EE2410 Data Structure Hw #6 (Chapter 7 Sorting, Chapter 8 Hashing)
due date 6/16/2024 (Sun.), 23:59

Format: Use MS Word to **edit this file** by directly typing your student number and name in above blanks and your answer to each homework problem right in the **Sol:** blanks as shown below. Then save your file as **Hw6-SNo.pdf**, where SNo is your student number. Submit the **Hw6-SNo.pdf** file via eLearn. The grading will be based on the correctness of your answers to the problems, and the **format requirement**. Fail to comply with the aforementioned format (file name, header, problem, answer, problem, answer,…), will certainly degrade your score. If you have any questions, please feel free to ask. Submit your homework before the deadline (midnight of 6/13 Sun.). Fail to comply (**late** homework) will have ZERO score. **Copy** homework will have ZERO score (both parties) and SERIOUS consequences.

Sorting:

1. (10%) 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)

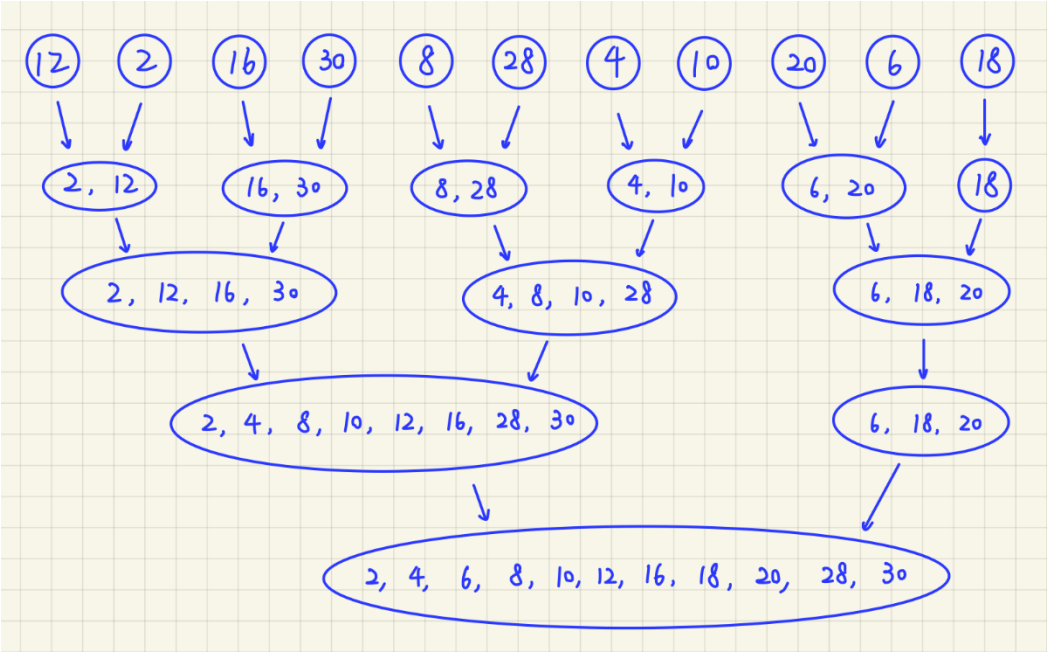
j	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]	[11]
1	12	2	16	30	8	28	4	10	20	6	18
2	2	12	16	30	8	28	4	10	20	6	18
3	2	12	16	30	8	28	4	10	20	6	18
4	2	12	16	30	8	28	4	10	20	6	18
5	2	8	12	16	30	28	4	10	20	6	18
6	2	8	12	16	28	30	4	10	20	6	18
7	2	4	8	12	16	28	30	10	20	6	18
8	2	4	8	10	12	16	28	30	20	6	18
9	2	4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

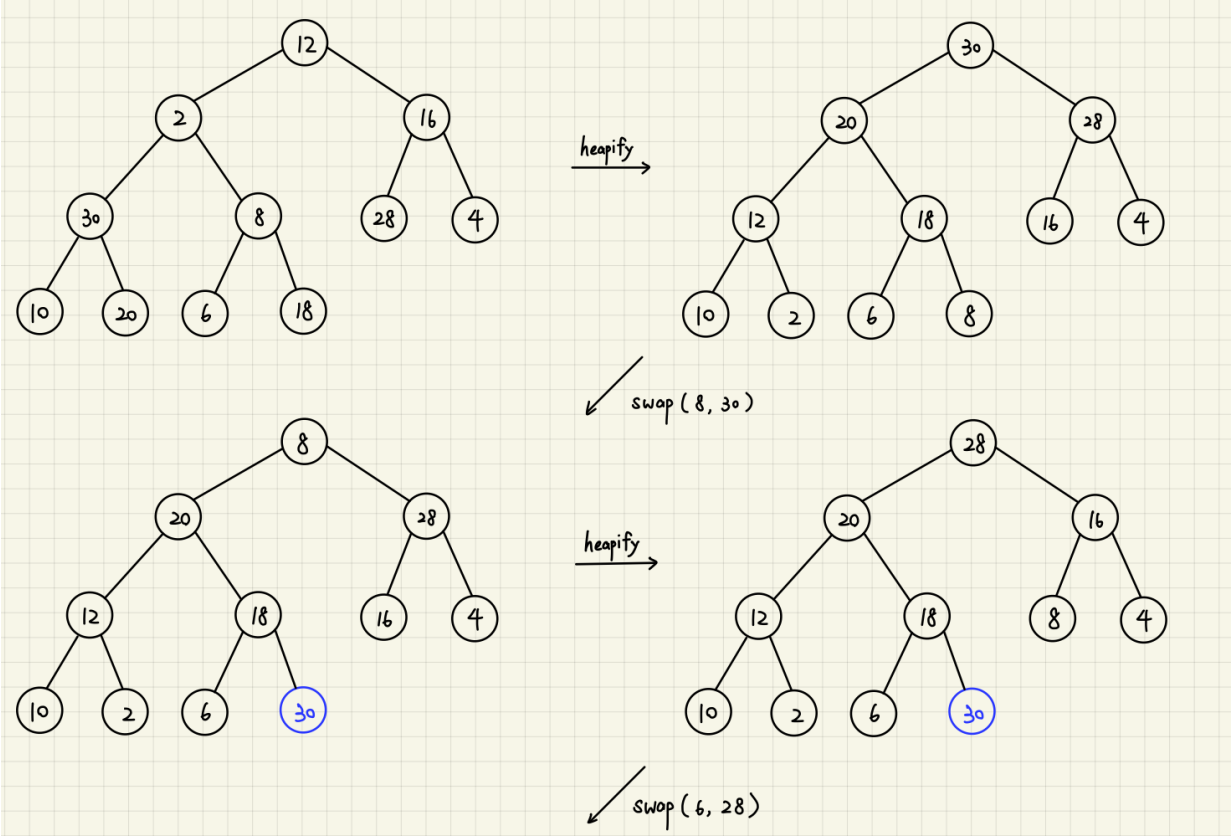
R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	R_{11}	left	right
[12	2	16	30	8	28	4	10	20	6	18]	1	11
[12	2	**16**	30	8	28	4	10	20	**6**	18]	1	11
[12	2	6	**30**	8	28	4	**10**	20	16	18]	1	11
[12	2	6	10	8	**28**	**4**	30	20	16	18]	1	11

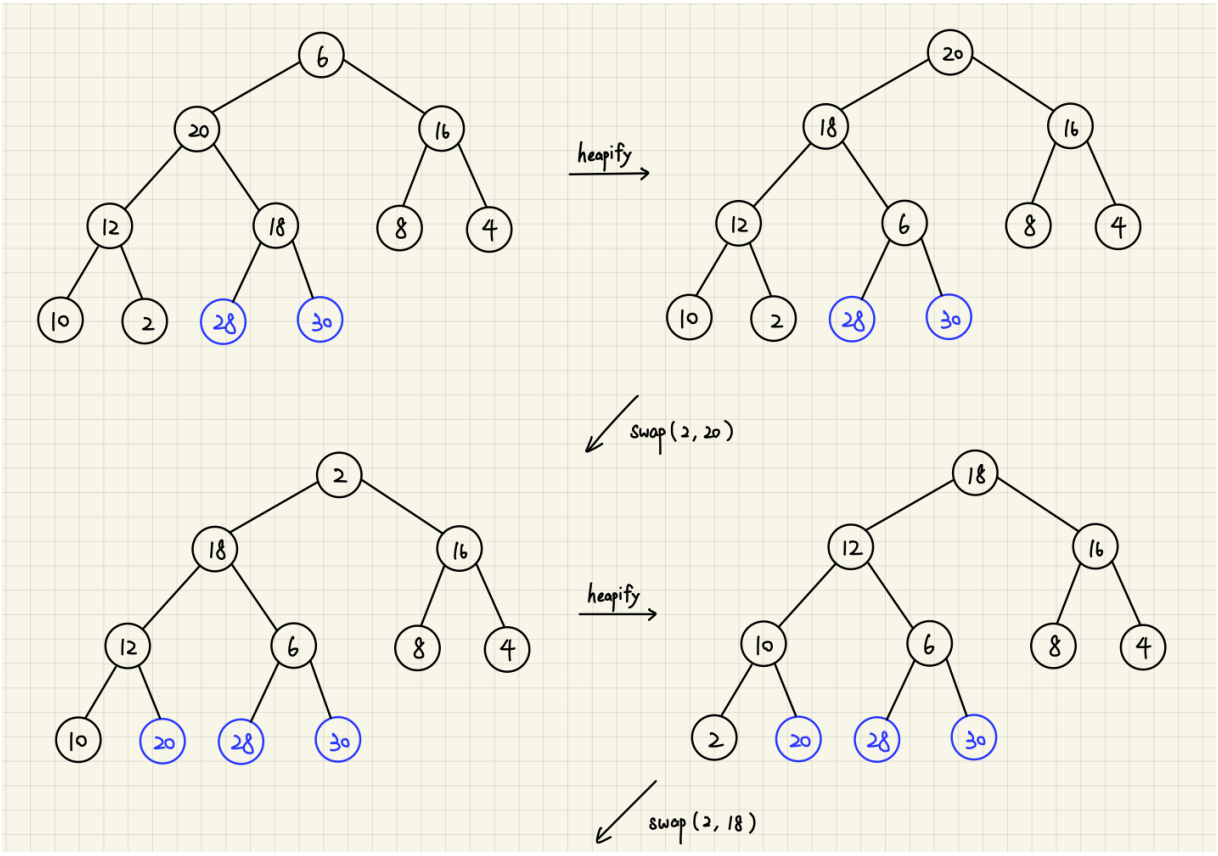
[12	2	6	10	8	**4**	**28**	30	20	16	18]	//i=7 j=6	//swap a[j] with pivot
[4	2	6	10	8]	12	[28	30	20	16	18]	1	5
[4	**2**	**6**	10	8]	12	[28	30	20	16	18]	//i=3 j=2	//swap a[j] with pivot
[2]	4	[6	10	8]	12	[28	30	20	16	18]	1	1
2	4	[6	10	8]	12	[28	30	20	16	18]	3	5
2	4	[**6**	10	8]	12	[28	30	20	16	18]	//i=3 j=3	//swap a[j] with pivot
2	4	6	[10	8]	12	[28	30	20	16	18]	4	5
2	4	6	[10	**8**]	12	[28	30	20	16	18]	//i=6 j=5	//swap a[j] with pivot
2	4	6	8	10	12	[28	30	20	16	18]	7	11
2	4	6	8	10	12	[28	**30**	20	16	**18**]	7	11
2	4	6	8	10	12	[28	18	20	**16**	**30**]	//i=11 j=10	//swap a[j] with pivot
2	4	6	8	10	12	[16	18	20]	28	[30]	7	9
2	4	6	8	10	12	[**16**	**18**	20]	28	[30]	//i=8 j=7	//swap a[j] with pivot
2	4	6	8	10	12	16	[18	20]	28	[30]	8	9
2	4	6	8	10	12	16	[**18**	**20**]	28	[30]	//i=9 j=8	//swap a[j] with pivot
2	4	6	8	10	12	16	18	[20]	28	[30]	9	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	//sort complete	

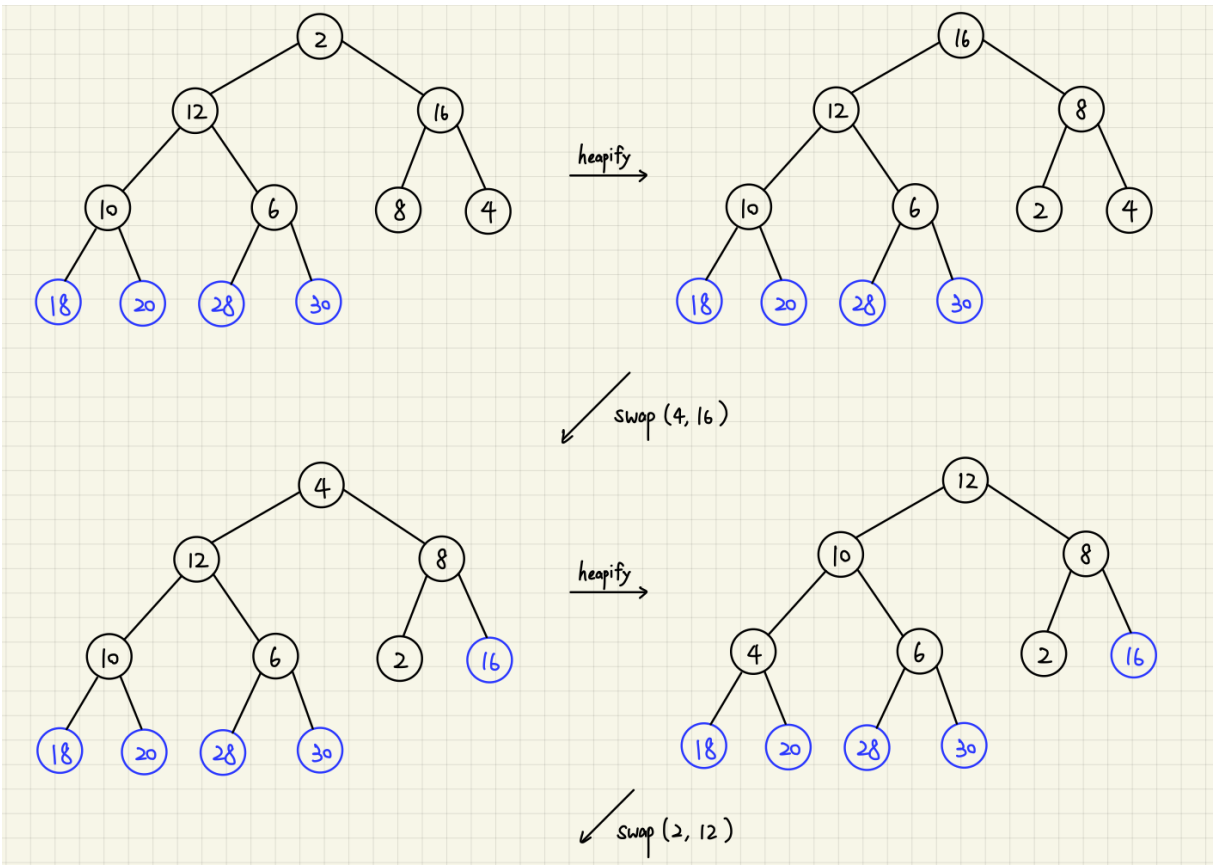
(c) Write the status of the list L at the end of each phase of MergeSort (Program 7.9), i.e., draw the Merge tree (similar to Figure 7.4 in textbook) of this problem.

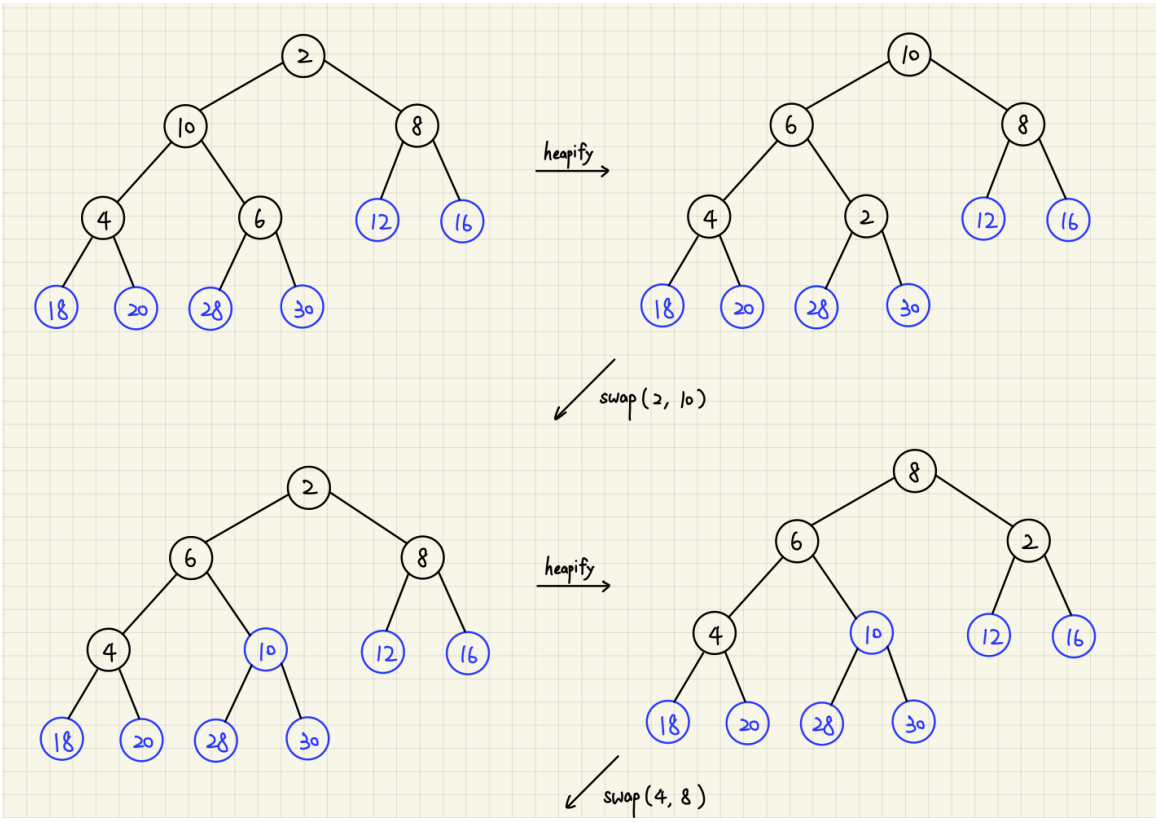


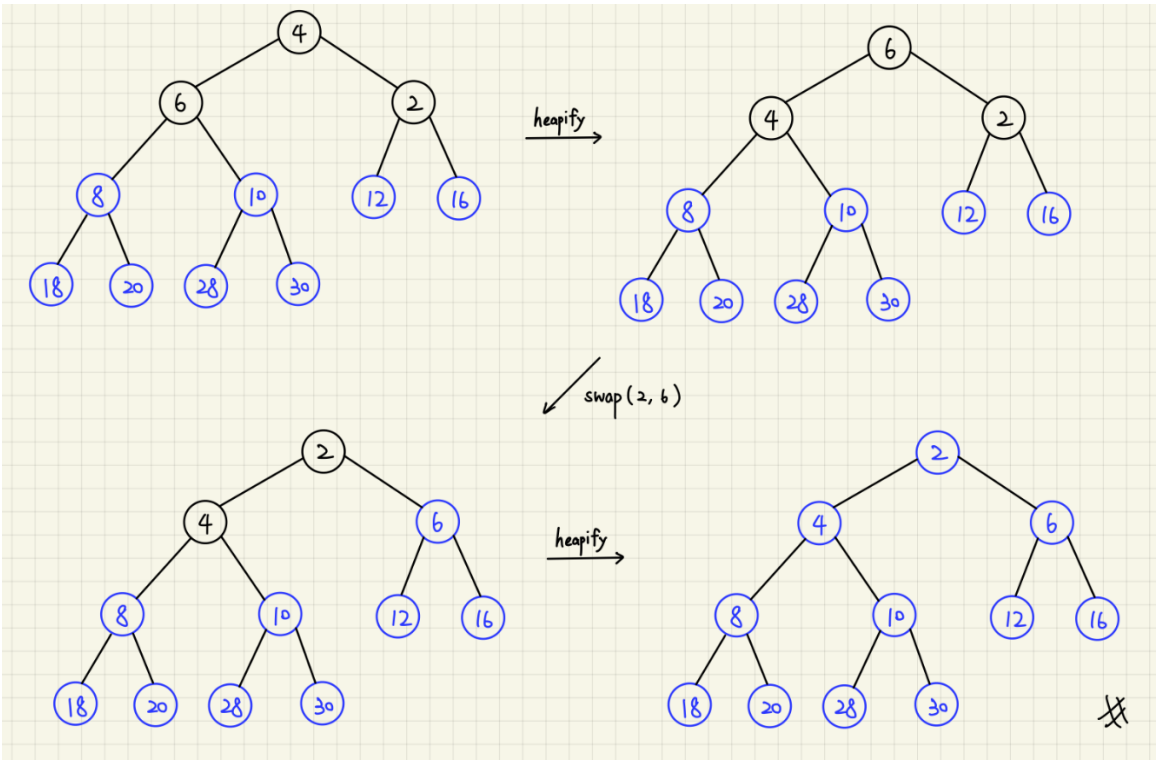
(d) Write the status of the list L at the end of the first **for** loop as well as at the end of the second **for** loop of HeapSort (Program 7.14), i.e., you need to draw the following trees for: 1) input array, 2) initial heap, and 9 more trees with heap size from 10 down to 2 with corresponding sorted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



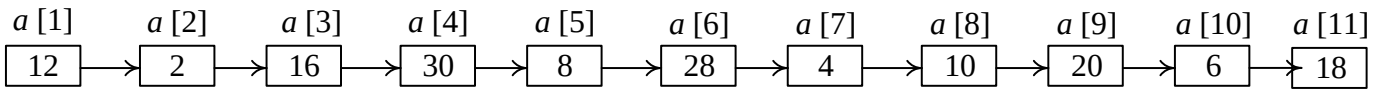

6
20
16
12
18
8
4
10
2
28
30
heapify
20
18
16
12
6
8
4
10
2
28
30
swap(2, 20)
2
18
16
12
6
8
4
10
20
28
30
heapify
18
12
16
10
6
8
4
2
20
28
30
swap(2, 18)


2
12
16
10
6
8
4
18
20
28
30
heapify
16
12
8
10
6
2
4
18
20
28
30
swap(4, 16)
4
12
8
10
6
2
16
18
20
28
30
heapify
12
10
8
4
6
2
16
18
20
28
30
swap(2, 12)

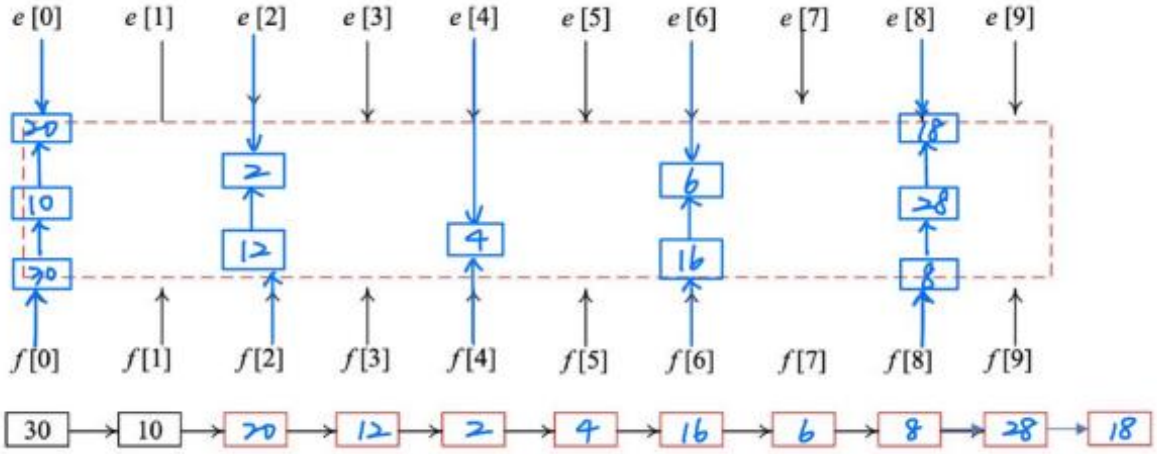

heapify
swap(2, 10)
heapify
swap(4, 8)


heapify
swap(2, 6)
heapify

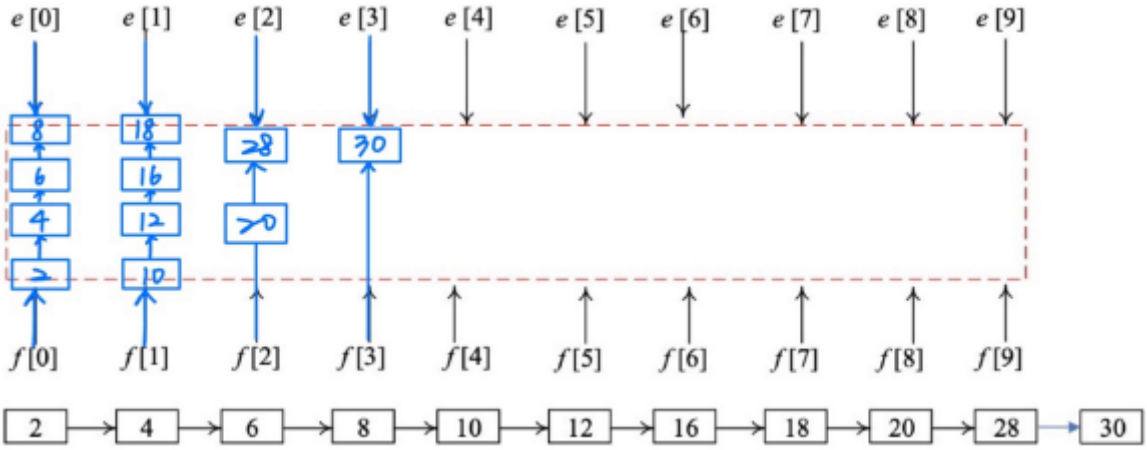
(e) Write the status of the list L at the end of each pass of RadixSort (Program 7.15), using r = 10. That is fill the missing parts (the node boxes with numbers and arrows between e[j] and f[j] enclosed by red dashed rectangle in (ii) and (iii) part of the following figure, and the missing numbers in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



(i) Initial input



(ii) First-pass queues and resulting chain\



(iii) Second-pass queues and resulting chain

Hashing:

2. (5%) (a) Briefly explain the one-way property, weak collision resistance, or strong collision resistance regarding hash function. (b) Show that the hash function h(k) = k%17 does not satisfy the one-way property, weak collision resistance, or strong collision resistance.

Sol:

(a) One-way property:

Given a value c and a hash function h(k), it's hard to find a key x such that h(x) = c.

Weak collision resistance:

Given h(x) = c, it's hard to find a y such that h(y) also equals to c.

Strong collision resistance:

It's hard to find a pair of keys (x, y) such that h(x) = h(y).

(b) One-way property:

For h(k) = k % 17, if h(k) = c, we can easily derive the inverse function k = 17n + c (n is an integer), thus it does not satisfy one-way property.

Weak collision resistance:

For h(k) = k % 17, if h(x) = c, we can easily construct y = 17x + c, then h(y) = c. Thus, it does not satisfy weak collision resistance.

Strong collision resistance:

Counterexample: we can easily find (18, 35) where h(18) = 1 = h(35). Thus, it does not satisfy strong collision resistance.

3. (5%) 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 $h = (\log_e 2)m/u$.

Sol:

$$\ln(P(u)) = -\frac{u}{n} + h\ln(1 - e^{-uh/m})$$

$$\frac{\partial}{\partial h}\ln(P(u)) = \frac{\frac{\partial}{\partial h}P(u)}{P(u)} = \ln(1 - e^{-uh/m}) + h \cdot \frac{\frac{u}{m}e^{-uh/m}}{1 - e^{-uh/m}}$$

$$\Rightarrow \frac{\partial}{\partial h}P(u) = P(u)\left[\ln(1 - e^{-uh/m}) + h \cdot \frac{\frac{u}{m}e^{-uh/m}}{1 - e^{-uh/m}}\right] = 0$$

$$\Rightarrow \ln(1 - e^{-uh/m}) + h \cdot \frac{\frac{u}{m}e^{-uh/m}}{1 - e^{-uh/m}} = 0 \quad \cdots ①$$

Let $x = 1 - e^{-uh/m} \Rightarrow e^{-uh/m} = 1 - x \Rightarrow h = -\frac{m}{u}\ln(1-x)$

$$① \Rightarrow \ln x + \frac{-(1-x)\ln(1-x)}{x} = 0,$$

$$x\ln x = (1-x)\ln(1-x) = 0, \quad x = \frac{1}{2}$$

$$\Rightarrow h = \frac{-m}{u}\ln(1-\tfrac{1}{2}) = \frac{m}{u}\ln 2$$

$\Rightarrow P(u)$ has minimum when $h = \frac{m}{u}\ln 2$ ※