FCE-hw2

Weikun Su

NUID: 001498198

October 2019

Question 1 (Randomized Quicksort)

The results are shown below.

```
The original array is:
1 2 3 4 5 6 7 8 9 10   11   12   13   14   15   16   17   18   19   20   21   22   23   24   25   26
    27   28   29   30   31   32   33   34   35   36   37   38   39   40   41   42   43   44   45   46   47
      48   49   50   51   52   53   54   55   56   57   58   59   60   61   62   63   64   65   66   67
    68   69   70   71   72   73   74   75   76   77   78   79   80   81   82   83   84   85   86   87   88
      89   90   91   92   93   94   95   96   97   98   99   100
The running time of trial 1 is: 1e-005 seconds
The running time of trial 2 is: 9.3e-006 seconds
The running time of trial 3 is: 9.1e-006 seconds
The running time of trial 4 is: 9e-006 seconds
The running time of trial 5 is: 8e-006 seconds
```

The code is shown below.

```
//Quick Sort
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <time.h>

int partition(int v[], int p, int q){
    int pivot = v[p];
    int i  = p;
    int j;
    for (j = p+1; j <= q; j++){
        if (v[j] < pivot){
            i++;
            std::swap(v[j], v[i]);
        }
    }
    std::swap(v[i],v[p]);
    return i;
}
int partition_rand(int v[], int p, int q){
    int pivot_ind;
    pivot_ind = rand() % (q-p+1) + p;
    std::swap(v[p],v[pivot_ind]);
    return partition(v, p, q);
}
void quickSort_rand(int v[],int p,int q){
    int r;
    if (p < q){
        r = partition_rand(v, p, q);
        quickSort_rand(v, p, r-1);
        quickSort_rand(v, r+1, q);
    }
}

int main(){
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nBeginTime;
    LARGE_INTEGER nEndTime;
    QueryPerformanceFrequency(&nFreq); // get the frequency of the counter
    double t;
    int j;
    int N = 100;
    int v[N];
    int i;
    srand(time(NULL));
    for (i=0; i<N; i++){
        v[i] = i+1;
    }
    std::cout << "The original array is:" << std::endl; // print original array
    for (i=0; i<N; i++){
        std::cout << v[i] << '\t';
    }
    std::cout << std::endl;
    for (j=0; j<5; j++){
        QueryPerformanceCounter(&nBeginTime); //begin time of quick sort
        quickSort_rand(v, 0, N-1);
        QueryPerformanceCounter(&nEndTime); //end time of quick sort
        t = (double)(nEndTime.QuadPart-nBeginTime.QuadPart)/(double)nFreq.QuadPart; // running time of quick sort
        std::cout << "The running time of trial "<< j+1 << " is: " << t << " seconds" << std::endl;
    }

    return 0;
}
```

Question 2 (Heapsort)

Result:

```
Original array:
92  20  14  97  47  12  24  66  64  41  42  43  76  91  69  77  13  34  81  36  33  87
      70  61  68  4 57  5 84  31  26  55  15  83  52  18  95  9 49  67  50  19  62  78
      11  79  32  75  35  53  22  71  74  51  29  23  82  2 54  96  90  30  10  44  6
    100 25  40  98  56  94  65  7 99  16  58  80  63  1 38  17  60  27  21  88  86  28
      59  85  48  37  46  39  8 73  72  89  93  45  3
Sorted array:
1 2 3 4 5 6 7 8 9 10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26
    27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47
      48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67
    68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88
      89  90  91  92  93  94  95  96  97  98  99  100
The running time is: 1.41e-005 seconds
```

Code:

```
//Heap Sort
```

```
#include <iostream>
#include <windows.h>
#include <stdlib.h>
#include <time.h>

void randperm(int a[], int N){
    int i, j;
    for (i=0; i<N; i++){
        j = rand() % (N-i) + i;
        std::swap(a[i], a[j]);
    }
}
void max_heapify(int a[], int N, int i){
    int left, right, largest;
    left = 2*i + 1;
    right = 2*i + 2;
    if (left < N && a[left] > a[i]){
        largest = left;
    }
    else
    {
        largest = i;
    }
    if (right < N && a[right] > a[largest]){
        largest = right;
    }
    if (largest != i){
        std::swap(a[i], a[largest]);
        max_heapify(a, N, largest);
    }
}
void build_max_heap(int a[], int N){
    int i;
    for (i = N/2 - 1; i >= 0; i--){
        max_heapify(a, N, i);
    }
}
void heap_sort(int a[], int N){
    build_max_heap(a, N);
    int i;
    for (i = N - 1; i >= 0; i--){
        std::swap(a[i], a[0]);
        max_heapify(a, i, 0);
    }
}
int main(){
    int i;
    int N = 100;
    int v[N];
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nBeginTime;
    LARGE_INTEGER nEndTime;
    QueryPerformanceFrequency(&nFreq); // get the frequency of the counter
    double t;
    srand(time(NULL));
    for (i=0; i<N; i++){
        v[i] = i+1;
    }
    randperm(v, N);
    std::cout << "Original array:" << std::endl;
    for (i=0; i<N; i++){
        std::cout << v[i] << '\t' ;
    }
    std::cout << std::endl;
    QueryPerformanceCounter(&nBeginTime);
    heap_sort(v, N);
    QueryPerformanceCounter(&nEndTime);
    t = (double)(nEndTime.QuadPart-nBeginTime.QuadPart)/(double)nFreq.QuadPart;
    std::cout << "Sorted array:" << std::endl;
    for (i=0; i<N; i++){
        std::cout << v[i] << '\t' ;
    }
    std::cout << std::endl;
    std::cout << "The running time is: " << t << " seconds" << std::endl;
    return 0;
}
```

Question 3 (Counting Sort)

Result:

```
Original array is:
20  18  5 7 16  10  9 3 12  14  0
Sorted array is:
0 3 5 7 9 10  12  14  16  18  20
The running time is: 5e-007 seconds
```

Code:

```
//Counting sort
#include <iostream>
#include <windows.h>
#include <algorithm>

void counting_sort(int a[],int n){
    int k;
    k = *std::max_element(a, a + n);
    int i;
    int c[k+1];
    int b[n];
    for (i = 0; i <= k; i++){
        c[i] = 0;
    }
    for (i = 0; i < n; i++){
        c[a[i]] = c[a[i]] + 1;
    }
    for (i = 1; i <= k; i++){
        c[i] = c[i] + c[i-1];
    }
    for (i = n-1; i >= 0; i--){
        b[c[a[i]]-1] = a[i];
        c[a[i]] = c[a[i]]-1;
    }
    for (i = 0; i < n; i++){
        a[i] = b[i];
    }
}
int main(){
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nBeginTime;
    LARGE_INTEGER nEndTime;
    QueryPerformanceFrequency(&nFreq); // get the frequency of the counter
    double t;
    int a[] = {20,18,5,7,16,10,9,3,12,14,0};
    int n;
    n = sizeof(a)/sizeof(a[0]);
    int i;
    std::cout << "Original array is:" << std::endl;
    for (i = 0; i < sizeof(a)/sizeof(a[0]); i++){
        std::cout << a[i] << '\t';
    }
    std::cout << std::endl;
    QueryPerformanceCounter(&nBeginTime);
    counting_sort(a, n);
    QueryPerformanceCounter(&nEndTime);
    t = (double)(nEndTime.QuadPart-nBeginTime.QuadPart)/(double)nFreq.QuadPart;
    std::cout << "Sorted array is:" << std::endl;
    for (i = 0; i < sizeof(a)/sizeof(a[0]); i++){
        std::cout << a[i] << '\t';
    }
    std::cout << std::endl;
    std::cout << "The running time is: " << t << " seconds" << std::endl;
}
```

Question 4 (Radix Sort)

Result:

```
Original array is:
329 457 657 839 436 720 353
Sorted array is:
329 353 436 457 657 720 839
The running time is: 1.3e-006 seconds
```

Code:

```
//Radix Sort
#include <iostream>
#include <windows.h>
#include <algorithm>

void counting_sort_d(int a[], int n, int digit){
    int b[n];
    int c[10];
    int i;
    for (i = 0; i < 10; i++){
        c[i] = 0;
    }
    for (i = 0; i < n; i++){
        c[a[i]/digit % 10] = c[a[i]/digit % 10] + 1;
    }
    for (i = 1; i < 10; i++){
        c[i] = c[i] + c[i-1];
    }
    for (i = n-1; i >= 0; i--){
        b[c[a[i]/digit % 10]-1] = a[i];
        c[a[i]/digit % 10] = c[a[i]/digit % 10] - 1;
    }
    for (i = 0; i < n; i++){
        a[i] = b[i];
    }
}
void radix_sort(int a[], int n){
    int digit;
    for (digit = 1; digit <= 100; digit = digit*10){
        counting_sort_d(a, n, digit);
    }
}
int main(){
    LARGE_INTEGER nFreq;
    LARGE_INTEGER nBeginTime;
    LARGE_INTEGER nEndTime;
    QueryPerformanceFrequency(&nFreq); // get the frequency of the counter
    double t;
    int a[] = {329,457,657,839,436,720,353};
    int n;
    int i;
    n = sizeof(a)/sizeof(a[0]);
    std::cout << "Original array is:" << std::endl;
    for (i = 0; i < n; i++){
        std::cout << a[i] << '\t';
    }
    std::cout << std::endl;
    QueryPerformanceCounter(&nBeginTime);
    radix_sort(a, n);
    QueryPerformanceCounter(&nEndTime);
    t = (double)(nEndTime.QuadPart-nBeginTime.QuadPart)/(double)nFreq.QuadPart;
    std::cout << "Sorted array is:" << std::endl;
    for (i = 0; i < n; i++){
        std::cout << a[i] << '\t';
    }
    std::cout << std::endl;
    std::cout << "The running time is: " << t << " seconds" << std::endl;
}
```