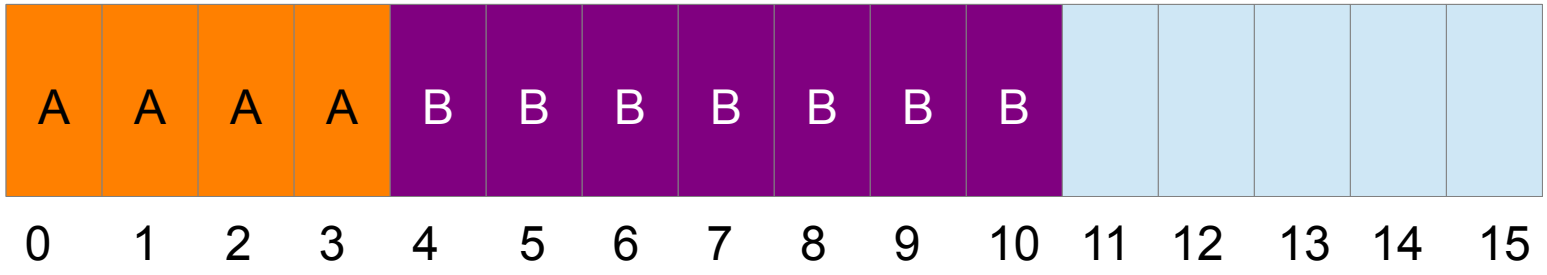
Theoriefrage 1- Best Fit

1) A fordert 4 MB an



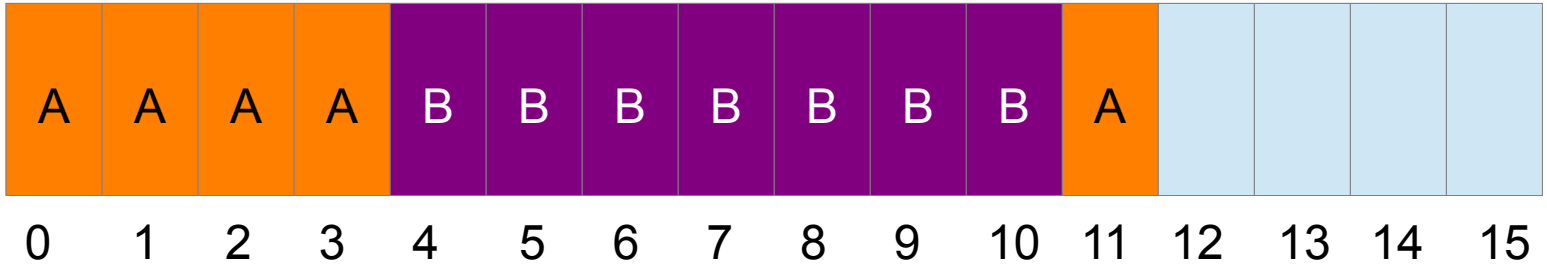
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2)B fordert 7 MB an



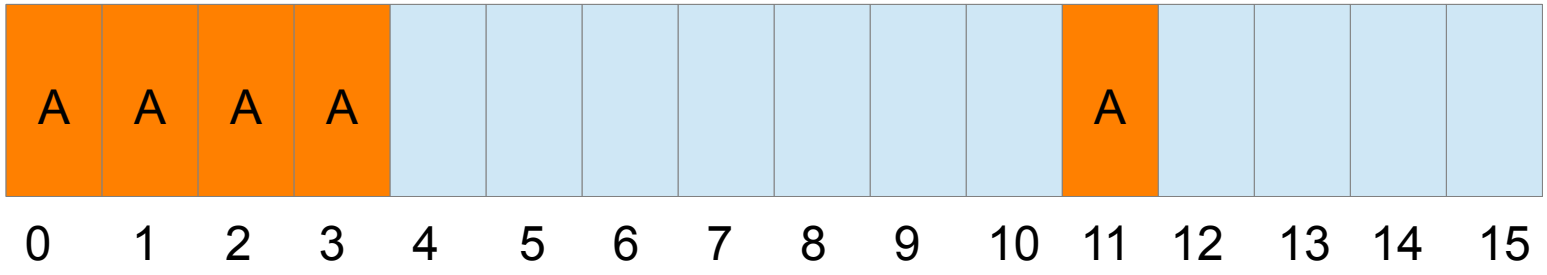
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an



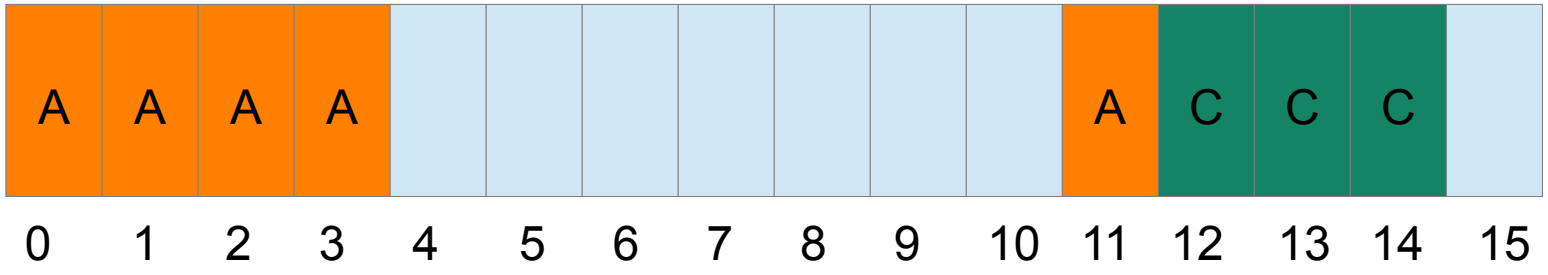
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei



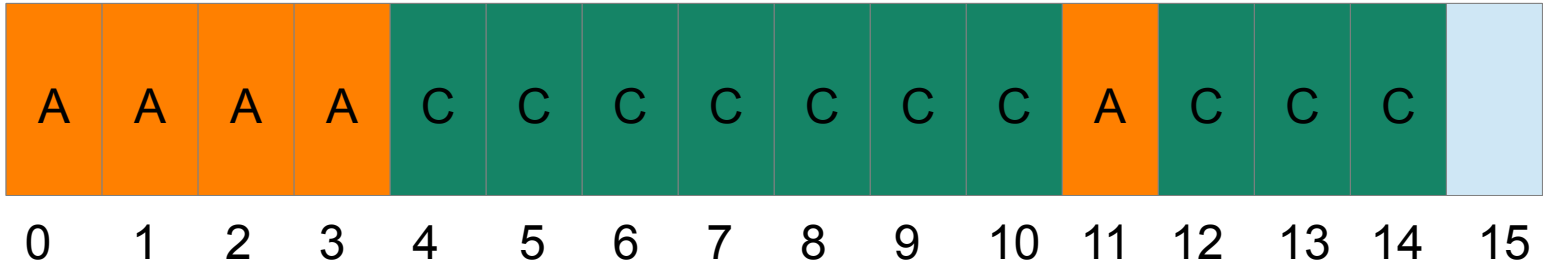
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an



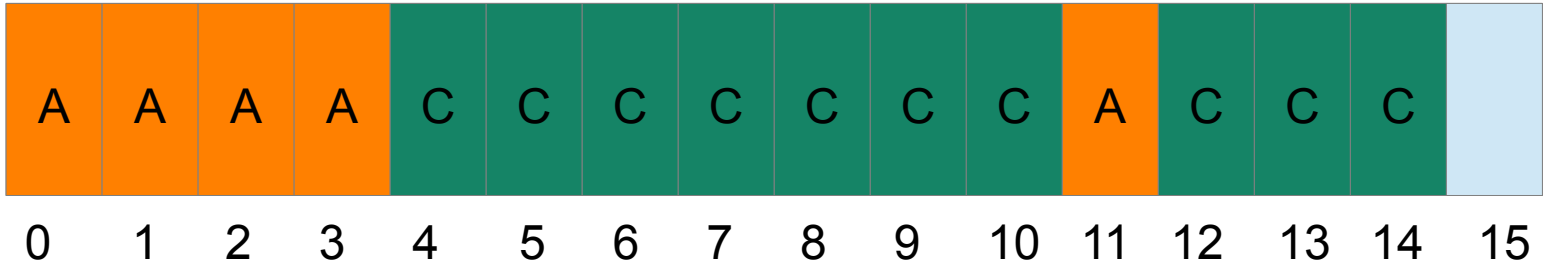
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an
6) C fordert 7 MB an



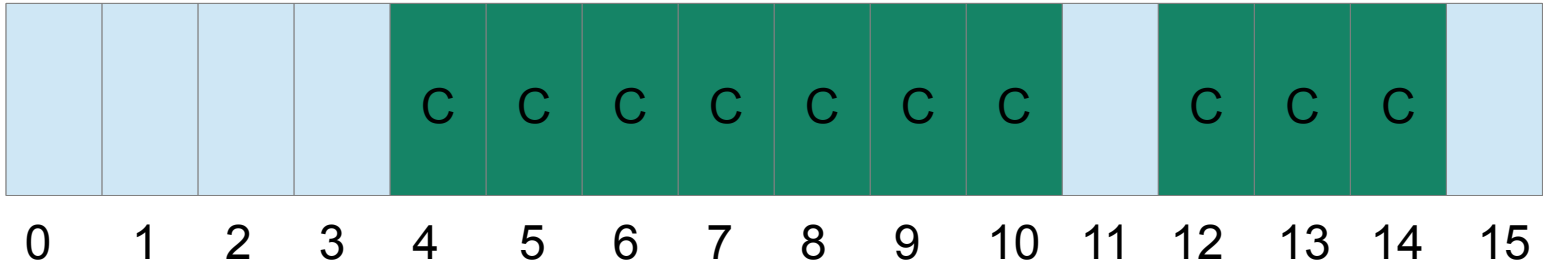
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an
6) C fordert 7 MB an
7) D fordert 2 MB an. -> Keine 2 MB frei!



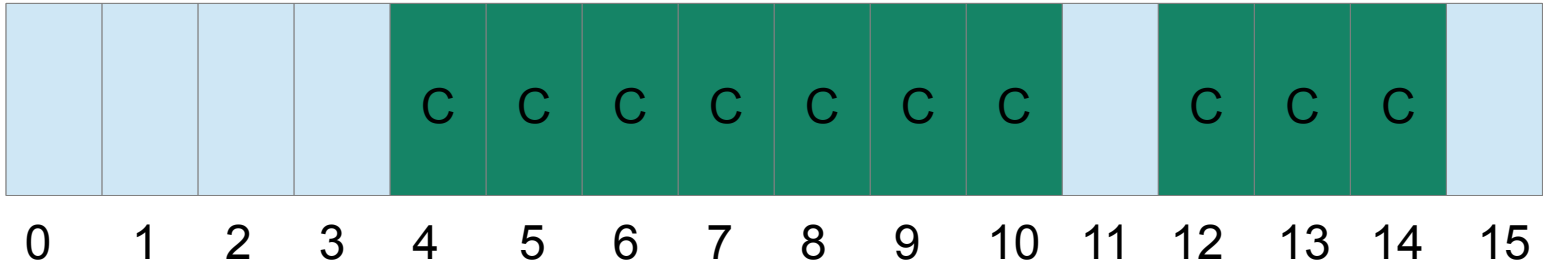
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an
6) C fordert 7 MB an
7) D fordert 2 MB an. -> Keine 2 MB frei!
8) A gibt seinen gesamten Speicher frei



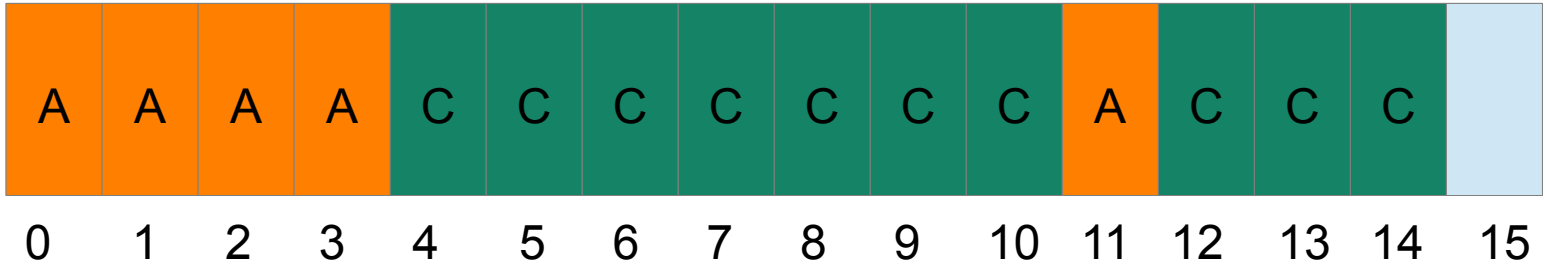
Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an
6) C fordert 7 MB an
7) D fordert 2 MB an. -> Keine 2 MB frei!
8) A gibt seinen gesamten Speicher frei
9) D fordert 6 MB an -> Keine 6 MB *zusammenhängend* frei!



Theoriefrage 1- Best Fit

1) A fordert 4 MB an
2) B fordert 7 MB an
3) A fordert 1 MB an
4) B gibt seinen gesamten Speicher frei
5) C fordert 3 MB an
6) C fordert 7 MB an

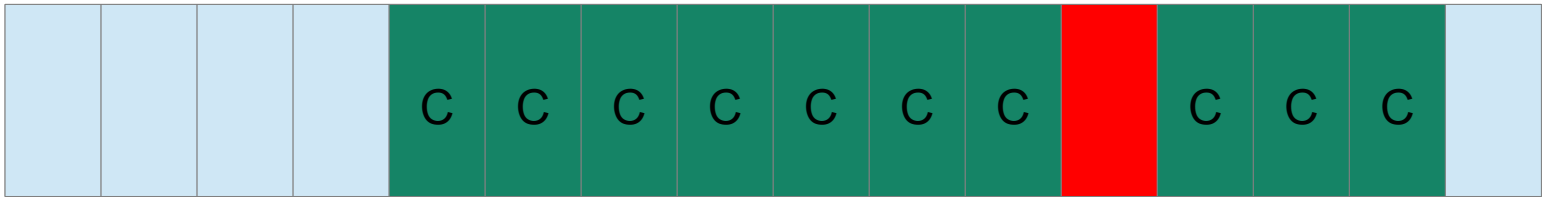


Theoriefrage 2- Verschnitt

Erklären Sie anhand der Best Fit-Strategie aus Aufgabe 1, um welche Art von Verschnitt es sich handelt und wie er zustandekommt.

Theoriefrage 2- Verschnitt

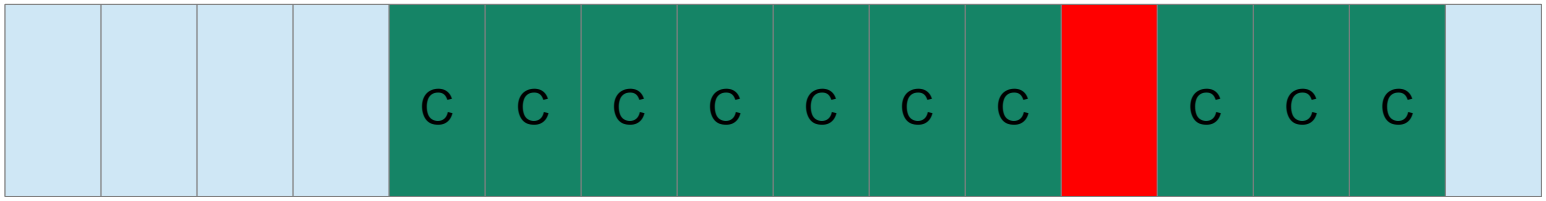
Erklären Sie anhand der Best Fit-Strategie aus Aufgabe 1, um welche Art von Verschnitt es sich handelt und wie er zustandekommt.



Theoriefrage 2- Verschnitt

Erklären Sie anhand der Best Fit-Strategie aus Aufgabe 1, um welche Art von Verschnitt es sich handelt und wie er zustandekommt.

- Es handelt sich um externen Verschnitt.
- Kleine, ungenutzte und meist unbenutzbare Speicherbereiche, die entstehen, wenn eine Lücke nicht komplett ausgenutzt wird.



Theoriefrage 2- Verschnitt

In den folgenden Aufgaben wird das Buddy-Verfahren betrachtet. Kann es bei diesem Verfahren auch zu Verschnitt kommen?

Theoriefrage 2- Verschnitt

In den folgenden Aufgaben wird das Buddy-Verfahren betrachtet. Kann es bei diesem Verfahren auch zu Verschnitt kommen?

- Ja, interner Verschnitt.
- Durch das Aufrunden der angefragten Speichermenge auf die nächste Zweierpotenz wir gegebenfalls zu viel reserviert.

Theoriefrage 2- Verschnitt

In den folgenden Aufgaben wird das Buddy-Verfahren betrachtet. Kann es bei diesem Verfahren auch zu Verschnitt kommen?

- Ja, interner Verschnitt.
- Durch das Aufrunden der angefragten Speichermenge auf die nächste Zweierpotenz wir gegebenfalls zu viel reserviert.

- 3MB angefragt -> 4 MB reserviert.
- 5 MB angefragt -> 8 MB reserviert.

Theoriefrage 3 - Baumstruktur

Warum nutzt Buddy eine Baumstruktur?

Theoriefrage 3 - Baumstruktur

Warum nutzt Buddy eine Baumstruktur?

- Erleichtert die Berechnung der Speicherbereiche / Adressen

Theoriefrage 3 - Baumstruktur

Warum nutzt Buddy eine Baumstruktur?

- Erleichtert die Berechnung der Speicherbereiche / Adressen
  - Ein Knoten -> genau ein Speicherbereich
  - Kinder -> jeweils der halbe Speicherbereich
  - Blatt -> ein Speicherblock

Theoriefrage 3 - Baumstruktur

Warum nutzt Buddy eine Baumstruktur?

- Erleichtert die Berechnung der Speicherbereiche / Adressen
  - Ein Knoten -> genau ein Speicherbereich
  - Kinder -> jeweils der halbe Speicherbereich
  - Blatt -> ein Speicherblock
- Ein Baum ist effizienter durchsuchbar als eine Liste

Theoriefrage 3 - Baumstruktur

In der Programmieraufgabe wird ein Baum der Höhe 7 verwendet, d.h. unterhalb der Wurzel befinden sich sechs Ebenen mit Knoten. Was würde bei gleichbleibender Speichergröße eine abweichende Wahl der Baumhöhe bewirken?

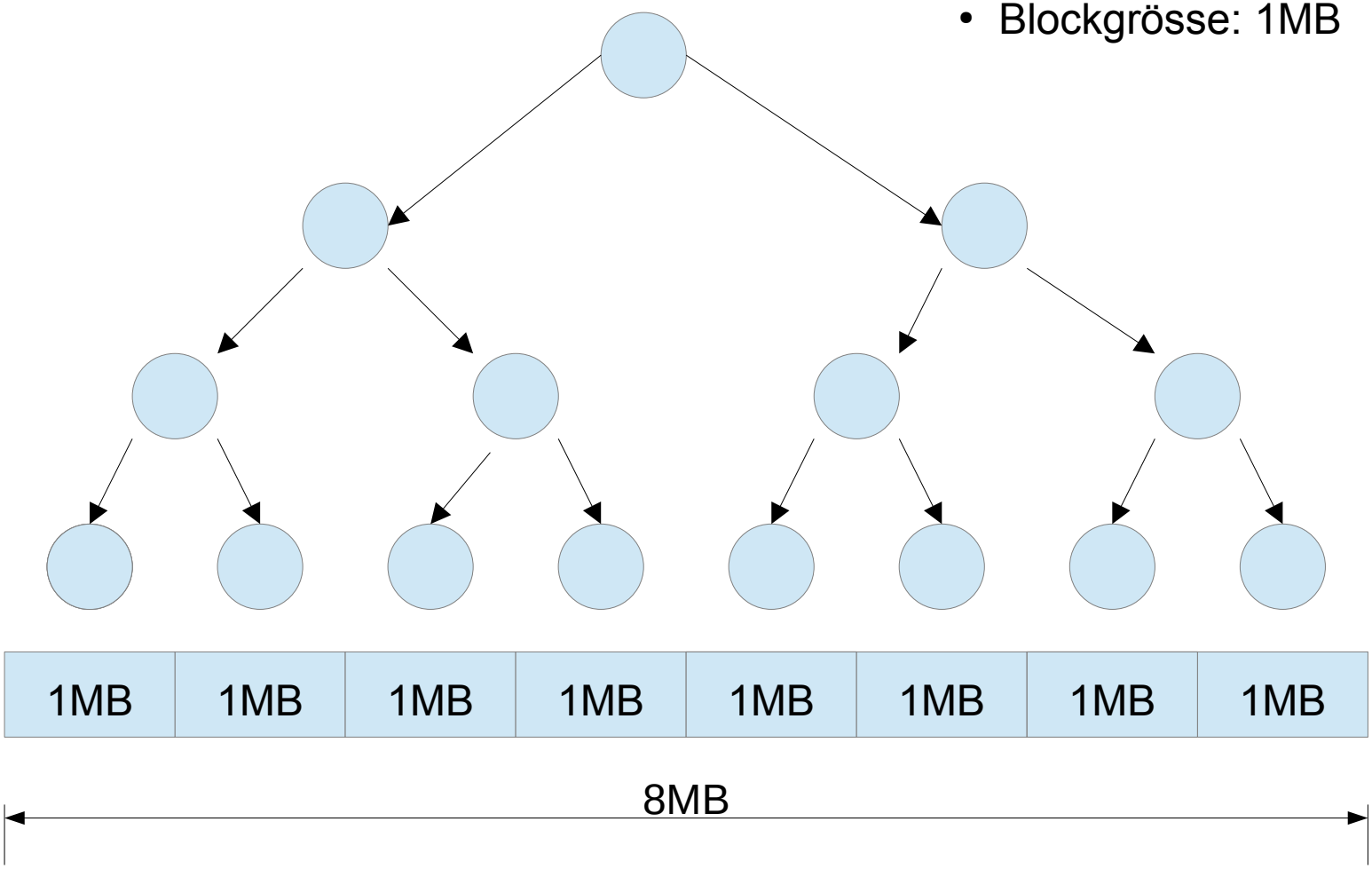
Theoriefrage 3 - Baumstruktur

In der Programmieraufgabe wird ein Baum der Höhe 7 verwendet, d.h. unterhalb der Wurzel befinden sich sechs Ebenen mit Knoten. Was würde bei gleichbleibender Speichergröße eine abweichende Wahl der Baumhöhe bewirken?

- Je höher der Baum, desto kleinere Speicherblöcke können verwaltet werden

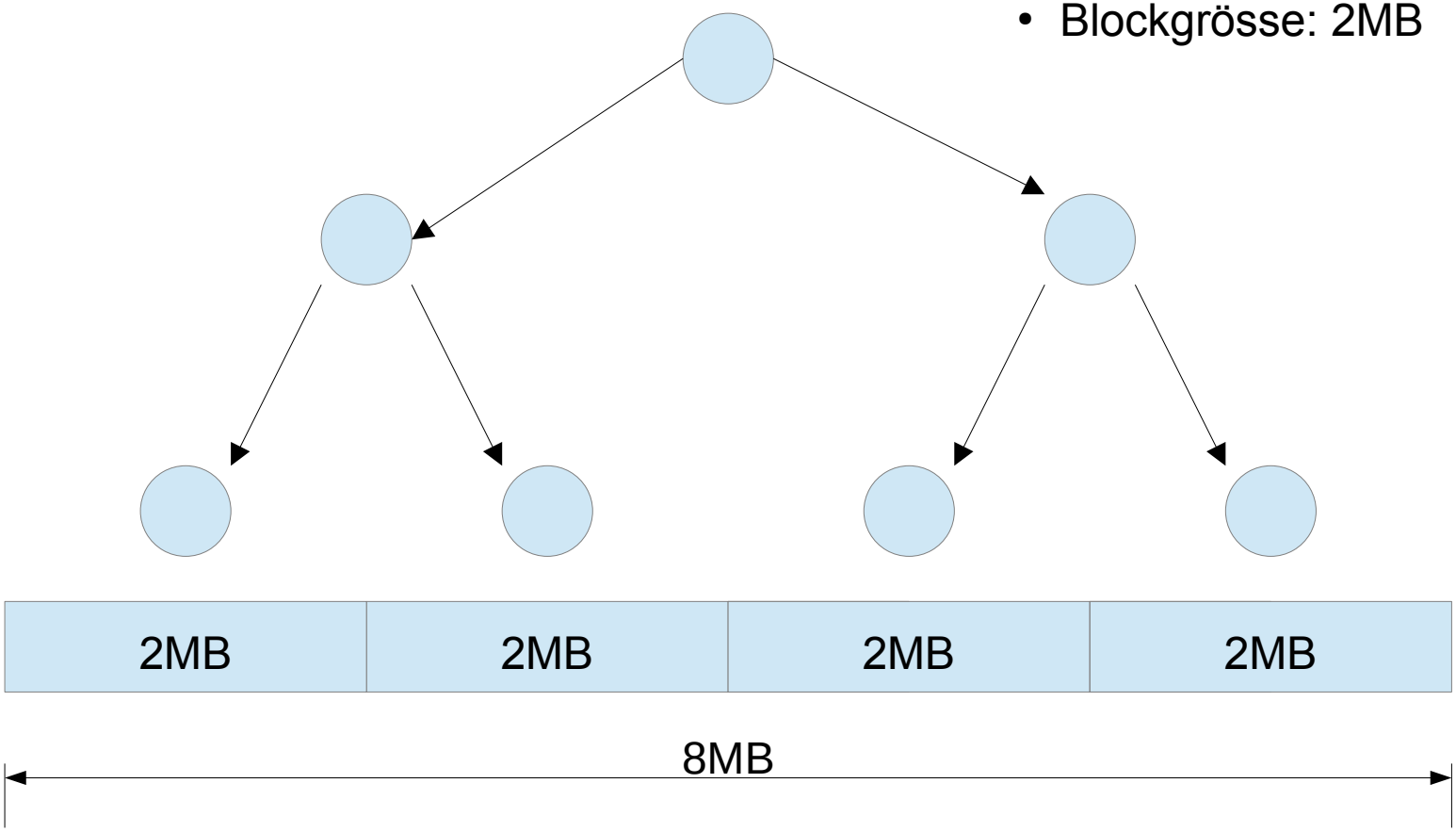
Baumstruktur Beispiel

- Höhe: 4
- Blockgrösse: 1MB



Baumstruktur Beispiel

- Höhe: 3
- Blockgrösse: 2MB



Programmieraufgabe a)

```
if (addr == NULL){
    return;
} // Fehlerbehandlung
node_t node = bst_root();

// Solange noch kein belegter Speicherbereich mit
// passender Startadresse gefunden wurde...
while ((node_start_addr(node) != addr) || (*node <= 0)) {
        // Wo geht es lang?
        if (addr < node_split_addr(node)) {
            node = bst_left(node);
        } else {
            node = bst_right(node);
        }
        if (node == NULL) {
            /* Fehlerbehandlung */
        }
}
// Markiere den gefundenen Knoten als frei
*node = NODE_FREE;
```

Programmieraufgabe b)

```
void *buddy_alloc(char request_id, size_t size) {
   if (size == 0) {
      return NULL;
    }
   if (request_id <= 0) {
      /* Fehlerbehandlung */
   }
   // Rekusiv (beginnend bei der Wurzel) nach einem passenden Speicherbereich
   // suchen
   return recursive_alloc(bst_root(), request_id, size_to_chunks(size));
}
```

Programmieraufgabe b)

```
static void *recursive_alloc(node_t node, char request_id, size_t
size) {
    // ...
   // Abbruch bei node == NULL oder *node > 0

   // Ist der Speicherbereich zu klein?
   if (node_width_chunks(node) < size) {
      return NULL;
   }

    // Ist der Knoten frei und hat er die richtige Größe?
   if ((*node == NODE_FREE) && (node_width_chunks(node) == size)) {
      // Knoten belegen ...
      *node = request_id;
      // ... und die Adresse zurückgeben
      return node_start_addr(node);
   }

   // Ansonsten rekursiv bei den Kindern weiterversuchen
    addr = recursive_alloc(bst_left(node), request_id, size);
   if (addr != NULL) {
      // Allokation im Teilbaum war erfolgreich
      *node = NODE_SPLIT;
      return addr;
   }
   // Analog beim rechten Kind
   return NULL;
}
```

Zusatzaufgabe c)

- Alle Knoten zwischen node und Wurzel können frei werden oder freie Buddies haben
- Ggf. vereinigen von Speicherbereichen
- Zunächst die beiden Kinder des Elternknotens von node:

```
// Zurück zur Wurzel und betrachte alle Knoten auf dem Weg
while ((node = bst_parent(node)) != NULL) {
    // Sind die beiden Buddies frei?
    if ((*bst_left(node) == NODE_FREE) &&
        (*bst_right(node) == NODE_FREE)) {
        // Dann ist ihr Elternknoten auch frei
        // und nicht mehr aufgeteilt
        *node = NODE_FREE;
    }
}
```