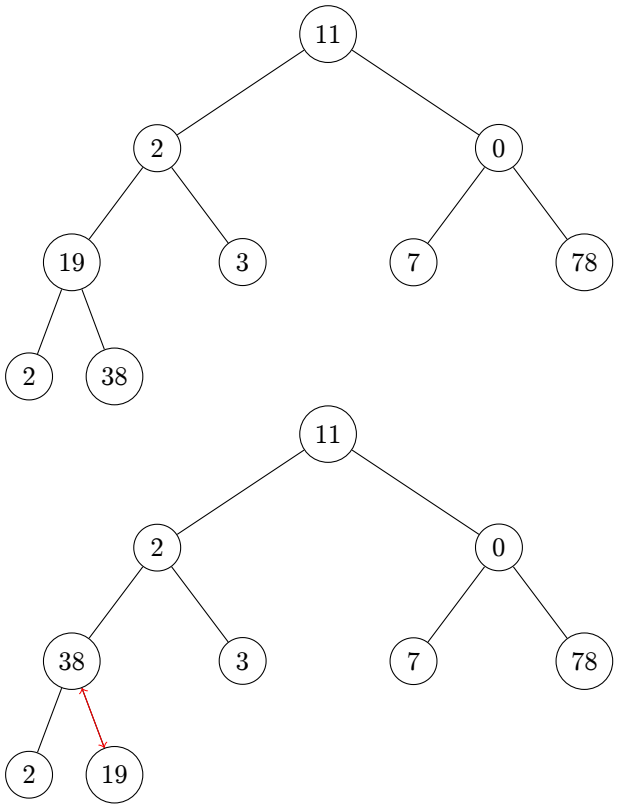
AuD Übung 09

PDF

14 Heapsort

1.

a)



b)

```
[11, 2, 0, 19, 3, 7, 78, 2, 38]
```

15 Hashing

1.

0	**100**
1	11
2	32
3	128 (da 8 schon belegt, linear Sondiert)
4	

0	100
5	
6	
7	
8	48
9	79

2.

Primzahlen, da

- Weniger Gemeinsame Teiler
- nicht so häufige Wiederholungen

Sei n die Anzahl der Einträge und m die Tabellengröße, dann:
mit $k \in \mathbb{R} \mid 1.5 \leq k \leq 2$
$m \geq \text{nächste Primzahl} \geq k \cdot n$

3.

a)

Hash	Key
0	Franz
1	
2	
3	Susi
4	
5	Ali -> Alfred -> Arno -> Alice -> Kurt -> Alex -> Angy -> Alf
6	Babsi -> Benno -> Bine
7	Max
8	
9	

```
names = [ "Ali", "Babsi", "Alfred", "Arno", "Alice", "Benno", "Kurt",
"Alex", "Angy", "Bine", "Max", "Franz", "Susi", "Alf"]

def calcHash(name):
        return ord(name[0])

def main():
```

```
        print(f"{'Name':<10} {'Hash':<10} {'Mod 10':<10}")
        for n in names:
                name_hash = calcHash(n)
                mod_10 = name_hash % 10

                print(f"{n:<10} {name_hash:<10} {mod_10:<10}")


if __name__ == "__main__":
        for i in names:
                print(f"{i}: {ord(i[0])}, {ord(i[-1])}")
        main()
```

b)

Hash	Key
0	Ali
1	Babsi -> Alex
2	Alfred -> Angy
3	Alice -> Max -> Franz -> Susi
4	Kurt -> Bine -> Alf
5	
6	Arno
7	Benno
8	
9	

```
names = [ "Ali", "Babsi", "Alfred", "Arno", "Alice", "Benno", "Kurt",
"Alex", "Angy", "Bine", "Max", "Franz", "Susi", "Alf"]

def calcHash(name):
        return (ord(name[0]) % 5), (ord(name[-1]) % 7)

def main():
        print(f"{'Name':<10} {'Hash':<20}")
        for n in names:
                first_hash, second_hash = calcHash(n)

                print(f"{n:<10} {first_hash:<10} + {second_hash:<10} =
{first_hash + second_hash:<10}")
```

```
if __name__ == "__main__":
        main()
```

4.

a)

Summe der Suchschritte = $1 + 1 + 2 + 3 + 4 + 2 + 5 + 6 + 7 + 3 + 1 + 1 + 1 + 8 = 45$

b)

Summe der Suchschritte = $1 + 1 + 1 + 1 + 1 + 1 + 1 + 2 + 2 + 2 + 2 + 3 + 4 + 3 = 25$

5.

Pro:

- Bessere Verteilung
- Bessere Effizienz bei der suchen

Contra:

- Höherer Berechnungsaufwand
- Mehr Speicherbedarf