CS 477: Homework #4

Due on October 4th, 2016 at 2:30pm

Monica Nicolescu

Matthew J. Berger

Problem 1

(U & G-required)[20 points]

(a) [20 points] Assuming that the set of possible list values is a, b, c, d, sort the following list in alphabetical order using counting sort: b, c, d, c, b, a, a, b. At each step show how the values of your counting array and the final array.

(b) [10 points] Illustrate the operation of Radix sort on the following list of English words: COW, DOG, SEA, RUG, ROW, MOB, BOX, TAB, BAR, EAR, TAR, DIG, BIG, TEA, NOW, FOX.

Solution

(a) 1.) Make a counting array to store the number of occurrences of each unique item.

Original	b c d c b a a b
Order	a b c d
Count	2 3 2 1

2.) Modify the count array such that each element at each index stores the sum of previous counts.

Order	a b c d
Count	2 5 7 8

The modified count array indicates the position of each unique item in the output sequence.

3.) Output each object from the input at the position indicated by the count array, then decrease its count by 1. Assume the output array is empty at the start. The index is zero-based so the count is one greater than the actual index of the item.

Original	***b*** c d c b a a b
Count	2 ***5*** 7 8
Index	0 1 2 3 4 5 6 7
Output	b

→

Original	b ***c*** d c b a a b
Count	2 4 ***7*** 8
Index	0 1 2 3 4 5 6 7
Output	b c

→

Original	b c ***d*** c b a a b
Count	2 4 6 ***8***
Index	0 1 2 3 4 5 6 7
Output	b c d

→

Original	b c d ***c*** b a a b
Count	2 4 ***6*** 7
Index	0 1 2 3 4 5 6 7
Output	b c c d

→

Original	b c d c ***b*** a a b
Count	2 ***4*** 7 14
Index	0 1 2 3 4 5 6 7
Output	b b c c d

→

Original	b c d c b ***a*** a b
Count	***2*** 3 7 14
Index	0 1 2 3 4 5 6 7
Output	a b b c c d

→

Original	b c d c b a ***a*** b
Count	***1*** 3 7 14
Index	0 1 2 3 4 5 6 7
Output	a a b b c c d

→

Original	b c d c b a a ***b***
Count	0 ***3*** 7 14
Index	0 1 2 3 4 5 6 7
Output	<u>a a b b b c c d</u>

Problem 1 continued on next page. . .

(b) Radix sort sorts items by using the least significant 'digit' for comparison, working its way up to the most significant digit. Radix sort uses a counting sort as its sorting subroutine.

- CO$\underline{\text{W}}$, DO$\underline{\text{G}}$, SE$\underline{\text{A}}$, RU$\underline{\text{G}}$, RO$\underline{\text{W}}$, MO$\underline{\text{B}}$, BO$\underline{\text{X}}$, TA$\underline{\text{B}}$, BA$\underline{\text{R}}$, EA$\underline{\text{R}}$, TA$\underline{\text{R}}$, DI$\underline{\text{G}}$, BI$\underline{\text{G}}$, TE$\underline{\text{A}}$, NO$\underline{\text{W}}$, FO$\underline{\text{X}}$.
  $\rightarrow$ SEA, TEA, MOB, TAB, DOG, RUG, DIG, BIG, BAR, EAR, TAR, COW, ROW, NOW, BOX, FOX.
- S$\underline{\text{E}}$A, T$\underline{\text{E}}$A, M$\underline{\text{O}}$B, T$\underline{\text{A}}$B, D$\underline{\text{O}}$G, R$\underline{\text{U}}$G, D$\underline{\text{I}}$G, B$\underline{\text{I}}$G, B$\underline{\text{A}}$R, E$\underline{\text{A}}$R, T$\underline{\text{A}}$R, C$\underline{\text{O}}$W, R$\underline{\text{O}}$W, N$\underline{\text{O}}$W, B$\underline{\text{O}}$X, F$\underline{\text{O}}$X
  $\rightarrow$ TAB, BAR, EAR, TAR, SEA, TEA, DIG, BIG, MOB, DOG, COW, ROW, NOW, BOX, FOX, RUG.
- $\underline{\text{T}}$AB, $\underline{\text{B}}$AR, $\underline{\text{E}}$AR, $\underline{\text{T}}$AR, $\underline{\text{S}}$EA, $\underline{\text{T}}$EA, $\underline{\text{D}}$IG, $\underline{\text{B}}$IG, $\underline{\text{M}}$OB, $\underline{\text{D}}$OG, $\underline{\text{C}}$OW, $\underline{\text{R}}$OW, $\underline{\text{N}}$OW, $\underline{\text{B}}$OX, $\underline{\text{F}}$OX, $\underline{\text{R}}$UG.
  $\rightarrow$ BAR, BIG, BOX, COW, DIG, DOG, EAR, FOX, MOB, NOW, ROW, RUG, SEA, TAB, TAR, TEA.

  This successfully sorts the list.

Problem 2

(U & G-required)[40 points] Implement in C or C++ an algorithm that given n integers in the range 0 to k returns how many of the n integers fall into a range $[a..b]$, given as an input. Your algorithm should use $\theta(n + k)$ processing time. Show the answer returned by your algorithm on the following input A = [5 6 9 4 4 2 2 9 5 3 4 0 1 8 5 3 8 7], and $a = 2$, $b = 7$.

Solution

The algorithm I've created for this problem is as follows:

```
#include <iostream>

int CountOfNumbersInRange(int* arr, int min, int max, int size)
{
    int count = 0;
    for(int i = 0; i < size; i++)
        if ((min <= arr[i]) && (arr[i] <= max)) count++;
    return count;
}

int main(int argc, char** argv)
{
    int arr[] = { 5, 6, 9, 4, 4, 2, 2, 9, 5, 3, 4, 0, 1, 8, 5, 3, 8, 7 };
    int size = 18;
    int min = 2;
    int max = 7;
    int output = CountOfNumbersInRange(arr, min, max, size);
    std::cout << "Input: ";
    for(int i = 0; i < size; i++)
    {
        std::cout << arr[i];
        if (i < size - 1) std::cout << ",";
        else std::cout << std::endl;
    }
std::cout << "Numbers between range " << min << " and " << max << ": " << output << std::endl;
}
```

This program will run the algorithm against the test input stated in the question. Its output is:

```
Input: 5,6,9,4,4,2,2,9,5,3,4,0,1,8,5,3,8,7
Numbers between range 2 and 7: 12
```

Problem 3

(U & G-required)[**40 points**] Consider a red-black tree formed by inserting n nodes with RB-INSERT. Give a justification for the fact that if $n > 1$, then the three has at least one red node.

Solution

There are five rules for Red-Black Trees:

1. Every node is either black or red.
2. The root is always black.
3. Every leaf is black.
4. If a node is red, then both its sons are black.
5. All paths from a single node to any leaf must contain an equal amount of black nodes.

Any Red-Black tree containing at least two elements will have at least one red node, because the inserted node is always colored red initially but any later modifications to the tree (such as fixing the colors) either leave the color of the newly added node as is (it stays red), or makes at least one other node red.

Problem 5

Extra Credit (I'm an undergraduate)

[**20 points**] Explain how you would sort n integers that span the range between 0 to $n^3 - 1$ in $\mathcal{O}(n)$ time.

Solution

One possible solution is to represent and sort the numbers in a different base. For a base-k number system, any number n can be represented by using $log_k(R)$ digits. Setting $k = n$ and $R = n^3$, it's clear that only $log_n(n^3) = \underline{3}$ digits are necessary to represent all the integers that span the range between 0 to $n^3 - 1$. To perform the conversion, you must solve for a and b in $x = a * n^2 + b * n + c$, where a,b,c are within our range and the actual number is equivalent to the product of a, b, and c. The largest number $(n^3 - 1)$ can be written to fit the form of the geometric sum formula.

$$(n-1) * n^2 + (n-1) * n + n - 1 = (n-1)\frac{1-(-n)^3}{1-(-n)}$$

This makes the n-base system representation $(-n-1)$. Applying radix sort to the converted 3-digit base-n numbers makes the complexity $\theta(d(n+k)) = \theta(3(n+n)) = \theta(n)$ which satisfies the requirement for this problem.