Question : 2

* Given Date:

87	92	78	95	88	90	84	89	93	91

* Algorithm :

Insertion sort is best in this because is smaller & Insertion sort easy for the teacher. it to implement.

* Solution :

→ 87, 92, 78, 95, 88, 90, 84, 89, 93, 91

92 > 87 → don't swap

78 ≠ 92 → swap with index 2 to 1

→ 87, 78, 92, 95, 88, 90, 84, 89, 93, 91

78 ≠ 87 → swap it

→ 78, 87, 92, 95, 88, 90, 84, 89, 93, 91

95 > 92 → don't swap

88 ≠ 95 → swap it

→ 78, 87, 92, 88, 95, 90, 84, 89, 93, 91

88 ≠ 92 → swap it.

→ 78, 87, 88, 92, 95, 90, 84, 89, 93, 91

90 ≠ 95 → swap it

→ 78, 87, 88, 92, 90, 95 . . . .

90 ≠ 92 → swap it

→ 78, 87, 88, 90, 92, 95, 84, 89, 93, 91

84 ≠ 95 → swap

84 ≠ 92 → again swap

84 ≠ 90 → again swap

84 ≠ 88 → again swap

84 ≠ 87 → again swap

84 > 78 → don't swap

→ 78, 84, 87, 88, 90, 92, 95, 89, 93, 91

89 ≠ 95 → swap

89 ≠ 92 → swap

89 ≠ 90 → swap

89 > 88 → don't swap

→ 78, 84, 87, 88, 89, 90, 92, 95, 93, 91.

93 ≠ 95 → swap

→ 78, 84, 87, 88, 89, 90, 92, 93, 95, 91

91 ≠ 95 → swap
91 ≠ 93 → swap
91 ≠ 92 → swap
91 > 90 → don't swap.

→ 78, 84, 87, 88, 89, 90, 91, 92, 93, 95

Hence, Array is completely sorted.

Question : 2 :-

* Given data.

1023	1018	1025	1020	1019	1022	1021	1017	1024	1016

* Algorithm:

→ And this case ~~we~~ use Radix sort because No of input larger so ~~it~~ have quickly sorting it easy.

Pass: 1 : check last digit

1020	1021	1022	1023	1024	1025	1016	1017	1018	1019
0	1	2	3	4	5	6	7	8	9

Pass: 2 : check 2nd last digit.

0	1	2	3	4	5	6	7	8	9
1020	1019 1018 1017 1016	1025 1024 1023 1022 1021							

Pass: 3 : check 3rd last digit

index : 0	1	2	3	4	5	6	7	8	9
1025 1024 1023 1022 1021 1020 1019 1018 1017 1016									

Pass: 4 : check first digit

All input on index (1) same order.

Final Sorted Array:

1016	1017	1018	1019	1020	1021	1022	1023	1024	1025

Question: 3 :~

* Given Data.

22.5	24.3	23.1	21.9	25.7	24.8	26.2	22.0	23.5	25.0

* Algorithm:

↳ In this case I use Bucket sort for sorting because No: of input in decimal & smaller so. Bucket sort to implement & understand. in Bucket sub array use insertion sort.

Bucket's

0 → ~~21.9~~

1 → ~~21.9~~ sorted

2 → 22.5, 22.0 → ~~22.0~~, ~~22.5~~

3 → ~~23.1~~, ~~23.5~~ sorted

4 → ~~24.3~~, ~~24.8~~ sorted

5 → ~~25.7~~, 25.0 → ~~25.0~~, ~~25.7~~

6 → ~~26.2~~ sorted

7

8

9

Final sorted Array:

21.9	22.0	22.5	23.1	23.5	24.3	24.8	25.0	25.7	26.2

Question: 4 :~

* Given Data.

Emma	Rushford	franklin	jack	Alice	frank	Grace	hannah	charlie	david	isaac	bob

~~Pass~~

* Algorithm:

↳ In this case I use Radix sort for sorting because input in Alphbatic & it easy in that case.

Pass 1: check last alphabate.

Emma, bob, isaac, rushford, david, alice, grace, charlie, hannah, jack, frank, franklin

Pass 2: check 2nd alphabete of Pass 1:

isaac, hannah, alice, grace, jack, david, charlie, franklin, Emma, frank, bob, rushford

Pass 3: check 3rd alphabate of Pass 2:

isaac, grace, jack, frank, bob, alice, charlie, franklin, emma, hannah, rushford, david

Pass 4: check 4th alphabate of Pass 3:

bob, david, emma, rushford, jack, franklin, alice, hannah, grace, frank, charlie, isaac.

Pass 5: check 5th alphabate of Pass 4:

Bob, Emma, jack, alice, hannah, charlie, david, frank, grace, rushford, isaac, franklic

Pass 6: check 6th alphabate of Pass 5:

Bob, Emma, jack, alice, david, frank, grace, isaac, franklin, hannah, charlie, rushford

Pass 7: check 7th alphabate of Pass 6:

Bob, Emma, jack, alice, david, frank, grace, isaac, charlie, franklin, rushford.

Pass 8: check 8th alphabate of Pass 7:

final sorted list.

Bob, Emma, jack, Alice, David, frank, Grace, Isaac, charlie, franklin, Rushford.

Question: 5 :~

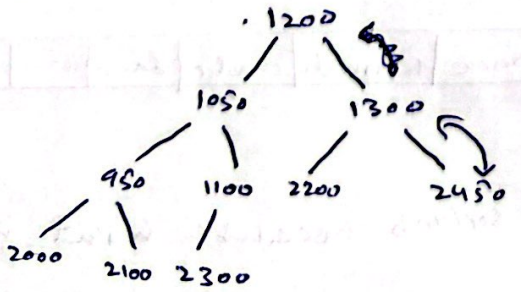
* Give Data.

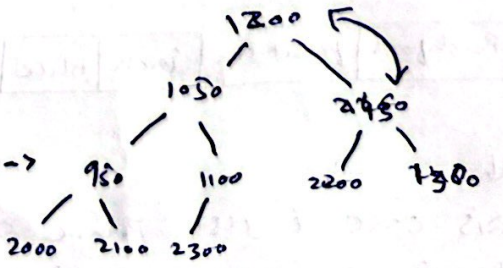
1200	1050	1300	950	1100	2200	2450	2000	2100	2300

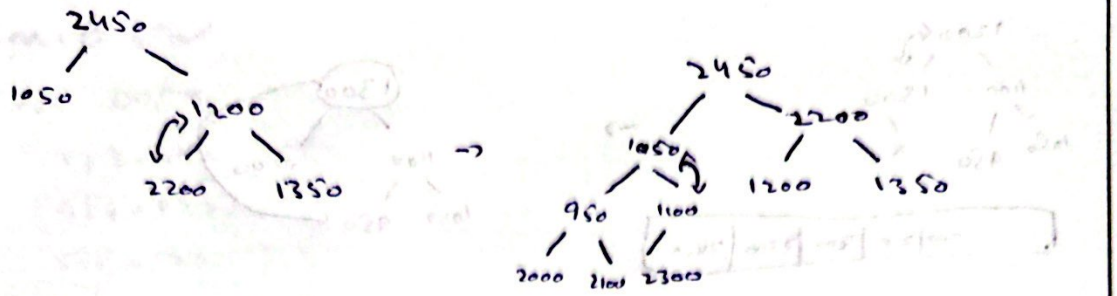
* Algorithm:

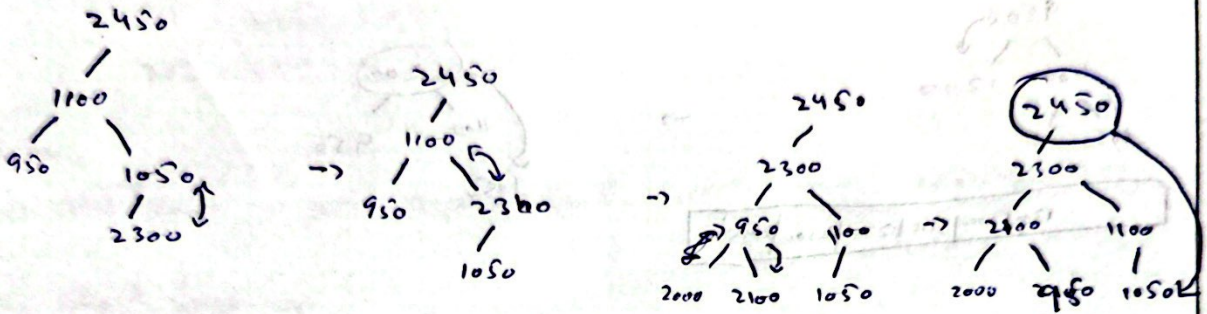
↳ In this case I use Max heap sort which perfectly suitable for this & easy to sort.

Tree:

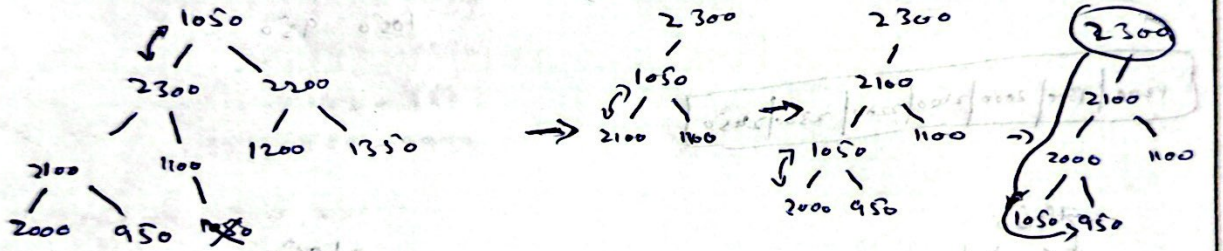


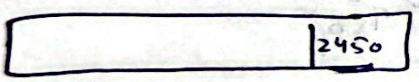




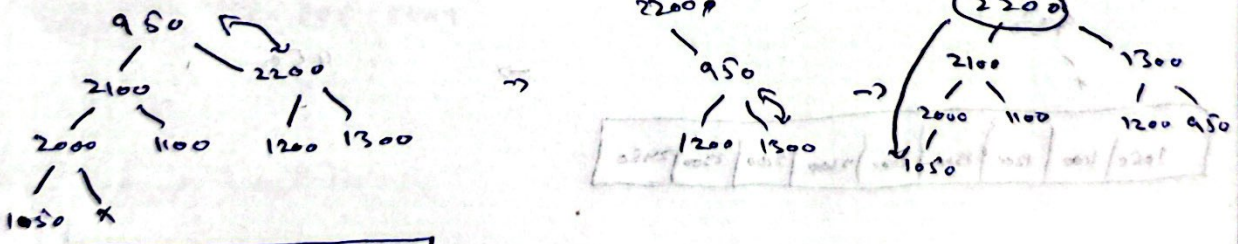


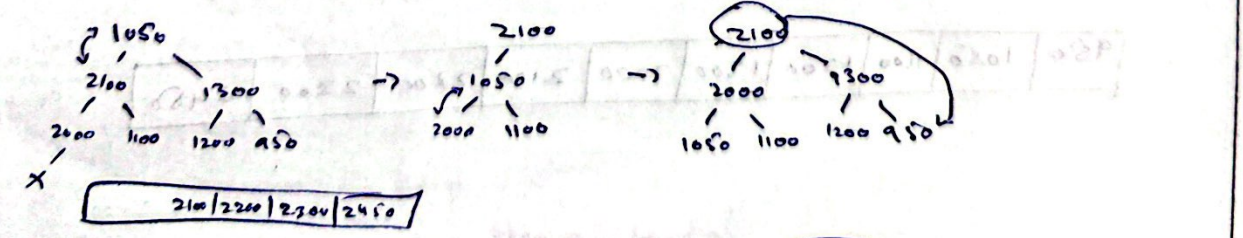
Swap with last & delete.

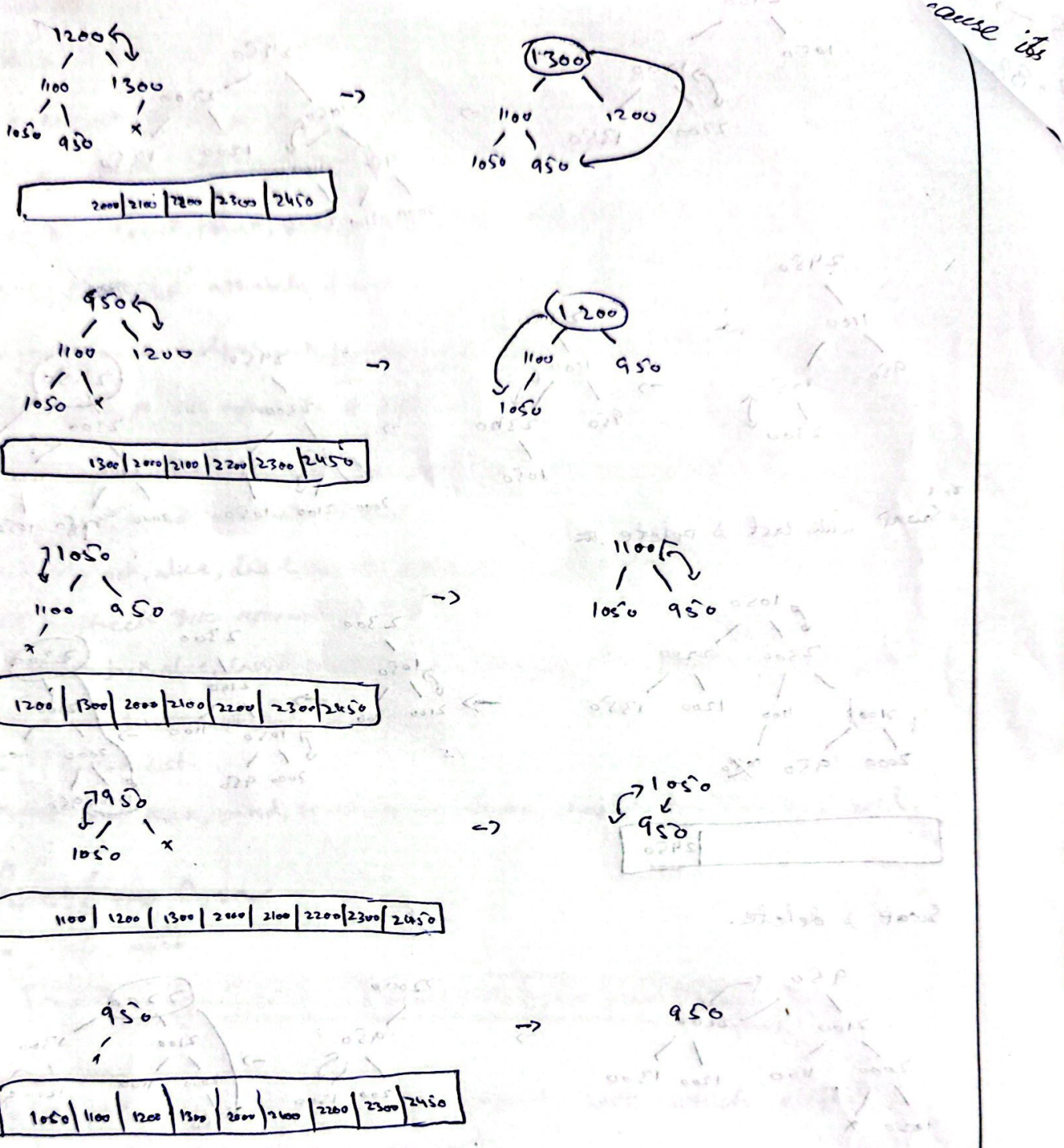




Swap & delete.







Final sorted:

950	1050	1100	1200	1300	2000	2100	2200	2300	2450

Question: 6 :~

* Given data.

123-456-7890
987-654-3210
555-123-4567
777-888-9999
111-222-3337
323-423-7634

* Algorithm.
  ↳ in this case, i use Radix Sort which more appropriative.

Pass 1 : check last No:

123-456-7890
987-654-3210
111-222-3337
323-423-7634
555-123-4567
777-888-9999

Pass 2 : check 2nd No:

987-654-3210
111-222-337
323-423-7634
555-123-4567
123-456-7890
777-888-9999

:

Pass 10 : check first No:

111-222-333
123-456-7890
323-423-7634
555-123-4567
777-888-9999
987-654-3210

Hence, sorted.

Question : 7 :~

Algorithm	No: of comparison	No: of swaps	Best case	Average case	Worst case	in place	stable
Insertion sort	$O(n^2)$	$O(n^2)$	$O(n)$	$O(n^2)$	$O(n^2)$	Yes	Yes
Bucket sort	$O(n^2)$	$O(n^2)$	$O(n+k)$	$O(n+k)$	$O(n^2)$	No	Yes
Radix sort	$O(nk)$	$O(n+k)$	$O(nk)$	$O(nk)$	$O(nk)$	No	Yes
Merge sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	No	Yes
Max-Heap sort	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	$O(n\log n)$	Yes	No