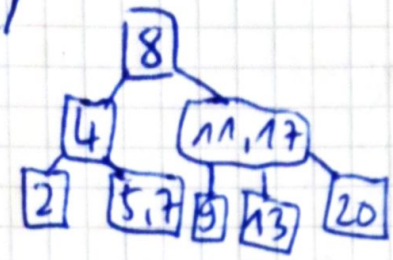
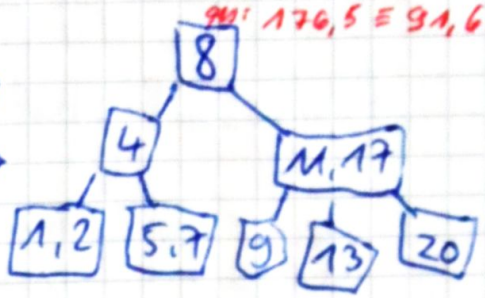
7 + 5 + 16 = 28/32
gut: 176,5 = 91,6 %.

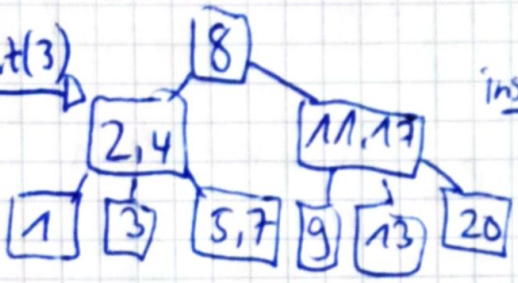
6. 1a)



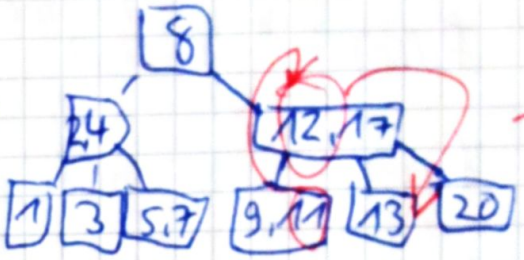
insert (1)



insert(3)

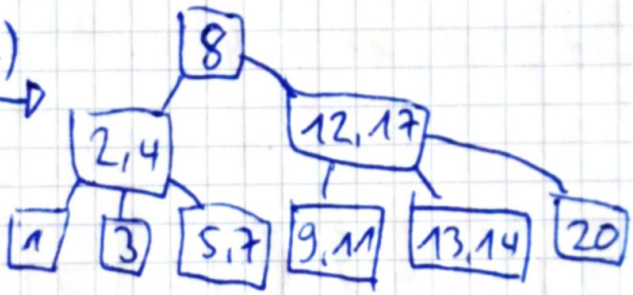


insert(12)

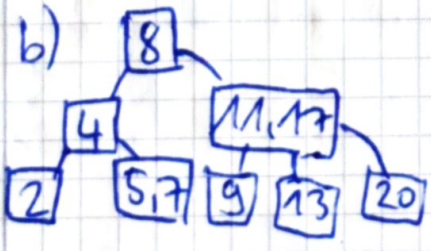


– 1 p.

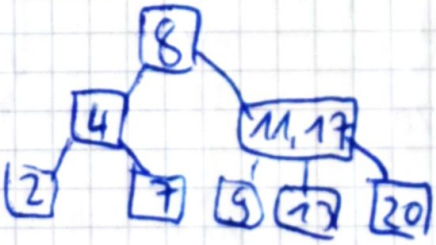
insert (14)



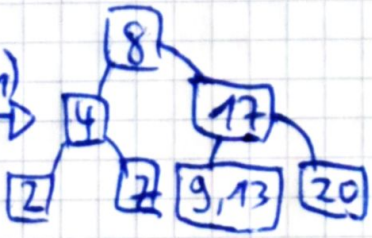
b)



remove(5)



remove(11)



remove (4)

k und t ? – 1p.

6.2a) $f(20) = 20 \bmod 11 = 9$

$f(32) = 32 \bmod 11 = 10$

$f(220) = 220 \bmod 11 = 0$

→ $200 + 7 \cdot 1 = 207$ – 1p.

$f(359) = 359 \bmod 11 = 7$

→ $360 - 20 = 340$ – 1p.

$f(60) = 60 \bmod 11 = 5$

$f(35) = 35 \bmod 11 = 2$

Wo war denn die los?

$f(101) = 101 \bmod 11 = 2$

$f(37) = 37 \bmod 11 = 4$

A =

0	1	2	3	4	5	6	7	8	9	10
220		35		37	60		359		20	32
		101								

b) $h_0(20) = (f(20)+0) \cdot 9$

$h_0(32) = (f(32)+0) = 10$

$h_0(220) = f(220)+0) = 0$

$h_0(359) = f(359)+0 = 7$

$h_0(60) = f(60)+0 = 5$

$h_0(35) = f(35)+0 > 2$

$h_0(101) = f(101)+0 = 2$

$h_1(101) = f(101)+1 = 3$

$h_0(37) = f(37)+0 = 4$

A =

0	→ 220
1	
2	→ 35
3	→ 101
4	→ 37
5	→ 60
6	
7	→ 359
8	
9	→ 20
10	→ 32

c) $h_0(20) = (f(20) + 0 \cdot g(20)) \bmod 11 = 9$

$h_0(32) = (f(32) + 0 \cdot g(32)) \bmod 11 = 10$

$h_0(32) = (f(220) + 0 \cdot g(220)) \bmod 11 = 0$

$h_0(359) = (f(359) + 0 \cdot g(359)) \bmod 11 = 7$

$h_0(60) = (f(60) + 0 \cdot g(60)) \bmod 11 = 5$

$h_0(35) = (f(35) + 0 \cdot g(35)) \bmod 11 = 2$

$h_0(101) = (f(101) + 0 \cdot g(101)) \bmod 11 = 2$

$h_1(101) = (f(101) + 1 \cdot g(101)) \bmod 11 = 6$

$h_0(37) = (f(37) + 0 \cdot g(37)) \bmod 11 = 4$

A =

0	→ 32
1	
2	→ 35
3	
4	→ 37
5	→ 60
6	→ 101
7	→ 359
8	
9	→ 20
10	→ 32

6.3ai)
```
insert (p) {
    insterthilf ( p, root);
}
insterthilf ( p, pos ){
    if (! kind ( pos, p) && (kind (pos, p)).isLeaf ())
        kind (pos, p) = p;  // Ziel ist ein leeres Blatt
    if (! kind (pos, p) && ! ( kind (pos, p)).isLeaf ())
        insterthilf (p, kind (pos, p));
        // Ziel ist unter einem Knoten
    if ( kind (pos, p) [illegible] )
        rette = kind (pos, p);   // Werte, 4 Pointer
        kind ( pos, p) = new Knoten (0,0,0,0,0);
        insterthilf (rette, kind (p, pos));
        insterthilf (p, kind(p, pos));
        // Das Ziel liegt in einem Blatt. Wir müssen
            das Ziel in einen Knoten aufteilen.
}
remove ( q){
    if (q. hat Knoten Als Geschwister) delete q;
        // ist ein Knoten auf der Ebene von q, kann
        // q einfach fliegen
    else
        if (q. [illegible] Der Generation > 2) delete q;
        // sind mehr als 2 nicht-leere Blätter in
            der Generation von q, kann q einfach fliegen
        else
            q. Elter = finde Letztes Blatt (q);
            Lösche Generation (q);
            schiebe Knoten (q. Elter);   }
```

// zu remove
// Ist das zu löschende Blatt in einer Generation, in der es nur ein weiteres nicht-leere Blatt gibt, und keine Knoten, so genügt es, das andere Blatt als Wurzel zu setzen.
Allerdings muss die Wurzel noch weiter „nach oben" geschoben werden, sollte sie das einzige nicht-leere Blatt in ihrer neuen Generation sein.
Dazu dient schiebeKnoten(p)

```
schiebeKnoten(p){
    while(BlätterInDerGeneration() == 0){
        p = p.Elter
    }
}
```

bi) Wenn der Abstand zwischen p_1 und p_2 auf den Einheitsquadrat ε -gross ist, wird da entsprechende Baum sehr tief. $\varepsilon > 0$, $\varepsilon \approx 0$.

ii) Idee: Sei p_0 der erste Knoten im „leeren Pfad". Seien p_1, p_2, p_3 und p_4 die Blätter (von denen 2 nicht-leer sein müssen) am tiefsten Ende des leeren Pfades.
Wir setzen die 4 Kind-Pointer von p_0 auf p_{1-4}.

Pseudocode auf dieser Rückseite

```
kürze Pfad (p0)
    kindPointer 0 = lookup (p1);
    kind Pointer 1 = lookup (p2);
    kind Pointer 2 = lookup (p3);
    kind Pointer 3 = lookup (p4);
```

6.3a ii)

```
anzahl (z) {
    if (z == Knoten) {
return  anzahl (z.kindPointer 0)+
        anzahl (z.kindPointer 1)+
        anzahl (z.kindPointer 2)+
        anzahl (z.kindPointer 3);
        }
    else {
        if (!z) return 0;
        else {return 1;}
        }
    }
}
```

Idee: Es werden rekursiv alle Kinder im Knoten abgeklappert. Jedes nicht-leere Blatt gibt 1 zurück. Alles wird aufsummiert.