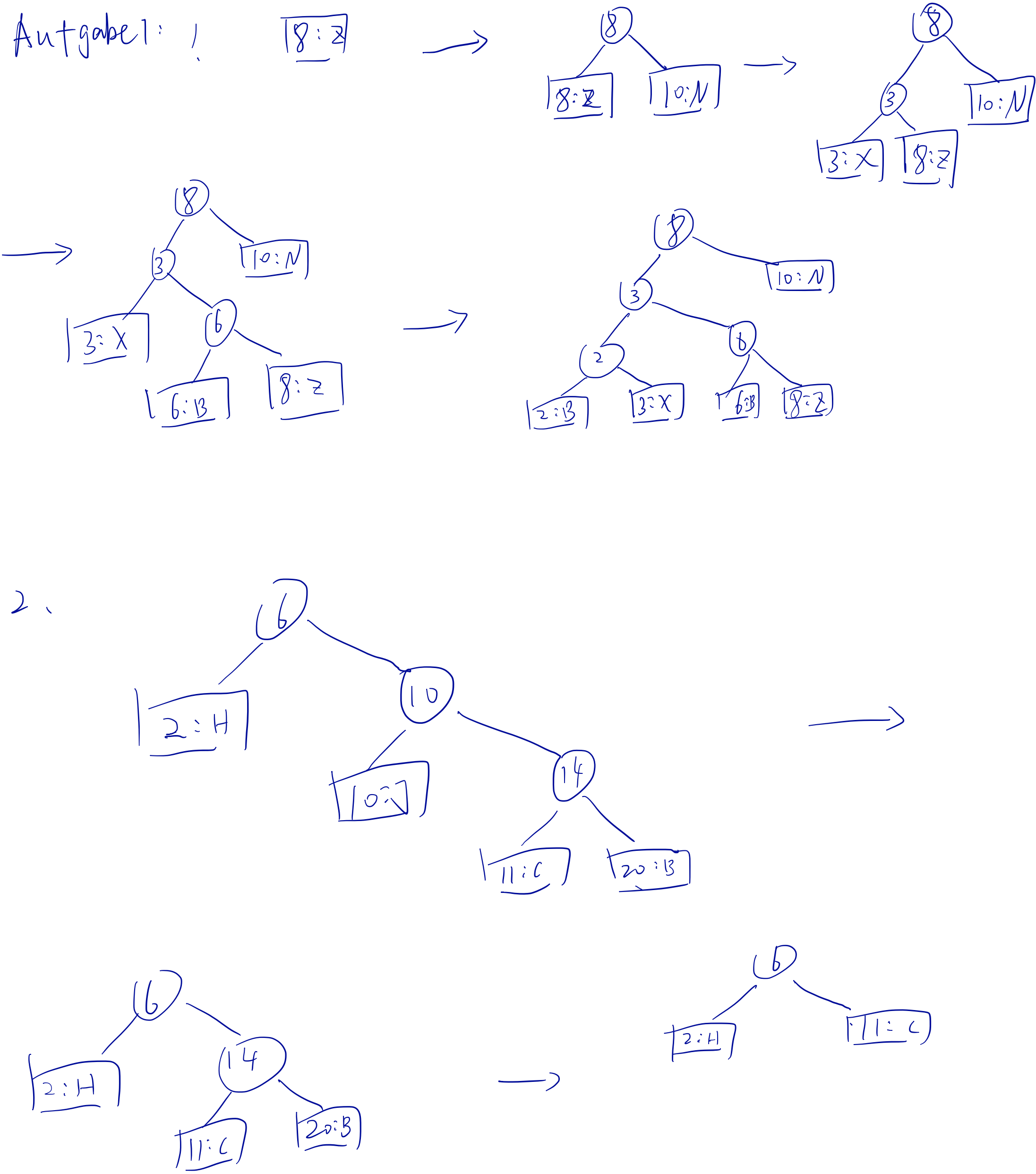
Aufgabe 1:


8:Z
8
8:Z
10:N
8
3
10:N
3:X
8:Z
8
3
10:N
3:X
6
6:B
8:Z
8
3
10:N
2
6
2:B
3:X
6:B
8:Z

2.

6
2:H
10
0:
14
11:C
20:B
6
2:H
14
11:C
20:B
6
2:H
11:C
2:H
leer

	Löschen		Suchen		Einfügen	
	Average	Worst	Average	Worst	Average	Worst
Binärer Suchbaum	O(log n)	O(n)	O(log n)	O(n)	O(log n)	O(n)
Binärer Suchbaum (degeneriert)	O(n)	O(n)	O(n)	O(n)	O(n)	O(n)
Binärer Suchbaum (vollständig)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)
Array und binäre Suche	O(n)	O(n)	O(n)	O(n)	O(log n)	O(log n)

Frage 1:Wenn oft einfügen oder gelöscht werden soll,ist ein binärer Suchbaum sinnvoller.Ansonsten ist ein Array besser,weil beim worst case oder degeneriert binäre suchbaum ist suchen-Operation langsamer.

Frage 2.1:Einfügen-Methode wird n-mal aufgerufen,deswegen hat Average-Case-Komplexität O(nlogn) und Worst-case O(n^2).

Frage 2.2:Das Array muss sortiert sein,damit eine binäre Suche durchgeführt werden kann. Z.b für Quicksot hat Average-case-Komplexität O(nlogn).

Frage 3:Um balanciert suchbaum zu erzeugen,muss erst das Element in der Mitte des Arrays einzufügen,dann das Element in der Mitte des rechten Teilarrays,danach das Element in der Mitte des linken Teilarrays usw. bis zum Ende.

Frage 4:Alle drei Traversieren-Methode Pre/In/Post-Order können ein binäre suchbaum in eine sortiert Array unwandeln,weil nur blätten(leaf) Daten sprich ein.