Arrays
Structures and Classes
Week 5 & 7
2017 Fall
Problem 1
Count Sort
Count Sort

• One of sorting algorithms
• Time Complexity is $O(n)$
  • A fast sorting algorithm, Quick Sort has $O(n \log n)$
• Only available in limited conditions.
  • The range of elements should be bounded
  • The range of elements should be already known
Count Sort

Data

| 4 | 3 | 8 | 2 | 3 | 6 | 1 | 1 | 7 | 3 |

Counter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Count Sort

Data

4  3  8  2  3  6  1  1  7  3

Counter

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Count Sort

Data

4  3  8  2  3  6  1  1  7  3

Counter

1  2  3  4  5  6  7  8  9  10

0  0  1  1  0  0  0  0  0  0
Count Sort

Data

| 4 | 3 | 8 | 2 | 3 | 6 | 1 | 1 | 7 | 3 |

Counter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Count Sort

Sorted Data

Counter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Count Sort

Sorted Data

Counter
Count Sort

Sorted Data

Counter
Count Sort

Sorted Data

| 1 | 1 | 2 |

Counter

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Count Sort

Sorted Data

1  1  2  3  3  3  3  4  6  7  8

Counter

1  2  3  4  5  6  7  8  9  10
2  1  3  1  0  1  1  1  0  0
Count Sort

• Input
  • 20 numbers (0~99)

• Output
  • Sorted numbers in ascending order

• Example of input and output
  > 6 4 7 65 2 6 3 7 4 1 87 34 1 6 3 2 3 54 32
  1 1 2 2 3 3 3 3 3 4 4 6 6 6 7 7 32 34 54 65 87
Count Sort

```cpp
#include <iostream>

#define DATA_CNT 20

using namespace std;

void count_sort(int data[]) {
    int cnt_arr[100] = {0,};
    int cnt = 0;

    for (int i=0; i<DATA_CNT; i++)
        cnt_arr[data[i]]++; 

    for (int i=0; i<100; i++)
        for (int j=0; j<cnt_arr[i]; j++)
            data[cnt++] = i;
}

int main() {
    int data[DATA_CNT] = {0,};

    for (int i=0; i<DATA_CNT; i++)
        cin >> data[i];

    count_sort(data);

    for (int i=0; i<DATA_CNT; i++) {
        if (i < DATA_CNT-1)
            cout << data[i] << " ";
        else
            cout << data[i] << "\n";
    }

    return 0;
}
```
Problem 2
Distance between the Points
Distance between the Points

- Printing the ‘Euclidean distance’ between the given two 2D points in two versions
  - Structure version
  - Class version
- Point consists of two double variables
- Distance is calculated as a function
  - Structure version: get_distance_between()
  - Class version: get_distance_to()
  - Use hypot() in the library <cmath>
Distance between the Points

• Refer to the following code structure
  • c.f.) Generally, the header file contains this

```c
typedef struct Point{
    double x, y;
} Point_s;

double get_distance_between(Point_s, Point_s);

class Point_c{
public:
    double x, y;
    double get_distance_to(Point_c);
}
```
Distance between the Points

• Input
  • The 1st line: The number of test cases
  • The other lines: The test cases consisting of 4 coordinates of type double
    • x1, y1, x2, y2 in order

• Output
  • Calculated distances for given coordinates
  • Two lines for each test case
    • One for the structure version
    • The other for the class version
Distance between the Points

- An example

\[ \begin{pmatrix} 3 \\ 1 & 2 & 3 & 2 \\ 2 & 4 & 2 & 1 \\ 1 & 1 & 2 & 2 \end{pmatrix} \]

\[ 1.41421 \]

\[ 1.41421 \]
Distance between the Points

```cpp
#include <iostream>
#include <cmath>

using namespace std;

typedef struct Point {
   double x, y;
} Point_s;

double get_distance_between(Point_s A, Point_s B){
    return hypot(A.x-B.x, A.y-B.y);
}

class Point_c {
   public:
      double x, y;
      double get_distance_to(Point_c);
};

double Point_c::get_distance_to(Point_c other){
    return hypot(x-other.x, y-other.y);
}
```
Distance between the Points

```cpp
int main(){
    Point_s Ps1, Ps2;
    Point_c Pc1, Pc2;
    int N;
    double x1, y1, x2, y2;

    cin >> N;
    for(int i=0; i<N; i++){
        cin >> x1 >> y1 >> x2 >> y2;

        Ps1.x = x1;
        Ps1.y = y1;
        Ps2.x = x2;
        Ps2.y = y2;

        Pc1.x = x1;
        Pc1.y = y1;
        Pc2.x = x2;
        Pc2.y = y2;

        cout << get_distance_between(Ps1, Ps2) << endl;
        cout << Pc1.get_distance_to(Pc2) << endl;
    }
}
```

If the way to use the member function of the class is different with this, The submission can be deducted although the ‘goorm’ result is full score.