**EECS2040 Data Structure Hw #6 (Chapter 7 Sorting, Chapter 8 Hashing)**

**due date 6/13/2022 by 109070025 林泓錩**

**Part 1 (2% of final Grade)**

1. (50%) The list L: (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18) is to be sorted by various sorting algorithm.
2. 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)

**Sol:**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 |

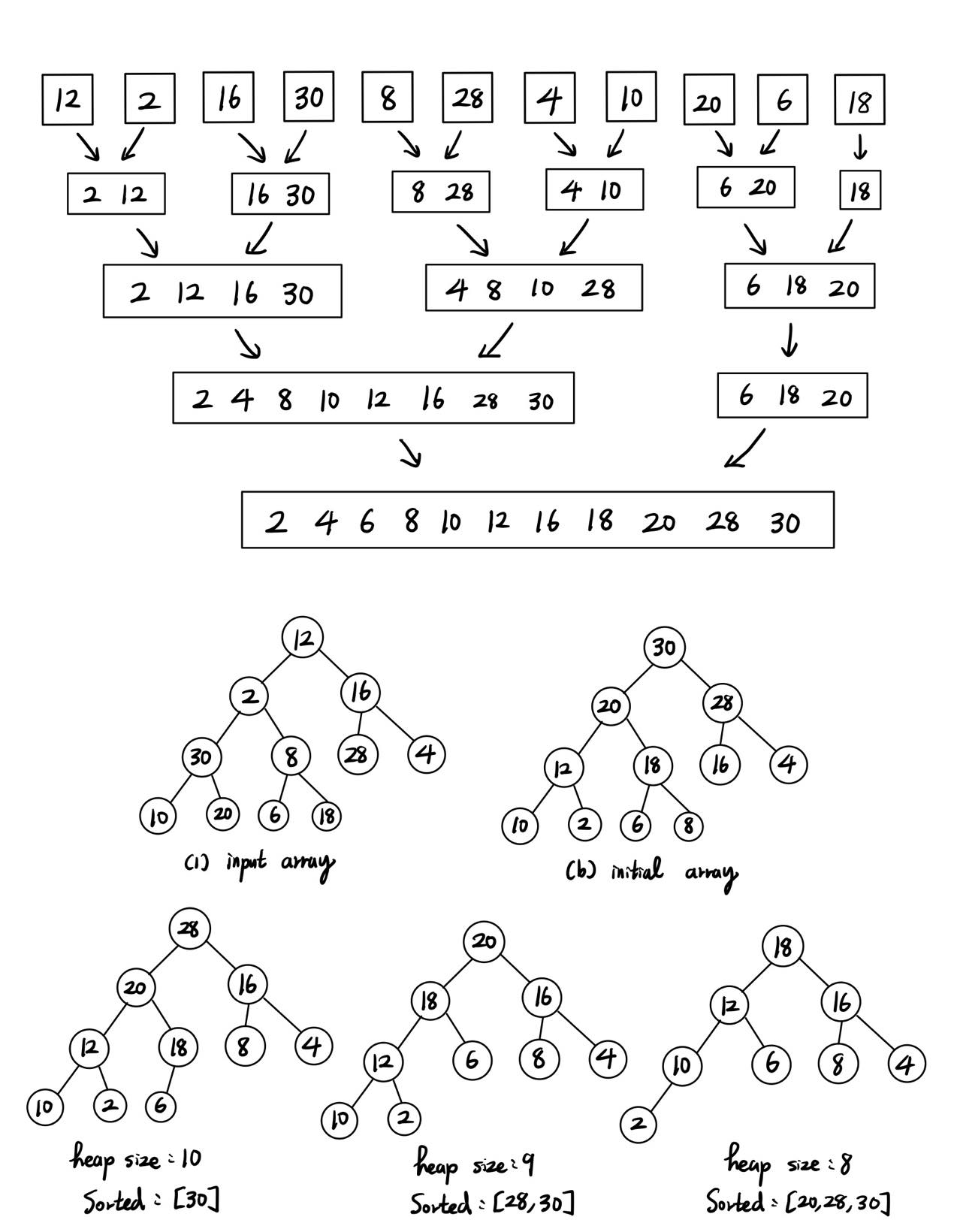
1. Trace Program 7.6 QuickSort, use it on the list L, and draw a figure similar to Figure 7.1 Quick Sort example starting with the list L. Put your answer in the following table. (add necessary rows for your answer)

**Sol:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 | 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 | 4 | 4 |
| 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 |  |  |

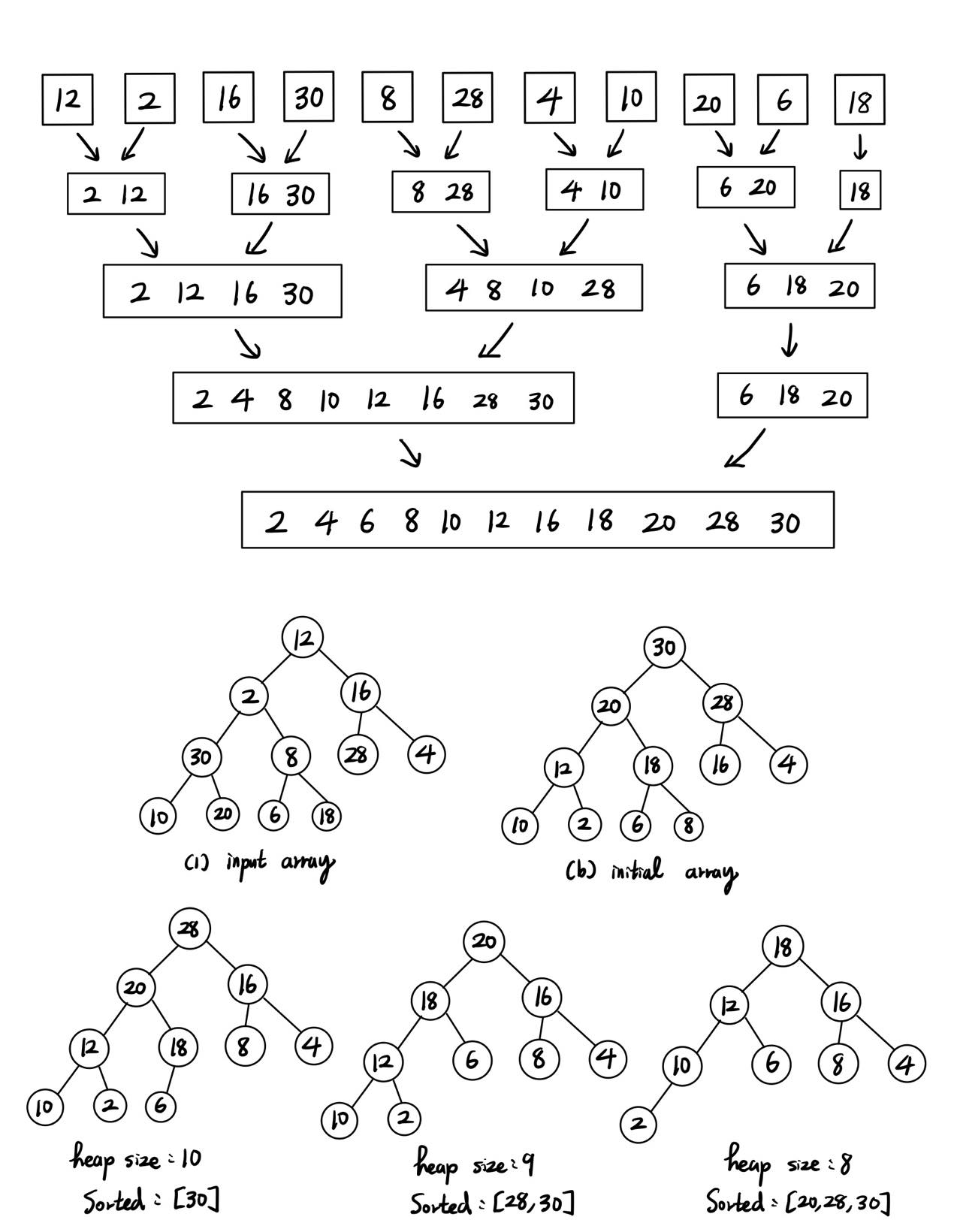
1. Write the status of the list L at the end of each phase of MergeSort (Program 7.9), i.e., draw the Merge tree (similar to Figure 7.4 in textbook) of this problem.

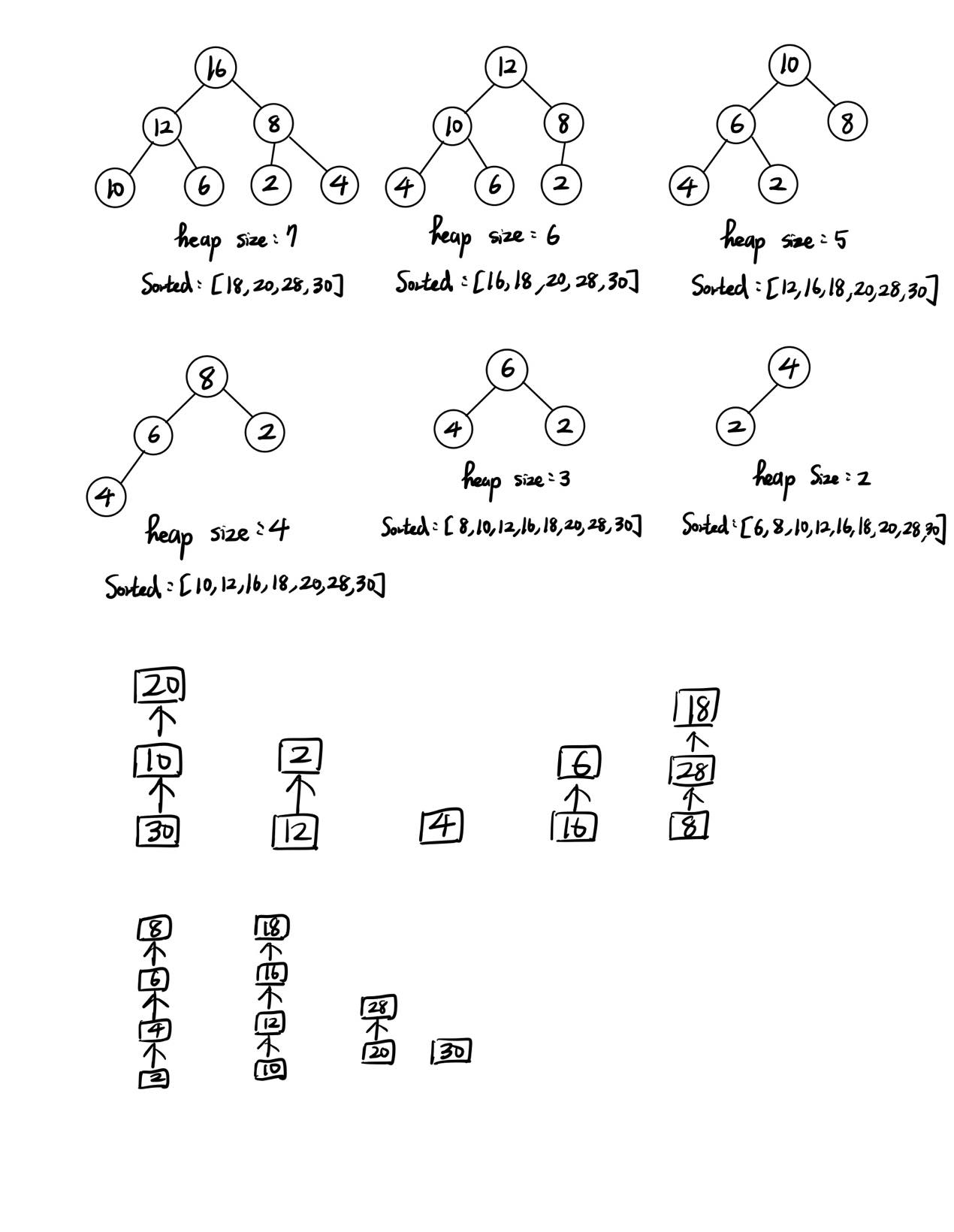
**Sol:**



1. 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.

**Sol:**





Finally sorted sequence is: [2,4,6,8,10,12,16,18,20,28,30]

1. 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).)

**Sol:**



20

28

10

12

16

18

30

8

2

4

6

8

16

6

4

2

28

18

12

30

10

20

18

28

8

4

16

6

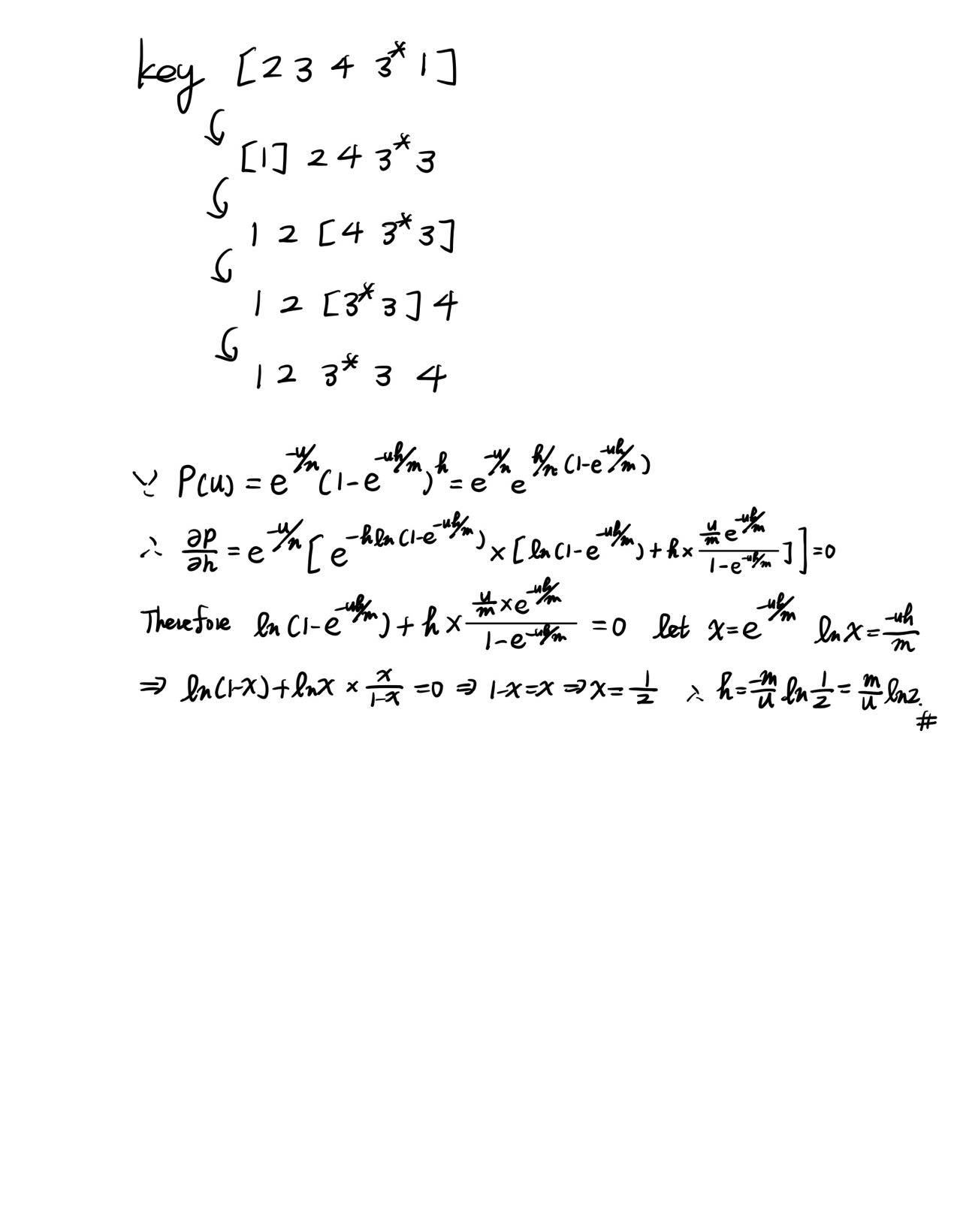
2

12

20

1. (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.

**Sol:**



1. (10%) Show that MergeSort (Program 7.9) is stable

**Sol:**

Because when we Merge two list, we use “if (initList[i1] <= initList[i2])”, since it is“<=”, we can sure that the relative order of same key would not be changed.

1. (10%) If we have n records with integer keys in the range [0,n2),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 = n2) takes O(n2) 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, Ki, may be written as Ki = Ki1\*n + Ki2 with Ki1 and Ki2 integers in the range [0,n).)

**Sol:**

Interpret Ki = Ki1\*n + Ki2, where Ki1= [(Ki)/n], then Ki1 is in the range [0,n)

And Ki2 = Ki % n, Ki2 also in the range [0,n). therefore complexity = O(d(n+N)) = O(2(n+n)) = O(4n) = O(n)。

1. (10%) (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:**

1. One way property: if h(x) = c and I give you c, it is impossible calculate x from c. Weak collision resistance: suppose hash function is h(x). If give a x, it is hardly to find a x’ different from x such that h(x) = (x’). Strong collision resistance: suppose hash function is h(x). it is hardly to find a pair x1 and x2 such that h(x1) = h(x2).
2. For one-way property, given c, it’s easy find the x such that h(x) = c, where

x=c+17\*n, so it doesn’t satisfy one-way property. For weak collision resistance, if given x, it’s easy to find y>0 and y = x+n×17, where n is integer, such that h(x) = h(y), so it doesn’t satisfy the weak collision resistance.

For strong collision resistance, it’s easy to find a pair (x,y) that x,y>0 and

x=17×n+c, y=m×17+c, where n,m is integer, such that h(x) = h(y),so it doesn’t

satisfy the strong collision resistance.

1. (10%) The probability *P*(*u*) that an arbitrary query made after *u* updates results in a filter error is given by . By differentiating *P*(*u*) with respect to *h*, show that *P*(*u*) is minimized when *h* = (log*e*2)*m*/*u*.

**Sol:**

