**Homework lecture 1**

**Sort**

1. Given a list of integer numbers: 2, 1, 10, 6, 3, 8, 7, 13, 20. Draw the steps to sort (ascending sort) the list by following methods:

* Selection sort

step 0: 2 1 10 6 3 8 7 13 20

step 1: 1 2 10 6 3 8 7 13 20

step 2: 1 2 10 6 3 8 7 13 20

step 3: 1 2 3 6 10 8 7 13 20

step 4: 1 2 3 6 10 8 7 13 20

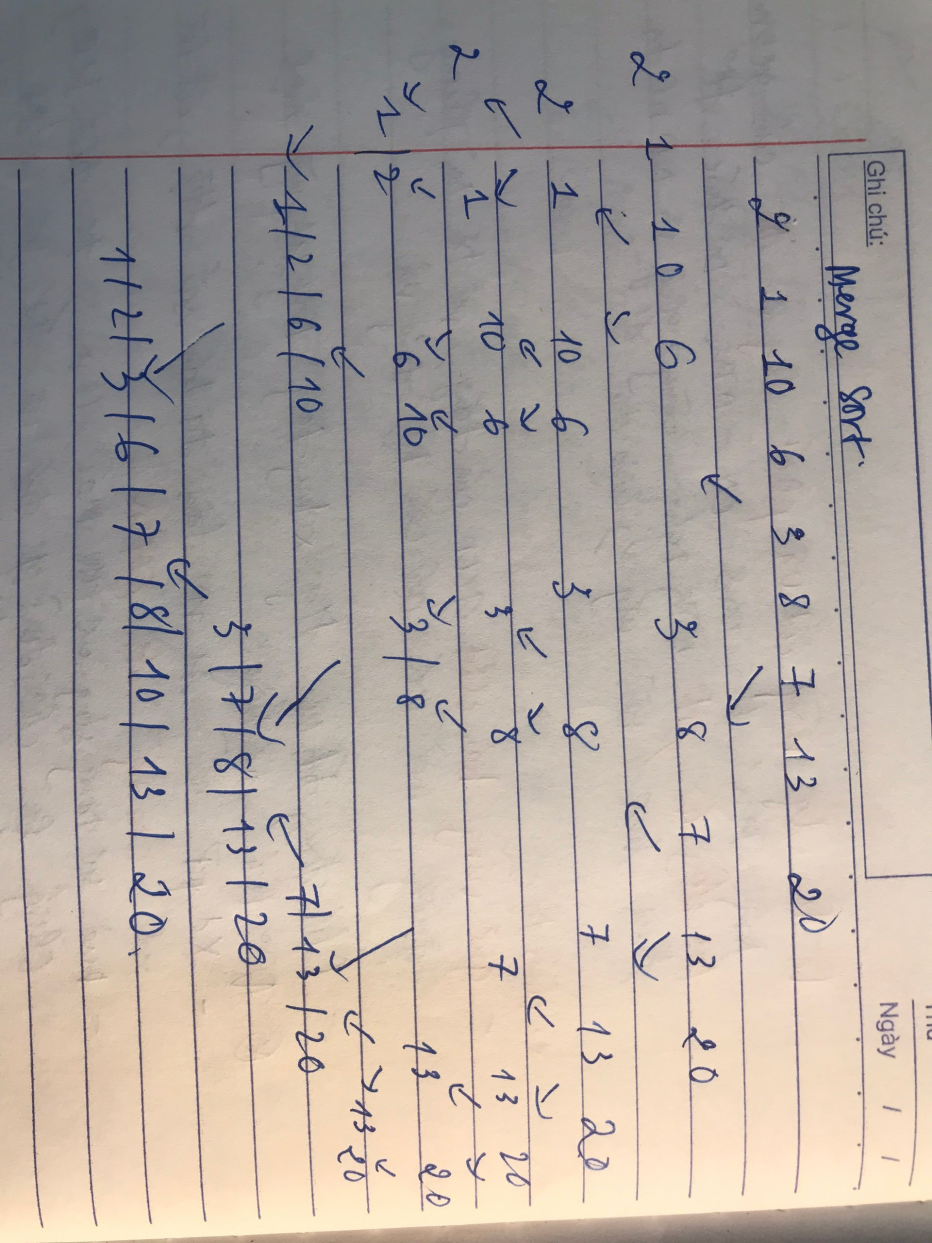
step 5: 1 2 3 6 7 8 10 13 20

step 6: 1 2 3 6 7 8 10 13 20

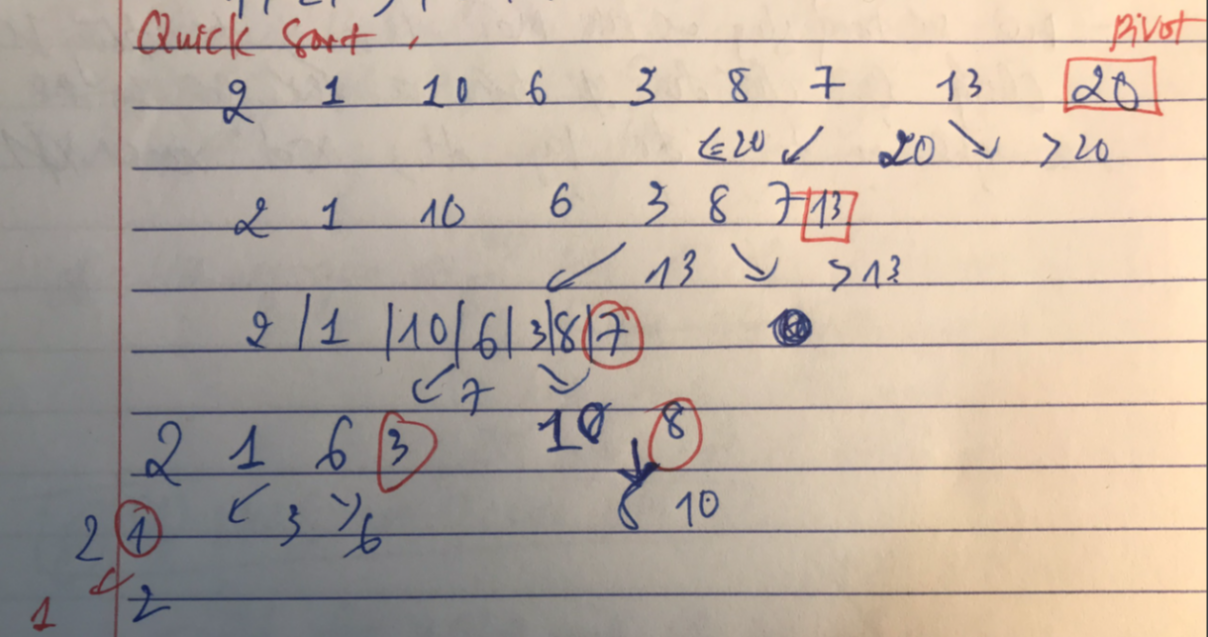
step 7: 1 2 3 6 7 8 10 13 20

step 8: 1 2 3 6 7 8 10 13 20

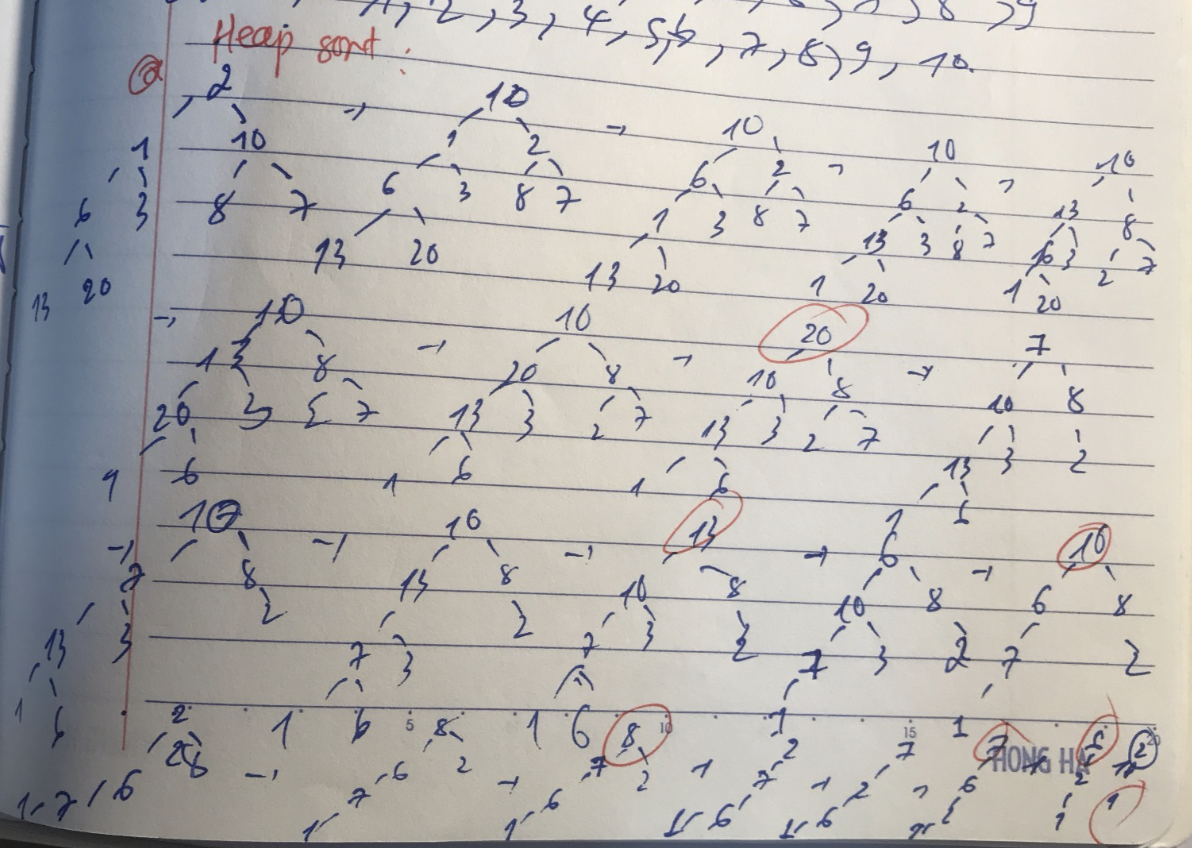
* Insertion sort
  + step 1: 2 1 10 6 3 8 7 13 20
  + step 2: 1 2 10 6 3 8 7 13 20
  + step 3: 1 2 10 6 3 8 7 13 20
  + step 4: 1 2 6 10 3 8 7 13 20
  + step 5: 1 2 3 6 10 8 7 13 20
  + step 6: 1 2 3 6 8 10 7 13 20
  + step 7: 1 2 3 6 7 8 10 13 20
  + step 8: 1 2 3 6 7 8 10 13 20
  + step 9: 1 2 3 6 7 8 10 13 20
* Bubble sort
  + step0: 2 1 10 6 3 8 7 13 20
  + step1:
  + step1.1: 1 2 10 6 3 8 7 13 20
  + step1.2: 1 2 10 6 3 8 7 13 20
  + step1.3: 1 2 6 10 3 8 7 13 20
  + step1.4: 1 2 6 3 10 8 7 13 20
  + step1.5: 1 2 6 3 8 10 7 13 20
  + step1.6: 1 2 6 3 8 7 10 13 20
  + step1.7: 1 2 6 3 8 7 10 13 20
  + step1.8: 1 2 6 3 8 7 10 13 20
  + step2:
  + step2.1: 1 2 6 3 8 7 10 13 20
  + step2.2: 1 2 6 3 8 7 10 13 20
  + step2.3: 1 2 3 6 8 7 10 13 20
  + step2.4: 1 2 3 6 8 7 10 13 20
  + step2.5: 1 2 3 6 7 8 10 13 20
  + step2.6: 1 2 3 6 7 8 10 13 20
  + step2.7: 1 2 3 6 7 8 10 13 20
  + step2.8: 1 2 3 6 7 8 10 13 20
  + step3:
  + step3.1: 1 2 3 6 7 8 10 13 20
  + step3.2: 1 2 3 6 7 8 10 13 20
  + step3.3: 1 2 3 6 7 8 10 13 20
  + step3.4: 1 2 3 6 7 8 10 13 20
  + step3.5: 1 2 3 6 7 8 10 13 20
  + step3.6: 1 2 3 6 7 8 10 13 20
  + step3.7: 1 2 3 6 7 8 10 13 20
  + step3.8: 1 2 3 6 7 8 10 13 20
* Merge sort



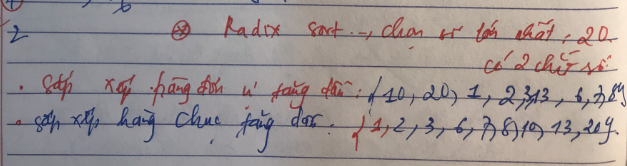
* Quick sort



* Heap sort

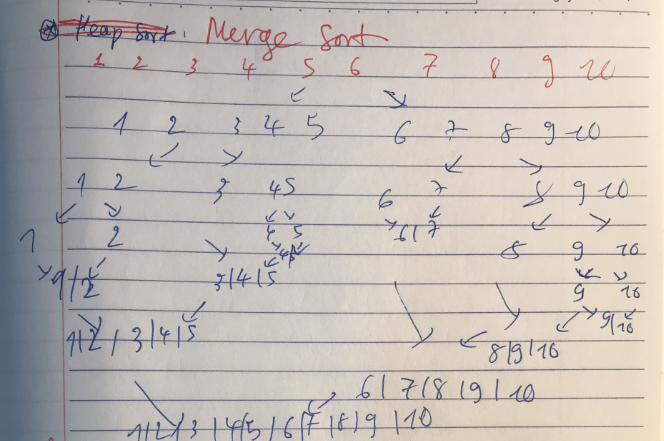


* Radix sort

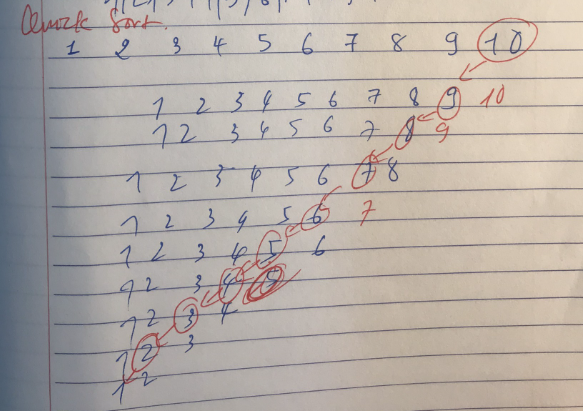


1. Same question for these lists:

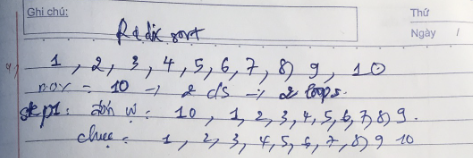
* 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  + Selection sort
    - step 0: 1 2 3 4 5 6 7 8 9 10
    - step 1: 1 2 3 4 5 6 7 8 9 10
    - step 2: 1 2 3 4 5 6 7 8 9 10
    - step 3: 1 2 3 4 5 6 7 8 9 10
    - step 4: 1 2 3 4 5 6 7 8 9 10
    - step 5: 1 2 3 4 5 6 7 8 9 10
    - step 6: 1 2 3 4 5 6 7 8 9 10
    - step 7: 1 2 3 4 5 6 7 8 9 10
    - step 8: 1 2 3 4 5 6 7 8 9 10
    - step 9: 1 2 3 4 5 6 7 8 9 10
  + Insertion sort
    - step 1: 1 2 3 4 5 6 7 8 9 10
    - step 2: 1 2 3 4 5 6 7 8 9 10
    - step 3: 1 2 3 4 5 6 7 8 9 10
    - step 4: 1 2 3 4 5 6 7 8 9 10
    - step 5: 1 2 3 4 5 6 7 8 9 10
    - step 6: 1 2 3 4 5 6 7 8 9 10
    - step 7: 1 2 3 4 5 6 7 8 9 10
    - step 8: 1 2 3 4 5 6 7 8 9 10
    - step 9: 1 2 3 4 5 6 7 8 9 10
    - step 10: 1 2 3 4 5 6 7 8 9 10
  + Bubble sort
    - step0: 1 2 3 4 5 6 7 8 9 10
    - step1:
    - step1.1: 1 2 3 4 5 6 7 8 9 10
    - step1.2: 1 2 3 4 5 6 7 8 9 10
    - step1.3: 1 2 3 4 5 6 7 8 9 10
    - step1.4: 1 2 3 4 5 6 7 8 9 10
    - step1.5: 1 2 3 4 5 6 7 8 9 10
    - step1.6: 1 2 3 4 5 6 7 8 9 10
    - step1.7: 1 2 3 4 5 6 7 8 9 10
    - step1.8: 1 2 3 4 5 6 7 8 9 10
    - step1.9: 1 2 3 4 5 6 7 8 9 10
  + Merge sort



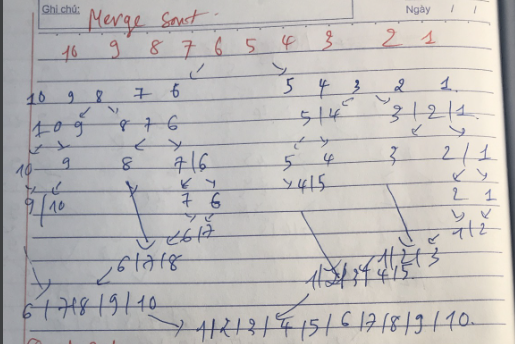
* + Quick sort



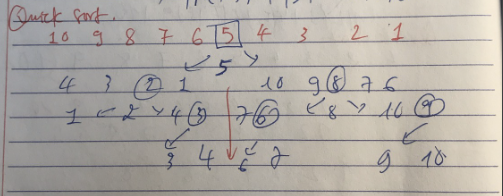
* + Heap sort
  + Radix sort



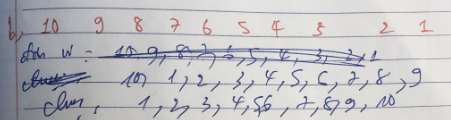
* 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
  + Selection sort
    - step 0: 10 9 8 7 6 5 4 3 2 1
    - step 1: 1 9 8 7 6 5 4 3 2 10
    - step 2: 1 2 8 7 6 5 4 3 9 10
    - step 3: 1 2 3 7 6 5 4 8 9 10
    - step 4: 1 2 3 4 6 5 7 8 9 10
    - step 5: 1 2 3 4 5 6 7 8 9 10
    - step 6: 1 2 3 4 5 6 7 8 9 10
    - step 7: 1 2 3 4 5 6 7 8 9 10
    - step 8: 1 2 3 4 5 6 7 8 9 10
    - step 9: 1 2 3 4 5 6 7 8 9 10
  + Insertion sort
    - step 1: 10 9 8 7 6 5 4 3 2 1
    - step 2: 9 10 8 7 6 5 4 3 2 1
    - step 3: 8 9 10 7 6 5 4 3 2 1
    - step 4: 7 8 9 10 6 5 4 3 2 1
    - step 5: 6 7 8 9 10 5 4 3 2 1
    - step 6: 5 6 7 8 9 10 4 3 2 1
    - step 7: 4 5 6 7 8 9 10 3 2 1
    - step 8: 3 4 5 6 7 8 9 10 2 1
    - step 9: 2 3 4 5 6 7 8 9 10 1
    - step 10: 1 2 3 4 5 6 7 8 9 10
  + Bubble sort
    - step0: 10 9 8 7 6 5 4 3 2 1
    - step1:
    - step1.1: 9 10 8 7 6 5 4 3 2 1
    - step1.2: 9 8 10 7 6 5 4 3 2 1
    - step1.3: 9 8 7 10 6 5 4 3 2 1
    - step1.4: 9 8 7 6 10 5 4 3 2 1
    - step1.5: 9 8 7 6 5 10 4 3 2 1
    - step1.6: 9 8 7 6 5 4 10 3 2 1
    - step1.7: 9 8 7 6 5 4 3 10 2 1
    - step1.8: 9 8 7 6 5 4 3 2 10 1
    - step1.9: 9 8 7 6 5 4 3 2 1 10
    - step2:
    - step2.1: 8 9 7 6 5 4 3 2 1 10
    - step2.2: 8 7 9 6 5 4 3 2 1 10
    - step2.3: 8 7 6 9 5 4 3 2 1 10
    - step2.4: 8 7 6 5 9 4 3 2 1 10
    - step2.5: 8 7 6 5 4 9 3 2 1 10
    - step2.6: 8 7 6 5 4 3 9 2 1 10
    - step2.7: 8 7 6 5 4 3 2 9 1 10
    - step2.8: 8 7 6 5 4 3 2 1 9 10
    - step2.9: 8 7 6 5 4 3 2 1 9 10
    - step3:
    - step3.1: 7 8 6 5 4 3 2 1 9 10
    - step3.2: 7 6 8 5 4 3 2 1 9 10
    - step3.3: 7 6 5 8 4 3 2 1 9 10
    - step3.4: 7 6 5 4 8 3 2 1 9 10
    - step3.5: 7 6 5 4 3 8 2 1 9 10
    - step3.6: 7 6 5 4 3 2 8 1 9 10
    - step3.7: 7 6 5 4 3 2 1 8 9 10
    - step3.8: 7 6 5 4 3 2 1 8 9 10
    - step3.9: 7 6 5 4 3 2 1 8 9 10
    - step4:
    - step4.1: 6 7 5 4 3 2 1 8 9 10
    - step4.2: 6 5 7 4 3 2 1 8 9 10
    - step4.3: 6 5 4 7 3 2 1 8 9 10
    - step4.4: 6 5 4 3 7 2 1 8 9 10
    - step4.5: 6 5 4 3 2 7 1 8 9 10
    - step4.6: 6 5 4 3 2 1 7 8 9 10
    - step4.7: 6 5 4 3 2 1 7 8 9 10
    - step4.8: 6 5 4 3 2 1 7 8 9 10
    - step4.9: 6 5 4 3 2 1 7 8 9 10
    - step5:
    - step5.1: 5 6 4 3 2 1 7 8 9 10
    - step5.2: 5 4 6 3 2 1 7 8 9 10
    - step5.3: 5 4 3 6 2 1 7 8 9 10
    - step5.4: 5 4 3 2 6 1 7 8 9 10
    - step5.5: 5 4 3 2 1 6 7 8 9 10
    - step5.6: 5 4 3 2 1 6 7 8 9 10
    - step5.7: 5 4 3 2 1 6 7 8 9 10
    - step5.8: 5 4 3 2 1 6 7 8 9 10
    - step5.9: 5 4 3 2 1 6 7 8 9 10
    - step6:
    - step6.1: 4 5 3 2 1 6 7 8 9 10
    - step6.2: 4 3 5 2 1 6 7 8 9 10
    - step6.3: 4 3 2 5 1 6 7 8 9 10
    - step6.4: 4 3 2 1 5 6 7 8 9 10
    - step6.5: 4 3 2 1 5 6 7 8 9 10
    - step6.6: 4 3 2 1 5 6 7 8 9 10
    - step6.7: 4 3 2 1 5 6 7 8 9 10
    - step6.8: 4 3 2 1 5 6 7 8 9 10
    - step6.9: 4 3 2 1 5 6 7 8 9 10
    - step7:
    - step7.1: 3 4 2 1 5 6 7 8 9 10
    - step7.2: 3 2 4 1 5 6 7 8 9 10
    - step7.3: 3 2 1 4 5 6 7 8 9 10
    - step7.4: 3 2 1 4 5 6 7 8 9 10
    - step7.5: 3 2 1 4 5 6 7 8 9 10
    - step7.6: 3 2 1 4 5 6 7 8 9 10
    - step7.7: 3 2 1 4 5 6 7 8 9 10
    - step7.8: 3 2 1 4 5 6 7 8 9 10
    - step7.9: 3 2 1 4 5 6 7 8 9 10
    - step8:
    - step8.1: 2 3 1 4 5 6 7 8 9 10
    - step8.2: 2 1 3 4 5 6 7 8 9 10
    - step8.3: 2 1 3 4 5 6 7 8 9 10
    - step8.4: 2 1 3 4 5 6 7 8 9 10
    - step8.5: 2 1 3 4 5 6 7 8 9 10
    - step8.6: 2 1 3 4 5 6 7 8 9 10
    - step8.7: 2 1 3 4 5 6 7 8 9 10
    - step8.8: 2 1 3 4 5 6 7 8 9 10
    - step8.9: 2 1 3 4 5 6 7 8 9 10
    - step9:
    - step9.1: 1 2 3 4 5 6 7 8 9 10
    - step9.2: 1 2 3 4 5 6 7 8 9 10
    - step9.3: 1 2 3 4 5 6 7 8 9 10
    - step9.4: 1 2 3 4 5 6 7 8 9 10
    - step9.5: 1 2 3 4 5 6 7 8 9 10
    - step9.6: 1 2 3 4 5 6 7 8 9 10
    - step9.7: 1 2 3 4 5 6 7 8 9 10
    - step9.8: 1 2 3 4 5 6 7 8 9 10
    - step9.9: 1 2 3 4 5 6 7 8 9 10
    - step10:
    - step10.1: 1 2 3 4 5 6 7 8 9 10
    - step10.2: 1 2 3 4 5 6 7 8 9 10
    - step10.3: 1 2 3 4 5 6 7 8 9 10
    - step10.4: 1 2 3 4 5 6 7 8 9 10
    - step10.5: 1 2 3 4 5 6 7 8 9 10
    - step10.6: 1 2 3 4 5 6 7 8 9 10
    - step10.7: 1 2 3 4 5 6 7 8 9 10
    - step10.8: 1 2 3 4 5 6 7 8 9 10
    - step10.9: 1 2 3 4 5 6 7 8 9 10
  + Merge sort



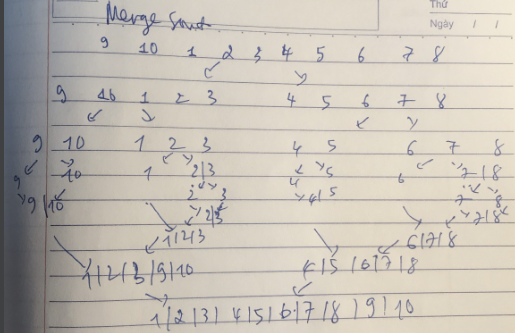
* + Quick sort



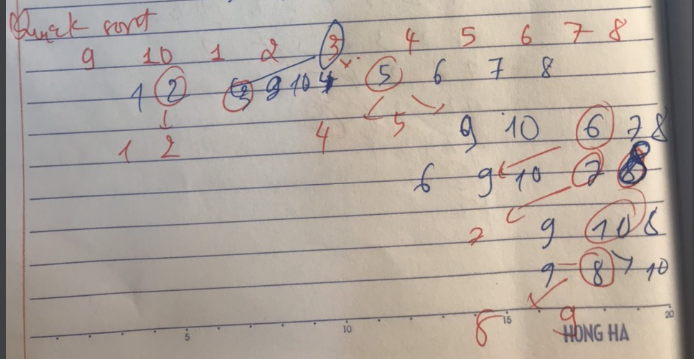
* + Heap sort
  + Radix sort



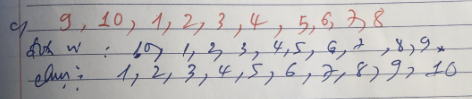
* 9, 10, 1, 2, 3, 4, 5, 6, 7, 8
  + Selection sort
    - step 0: 9 10 1 2 3 4 5 6 7 8
    - step 1: 1 10 9 2 3 4 5 6 7 8
    - step 2: 1 2 9 10 3 4 5 6 7 8
    - step 3: 1 2 3 10 9 4 5 6 7 8
    - step 4: 1 2 3 4 9 10 5 6 7 8
    - step 5: 1 2 3 4 5 10 9 6 7 8
    - step 6: 1 2 3 4 5 6 9 10 7 8
    - step 7: 1 2 3 4 5 6 7 10 9 8
    - step 8: 1 2 3 4 5 6 7 8 9 10
    - step 9: 1 2 3 4 5 6 7 8 9 10
  + Insertion sort
    - step 1: 9 10 1 2 3 4 5 6 7 8
    - step 2: 9 10 1 2 3 4 5 6 7 8
    - step 3: 1 9 10 2 3 4 5 6 7 8
    - step 4: 1 2 9 10 3 4 5 6 7 8
    - step 5: 1 2 3 9 10 4 5 6 7 8
    - step 6: 1 2 3 4 9 10 5 6 7 8
    - step 7: 1 2 3 4 5 9 10 6 7 8
    - step 8: 1 2 3 4 5 6 9 10 7 8
    - step 9: 1 2 3 4 5 6 7 9 10 8
    - step 10: 1 2 3 4 5 6 7 8 9 10
  + Bubble sort
    - step0: 10 9 1 2 3 4 5 6 7 8
    - step1:
    - step1.1: 9 10 1 2 3 4 5 6 7 8
    - step1.2: 9 1 10 2 3 4 5 6 7 8
    - step1.3: 9 1 2 10 3 4 5 6 7 8
    - step1.4: 9 1 2 3 10 4 5 6 7 8
    - step1.5: 9 1 2 3 4 10 5 6 7 8
    - step1.6: 9 1 2 3 4 5 10 6 7 8
    - step1.7: 9 1 2 3 4 5 6 10 7 8
    - step1.8: 9 1 2 3 4 5 6 7 10 8
    - step1.9: 9 1 2 3 4 5 6 7 8 10
    - step2:
    - step2.1: 1 9 2 3 4 5 6 7 8 10
    - step2.2: 1 2 9 3 4 5 6 7 8 10
    - step2.3: 1 2 3 9 4 5 6 7 8 10
    - step2.4: 1 2 3 4 9 5 6 7 8 10
    - step2.5: 1 2 3 4 5 9 6 7 8 10
    - step2.6: 1 2 3 4 5 6 9 7 8 10
    - step2.7: 1 2 3 4 5 6 7 9 8 10
    - step2.8: 1 2 3 4 5 6 7 8 9 10
    - step2.9: 1 2 3 4 5 6 7 8 9 10
    - step3:
    - step3.1: 1 2 3 4 5 6 7 8 9 10
    - step3.2: 1 2 3 4 5 6 7 8 9 10
    - step3.3: 1 2 3 4 5 6 7 8 9 10
    - step3.4: 1 2 3 4 5 6 7 8 9 10
    - step3.5: 1 2 3 4 5 6 7 8 9 10
    - step3.6: 1 2 3 4 5 6 7 8 9 10
    - step3.7: 1 2 3 4 5 6 7 8 9 10
    - step3.8: 1 2 3 4 5 6 7 8 9 10
    - step3.9: 1 2 3 4 5 6 7 8 9 10
  + Merge sort



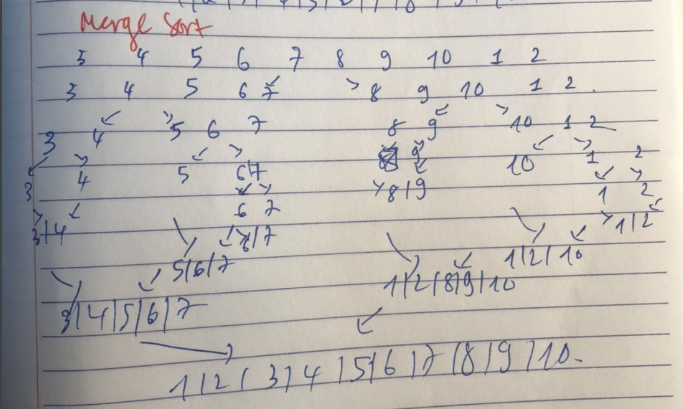
* + Quick sort



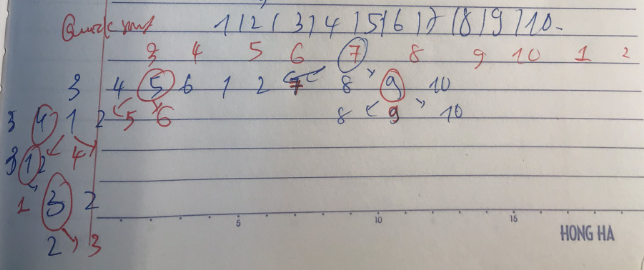
* + Heap sort
  + Radix sort



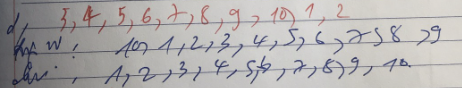
* 3, 4, 5, 6, 7, 8, 9, 10, 1, 2
  + Selection sort
    - step 0: 3 4 5 6 7 8 0 10 1 2
    - step 1: 0 4 5 6 7 8 3 10 1 2
    - step 2: 0 1 5 6 7 8 3 10 4 2
    - step 3: 0 1 2 6 7 8 3 10 4 5
    - step 4: 0 1 2 3 7 8 6 10 4 5
    - step 5: 0 1 2 3 4 8 6 10 7 5
    - step 6: 0 1 2 3 4 5 6 10 7 8
    - step 7: 0 1 2 3 4 5 6 10 7 8
    - step 8: 0 1 2 3 4 5 6 7 10 8
    - step 9: 0 1 2 3 4 5 6 7 8 10
  + Insertion sort
    - step 1: 0 1 2 3 4 5 6 7 8 10
    - step 2: 0 1 2 3 4 5 6 7 8 10
    - step 3: 0 1 2 3 4 5 6 7 8 10
    - step 4: 0 1 2 3 4 5 6 7 8 10
    - step 5: 0 1 2 3 4 5 6 7 8 10
    - step 6: 0 1 2 3 4 5 6 7 8 10
    - step 7: 0 1 2 3 4 5 6 7 8 10
    - step 8: 0 1 2 3 4 5 6 7 8 10
    - step 9: 0 1 2 3 4 5 6 7 8 10
    - step 10: 0 1 2 3 4 5 6 7 8 10
  + Bubble sort
    - step 0: 3 4 5 6 7 8 9 10 1 2
    - step1:
    - step 1.1: 3 4 5 6 7 8 9 10 1 2
    - step 1.2: 3 4 5 6 7 8 9 10 1 2
    - step 1.3: 3 4 5 6 7 8 9 10 1 2
    - step 1.4: 3 4 5 6 7 8 9 10 1 2
    - step 1.5: 3 4 5 6 7 8 9 10 1 2
    - step 1.6: 3 4 5 6 7 8 9 10 1 2
    - step 1.7: 3 4 5 6 7 8 9 10 1 2
    - step 1.8: 3 4 5 6 7 8 9 1 10 2
    - step 1.9: 3 4 5 6 7 8 9 1 2 10
    - step2:
    - step 2.1: 3 4 5 6 7 8 9 1 2 10
    - step 2.2: 3 4 5 6 7 8 9 1 2 10
    - step 2.3: 3 4 5 6 7 8 9 1 2 10
    - step 2.4: 3 4 5 6 7 8 9 1 2 10
    - step 2.5: 3 4 5 6 7 8 9 1 2 10
    - step 2.6: 3 4 5 6 7 8 9 1 2 10
    - step 2.7: 3 4 5 6 7 8 1 9 2 10
    - step 2.8: 3 4 5 6 7 8 1 2 9 10
    - step 2.9: 3 4 5 6 7 8 1 2 9 10
    - step3:
    - step 3.1: 3 4 5 6 7 8 1 2 9 10
    - step 3.2: 3 4 5 6 7 8 1 2 9 10
    - step 3.3: 3 4 5 6 7 8 1 2 9 10
    - step 3.4: 3 4 5 6 7 8 1 2 9 10
    - step 3.5: 3 4 5 6 7 8 1 2 9 10
    - step 3.6: 3 4 5 6 7 1 8 2 9 10
    - step 3.7: 3 4 5 6 7 1 2 8 9 10
    - step 3.8: 3 4 5 6 7 1 2 8 9 10
    - step 3.9: 3 4 5 6 7 1 2 8 9 10
    - step4:
    - step 4.1: 3 4 5 6 7 1 2 8 9 10
    - step 4.2: 3 4 5 6 7 1 2 8 9 10
    - step 4.3: 3 4 5 6 7 1 2 8 9 10
    - step 4.4: 3 4 5 6 7 1 2 8 9 10
    - step 4.5: 3 4 5 6 1 7 2 8 9 10
    - step 4.6: 3 4 5 6 1 2 7 8 9 10
    - step 4.7: 3 4 5 6 1 2 7 8 9 10
    - step 4.8: 3 4 5 6 1 2 7 8 9 10
    - step 4.9: 3 4 5 6 1 2 7 8 9 10
    - step5:
    - step 5.1: 3 4 5 6 1 2 7 8 9 10
    - step 5.2: 3 4 5 6 1 2 7 8 9 10
    - step 5.3: 3 4 5 6 1 2 7 8 9 10
    - step 5.4: 3 4 5 1 6 2 7 8 9 10
    - step 5.5: 3 4 5 1 2 6 7 8 9 10
    - step 5.6: 3 4 5 1 2 6 7 8 9 10
    - step 5.7: 3 4 5 1 2 6 7 8 9 10
    - step 5.8: 3 4 5 1 2 6 7 8 9 10
    - step 5.9: 3 4 5 1 2 6 7 8 9 10
    - step6:
    - step 6.1: 3 4 5 1 2 6 7 8 9 10
    - step 6.2: 3 4 5 1 2 6 7 8 9 10
    - step 6.3: 3 4 1 5 2 6 7 8 9 10
    - step 6.4: 3 4 1 2 5 6 7 8 9 10
    - step 6.5: 3 4 1 2 5 6 7 8 9 10
    - step 6.6: 3 4 1 2 5 6 7 8 9 10
    - step 6.7: 3 4 1 2 5 6 7 8 9 10
    - step 6.8: 3 4 1 2 5 6 7 8 9 10
    - step 6.9: 3 4 1 2 5 6 7 8 9 10
    - step7:
    - step 7.1: 3 4 1 2 5 6 7 8 9 10
    - step 7.2: 3 1 4 2 5 6 7 8 9 10
    - step 7.3: 3 1 2 4 5 6 7 8 9 10
    - step 7.4: 3 1 2 4 5 6 7 8 9 10
    - step 7.5: 3 1 2 4 5 6 7 8 9 10
    - step 7.6: 3 1 2 4 5 6 7 8 9 10
    - step 7.7: 3 1 2 4 5 6 7 8 9 10
    - step 7.8: 3 1 2 4 5 6 7 8 9 10
    - step 7.9: 3 1 2 4 5 6 7 8 9 10
    - step8:
    - step 8.1: 1 3 2 4 5 6 7 8 9 10
    - step 8.2: 1 2 3 4 5 6 7 8 9 10
    - step 8.3: 1 2 3 4 5 6 7 8 9 10
    - step 8.4: 1 2 3 4 5 6 7 8 9 10
    - step 8.5: 1 2 3 4 5 6 7 8 9 10
    - step 8.6: 1 2 3 4 5 6 7 8 9 10
    - step 8.7: 1 2 3 4 5 6 7 8 9 10
    - step 8.8: 1 2 3 4 5 6 7 8 9 10
    - step 8.9: 1 2 3 4 5 6 7 8 9 10
    - step9:
    - step 9.1: 1 2 3 4 5 6 7 8 9 10
    - step 9.2: 1 2 3 4 5 6 7 8 9 10
    - step 9.3: 1 2 3 4 5 6 7 8 9 10
    - step 9.4: 1 2 3 4 5 6 7 8 9 10
    - step 9.5: 1 2 3 4 5 6 7 8 9 10
    - step 9.6: 1 2 3 4 5 6 7 8 9 10
    - step 9.7: 1 2 3 4 5 6 7 8 9 10
    - step 9.8: 1 2 3 4 5 6 7 8 9 10
    - step 9.9: 1 2 3 4 5 6 7 8 9 10
  + Merge sort



* + Quick sort



* + Heap sort
  + Radix sort



1. Write a program to sort (ascending sort) the list by following methods:

* Selection sort

```*void* SelectionSort() {

    for (*int* i = 0; i < n; i++) {

*int* id = i;

        for (*int* j = i+1; j < n; j++) {

            if (a[j] < a[id]) {

                id = j;

            }

        }

        swap(a[i],a[id]);

        for (*int* k = 0; k < n; k++) cout << a[k] << " ";

        cout << endl;

}

}

* Insertion sort

```*void* InsertionSort() {

    for (*int* i = 1; i < n; i++) {

*int* pos = i - 1, x = a[i];

        while (pos >= 0 && a[pos] > x) {

            a[pos+1] = a[pos];

            --pos;

        }

        a[pos+1] = x;

        for (*int* j = 0; j < n; j++) cout << a[j] << " ";

        cout << endl;

    }

}

* Bubble sort

```*void* Bubble\_Sort() {

    for (*int* i = 0; i < n; i++) {

        for (*int* j = i+1; j < n; j++) {

            if (a[j] < a[i]) {

                swap(a[i],a[j]); *//ptu max se chuyen xuong cuoi dong*

            }

        }

        for (*int* k = 0; k < n; k++) cout << a[k] << " ";

        cout << endl;

    }

}

* Merge sort

```*void* merge(*int* *a*[], *int* *left*, *int* *mid*, *int* *right*) {*//Thao tac tron*

*int* n1 = *mid*-*left*+1, n2 = *right*-*mid*;

*int* l[n1], r[n2];

    for (*int* i = *left*; i <= *mid*; i++) {

        l[i-*left*] = *a*[i];

    }

    for (*int* i = *mid*+1; i <= *right*; i++) {

        r[i-*mid*-1] = *a*[i];

    }

*int* i = 0, j = 0, cnt = *left*;

    while (i < n1 && j < n2) {

        if (l[i] <= r[j]) {

*a*[cnt++] = l[i++];

        }

        else *a*[cnt++] = r[j++];

    }

    while (i < n1) *a*[cnt++] = l[i++];

while (j < n2) *a*[cnt++] = r[j++];

}

*void* mergesort(*int* *a*[], *int* *l*, *int* *r*) {*//chia mang*

    if (*l* < *r*) {

*int* m = *l* + (*r*-*l*)/2;

        mergesort(*a*,*l*,m);

        mergesort(*a*,m+1,*r*);

        merge(*a*,*l*,m, *r*);

    }

}

* Quick sort

```*int* partition(*int* *a*[], *int* *l*, *int* *r*) {

*int* pivot = *a*[*r*];*//phan tu cuoi cung*

*int* i = *l*-1;

    for (*int* j = *l*; j < *r*; j++) {

        if (*a*[j] <= pivot) {

            ++i;

            swap(*a*[i], *a*[j]);

        }

    }

    ++i;

    swap(*a*[i], *a*[*r*]);

    return i;

}

*void* quick\_Sort(*int* *a*[], *int* *l*, *int* *r*) {

    if (*l* < *r*) {

*int* m = *l* + (*r*-*l*)/2;

*int* p = partition(*a*,*l*,*r*);

        Quick\_Sort(*a*,*l*,p-1);

        Quick\_Sort(*a*,p+1,*r*);

    }

}