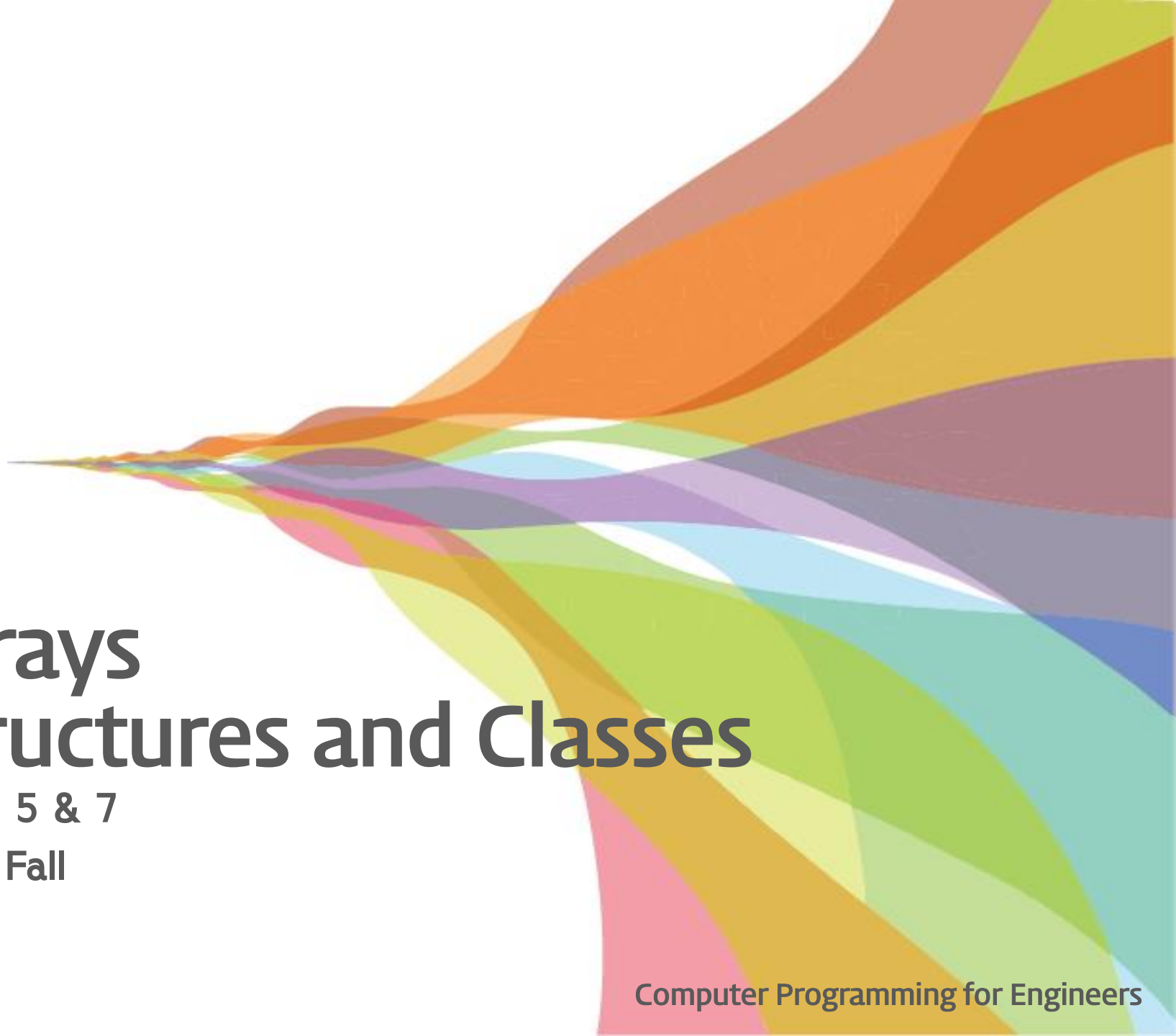


Arrays Structures and Classes

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

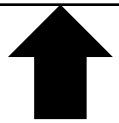
Counter

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

Count Sort

Data

4	3	8	2	3	6	1	1	7	3



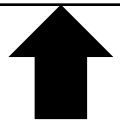
Counter

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

Count Sort

Data

4	3	8	2	3	6	1	1	7	3



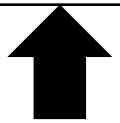
Counter

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

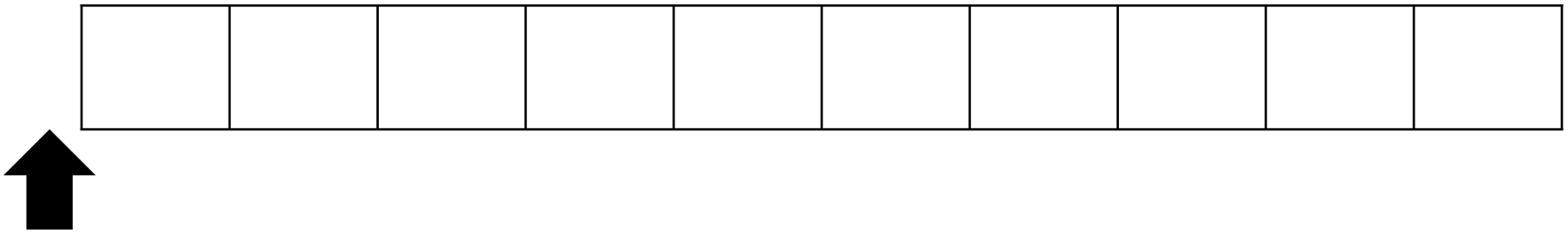


Counter

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

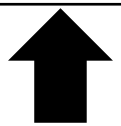
Count Sort

Sorted Data



Counter

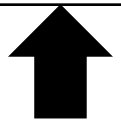
1	2	3	4	5	6	7	8	9	10
2	1	3	1	0	1	1	1	0	0



Count Sort

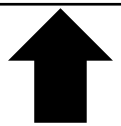
Sorted Data

1									



Counter

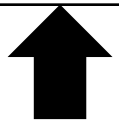
1	2	3	4	5	6	7	8	9	10
2	1	3	1	0	1	1	1	0	0



Count Sort

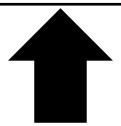
Sorted Data

1	1								



Counter

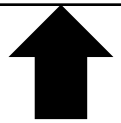
1	2	3	4	5	6	7	8	9	10
2	1	3	1	0	1	1	1	0	0



Count Sort

Sorted Data

1	1	2							



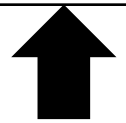
Counter

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

Count Sort

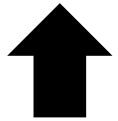
Sorted Data

1	1	2	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 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;
}
```

```cpp
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

```
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

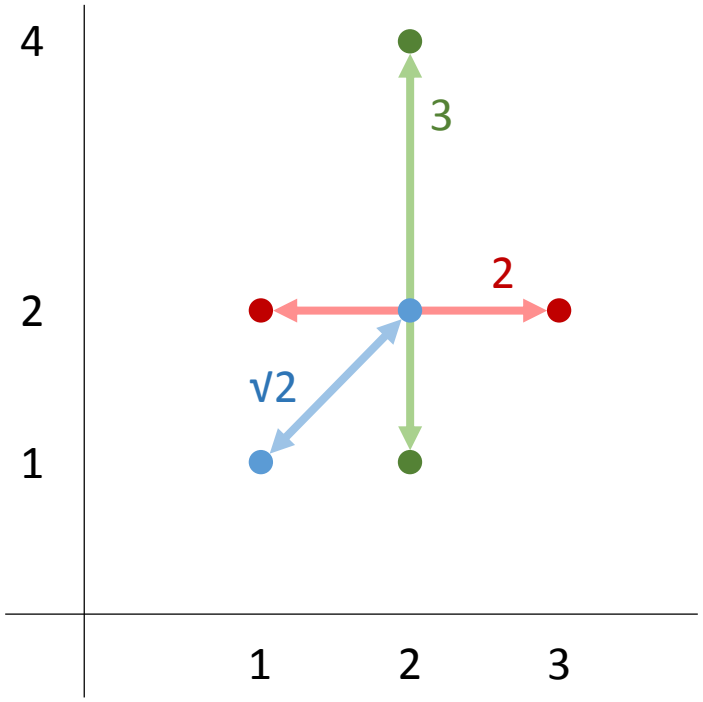
- Input
  - The 1st line: The number of test cases
  - The other lines: The test cases consisting of 4 coordinates of type `double`
    - x1, y1, x2, y2 in order
- Output
  - Calculated distances for given coordinates
  - Two lines for each test case
    - One for the structure version
    - The other for the class version

Distance between the Points

- An example

```
> 3
1 2 3 2
2 4 2 1
1 1 2 2

2
2
3
3
1.41421
1.41421
```



Distance between the Points

```
 1  #include <iostream>
 2  #include <cmath>
 3
 4  using namespace std;
 5
 6  typedef struct Point {
 7      double x, y;
 8  } Point_s;
 9
10  double get_distance_between(Point_s A, Point_s B){
11      return hypot(A.x-B.x, A.y-B.y);
12  }
13
14  class Point_c {
15  public:
16      double x, y;
17      double get_distance_to(Point_c);
18  };
19
20  double Point_c::get_distance_to(Point_c other){
21      return hypot(x-other.x, y-other.y);
22  }
```

Distance between the Points

```
24  int main(){
25      Point_s Ps1, Ps2;
26      Point_c Pc1, Pc2;
27      int N;
28      double x1, y1, x2, y2;
29
30      cin >> N;
31      for(int i=0; i<N; i++){
32          cin >> x1 >> y1 >> x2 >> y2;
33
34          Ps1.x = x1;
35          Ps1.y = y1;
36          Ps2.x = x2;
37          Ps2.y = y2;
38
39          Pc1.x = x1;
40          Pc1.y = y1;
41          Pc2.x = x2;
42          Pc2.y = y2;
43
44          cout << get_distance_between(Ps1, Ps2) << endl;
45          cout << Pc1.get_distance_to(Pc2) << endl;
46      }
47  }
```

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.