Ocean Lu
CS 4080.02
Professor Yang
09/19/2019

Assignment #2

1. For this problem our task is to store 8000 randomly generated integers in the range of 1 to 1000 inclusively using different languages/methods. Note: in all above tasks store element one by one.

Task 1:

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

/*C++. declare a stack dynamic array and store randomly generated numbers one by one to the array.*/
int main() {
  //Store starting time
  clock_t begin = clock();

  //CODE
  const int SIZE = 8000;
  int numbers[SIZE];
  for (int i = 0; i < SIZE; i++) {
    numbers[i] = (rand()%1000) + 1;
  }

  cout << (double(clock() - begin) / CLOCKS_PER_SEC) * 1000;
}
```

Task 2:

```cpp
#include <iostream>
#include <cstdlib>
#include <ctime>
using namespace std;

/*C++. Same as above but using a heap dynamic array, i.e.*/
int main() {
  //Store starting time
  clock_t begin = clock();

  const int SIZE = 8000;
  int *num_ref = new int [SIZE];
  for (int j = 0; j < SIZE; j++) {
    *num_ref = (rand()%1000) + 1;
  }

  cout << (double(clock() - begin) / CLOCKS_PER_SEC) * 1000;
}
```

Task 3:

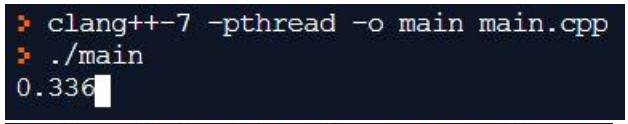
```java
/*Java. Same as above but now using a Java array. */
public class Assignment2 {
    public static void main(String[] args) {
        //Store starting time
        long startTime = System.currentTimeMillis();

        // CODE
        int SIZE = 8000;
        int[] arr = new int[SIZE];
        for (int i = 0; i < SIZE; i++) {
            int random = (int) (Math.random() * SIZE + 1);
            arr[i] = random;
        }

        System.out.println("Time taken : " + ( System.currentTimeMillis() - startTime ) + " millisecond(s)." );
    }
}
```

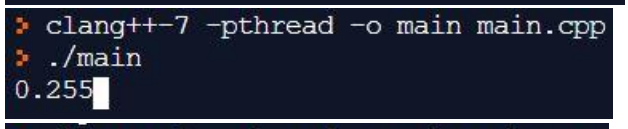
Task 4:

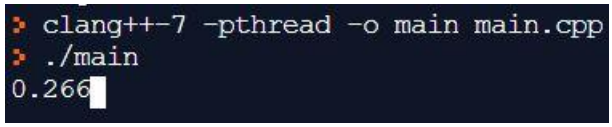
```java
/*Same as above but now using a Java ArrayList. */
import java.util.*;
public class Assignment2 {
    public static void main(String[] args) {
        //Store starting time
        long startTime = System.currentTimeMillis();

        //CODE
        int SIZE = 8000;
        ArrayList<Integer> arrli = new ArrayList<Integer>(SIZE);
        for (int i = 0; i < SIZE; i++) {
            int random = (int) (Math.random() * SIZE + 1);
            arrli.add(random);
        }

        System.out.println("Time taken : " + (System.currentTimeMillis() - startTime) + " millisecond(s).");
    }
}
```

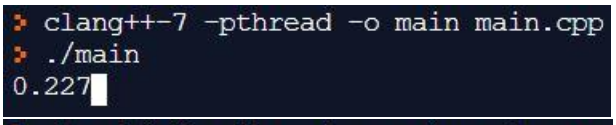
Calculate the average computation time for each of above tasks by running the code 10 times and then take the average of the 10 test runs.

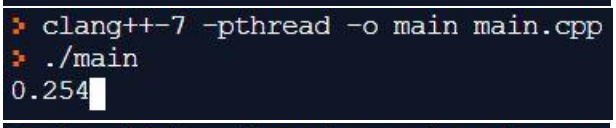
Task 1: Average: (0.336 + 0.225 + 0.266 + 0.227 + 0.254 + 0.219 + 0.255 + 0.221 + 0.215 + 0.326) / 10 = 0.544 milliseconds

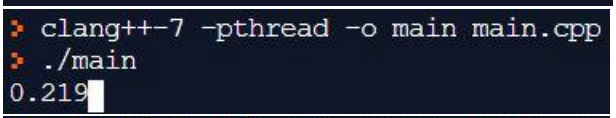
1. 

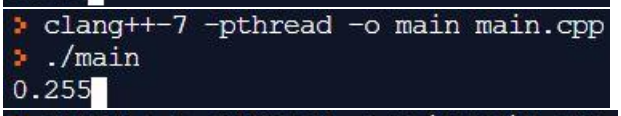
2. 

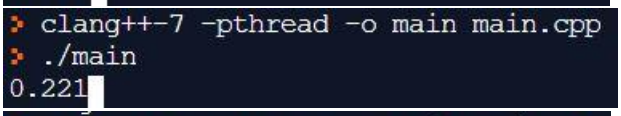
3. 

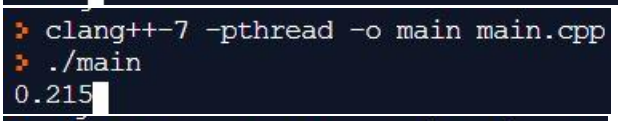
4. 

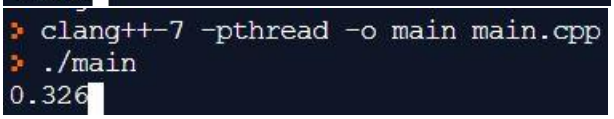
5. 

6. 

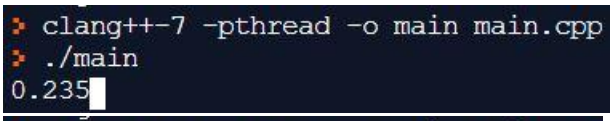
7. 

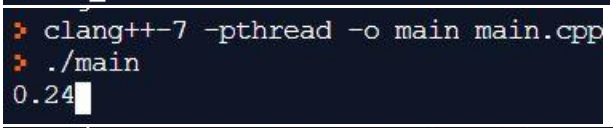
8. 

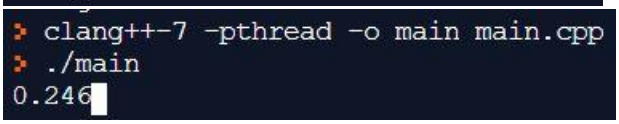
9. 

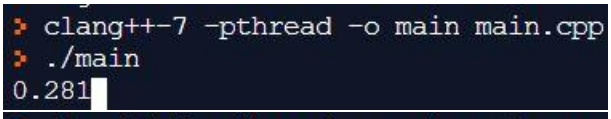
10. 

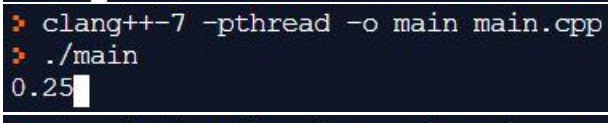
Task 2: Average: (0.235 + 0.24 + 0.246 + 0.281 + 0.25 + 0.239 + 0.201 + 0.322 + 0.253 + 0.255) / 10 = 0.2522 milliseconds

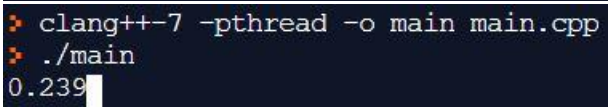
1. 

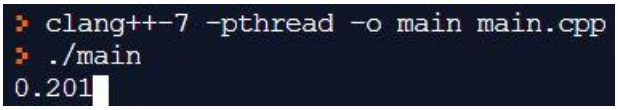
2. 

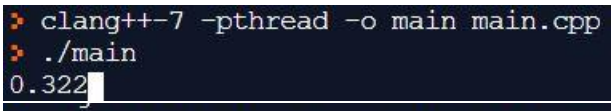
3. 

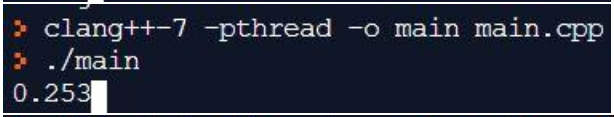
4. 

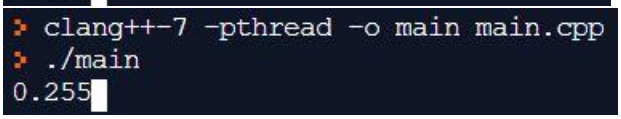
5. 

6. 

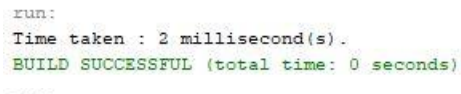
7. 

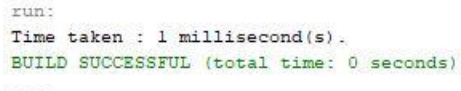
8. 

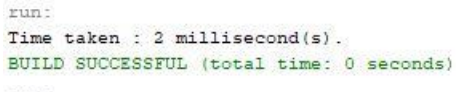
9. 

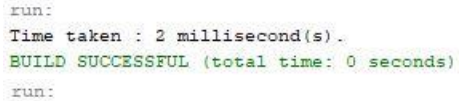
10. 

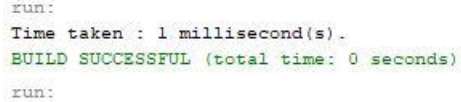
Task 3: Average: (2 + 1 + 2 + 2 + 1 + 1 + 2 + 1 + 1 + 1) / 10 = 1.4 millisecond

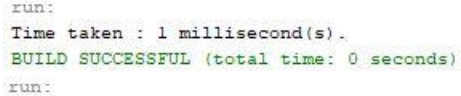
1. 

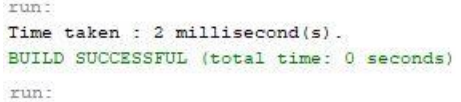
2. 

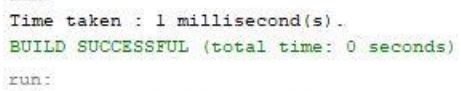
3. 

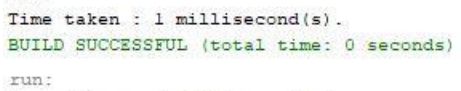
4. 

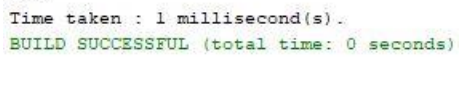
5. 

6. 

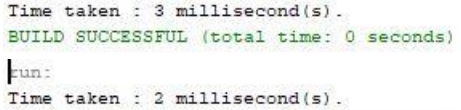
7. 

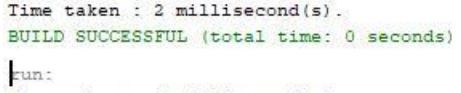
8. 

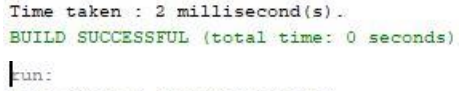
9. 

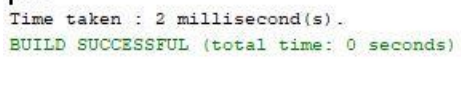
10. 

Task 4: Average: (3 + 2 + 2 + 2 +2 + 3 + 2 + 3 + 3 + 3) / 10 = 2.5 milliseconds

1. 

2. 

3. 

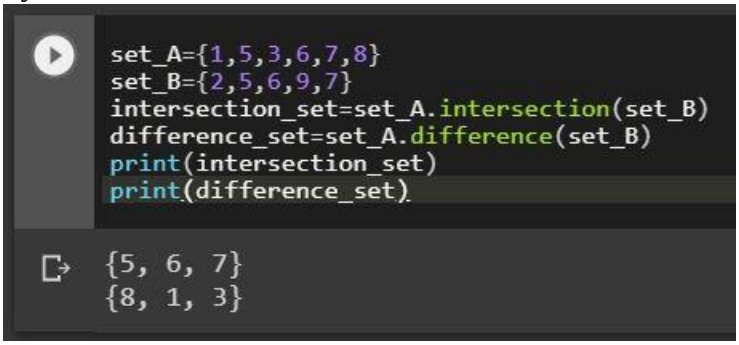
4. 

5.
```
run:
Time taken : 2 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```
6.
```
run:
Time taken : 3 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```
7.
```
run:
Time taken : 2 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```
8.
```
run:
Time taken : 3 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```
9.
```
run:
Time taken : 3 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```
10.
```
run:
Time taken : 3 millisecond(s).
BUILD SUCCESSFUL (total time: 0 seconds)
```

(1) Task 1 vs. Task 2. Which one is easier to program? Which task runs faster or are they similar in performance? Why? i.e. Describe your observation and then briefly justify your answer.
C++'s heap dynamic array was most difficult to program, but because it directly goes to the addresses, it is most effective of the four programs.

(2) Task 3 vs. Task 4. Which one is easier to program? What's the advantage of using ArrayList? Which task runs faster or are they similar in performance? Why? i.e. Describe your observation and then briefly justify your answer.
Java's ArrayLists were very intuitive and very simple to use. Its easy functions are easy to understand and very flexible. However, it is the longest run time of these four programs.

C++, Java, Python. Assume we have two sequences of values S1 containing 1, 5, 3, 6, 7, 8 while S2 containing 2, 5, 6, 9, 7. We'd store these two sequences as sets and performance set intersection and set difference. Write C++, Java, Python codes to do that respectively. Compare and contrast the readability and writability.
Python:



Java:

```java
import java.util.*;
public class Assignment2 {
    public static void main(String args[]) {
        Set<Integer> set1 = new HashSet<Integer>();
        Set<Integer> set2 = new HashSet<Integer>();
        set1.addAll(Arrays.asList(new Integer[]{1, 5, 3, 6, 7, 8}));
        set2.addAll(Arrays.asList(new Integer[]{2, 5, 6, 9, 7}));
        Set<Integer> Inter = new HashSet<Integer>(set1);
        Inter.retainAll(set2);
        System.out.println(Inter);
        Set<Integer> difference_set = new HashSet<Integer>(set1);
        difference_set.removeAll(set2);
        System.out.println(difference_set);
    }
}
```

Output - [CS4080]Assignment2 (run)

```
run:
[5, 6, 7]
[1, 3, 8]
BUILD SUCCESSFUL (total time: 0 seconds)
```

C++:

```cpp
#include <iostream>
#include <vector>
#include <algorithm>
#include <set>
using namespace std;

int main() {
    set<int> set1;
    set<int> set2;
    set1.insert(1);
    set1.insert(5);
    set1.insert(3);
    set1.insert(6);
    set1.insert(7);
    set1.insert(8);
    set2.insert(2);
    set2.insert(5);
    set2.insert(6);
    set2.insert(9);
    set2.insert(7);
    set<int> output;
    set<int> output2;
    set_intersection(set1.begin(),set1.end(),set2.begin(),set2.end(),std::inserter(output,output.begin()));
    set_difference(set1.begin(),set1.end(), set2.begin(), set2.end(),inserter(output2,output2.end()));
}
```

I am more proficient, so I felt that Java had a better writability, although Python has very strong readability and writability for those who are new to coding. Python's offers uncluttered simple-to-learn syntax which would help programmers that are utilizing it.