EECS2040 Data Structure Hw #6 (Chapter 7 Sorting, Chapter 8 Hashing)
due date 6/20/2021
by 107061123, 孫元駿

Part 1

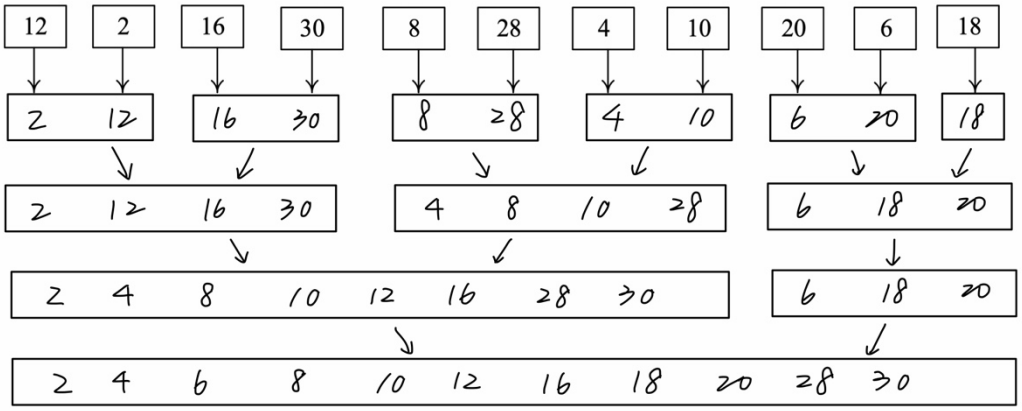
1. (50%) The list L: (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) is to be sorted by various sorting algorithm.
   (a) Write the status of the list at the end of each iteration of the **for** loop of InsertionSort (Program 7.5). Trace the program; understand it. Put your answer in the following table. (add necessary rows for your answer)

j	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
-	**12**	2	16	30	8	28	4	10	20	6	18
2	**2**	**12**	16	30	8	28	4	10	20	6	18
3	**2**	**12**	**16**	30	8	28	4	10	20	6	18
4	**2**	**12**	**16**	**30**	8	28	4	10	20	6	18
5	**2**	**8**	**12**	**16**	**30**	28	4	10	20	6	18
6	**2**	**8**	**12**	**16**	**28**	**30**	4	10	20	6	18
7	**2**	**4**	**8**	**12**	**16**	**28**	**30**	10	20	6	18
8	**2**	**4**	**8**	**10**	**12**	**16**	**28**	**30**	20	6	18
9	**2**	**4**	**8**	**10**	**12**	**16**	**20**	**28**	**30**	6	18
10	**2**	**4**	**6**	**8**	**10**	**12**	**16**	**20**	**28**	**30**	18
11	**2**	**4**	**6**	**8**	**10**	**12**	**16**	**18**	**20**	**28**	**30**

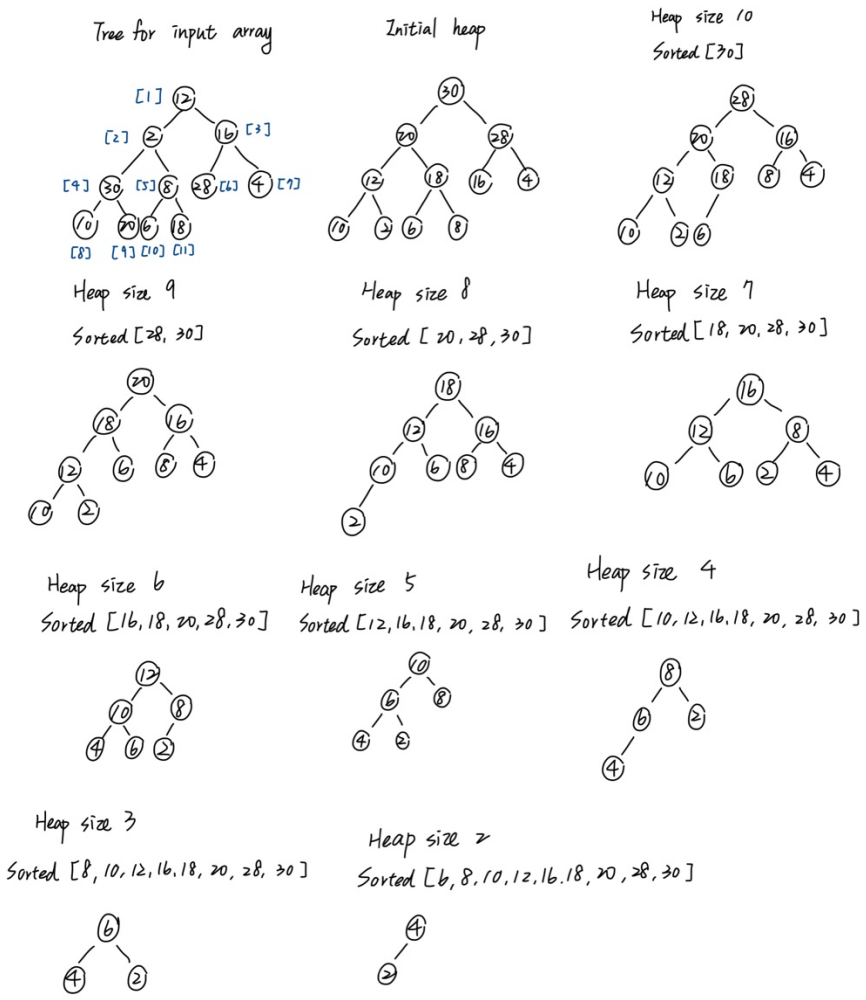
   (b) Trace Program 7.6 QuickSort, use it on the list L, and draw a figure similar to Figure 7.1 Quick Sort example starting with the list L. Put your answer in the following table. (add necessary rows for your answer)

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	left	right
[12	2	16	30	8	28	4	10	20	6	18]	1	11
[4	2	6	10	8]	12	[28	30	20	16	18]	1	5
[2]	4	[6	10	8]	12	[28	30	20	16	18]	1	1
2	4	[6	10	8]	12	[28	30	20	16	18]	3	5
2	4	6	[10	8]	12	[28	30	20	16	18]	4	5
2	4	6	8	10	12	[28	30	20	16	18]	7	11
2	4	6	8	10	12	[16	18	20]	28	[30]	7	9
2	4	6	8	10	12	16	[18	20]	28	[30]	8	9
2	4	6	8	10	12	16	18	20	28	[30]	11	11
2	4	6	8	10	12	16	18	20	28	30		

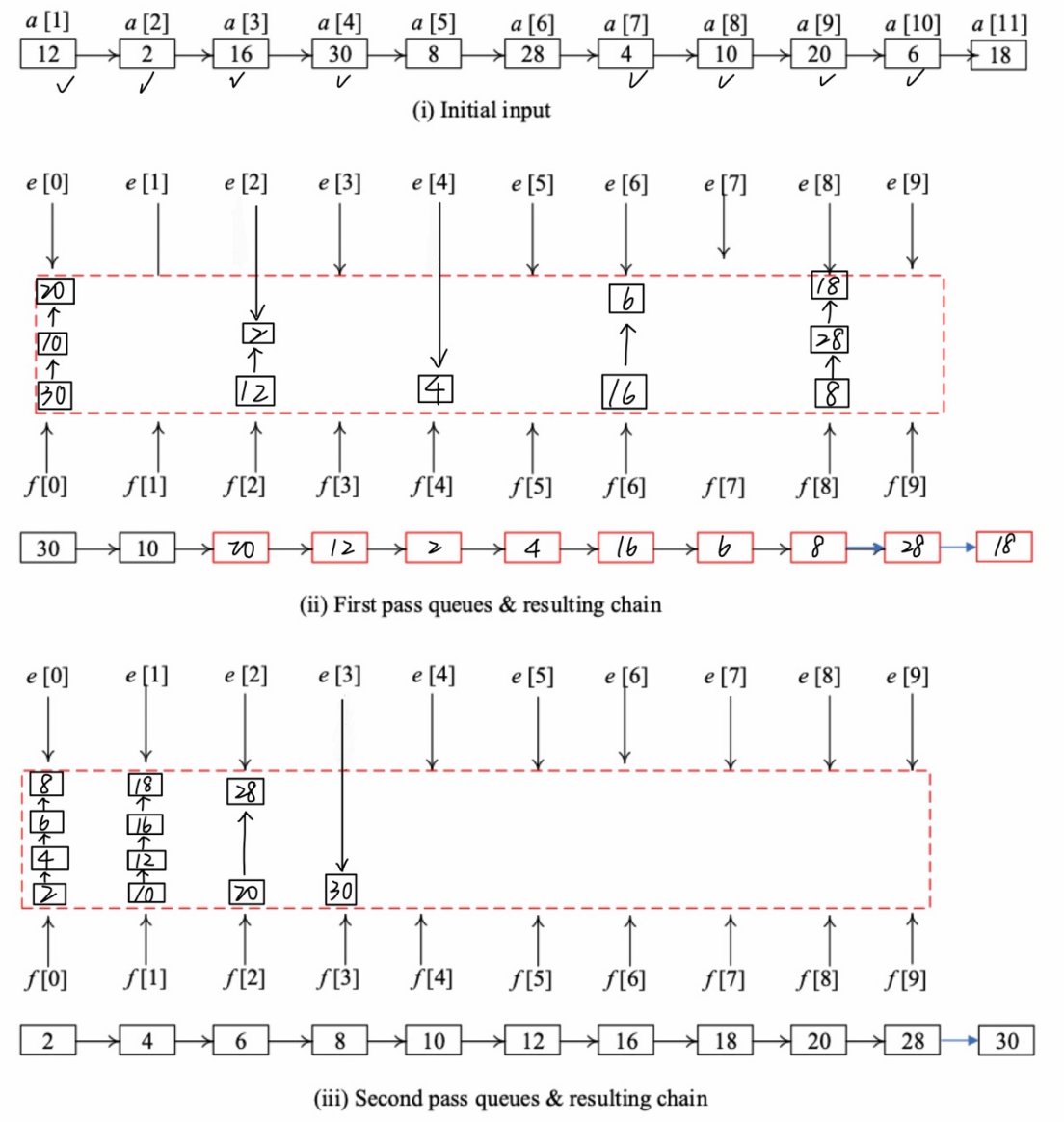
(c) Write the status of the list L at the end of each phase of MergeSort (Program 7.9), i.e., draw the Merge tree (similar to Figure 7.4 in textbook) of this problem.



(d) Write the status of the list L at the end of the first **for** loop as well as at the end of the second **for** loop of HeapSort (Program 7.14), i.e., you need to draw the following trees for: 1) input array, 2) initial heap, and 9 more trees with heap size from 10 down to 2 with corresponding sorted array. You can refer to similar results shown in Figure 7.8 in textbook.



(e) Write the status of the list L at the end of each pass of RadixSort (Program 7.15), using r = 10. That is fill the missing parts (the node boxes with numbers and arrows between e[j] and f[j] enclosed by red dashed rectangle in (ii) and (iii) part of the following figure, and the missing numbers in the resulting chain (red boxes) in (ii).)



2. (10%) QuickSort (Program 7.6) is an unstable sorting method. Give an example of an input list in which the order of records with equal keys is not preserved.

If we quick sort a key-value list

Ex. (2,a) (3,b) (3,c) (1,d)

swap

[(1,d)] (2,a) [(3,c) (3,b)]

(1,d) (2,a) [(3,c) (3,b)]

(1,d) (2,a) (3,c) [(3,b)]

(1,d) (2,a) (3,c) (3,b)

By the example, we could know QuickSort is unstable, because the same key with different value might be able to change after sort.

3. (10%) Show that MergeSort (Program 7.9) is stable
   If we merge two lists A and B, and two lists are all been sorted. If there are two same key, then there are two possible situations, the first one is two same keys are in the same list, then by the code in Program 7.7, we could know that we after merge they will still in order, because the key will store in mergeList one by one after comparing the key in two list. The second situation is two keys are in different lists, then also by Program 7.7, we know that "if(initList[i1] <= initList[i]2) mergedList[iResult] = initList[i1];" so we know that the same key in both list, one in the front list will be recorded first. So that we know whether in which situation the same list will still be stable, so MergeSort in Program 7.9 is stable.

4. (10%) If we have n records with integer keys in the range $[0,n^2)$,then they can be sorted in O(nlogn) time using Heap Sort or Merge Sort. Radix Sort on a single key (i.e., d = 1 and $r = n^2$) takes $O(n^2)$ time. Show how to interpret the keys as two subkeys so that Radix Sort will take only O(n) time to sort n records. (Hint: Each key, K_i, may be written as $K_i = K_i^1 * n + K_i^2$ with K_i^1 and K_i^2 integers in the range [0,n).)

   By $K_i = K_i' n + K_i^>$, we could know that $K_i' = \frac{K_i}{n}$ and $K_i^> = K_i \% n$
   then we can see K_i as a 2-tuple key $(K_i', K_i^>)$ with radix = n
   And we know that we need $O(d(n+r))$ time to sort by RadixSort
   Thus, we can sort n's K_i records in $O(d(n+r)) = O(2(n+n)) = O(4n) = O(n)$
   So, it only takes $O(n)$ time to sort n records. ※

5. (10%) Show that the hash function h(k) = k%17 does not satisfy the one-way property, weak collision resistance, or strong collision resistance.

$h(k) = k\%17 = C \quad C = [0, 16]$

1. One-way property : If given 'C' in a hash table
   We can find $k = C + 17 \cdot n \quad n = [0, \infty)$

2. Weak collision resistance : if given a number X, $h(k) = p\%17$
   we can find X' symonym of X, that $x' = x + 17$
   that $h(x) = h(x')$
   ⇒ h(x) is not weak collision resistance

3. Strong collision resistance : We can find a pair $(x, y) = (17m + c, 17n + c)$
   that $x \neq y$, but $h(x) = h(y) = C$
   ⇒ h(x) is not strong collision resistance

6. (10%) The probability $P(u)$ that an arbitrary query made after u updates results in a filter error is given by $P(u) = e^{-u/n}\left(1 - e^{-uh/m}\right)^h$. By differentiating $P(u)$ with respect to h, show that $P(u)$ is minimized when $h = (\log_e 2)m/u$.

$$P(u) = e^{-\frac{u}{n}}\left(1 - e^{-\frac{u}{m}h}\right)^h$$
$$= e^{-\frac{u}{n}} e^{h\ln\left(1 - e^{-\frac{u}{m}h}\right)}$$
$$\frac{\partial}{\partial h}P(u) = e^{-\frac{u}{n}} \frac{\partial}{\partial h} e^{h\ln\left(1 - e^{-\frac{u}{m}h}\right)}$$
$$= e^{-\frac{u}{n}} \cdot e^{h\ln\left(1 - e^{-\frac{u}{m}h}\right)} \left[\frac{\partial}{\partial h} h\ln\left(1 - e^{-\frac{u}{m}h}\right)\right]$$
$$= e^{-\frac{u}{n}} \cdot e^{h\ln\left(1 - e^{-\frac{u}{m}h}\right)} \left[\ln\left(1 - e^{-\frac{u}{m}h}\right) + h\frac{1}{1 - e^{-\frac{u}{m}h}} \cdot \frac{u}{m} e^{-\frac{u}{m}h}\right]$$

Find minimized set $\frac{\partial}{\partial h}P(u) = 0$

$$\ln\left(1 - e^{-\frac{u}{m}h}\right) + h\frac{1}{1 - e^{-\frac{u}{m}h}}\frac{u}{m}e^{-\frac{u}{m}h} = 0$$

Let $e^{-\frac{u}{m}h} = X \quad h = -\frac{m}{u}\ln X$

$$\ln(1 - X) + h\frac{1}{1 - X}\frac{u}{m}X = \ln(1 - X) - \frac{X}{1 - X}\ln X = 0$$

$$(1 - X)\ln(1 - X) = X\ln X$$

$$X = \frac{1}{2}$$

$$h = -\frac{m}{u}\ln\frac{1}{2} = \frac{m}{u}\ln 2$$

show P(u) is minimized that $h = \frac{m}{u}\ln 2$