

HW2_YIXIAO CHEN_002198256

Q1:

Question 1 (2 pt.) Randomized Quicksort: Write codes for randomized quicksort. You may need rand() to generate random numbers. Run the randomized quicksort 5 times for input array A = {1, 2, 3, ..., 99, 100} and report the 5 running times.

-----Codes after result for every question below-----

【program running result】

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Quicksort - main.cpp
Project
CMakeLists.txt main.cpp
Run: Quicksort
"D:\Documents\Northeastern University\EECE7205\Project\Quicksort\cmake-build-debug\Quicksort.exe"
>>> [Quick_Sort] <<<<
---- please give your n: ----
100
----Original 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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
----Random Pivot Position a[R]:----
[41] [17] [10] [0] [9] [5] [3] [1] [2] [4] [8] [7] [6] [14] [12] [11] [13] [15] [16] [26] [25] [20] [18] [19] [21] [23]
[22] [24] [33] [32] [29] [27] [28] [30] [31] [38] [37] [35] [34] [36] [40] [39] [44] [43] [42] [89] [50] [46] [45] [48]
[47] [49] [71] [68] [62] [51] [55] [52] [54] [53] [60] [59] [58] [56] [57] [61] [63] [66] [64] [65] [67] [70] [69] [86]
[82] [72] [76] [73] [75] [74] [81] [80] [77] [78] [79] [85] [83] [84] [88] [87] [94] [93] [92] [91] [90] [98] [97] [95]
[96] [99]
----QuickSort Result:----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
----QuickSort Running Time:----
0.032007ms Count:1
----Random Pivot Position a[R]:----
[33] [29] [28] [6] [0] [1] [3] [2] [5] [4] [9] [8] [7] [20] [10] [17] [16] [14] [11] [12] [13] [15] [19] [18] [25] [24]
[21] [22] [23] [26] [27] [31] [30] [32] [74] [64] [61] [39] [36] [35] [34] [38] [37] [49] [45] [44] [41] [40] [43] [42]
[48] [46] [47] [56] [53] [52] [50] [51] [54] [55] [59] [58] [57] [60] [62] [63] [66] [65] [71] [69] [68] [67] [70] [72]
[73] [96] [81] [78] [75] [76] [77] [79] [80] [93] [89] [83] [82] [87] [85] [84] [86] [88] [90] [92] [91] [95] [94] [99]
[98] [97]
----QuickSort Result:----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
Process finished with exit code 0
```

```
----Random Pivot Position a[R]:----
[73] [4] [3] [1] [0] [2] [47] [43] [34] [5] [20] [17] [8] [7] [6] [13] [12] [11] [10] [9] [15] [14] [16] [19] [18] [26]
[23] [21] [22] [24] [25] [31] [30] [29] [28] [27] [32] [33] [38] [37] [36] [35] [39] [42] [41] [40] [46] [44] [45] [72]
[59] [50] [48] [49] [53] [51] [52] [55] [54] [56] [58] [57] [70] [64] [63] [61] [60] [62] [65] [68] [66] [67] [69] [71]
[82] [77] [75] [74] [76] [79] [78] [80] [81] [85] [84] [83] [93] [86] [88] [87] [92] [91] [89] [90] [97] [96] [95] [94]
[99] [98]
----QuickSort Result:----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
----QuickSort Running Time:----
0.033007ms Count:5
Process finished with exit code 0
```

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help QuickSort - main.cpp
QuickSort
CMakeLists.txt main.cpp
Run: QuickSort
-----QuickSort Result:-----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
-----QuickSort Running Time:-----
0.029007ms Count:2
-----Random Pivot Position a[R]:-----
[9] [2] [1] [0] [5] [4] [3] [8] [6] [7] [24] [14] [10] [13] [12] [11] [15] [21] [19] [16] [18] [17] [20] [23] [22] [25]
[90] [52] [40] [26] [31] [28] [27] [29] [30] [37] [34] [32] [33] [35] [36] [39] [38] [42] [41] [51] [50] [48] [44] [43]
[47] [45] [46] [49] [68] [64] [59] [55] [54] [53] [57] [56] [58] [60] [63] [61] [62] [65] [67] [66] [77] [73] [69] [72]
[71] [70] [76] [74] [75] [79] [78] [88] [81] [80] [85] [83] [82] [84] [86] [87] [89] [98] [92] [91] [95] [93] [94] [96]
[97] [99]
-----QuickSort Result:-----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
-----QuickSort Running Time:-----
0.030005ms Count:3
-----Random Pivot Position a[R]:-----
[76] [17] [5] [2] [1] [0] [3] [4] [11] [8] [7] [6] [9] [10] [16] [15] [13] [12] [14] [19] [18] [41] [20] [30] [25] [24]
[21] [22] [23] [27] [26] [28] [29] [35] [33] [32] [31] [34] [37] [36] [40] [39] [38] [69] [63] [56] [54] [47] [45] [42]
[44] [43] [46] [52] [49] [48] [50] [51] [53] [55] [60] [59] [58] [57] [61] [62] [66] [65] [64] [67] [68] [72] [71] [70]
[74] [73] [75] [85] [77] [82] [81] [80] [78] [79] [83] [84] [89] [87] [86] [88] [94] [92] [90] [91] [93] [95] [99] [97]
[96] [98]
-----QuickSort Result:-----
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, 3
3, 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, 6
3, 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, 9
3, 94, 95, 96, 97, 98, 99, 100,
-----QuickSort Running Time:-----
0.025132ms Count:4
Process finished with exit code 0
```

【randomized quicksort code】_by cyx

```
#include <iostream>
#include <ratio>
#include <chrono>
#include <random>
using namespace std;
using namespace chrono;
int partition(int a[], int l, int r)
{
    int R = rand()%(r - l + 1) + l;//随机生成[l,r]之间整数
    cout<< "[" << R << "]" ";
    int pivot = a[R];
    a[R] = a[r];
    a[r] = pivot;
    int e;
    int i = l - 1;
    for(int j = l; j < r; j++)
    {
```

```

        if(a[j] <= pivot)
        {
            i = i + 1;
            e = a[i];
            a[i] = a[j];
            a[j] = e;
        }
    }
    a[r] = a[i + 1];
    a[i + 1] = pivot;
    return i+1;
}

void quicksort(int a[], int l, int r)
{
    int p;
    if(l <= r)
    {
        p = partition(a, l, r);
        quicksort(a, l, p-1);
        quicksort(a, p+1, r);
    }
}

void print_array(int a[], int n)
{
    for(int i = 0; i < n; i++)
    {
        cout << a[i] << " ";
    }
    cout << endl;
}

int main() {
    cout << ">>>> [Quick_Sort] <<<<" << endl;
    int n;
    cout << "---- please give your n: ----" << endl;
    cin >> n;
    int * a, * v;
    a = new int[n];
    v = new int[n];

```

```

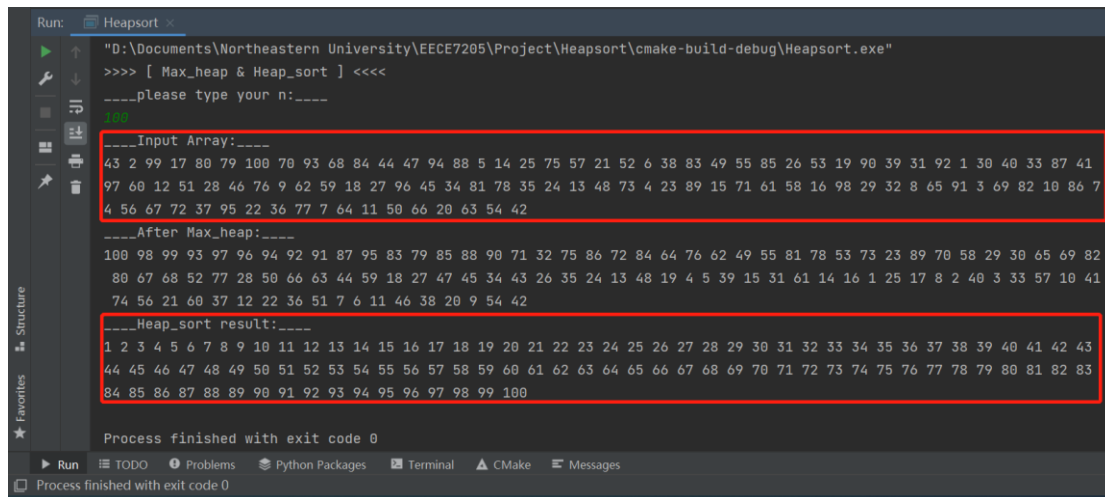
for (int i = 0; i < n; i++)
{
    a[i] = i + 1;
    v[i] = a[i];
}
cout << "____Original Array:____" << endl;
print_array(v, n);
for(int j = 1; j <= 5; j++)
{
    cout << "____Random Pivot Position a[R]:____" << endl;
    high_resolution_clock::time_point t1 = high_resolution_clock::now();
    quicksort(a, 0, n-1);
    cout << endl;
    high_resolution_clock::time_point t2 = high_resolution_clock::now();
    duration<double, ratio<1, 1000>>> duration_QS =
duration_cast<duration<double, ratio<1, 1000>>>(t2 - t1);
    cout << "____QuickSort Result:____" << endl;
    print_array(a, n);
    cout << "____QuickSort Running Time:____" << endl;
    cout << duration_QS.count() << "ms" << "    Count:" << j << endl;
}
return 0;
}

```

Q2:

Question 2 (2pt.) Heapsort: Write codes for heapsort. The input array is a random permutation of $A=\{1, 2, 3, \dots, 99, 100\}$. You should write codes to generate and print the random permutation first.

【program running result】



```
Run: Heapsort x
"D:\Documents\Northeastern University\EECE7205\Project\Heapsort\cmake-build-debug\Heapsort.exe"
>>> [ Max_heap & Heap_sort ] <<<<
----please type your n:----
100
----Input Array:----
43 2 99 17 80 79 100 70 93 68 84 44 47 94 88 5 14 25 75 57 21 52 6 38 83 49 55 85 26 53 19 90 39 31 92 1 30 40 33 87 41
97 60 12 51 28 46 76 9 62 59 18 27 96 45 34 81 78 35 24 13 48 73 4 23 89 15 71 61 58 16 98 29 32 8 65 91 3 69 82 10 86 7
4 56 67 72 37 95 22 36 77 7 64 11 50 66 20 63 54 42
----After Max_heap:----
100 98 99 93 97 96 94 92 91 87 95 83 79 85 88 90 71 32 75 86 72 84 64 76 62 49 55 81 78 53 73 23 89 70 58 29 30 65 69 82
80 67 68 52 77 28 50 66 63 44 59 18 27 47 45 34 43 26 35 24 13 48 19 4 5 39 15 31 61 14 16 1 25 17 8 2 40 3 33 57 10 41
74 56 21 60 37 12 22 36 51 7 6 11 46 38 20 9 54 42
----Heap_sort result:----
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
Process finished with exit code 0
```

【heap sort code】_by cyx

```
#include <iostream>
// #include <cmath>
#include <random>
#include <algorithm>
using namespace std;
void max_heapify(int a[], int i, int n) //heap 按 max 规则调整
{
    int l = 2 * i;
    int r = 2 * i + 1;
    int largest;
    if(l <= n && a[l] > a[i]){
        largest = l;
    }
    else{
        largest = i;
    }
    if(r <= n && a[r] > a[largest]){
        largest = r;
    }
    int value;
    if(largest != i){
```

```

        value = a[i];
        a[i] = a[largest];
        a[largest] = value;
        max_heapify(a, largest, n);
    }
}

void build_max_heap(int a[], int n) //自身调用 构建 max hea{
    for (int i = n/2; i >= 1 ; i--)
    {
        max_heapify(a, i, n);
    }
}

void print(int a[], int n) //打印输入序列{
    for (int i = 1; i <= n; i++)
    {
        cout << a[i] <<" ";
    }
    cout<< endl;
}

/*void print_heap_tree(int a[], int n){
    cout << "Max_Heap_Tree:" << endl;
    int total_level = log(n)/log(2);
    for(int l = 0; l <= total_level; l++)
    {
        for(int i = pow(2,l); i < pow(2,l+1) && i <= n ; i++)
        {
            if(a[i] <= 9)
            {
                cout << a[i] <<"    ";
            }
            else
            {
                cout << a[i] <<"    ";
            }
        }
        cout << endl;
    }
}*/

```

```

void heap_sort(int a[], int n){
    int * H;
    H = new int[n+1];
    for(int i = n; i >= 1; i--){
        {
            H[i] = a[1];
            a[1] = a[i];
            max_heapify(a, 1, i-1);
        }
    }
    cout << " ____Heap_sort result: ____" << endl;
    print(H, n);
}

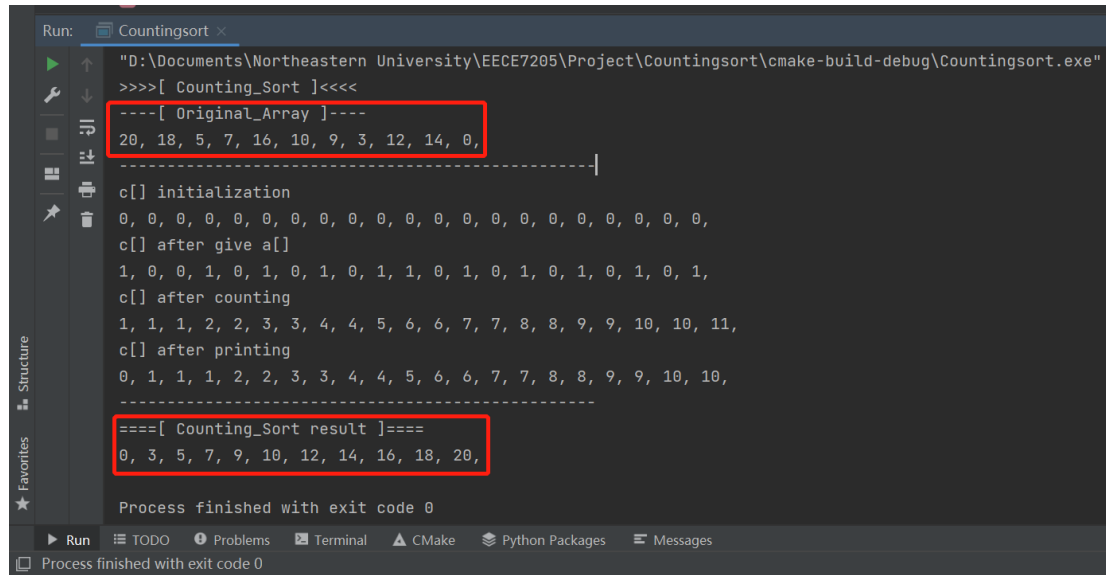
int main() {
    cout << ">>>> [ Max_heap & Heap_sort ] <<<<" << endl;
    int n;
    cout << " ____please type your n: ____" << endl;
    cin >> n; //读取目标数列总长度
    int * a;
    a = new int[n+1]; //数组指针从 0 开始, 但 heap tree 里面 key 不能为 0
    a[0] = 0; //暂且设定数组 a[0]=0
    for (int i = 1; i <= n; i++){
        {
            a[i] = i;
        }
    }
    for(int j = n; j >= 1; --j)
    {
        swap(a[j], a[rand()%j+1]);
    }
    cout << " ____Input Array: ____" << endl;
    print(a, n);
    build_max_heap(a, n);
    cout << " ____After Max_heap: ____" << endl;
    print(a, n);
    //print_heap_tree(a, n);
    heap_sort(a, n);
    return 0;
}

```

Q3:

Question 3 (1pt.) Counting Sort: Write codes for counting sort. The input array is $A = \{20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0\}$.

【program running result】



```
Run: Countingsort x
"D:\Documents\Northeastern University\EECE7205\Project\Countingsort\cmake-build-debug\Countingsort.exe"
>>>[ Counting_Sort ]<<<<
----[ Original_Array ]----
20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0,
-----|
c[] initialization
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
c[] after give a[]
1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1,
c[] after counting
1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10, 11,
c[] after printing
0, 1, 1, 1, 2, 2, 3, 3, 4, 4, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10, 10,
-----|
====[ Counting_Sort result ]====
0, 3, 5, 7, 9, 10, 12, 14, 16, 18, 20,
-----|
Process finished with exit code 0
```

【Counting sort code】_by cyx

```
//#include<bits/stdc++.h>
#include <iostream>
#include <algorithm>
#include <cstring>
using namespace std;
void print(int a[], int n) //打印输入序列{
    for (int i = 0; i < n; i++){
        cout << a[i] << " ";
    }
    cout << endl;
}
void counting_sort(int a[], int n, int k){
    //int *b, *c;
    //b = new int[n];
    //c = new int[k + 1];
    int b[n];
    int c[k+1];
    memset(c,0, sizeof(c));
    //cout << "c[] initialization" << endl;
    //print(c, k+1);
```



```

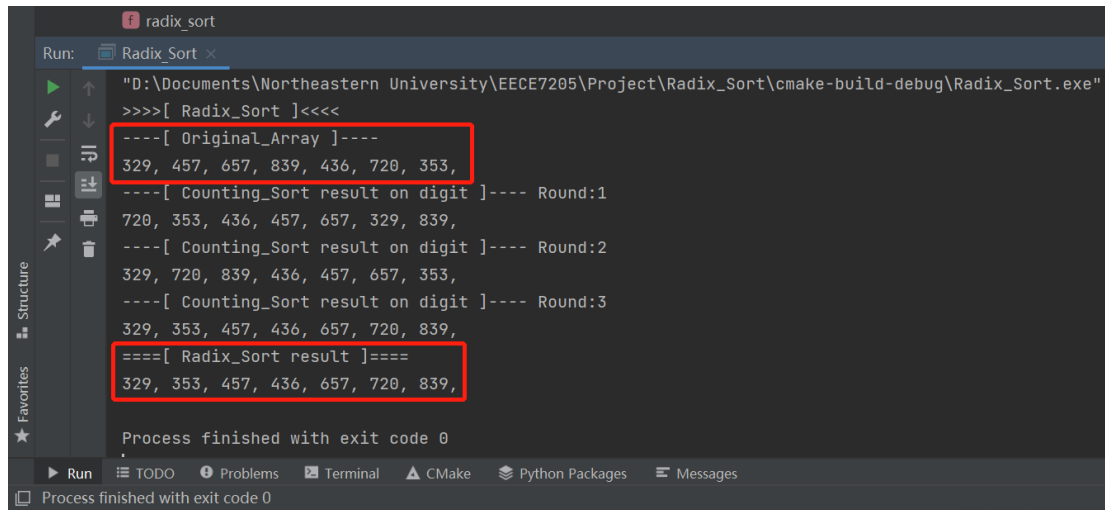
for(int j = 0; j < n; j++)
{
    c[a[j]] = c[a[j]] + 1;
}
//cout << "c[] after give a[]" << endl;
//print(c, k+1);
for(int i = 1; i <= k; i++)//数组 C 计数从指针第二位 (1) 开始累加.
{
    c[i] = c[i] + c[i-1];
}
//cout << "c[] after counting" << endl;
//print(c, k+1);
for(int j = n-1; j >= 0; j--)//此处需要考虑 A[0].
{
    b[c[a[j]]-1] = a[j];
    c[a[j]] = c[a[j]] - 1;
}
//cout << "c[] after printing" << endl;
//print(c, k+1);
//cout << "-----" << endl;
cout << "====[ Counting_Sort result ]====" << endl;
print(b, n);
//delete []b;
//delete []c;
}
int main(){
    cout << ">>>>[ Counting_Sort ]<<<<" << endl;
    int a[11] = {20, 18, 5, 7, 16, 10, 9, 3, 12, 14, 0};
    int n = 11;
    int k = 20;
    cout << "----[ Original_Array ]----" << endl;
    print(a, n);
    //cout << "-----" << endl;
    counting_sort(a, n, k);
    return 0;
}

```

Q4:

Question 4 (extra 1pt.) Radix Sort: Write codes for radix sort: use counting sort for decimal digits from the low order to high order. The input array is $A = \{329, 457, 657, 839, 436, 720, 353\}$.

【program running result】



```
radix_sort
Run: Radix_Sort x
"D:\Documents\Northeastern University\EECE7205\Project\Radix_Sort\cmake-build-debug\Radix_Sort.exe"
>>>[ Radix_Sort ]<<<<
----[ Original_Array ]----
329, 457, 657, 839, 436, 720, 353,
----[ Counting_Sort result on digit ]---- Round:1
720, 353, 436, 457, 657, 329, 839,
----[ Counting_Sort result on digit ]---- Round:2
329, 720, 839, 436, 457, 657, 353,
----[ Counting_Sort result on digit ]---- Round:3
329, 353, 457, 436, 657, 720, 839,
====[ Radix_Sort result ]====
329, 353, 457, 436, 657, 720, 839,
Process finished with exit code 0
```

【radix sort code】_by cyx

```
#include <iostream>
#include <cmath>
using namespace std;
void print(int a[], int n) //打印输入序列
{
    for (int i = 0; i < n; i++)
    {
        cout << a[i] << ", ";
    }
    cout << endl;
}
void radix_sort(int a[], int n, int k, int d)
{
    int *b, *c;
    b = new int[n];
    c = new int[k + 1];
    for(int dn = 10; dn <= pow(10,d); dn*=10)
    {
        for(int i = 0; i <= k; i++)//此处一定注意！ 课上伪代码数组取值范围 1-k;
        实际情况数组各项取值范围 0-K.
```

```

        {
            c[i] = 0;
        }
        for(int j = 0; j < n; j++)//数组 C 各位的指针与数组 C 各位对应，所以当
        计算 A[j]个数的时候，对应 C 的位置应该+1.
        {
            c[(a[j]%dn/(dn/10))] = c[(a[j]%dn/(dn/10))] + 1;
        }
        for(int i = 1; i <= k; i++)//数组 C 计数从指针第二位（1）开始累加.
        {
            c[i] = c[i] + c[i-1];
        }
        for(int j = n - 1; j >= 0; j--)//此处需要考虑 A[0].
        {
            b[c[(a[j]%dn/(dn/10))]-1] = a[j];
            c[(a[j]%dn/(dn/10))] = c[(a[j]%dn/(dn/10))] - 1;
        }
        cout << "----[ Counting_Sort result on digit ]---- Round:"<< log10(dn)
<<endl;
        print(b, n);
    }
    cout << "====[ Radix_Sort result ]===="<< endl;
    print(b, n);
}
int main()
{
    cout << ">>>>[ Radix_Sort ]<<<<" << endl;
    int a[] = {329, 457, 657, 839, 436, 720, 353};
    int n = 7;//number of elements in array a[]
    int k = 9;//max of each counting sort
    int d = 3;//relate to the number of digits
    cout << "----[ Original_Array ]----" << endl;
    print(a, n);
    radix_sort(a, n, k, d);
    return 0;
}

```