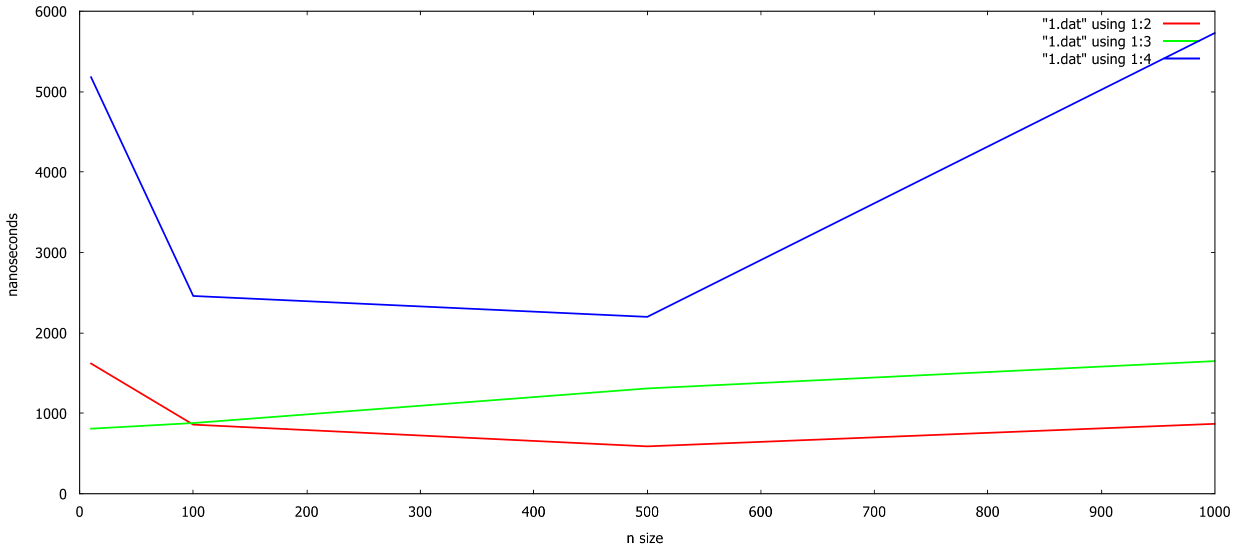
I have to use the data run my program from https://www.onlinegdb.com/online_c++_compiler.

My computer seems too fast, I get all zero at runtime, even use nanoseconds.

I will show printscreen of www.onlinegdb.com/online_c++_compiler at final



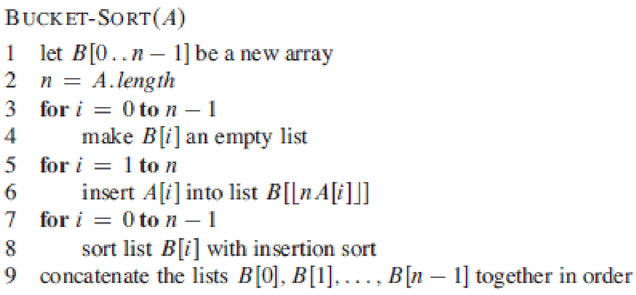
Red Line – randomized-select

Green Line – counting sort

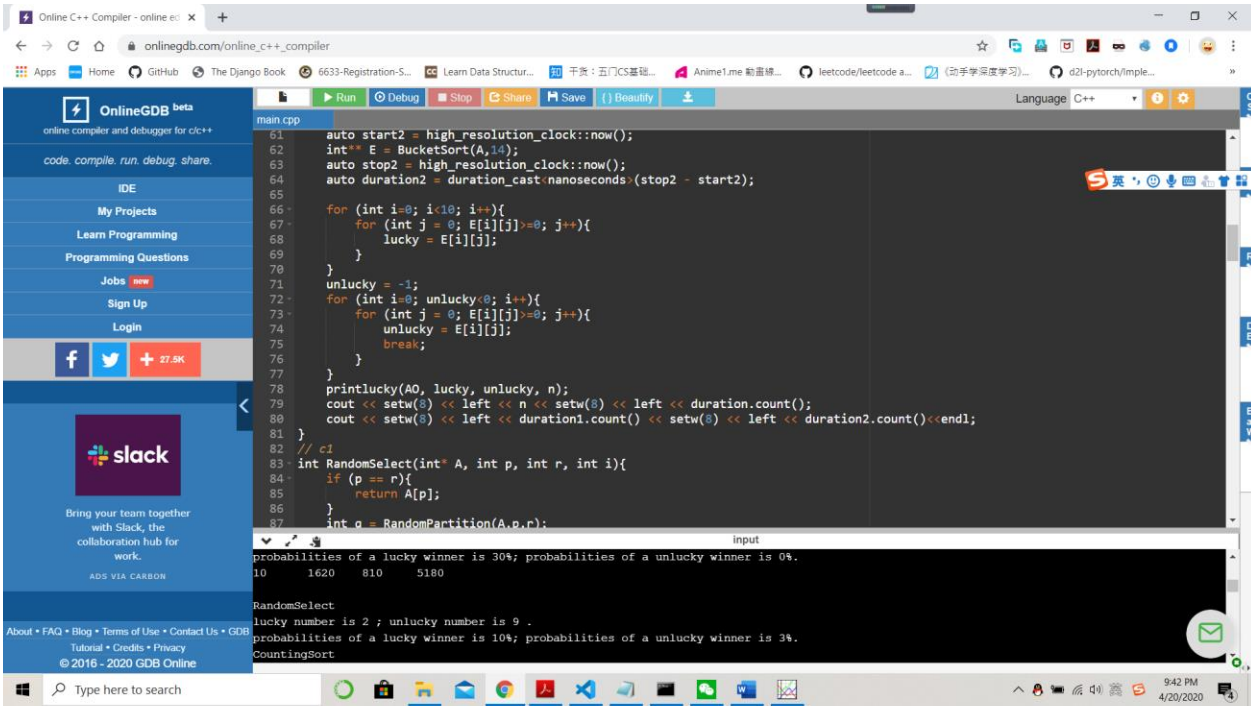
Blue Line – bucket sort

Explanation:

1. Time complexity of counting sort is O(n), when data closer, time complexity of randomized-select will be closer to O(n). When n increase, the randomness decrease, which make data closer
2. The reason why runtime of bucket sort is unstable may be that I use insertion sort to sort bucket as book written.
3. Probabilities of lucky and unlucky is closer when n grow higher, because as n increases, the randomness decreases

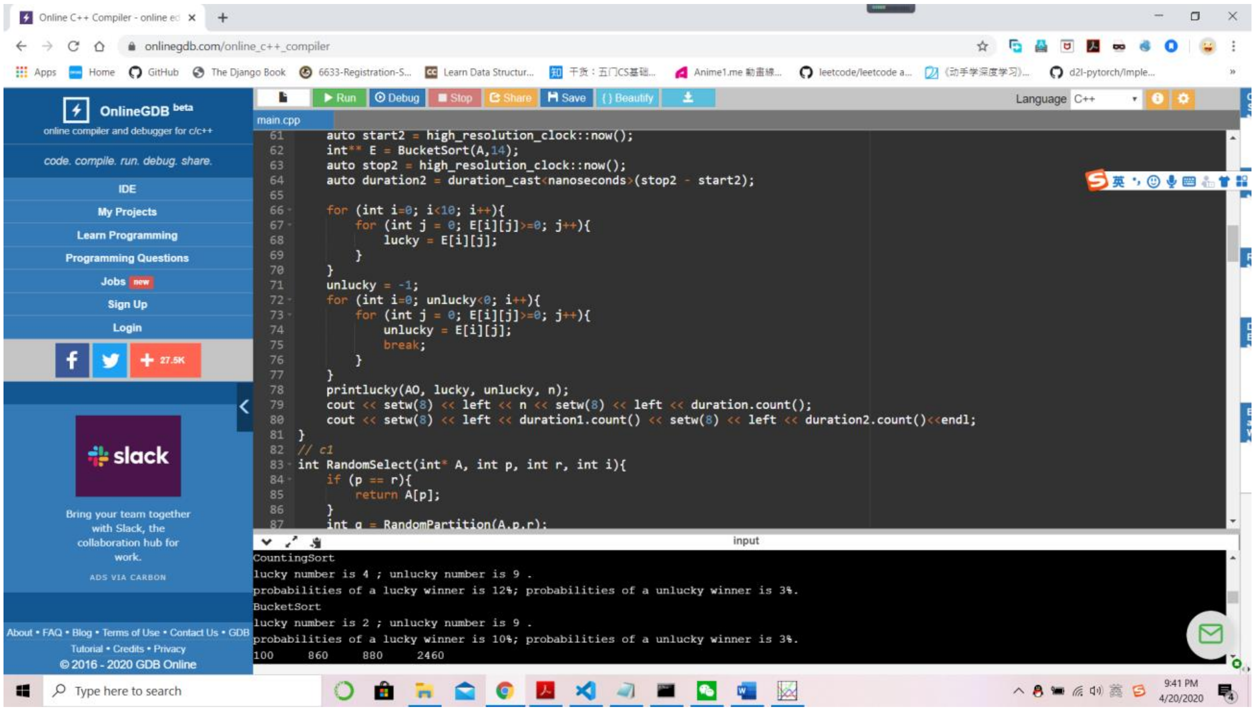


```
BUCKET-SORT($A$)
let $B[0\,..\,n-1]$ be a new array
$n = A.length$
for $i = 0$ to $n - 1$
    make $B[i]$ an empty list
for $i = 1$ to $n$
    insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$
for $i = 0$ to $n - 1$
    sort list $B[i]$ with insertion sort
concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order
```



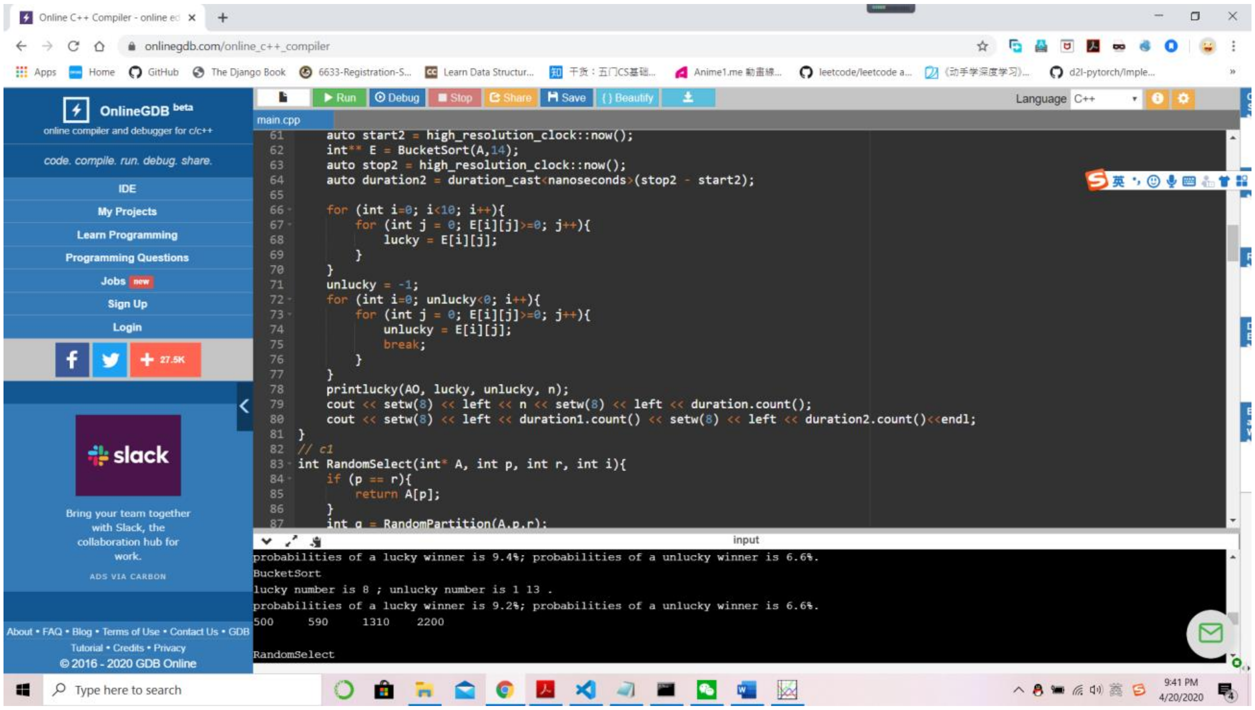
```
    auto start2 = high_resolution_clock::now();
    int** E = BucketSort(A,14);
    auto stop2 = high_resolution_clock::now();
    auto duration2 = duration_cast<nanoseconds>(stop2 - start2);

    for (int i=0; i<10; i++){
        for (int j = 0; E[i][j]>=0; j++){
            lucky = E[i][j];
        }
    }
    unlucky = -1;
    for (int i=0; unlucky<0; i++){
        for (int j = 0; E[i][j]>=0; j++){
            unlucky = E[i][j];
            break;
        }
    }
    printlucky(AO, lucky, unlucky, n);
    cout << setw(8) << left << n << setw(8) << left << duration.count();
    cout << setw(8) << left << duration1.count() << setw(8) << left << duration2.count()<<endl;
}
// c1
int RandomSelect(int* A, int p, int r, int i){
    if (p == r){
        return A[p];
    }
    int q = RandomPartition(A,p,r);
```

```
probabilities of a lucky winner is 30%; probabilities of a unlucky winner is 0%.
10      1620    810     5180

RandomSelect
lucky number is 2 ; unlucky number is 9 .
probabilities of a lucky winner is 10%; probabilities of a unlucky winner is 3%.
CountingSort
```



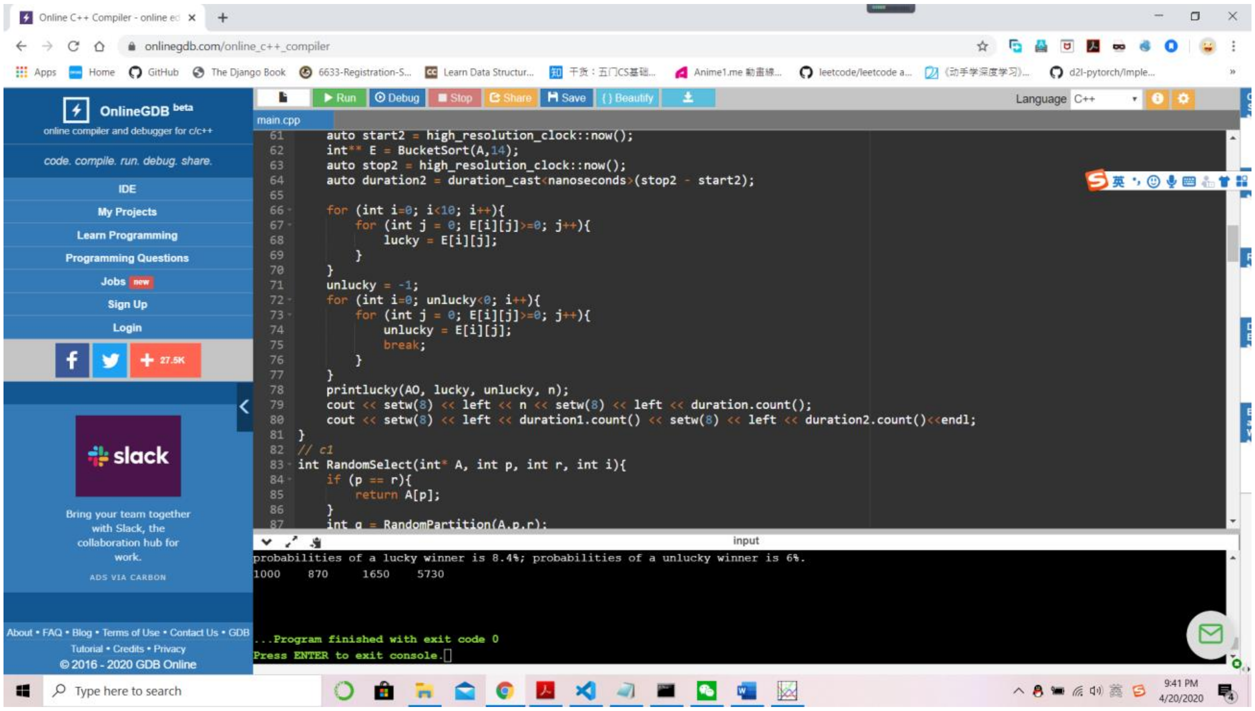
```
    auto start2 = high_resolution_clock::now();
    int** E = BucketSort(A,14);
    auto stop2 = high_resolution_clock::now();
    auto duration2 = duration_cast<nanoseconds>(stop2 - start2);

    for (int i=0; i<10; i++){
        for (int j = 0; E[i][j]>=0; j++){
            lucky = E[i][j];
        }
    }
    unlucky = -1;
    for (int i=0; unlucky<0; i++){
        for (int j = 0; E[i][j]>=0; j++){
            unlucky = E[i][j];
            break;
        }
    }
    printlucky(AO, lucky, unlucky, n);
    cout << setw(8) << left << n << setw(8) << left << duration.count();
    cout << setw(8) << left << duration1.count() << setw(8) << left << duration2.count()<<endl;
}
// c1
int RandomSelect(int* A, int p, int r, int i){
    if (p == r){
        return A[p];
    }
    int q = RandomPartition(A,p,r);
```

```
CountingSort
lucky number is 4 ; unlucky number is 9 .
probabilities of a lucky winner is 12%; probabilities of a unlucky winner is 3%.
BucketSort
lucky number is 2 ; unlucky number is 9 .
probabilities of a lucky winner is 10%; probabilities of a unlucky winner is 3%.
100     860     880     2460
```



```cpp
    auto start2 = high_resolution_clock::now();
    int** E = BucketSort(A,14);
    auto stop2 = high_resolution_clock::now();
    auto duration2 = duration_cast<nanoseconds>(stop2 - start2);

    for (int i=0; i<10; i++){
        for (int j = 0; E[i][j]>=0; j++){
            lucky = E[i][j];
        }
    }
    unlucky = -1;
    for (int i=0; unlucky<0; i++){
        for (int j = 0; E[i][j]>=0; j++){
            unlucky = E[i][j];
            break;
        }
    }
    printlucky(AO, lucky, unlucky, n);
    cout << setw(8) << left << n << setw(8) << left << duration.count();
    cout << setw(8) << left << duration1.count() << setw(8) << left << duration2.count()<<endl;
}
// c1
int RandomSelect(int* A, int p, int r, int i){
    if (p == r){
        return A[p];
    }
    int q = RandomPartition(A,p,r);
```

```
probabilities of a lucky winner is 9.4%; probabilities of a unlucky winner is 6.6%.
BucketSort
lucky number is 8 ; unlucky number is 1 13 .
probabilities of a lucky winner is 9.2%; probabilities of a unlucky winner is 6.6%.
500     590     1310    2200

RandomSelect
```



```cpp
    auto start2 = high_resolution_clock::now();
    int** E = BucketSort(A,14);
    auto stop2 = high_resolution_clock::now();
    auto duration2 = duration_cast<nanoseconds>(stop2 - start2);

    for (int i=0; i<10; i++){
        for (int j = 0; E[i][j]>=0; j++){
            lucky = E[i][j];
        }
    }
    unlucky = -1;
    for (int i=0; unlucky<0; i++){
        for (int j = 0; E[i][j]>=0; j++){
            unlucky = E[i][j];
            break;
        }
    }
    printlucky(AO, lucky, unlucky, n);
    cout << setw(8) << left << n << setw(8) << left << duration.count();
    cout << setw(8) << left << duration1.count() << setw(8) << left << duration2.count()<<endl;
}
// c1
int RandomSelect(int* A, int p, int r, int i){
    if (p == r){
        return A[p];
    }
    int q = RandomPartition(A,p,r);
```

```
probabilities of a lucky winner is 8.4%; probabilities of a unlucky winner is 6%.
1000    870     1650    5730



...Program finished with exit code 0
Press ENTER to exit console.
```