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

**due date 6/13/2022**

**by 108011235 陳昭維**

**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)

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

<answer>



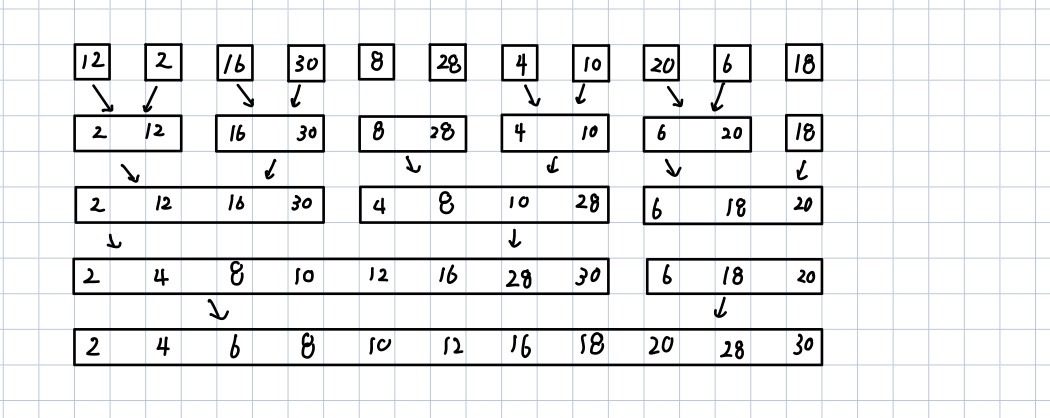
1. Trace Program 7.6 QuickSort, use it on the list L, and draw a figure similar to Figure 7.1 Quick Sort example starting with the list L. Put your answer in the following table. (add necessary rows for your answer)

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

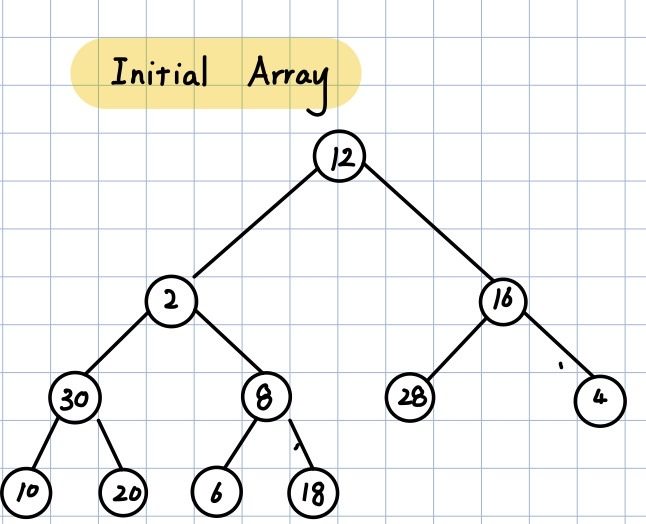
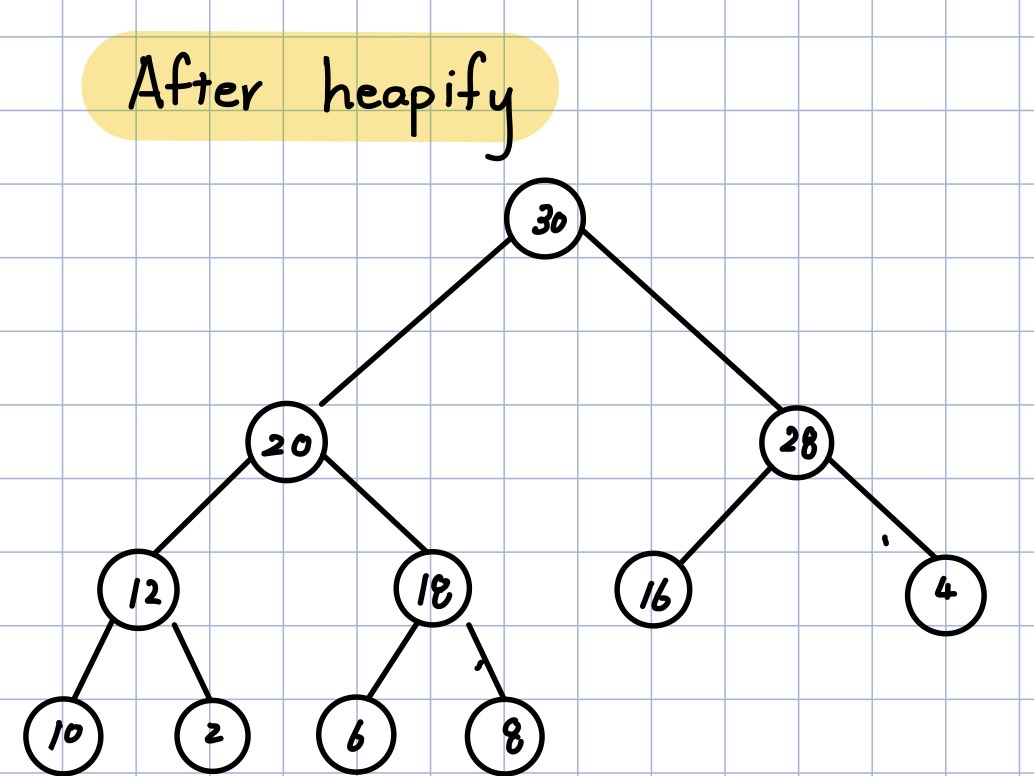
<answer>

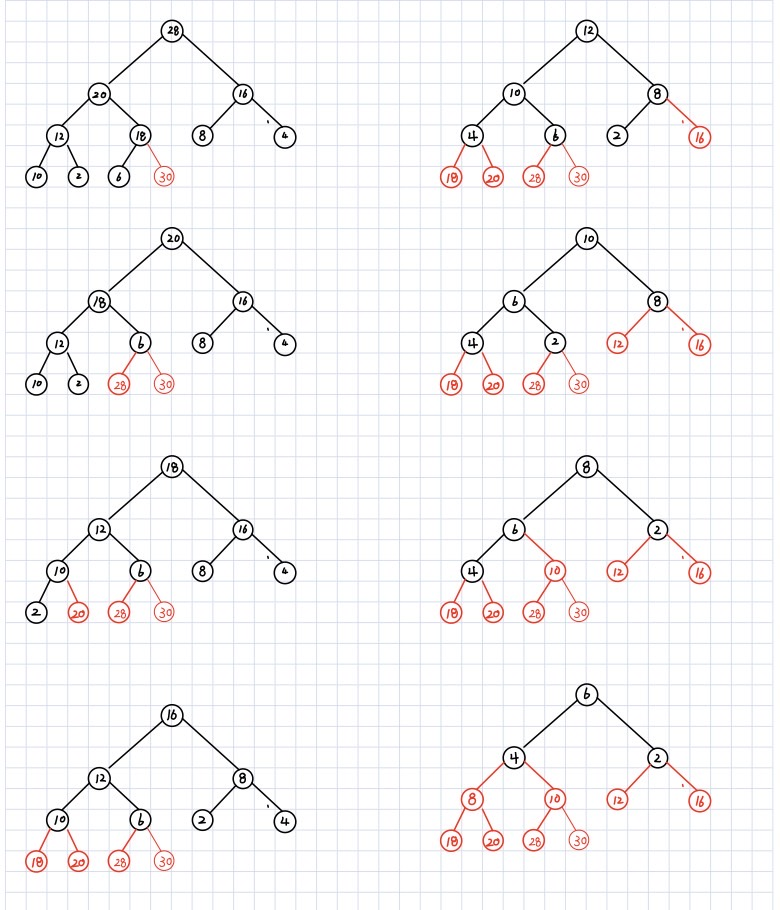


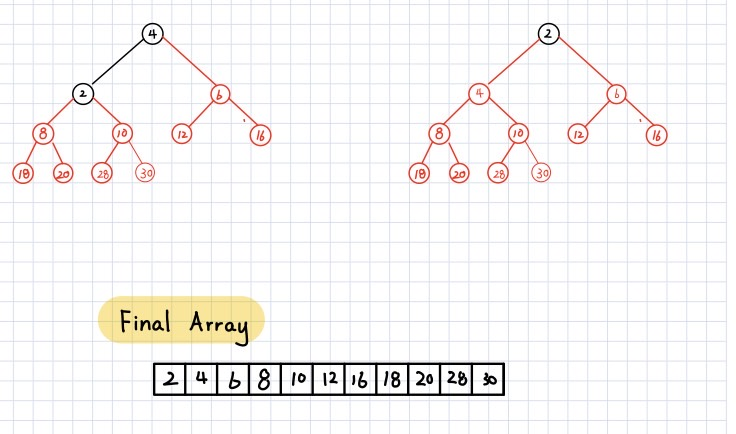
1. Write the status of the list L at the end of each phase of MergeSort (Program 7.9), i.e., draw the Merge tree (similar to Figure 7.4 in textbook) of this problem.



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.

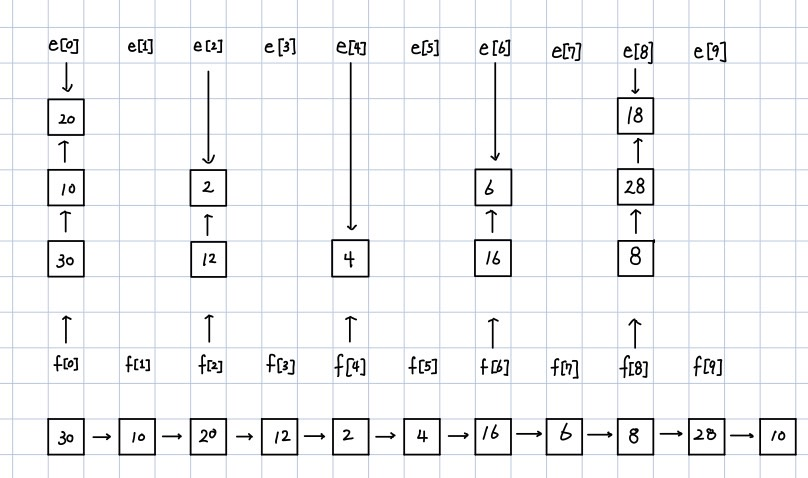
 

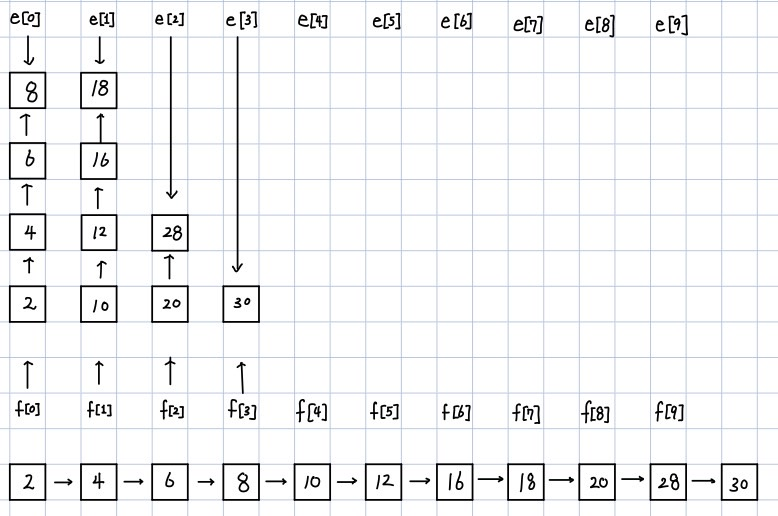




1. Write the status of the list L at the end of each pass of RadixSort (Program 7.15), using r = 10. That is fill the missing parts (the node boxes with numbers and arrows between e[j] and f[j] enclosed by red dashed rectangle in (ii) and (iii) part of the following figure, and the missing numbers in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







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.**

<Answer>

Consider the input sequence of (3,1,2,3,5,6,7,8,9), denote the first 3 as 3’, then the sequence becomes, (3’,1,2,3,5,6,7,8,9), the sequence will become ([1,2,3],3’,[5,6,7,8,9]), which changes the original order of the 3’s. This then shows that quick sort is unstable.

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

<Answer>

The stability issue only occurs when in the merging part. Since the passing part retains the original sequence. The cases might cause the sorting to be unstable. which is there exists the following [a, a, c] and [a, b, c], since the tie break mechanism in the algorithm is if there is a tie in the first block then the first block is sorted first which remains the input’s original unsorted order [a, a, a, b, c, c]. This indicates that the algorithm is indeed stable.

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).)**

<Answer>

Where and , let be the MSD and be the MSD, then we can do the radix sort by first and then do the radix sort by this takes two for loops of n times, thus the time complexity is O(2n) = 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.**

<Answer>

(a)

1. One-way property is that for a given c, it is computationally difficult to find a k such that h(k) = c.

2. Weak collision resistance is that it is difficult for whom has knowledge of h and M to determine a synonym of M where h is the hashing function and M is the transmitted message or data.

3. Strong collision resistance is that it is computationally difficult to find a pair (x, y) such that h(x) = h(y).

(b)

1. For hash function c = h(k) = k%17, it is easy to find an inverse hashing of k from c since the message must be in a form of Thus it is not computationally hard, thus violating the one-way property.

2. From above, consider there is M and h(M) given where , then consider . Thus , which leads to h(N) = h(M). This indicates that for adding arbitrary multiple of 17 to M can create synonym of M. This violates weak collision resistance property.

3. From above, we can see that it is not hard to find synonym for a given message, as long as a given M is known then we can create N that is M + multiple of 17, then this pair will guarantee to be synonym, which for the pair <M , N>, h(M) = h(N). This violates strong collision resistance property.

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*.**

<Answer>

