CS217 DSAA Homework 7

HONGLI YE 12311501

2024.11.05

Contents

1 Question 1

Illustrate the operation of CountingSort(A, B, 3) on the array

0	2	0	1	3	1	1

Table 1: Question 1

with input elements from the set $\{0, 1, 2, 3\}$

1. For each of the first three for loops, write down the contents of array C after the loop has ended.
2. For the last for loop, write down the contents of the arrays B and C at the end of each iteration of the loop.

Answer:

1. Answer to Question 1.1:

C	0	1	2	3
Value	2	5	6	7

Table 2: Question 1.1

2. Answer to Question 1.2:

Loop 1:

B	0	1	2	3	4	5	6
Value	0	0	0	0	1	0	0

Table 3: Loop 1

C	0	1	2	3
Value	2	4	6	7

Table 4: Loop 1

Loop 2:

B	0	1	2	3	4	5	6
Value	0	0	0	1	1	0	0

Table 5: Loop 2

C	0	1	2	3
Value	2	3	6	7

Table 6: Loop 2

Loop 3:

B	0	1	2	3	4	5	6
Value	0	0	0	1	1	0	3

Table 7: Loop 3

C	0	1	2	3
Value	2	3	6	6

Table 8: Loop 3

Loop 4:

B	0	1	2	3	4	5	6
Value	0	0	1	1	1	0	3

Table 9: Loop 4

C	0	1	2	3
Value	2	2	6	6

Table 10: Loop 4

Loop 5:

B	0	1	2	3	4	5	6
Value	0	0	1	1	1	0	3

Table 11: Loop 5

C	0	1	2	3
Value	1	2	6	6

Table 12: Loop 5

Loop 6:

B	0	1	2	3	4	5	6
Value	0	0	1	1	1	2	3

Table 13: Loop 6

C	0	1	2	3
Value	1	2	5	6

Table 14: Loop 6

Loop 7:

B	0	1	2	3	4	5	6
Value	0	0	1	1	1	2	3

Table 15: Loop 7

C	0	1	2	3
Value	0	2	5	6

Table 16: Loop 7

2 Question 2

Prove that after the **for** loop in lines 6-7 (in the pseudo-code provided at lecture) of CountingSort the array C contains in each position $C[i]$ the number of elements less than or equal to i (You can assume the previous two **for** loops are correct).

Answer: Write down the persudocode.

Algorithm 1 CountingSort(A,B,k)

```
1: Let C[0,1...,k] be a new array.
2: for i = 0 to k do
3:     C[i] = 0
4: end for
5: for j = 1 to A.length do
6:     C[A[j]] = C[A[j]] + 1
7: end for
8: for i = 1 to k do
9:     C[i] = C[i] + C[i − 1]
10: end for
11: for j = A.length downto 1 do
12:     B[C[A[j]]] = A[j]
13:     C[A[j]] = C[A[j]] − 1
14: end for
```

Before the **for** loop in line 6-7, in each $C[i]$ contains the number of elements with value i.
After running the loop:

$$C[i] = C[i] + C[i-1]$$

We suppose we get a new array $S[i]$, then we know:

$$S[i] = \Sigma_{i=1}^{n}(C[i])$$

By the relationship between $C[i]$ and number of elements with value i. We can easily prove that the array S contains in each position $S[i]$ the number of elements less than or equal to i

3 Question 3

Suppose that we were to rewrite the last for loop header of CountingSort as:
"$for\, j = 1\, to\, A.length$".
Then the algorithm:

1. Will not be stable and will not sort the numbers
2. Will be stable but will not sort the numbers
3. Will not be stable but will sort the numbers
4. Will be stable and will sort the numbers

Justify your answer.

Answer:

I think the answer is **3**.

It will sort the answer and since the B has stored all the situation information of the array and the visiting order doesn't bother the result.

But it has change the order of the element with same values, so when applying the counting sort to radix sort, it will cause problem.

4 Question 4

Describe an algorithm $CountingRange(A, k, a, b)$ that given n integers in the range 0 to k, preprocesses its input and and then answers any query about how many integers are present in the range $[a : b]$ in $O(1)$ time. The algorithm should use $O(n + k)$ preprocessing time.

Answer:

We use the first 3 for loops in the CountingSort algorithm. The array C is what we need. Using the result in Question 2, we know that:

$$C[i] = \text{How many integers} \leq i$$

So: $CountingRange(A, k, a, b) = \begin{cases} C[b] - C[a-1] & a \geq 1 \\ C[b] & a = 0 \end{cases}$

5 Question 5

Illustrate the operation of RadixSort on the following list of English words:

$$COW, DOG, TUG, ROW, MOB, BOX, TAB, BAR, CAR, TAR, PIG, BIG, WOW$$

Answer:

1. First we need to illustrate that the RadixSort need to find a upper bound, which is A t0 Z.
2. Second we do the first round loop: We do the sort based on the the last letter. So we got the array:

$$MOB, TAB, DOG, TUG, PIG, BIG, BAR, CAR, TAR, COW, ROW, WOW, BOX$$

3. Third we do the second round loop: We do the sort based on the the second letter. So we got the array:

$$TAB, BAR, CAR, TAR, PIG, BIG, MOB, DOG, COW, ROW, WOW, BOX, TUG$$

4. Fourth we do the third round loop: We do the sort based on the the first letter. So we got the array:

$$BAR, BIG, BOX, CAR, COW, DOG, MOB, PIG, ROW, TAB, TAR, TUG, WOW$$

6 Question 6

State which of the following algorithms are stable and which are not:

1. InsertionSort
2. MergeSort
3. HeapSort
4. QuickSort

For those that are stable argue why. For those that are not stable provide an example of an input that shows instability.

Answer:

1. InsertionSort is stable. Since we do the insertion from left to right, if meet elements with same values, the element to be inserted will be on the right side to keep the same order.
2. MergeSort is stable. MergeSort first divide the elements into pairs and then merge them use the order from left to right, so it doesn't change the order of elements with the same values.
3. HeapSort is not stable. We simple take $\{a_1, b, a_2\}$ and assume $a_1 = a_2 > b$ and the output of HeapSort is $\{b, a_2, a_1\}$
4. QuickSort is not stable. We simple take $\{c_1, c_2, a, b\}$ and assume $a < b < c_1 = c_2$ and the output of HeapSort is $\{a, b, c_2, c_1\}$

7 Question 7

Implement $CountingSort(A, B, k, n)$ that sorts n numbers between 0 and k and $RadixSort(A, d, n)$ that sorts an English Dictionary with n words of d letters each.

Answer:

I have finished the codes on OJ.