Ryan Gunawan
CS331-01
January 23, 2017

**Programming Assignment #1 – Analysis of Multiple Algorithms of Determining Intersections**

*Assignment Prompt:*

1. Write programs to implement the algorithm of HW#1 (part 1) and compute the CPU time for different sets of integers generated by a random number generator.
2. Write programs to implement the algorithm of HW#1 (part 2) using two different data structures (HashSet, BitSet, or TreeSet) and compute the CPU time for different sets of integers generated by a random number generator.

*Approach:*

In order to analyze the efficiency of the different intersection algorithms, a common set of randomly generated integers would need to be produced for use in each of the cases. Using a random number generator, two different integer arrays were filled. One constraint made against the requirements was to limit the range of the random number generator. This constraint was implemented because the random number generator would not likely produce any intersecting integers between the two sets, and by limiting the range, the algorithms would be able to produce results. To compare the CPU time for each of the algorithms, *System.nanoTime()* was used to record the time prior to starting the intersecting algorithm and the time after the algorithm was completed.

For the first part of this assignment, some of the different cases were noted prior to developing the algorithm for determining the intersection of the two sets of integers. Most notably were the cases that proved that there were no possible intersections between the two sets. By implementing these cases, the algorithm would become more efficient in determining the intersections. These cases included when the minimum integer value of one set is greater than the maximum integer value of the other set or vice versa.

The derived efficient intersection algorithm was constructed such that the function would increment through the first value in the first sorted integer array and compare it with the first value in the second sorted integer array. If the value from the first set is larger than the second set, the algorithm would increment through the second set until an integer value was greater than or equal to the value in the first set. This would also be true for the reverse case where the value from the second set is larger from the first. If the values are equal, the integer value would be written to a separate integer array that would be used to maintain the intersecting integer values from both sets. This would repeat until the algorithm completes processing through either of the integer sets.

The second part of this assignment required looking at different data structures and how they compared against the developed algorithm with unsorted integer arrays. The data structures implemented and analyzed in this program were HashSet and TreeSet. The HashSet implementation included creating a HashSet of the first set of integers and comparing the second set with the HashSet using the *contains()* function. The TreeSet implementation utilized a similar approach to the HashSet implementation where the first set of integers was inserted using *add()* and then the second set of integers were compared against the TreeSet using *contains()*.

Ryan Gunawan
CS331-01
January 23, 2017

*Results:*

The resulting computational times for several cases of differently filled integers are listed below in Table 1.

| Set 1 Integers | Set 2 Integers | Efficient Algorithm CPU Time (nanoseconds) | HashSet CPU Time (nanoseconds) | TreeSet CPU Time (nanoseconds) | Java Sort CPU Time (nanoseconds) |
|---|---|---|---|---|---|
| 1,000 | 1,000 | 84732 | 1758048 | 3122180 | 432676 |
| 1,000 | 3,000 | 94347 | 1959062 | 3943665 | 366874 |
| 2,000 | 2,000 | 120789 | 1777879 | 4358313 | 602441 |
| 1,000 | 1,000 | 78422 | 1000564 | 4337279 | 596432 |
| 1,000 | 1,000 | 54085 | 1189559 | 3124884 | 358461 |
| 10,000 | 10,000 | 677259 | 3852021 | 9241846 | 2537766 |
| 10,000 | 10,000 | 604846 | 3856829 | 9258071 | 2387231 |
| 10,000 | 10,000 | 520413 | 5639816 | 11888382 | 2553992 |
| 100,000 | 100,000 | 4550013 | 26362006 | 32172489 | 24456728 |
| 100,000 | 100,000 | 3516096 | 28704168 | 33780302 | 12428027 |
| 100,000 | 100,000 | 3478837 | 25916410 | 33430555 | 24173686 |
| 1,000,000 | 1,000,000 | 11515800 | 178008445 | 294560027 | 99556717 |
| 1,000,000 | 1,000,000 | 11519707 | 179074211 | 284104283 | 111087241 |
| 1,000,000 | 1,000,000 | 11917829 | 178154773 | 285212115 | 84450003 |
| 10,000,000 | 10,000,000 | 94637729 | 5588757086 | 4478809828 | 895491843 |
| 10,000,000 | 10,000,000 | 93611925 | 5478161714 | 4095932753 | 902480468 |
| 10,000,000 | 10,000,000 | 90301951 | 5180217798 | 3941837486 | 840637501 |

Table 1 – CPU Time Results for different intersection algorithms

For the multiple cases analyzed, it is evident that the efficient algorithm developed is much more efficient in determining the intersections between two different sets of integers. Some of the cases identified included both sets having the same amount of integer counts, one set having three times more integers than the other, and one set having ten times more integers than the other.

Ryan Gunawan
CS331-01
January 23, 2017

One interesting characteristic found is that when two different sets of integers contain a different number of integers, the CPU time required to determine the intersection are different. To determine the efficiency of the different algorithms and how they compare against different cases, a test was developed. The test was to compare 4,000 integers and how the different algorithms would be affected by the different sizes. The first instance utilized the same number of integers in both sets (2,000) and the second instance utilized different counts of integers (1,000 in Set 1 and 3,000 in Set 2). In this test case, the derived efficient algorithm and the TreeSet implementation of the intersection algorithm performed faster when using differently sized integer arrays instead of similarly sized integer arrays. For the HashSet implementation however, the algorithm proved faster when the two integer arrays were similarly sized.

Ryan Gunawan
CS331-01
January 23, 2017

```java
// Source Code

// Programming Assignment #1
// Ryan Gunawan
// CS331-01 | Winter 2017
import java.util.*;

public class PA1 {

	public static void main(String[] args) {
		Random generator = new Random();
		long initTime, phase1Time, phase2Time, phase3Time;

		// Establish the two sets of integers
		int[] set1 = new int[1000];
		int[] set2 = new int[1000];


		// Generation of random integer to fill the two sets
		for (int a = 0; a < set1.length; a++){
			set1[a] = generator.nextInt(100000);
		}
		for (int a = 0; a < set2.length; a++){
			set2[a] = generator.nextInt(100000);
		}

		// Sort integer sets
		int[] sorted1 = set1;
		Arrays.sort(sorted1);
		int[] sorted2 = set2;
		Arrays.sort(sorted2);

		System.out.println("Determine the intersection of two sets of integers.");
		/*
		// Output sets for debugging purposes
		System.out.println("\nSet 1: ");
		for (int ct = 0; ct < sorted1.length; ct++){
			System.out.print(sorted1[ct] + " ");
		}
		System.out.println("\nSet 2: ");
		for (int ct = 0; ct < sorted2.length; ct++){
			System.out.print(sorted2[ct] + " ");
		}
		*/
		// Record time to starting the determination of the intersection
		initTime = System.nanoTime();
		int[] sortedIntersection = intersect(sorted1,sorted2);
		// Subtract current time from initialized time to find the time it took
		//	to find the intersection of the two sets
		phase1Time = System.nanoTime() - initTime;

		System.out.println("\nUsing Efficient Intersection: ");
		System.out.println("Time: " + phase1Time);
		for (int ct = 0; ct < sortedIntersection.length; ct++){
			System.out.print(sortedIntersection[ct] + " ");
		}
```

```
		// Record time to starting the determination of the intersection
		initTime = System.nanoTime();
		int[] hashIntersection = intersect2(set1,set2);
		// Subtract current time from initialized time to find the time it took
		//	to find the intersection of the two sets
		phase2Time = System.nanoTime() - initTime;

		System.out.println("\n\nUsing HashSet Intersection: ");
		System.out.println("Time: " + phase2Time);
		for (int ct = 0; ct < hashIntersection.length; ct++){
			System.out.print(hashIntersection[ct] + " ");
		}

		// Record time to starting the determination of the intersection
		initTime = System.nanoTime();
		int[] treeIntersection = intersect3(set1,set2);
		// Subtract current time from initialized time to find the time it took
		//	to find the intersection of the two sets
		phase3Time = System.nanoTime() - initTime;

		System.out.println("\n\nUsing TreeSet Intersection: ");
		System.out.println("Time: " + phase3Time);
		for (int ct = 0; ct < treeIntersection.length; ct++){
			System.out.print(treeIntersection[ct] + " ");
		}

	}

	public static int[] intersect(int[] a1, int[] a2){
		// Best-Case scenario: no possible intersections
		if (a1[0] > a2[a2.length-1] || a2[0] > a1[a1.length-1]){
			return null;
		}

		int i = 0, j = 0, k= 0;
		int[] out;

		// Using worst-case: all integers match as output array
		if (a1.length > a2.length)
			out = new int[a2.length];
		else
			out = new int[a1.length];

		while(i < a1.length && j < a2.length){
			if (a1[i] > a2[j] ){
				j++;
			} else if(a1[i] < a2[j]) {
				i++;
			} else {
				out[k] = a1[i];
				i++;
				j++;
				k++;
			}
		}
```

```
		return out;
	}

	public static int[] intersect2(int[] input1, int[] input2){
		// Implementation of HashSet to find intersection
		HashSet<Integer> s = new HashSet<Integer>();
		int[] out = new int[input1.length];
		int k = 0;
		for (int i = 0; i < input1.length-1; i++){
			s.add(input1[i]);
		}

		for(int j = 0; j < input2.length-1; j++){
			if ( s.contains(input2[j]) ){
				out[k] = input2[j];
				k++;
			}

		}
		return out;

	}

	public static int[] intersect3(int[] input1, int[] input2){
		// Implementation of TreeSet to find intersection
		TreeSet<Integer> a = new TreeSet<Integer>();
		int[] out = new int[input1.length];
		int k = 0;
		for (int i = 0; i < input1.length-1; i++){
			a.add(input1[i]);
		}
		for (int j = 0; j < input2.length-1; j++){
			if ( a.contains(input2[j]) ){
				out[k] = input2[j];
				k++;
			}
		}
		return out;
	}

}
```

Ryan Gunawan
CS331-01
January 23, 2017

Console Output

// Using 1,000 integers (range: 0-10000) in both Set1 and Set2

```
Determine the intersection of two sets of integers.

Using Efficient Intersection:
Time: 119480
10918 16126 32298 35066 35597 40715 49219 51376 53999 62416 65999 75040 77069 80977 84327
85023 0 …

Using HashSet Intersection:
Time: 3111485
10918 16126 32298 35066 35597 40715 49219 51376 53999 62416 65999 75040 77069 80977 84327
85023 0 …

Using TreeSet Intersection:
Time: 5235170
10918 16126 32298 35066 35597 40715 49219 51376 53999 62416 65999 75040 77069 80977 84327
85023 0 …
```

// Set1: 1,000 integers; Set2: 2,000 integers

```
Determine the intersection of two sets of integers.

Using Efficient Intersection:
Time: 281848
11638 15050 16147 19264 21803 23990 24007 25202 27292 28235 32809 35008 38334 39581 42278
43787 43929 44840 52088 67817 71389 79167 80368 82377 82504 0 …

Using HashSet Intersection:
Time: 5996939
11638 15050 16147 19264 21803 23990 24007 25202 27292 28235 32809 35008 38334 39581 42278
43787 43929 44840 52088 67817 71389 79167 80368 82377 82504 0 …

Using TreeSet Intersection:
Time: 8567209
11638 15050 16147 19264 21803 23990 24007 25202 27292 28235 32809 35008 38334 39581 42278
43787 43929 44840 52088 67817 71389 79167 80368 82377 82504 0 …
```

// Set1: 1,000 integers; Set2: 10,000 integers

```
Determine the intersection of two sets of integers.

Using Efficient Intersection:
Time: 400049
4882 4991 5359 6989 7864 12288 14185 16373 16538 16939 17203 17630 18017 20578 21462 22759
25111 25720 26188 26860 27493 28150 29222 32266 34115 34522 34742 36265 37116 37379 39780
39865 41121 41208 42196 44237 48328 48346 48580 50317 50390 50510 51646 53867 53930 55697
57303 57739 59704 60939 61012 61256 61862 64807 66354 66996 67465 67712 69655 69992 70210
71546 71622 78340 78815 78981 79422 82433 84982 85301 85678 86131 86172 86216 87678 87823
88305 89265 91354 91882 93146 94634 97648 99964 0 …

Using HashSet Intersection:
```

Ryan Gunawan
CS331-01
January 23, 2017

```
Time: 4914452
4882 4991 5359 6989 7864 12288 14185 16373 16538 16939 17203 17630 17630 18017 20578 21462
22759 25111 25720 26188 26860 27493 28150 29222 32266 34115 34522 34742 34742 36265 37116
37379 39780 39865 41121 41208 42196 44237 44237 48328 48346 48580 50317 50390 50510 51646
53867 53930 55697 57303 57739 59704 59704 60939 61012 61256 61862 64807 66354 66996 67465
67712 69655 69992 70210 71546 71622 78340 78815 78981 79422 82433 84982 85301 85678 86131
86172 86216 87678 87823 88305 89265 91354 91882 93146 94634 97648 0 …

Using TreeSet Intersection:
Time: 15653392
4882 4991 5359 6989 7864 12288 14185 16373 16538 16939 17203 17630 17630 18017 20578 21462
22759 25111 25720 26188 26860 27493 28150 29222 32266 34115 34522 34742 34742 36265 37116
37379 39780 39865 41121 41208 42196 44237 44237 48328 48346 48580 50317 50390 50510 51646
53867 53930 55697 57303 57739 59704 59704 60939 61012 61256 61862 64807 66354 66996 67465
67712 69655 69992 70210 71546 71622 78340 78815 78981 79422 82433 84982 85301 85678 86131
86172 86216 87678 87823 88305 89265 91354 91882 93146 94634 97648 0 …
```

// Set1: 10,000 integers; Set2: 10,000 integers

```
Determine the intersection of two sets of integers.

Using Efficient Intersection:
Time: 756776
6 7 62 116 166 269 310 329 406 436 484 640 724 950 1332 1418 1523 1526 1599 1643 1658 1745
2044 2133 2173 2377 2379 2459 2469 2640 2649 2725 3148 3168 3219 3274 3281 3338 3392 3409
3454 3713 3717 3927 4106 4194 4269 4380 4613 4655 4764 5067 5118 5174 5192 5222 5252 5416
5592 5780 5982 6061 6083 6319 6385 6424 6477 6510 6566 6871 6896 7013 7038 7240 7285 7305
7428 7463 7489 7808 7894 8104 8151 8215 8574 8586 8831 9479 9513 9586 10122 10270 10296
10401 10408 10629 10722 10904 10942 10981 10984 11072 11142 11222 11365 11375 11382 11695
11800 11893 12035 12290 12300 12301 12361 12694 12908 13068 13111 13124 13132 13403 13447
13527 13637 13651 13670 13702 13784 13870 14107 14341 14406 14414 14465 14472 14504 14529
14683 14765 14781 14945 15005 15124 15131 15166 15192 15254 15275 15302 15395 15467 15515
15662 15690 15808 15878 16034 16146 16289 16329 16467 16492 16533 16599 16707 16710 16738
16989 17257 17357 17361 17363 17421 17429 17475 17527 17579 17598 17658 17870 17922 18141
18332 18447 18534 18622 18668 18672 18687 18858 19097 19323 19490 19580 19800 19839 19933
19958 19993 20396 20472 20664 20720 20943 20971 20999 21122 21223 21274 21332 21337 21450
21546 21566 21640 21658 21705 21758 21790 21934 21955 22061 22147 22181 22407 22592 22624
22663 22672 22683 22845 22887 22911 22957 23280 23284 23457 23575 23620 23873 23932 24029
24500 24503 24505 24521 24720 24943 25012 25017 25229 25379 25514 25623 25673 25842 25903
26015 26088 26243 26250 26253 26254 26259 26368 26402 26667 26697 26745 26750 27012 27033
27043 27225 27324 27723 27726 27764 27933 27962 27968 28242 28512 28517 28808 29175 29514
29646 29868 29954 30003 30103 30193 30211 30257 30308 30322 30440 30481 30627 30732 30801
30820 30840 31087 31163 31565 31795 32087 32204 32220 32368 32429 32643 32947 33047 33468
33473 33546 33569 33577 33721 33792 33839 33882 33945 34732 34972 35237 35295 35314 35351
35358 35608 35652 35717 35722 35961 36307 36332 36366 36498 36534 36584 36809 37291 37628
37641 37651 37810 38008 38132 38175 38193 38332 38427 38483 38488 38514 38520 38642 38890
39068 39080 39227 39254 39306 39338 39746 39826 40052 40184 40225 40228 40250 40303 40328
40383 40445 40529 40539 40796 40940 41151 41215 41393 41429 41525 41668 41673 41833 41958
42192 42209 42410 42468 42496 42718 42926 42931 42943 43217 43280 43497 43614 43697 43841
44126 44363 44370 44686 44705 44886 44922 44923 44936 44948 45047 45131 45163 45242 45306
45337 45441 45458 45476 45534 45638 45700 46000 46166 46240 46283 46338 46345 46427 46429
46515 46530 46532 46639 46745 46789 46860 46908 46975 47015 47039 47095 47160 47299 47359
47559 47710 47785 47799 47929 47991 48010 48013 48087 48107 48141 48295 48378 48541 48610
48740 49014 49172 49296 49310 49363 49384 49486 49544 49607 49651 49778 49864 50048 50113
50254 50526 50665 50743 50792 50803 50804 50974 51020 51066 51095 51247 51288 51387 51539
```

Ryan Gunawan
CS331-01
January 23, 2017

```
51543 51672 51754 51797 51856 51898 52274 52403 52505 52515 52590 52630 52699 52715 52887
53130 53334 53336 53665 53709 53899 53918 54086 54397 54422 54667 54858 54864 55238 55398
55462 55496 55540 55555 55562 55623 55625 55722 55957 56185 56258 56274 56356 56643 56750
56772 56790 56883 56929 57086 57094 57393 57447 57480 57483 57528 57588 57712 57767 57783
57888 57889 57932 57951 58306 58358 58423 58474 58659 58726 58804 59055 59114 59204 59311
59470 59906 60572 60582 60685 60791 60856 60869 60944 60962 61013 61133 61451 61662 61779
61781 62013 62077 62251 62344 62453 62466 62691 62699 62800 62818 62821 63067 63181 63402
63525 63527 63665 63761 63865 64033 64095 64105 64122 64132 64270 64458 64627 64680 64769
64949 64973 65080 65110 65151 65313 65340 65455 65550 65648 65779 65793 65810 65920 66283
66357 66483 66865 67103 67266 67357 67406 67510 67656 67794 68106 68413 68480 68537 68574
68600 68825 68866 68875 68893 68986 69111 69149 69220 69579 69603 69642 69738 69771 70082
70151 70236 70350 70407 70480 70567 70608 70614 70620 70688 70733 70956 71051 71281 71377
71542 71666 71843 71880 71928 72026 72069 72243 72263 72366 72448 72767 72875 72986 72987
73055 73166 73206 73310 73443 73446 73829 73893 73934 74507 74585 74728 74780 74794 74868
75005 75028 75127 75230 75502 76193 76221 76378 76424 76497 76547 76557 76670 76687 76759
76804 77066 77067 77295 77373 77486 78011 78157 78467 78475 78507 78714 78748 78838 78923
79044 79069 79146 79185 79222 79286 79399 79654 80072 80073 80082 80107 80262 80341 80370
80384 80621 80729 81130 81176 81212 81220 81239 81795 82048 82061 82067 82164 82276 82449
82633 82658 82662 83162 83208 83302 83569 83595 83695 83765 83802 83809 84042 84051 84078
84083 84159 84376 84770 84771 84782 84785 84794 84836 84915 85049 85076 85207 85274 85407
85523 85541 85549 85552 85707 85811 85827 85884 85915 86042 86088 86111 86250 86385 86450
86463 86660 86727 86736 86874 86991 87155 87227 87551 87587 87826 87851 87852 88242 88328
88379 88386 88469 88477 88866 89042 89055 89109 89260 89454 89475 89621 89623 89651 89767
89849 90073 90109 90872 90928 91164 91209 91267 91443 91459 91596 91746 92050 92143 92175
92192 92241 92457 92495 92699 92718 92749 93054 93081 93172 93176 93245 93256 93328 93476
94061 94137 94184 94331 94379 94448 94584 94611 94627 94669 94670 94710 94890 95038 95127
95214 95326 95464 95466 95566 95567 95583 95730 95753 95777 95877 95890 95892 95893 96159
96199 96296 96377 96393 96493 96723 96729 97175 97274 97285 97422 97909 98177 98208 98234
98245 98301 98495 98590 98692 98742 98780 98824 99041 99114 99131 99152 99166 99399 99432
99580 99597 99646 99664 99734 99745 99818 99891 99973 0 …

Using HashSet Intersection:
Time: 15115880
6 7 62 116 166 269 310 329 406 436 484 484 640 724 950 1332 1418 1523 1526 1599 1643 1643
1658 1745 2044 2133 2173 2173 2377 2379 2459 2469 2640 2649 2725 3148 3168 3219 3274 3281
3338 3392 3409 3454 3713 3717 3927 4106 4194 4269 4269 4380 4613 4655 4764 5067 5118 5118
5174 5192 5222 5252 5416 5592 5780 5982 6061 6083 6319 6319 6385 6424 6477 6510 6566 6871
6896 7013 7038 7240 7240 7285 7305 7428 7463 7489 7808 7894 8104 8151 8215 8574 8586 8831
9479 9513 9586 10122 10270 10296 10401 10408 10629 10722 10904 10942 10981 10984 11072
11142 11142 11222 11365 11375 11382 11695 11800 11893 12035 12290 12300 12301 12361 12694
12908 13068 13111 13124 13132 13403 13447 13527 13637 13651 13670 13702 13784 13870 14107
14341 14406 14414 14465 14472 14504 14529 14683 14765 14781 14945 15005 15124 15131 15166
15192 15254 15275 15302 15395 15467 15515 15662 15690 15808 15878 16034 16146 16289 16329
16467 16492 16492 16533 16599 16707 16710 16738 16989 17257 17357 17361 17363 17421 17429
17475 17475 17527 17579 17598 17658 17870 17922 18141 18141 18332 18447 18534 18622 18668
18672 18687 18858 19097 19323 19490 19490 19580 19800 19839 19933 19958 19993 19993 20396
20472 20664 20720 20943 20971 20999 21122 21223 21274 21274 21332 21337 21450 21546 21566
21640 21640 21658 21705 21758 21790 21934 21955 22061 22147 22181 22407 22592 22624 22663
22672 22683 22845 22887 22911 22957 23280 23284 23457 23575 23575 23620 23873 23932 24029
24500 24503 24505 24505 24521 24720 24943 25012 25017 25229 25229 25379 25514 25623 25673
25842 25842 25903 26015 26015 26088 26243 26250 26253 26253 26254 26259 26368 26402 26667
26697 26745 26750 27012 27033 27043 27225 27324 27723 27726 27764 27933 27962 27968 28242
28512 28517 28808 29175 29514 29646 29868 29954 30003 30103 30103 30193 30193 30211 30257
30308 30322 30440 30481 30627 30732 30801 30820 30840 31087 31163 31565 31795 32087 32204
32220 32368 32429 32643 32947 32947 33047 33468 33473 33546 33569 33577 33721 33792 33839
33882 33945 34732 34972 35237 35295 35314 35351 35358 35608 35652 35717 35722 35961 35961
```

Ryan Gunawan
CS331-01
January 23, 2017

```
36307 36332 36366 36498 36534 36584 36809 37291 37628 37641 37651 37810 38008 38132 38175
38193 38332 38427 38483 38488 38514 38520 38642 38890 39068 39080 39227 39227 39254 39306
39338 39746 39826 40052 40184 40184 40225 40228 40250 40303 40328 40383 40383 40445 40529
40539 40796 40940 41151 41215 41393 41429 41525 41668 41673 41833 41958 42192 42209 42410
42468 42496 42718 42926 42931 42931 42943 43217 43280 43497 43614 43697 43841 44126 44363
44363 44370 44686 44705 44886 44922 44923 44936 44948 45047 45131 45163 45242 45306 45337
45441 45458 45476 45534 45638 45700 46000 46166 46240 46283 46338 46345 46427 46429 46515
46530 46532 46532 46639 46745 46789 46860 46908 46908 46975 47015 47039 47095 47160 47299
47359 47559 47710 47710 47785 47799 47929 47991 48010 48013 48087 48107 48141 48295 48378
48541 48610 48740 49014 49172 49296 49310 49310 49363 49384 49486 49544 49607 49607 49651
49778 49864 50048 50113 50254 50526 50665 50743 50792 50803 50804 50974 51020 51066 51095
51247 51288 51387 51539 51543 51672 51754 51797 51856 51898 52274 52403 52505 52515 52590
52630 52699 52715 52887 53130 53130 53334 53334 53336 53665 53709 53899 53918 54086 54397
54422 54667 54858 54864 55238 55398 55462 55496 55540 55555 55562 55623 55625 55722 55957
56185 56258 56274 56356 56643 56750 56772 56790 56883 56929 56929 57086 57094 57393 57447
57480 57483 57528 57588 57712 57767 57783 57888 57889 57932 57951 58306 58358 58423 58474
58659 58659 58726 58804 59055 59114 59204 59311 59470 59906 60572 60582 60685 60791 60856
60856 60869 60944 60962 61013 61133 61451 61451 61662 61779 61781 62013 62077 62077 62251
62344 62453 62466 62691 62699 62800 62818 62821 63067 63181 63402 63525 63527 63527 63665
63761 63865 64033 64095 64105 64122 64122 64132 64270 64458 64458 64627 64680 64769 64949
64973 65080 65110 65151 65313 65340 65455 65550 65648 65779 65793 65810 65920 66283 66357
66483 66483 66865 67103 67266 67357 67357 67406 67510 67656 67794 68106 68413 68480 68537
68537 68574 68600 68600 68825 68866 68875 68893 68986 69111 69149 69220 69579 69603 69642
69738 69771 70082 70151 70236 70350 70407 70480 70567 70608 70614 70620 70688 70733 70956
71051 71281 71377 71542 71666 71843 71880 71928 72026 72069 72069 72243 72263 72366 72448
72767 72875 72986 72987 73055 73166 73206 73310 73443 73446 73829 73893 73934 74507 74585
74728 74780 74794 74868 75005 75028 75127 75230 75502 76193 76221 76378 76424 76497 76547
76557 76670 76687 76759 76804 76804 77066 77067 77295 77373 77486 78011 78157 78467 78475
78507 78714 78748 78838 78923 79044 79069 79146 79185 79222 79286 79399 79654 79654 80072
80073 80082 80107 80262 80341 80370 80384 80621 80729 81130 81176 81212 81220 81220 81239
81795 82048 82061 82067 82164 82276 82449 82633 82658 82662 83162 83208 83302 83569 83595
83695 83765 83802 83809 84042 84051 84078 84083 84159 84376 84770 84771 84782 84785 84794
84836 84915 85049 85076 85207 85274 85407 85523 85541 85549 85552 85707 85811 85827 85884
85915 85915 86042 86088 86111 86250 86385 86450 86463 86660 86727 86736 86874 86991 87155
87227 87551 87551 87587 87826 87851 87852 88242 88328 88379 88386 88469 88477 88866 89042
89055 89109 89260 89454 89475 89621 89623 89651 89767 89849 90073 90109 90872 90928 91164
91209 91267 91443 91459 91596 91746 92050 92143 92175 92192 92241 92457 92495 92699 92718
92749 93054 93081 93081 93172 93176 93245 93256 93328 93476 94061 94137 94184 94331 94379
94448 94584 94584 94611 94627 94669 94670 94710 94890 95038 95127 95214 95326 95464 95464
95466 95566 95567 95583 95730 95753 95777 95877 95890 95892 95893 96159 96199 96296 96377
96393 96493 96723 96729 97175 97274 97285 97422 97909 97909 98177 98208 98234 98245 98301
98495 98590 98692 98742 98780 98824 99041 99114 99114 99131 99152 99166 99399 99432 99580
99597 99646 99664 99664 99734 99745 99818 99891 99973 0 …

Using TreeSet Intersection:
Time: 37114689
6 7 62 116 166 269 310 329 406 436 484 484 640 724 950 1332 1418 1523 1526 1599 1643 1643
1658 1745 2044 2133 2173 2173 2377 2379 2459 2469 2640 2649 2725 3148 3168 3219 3274 3281
3338 3392 3409 3454 3713 3717 3927 4106 4194 4269 4269 4380 4613 4655 4764 5067 5118 5118
5174 5192 5222 5252 5416 5592 5780 5982 6061 6083 6319 6319 6385 6424 6477 6510 6566 6871
6896 7013 7038 7240 7240 7285 7305 7428 7463 7489 7808 7894 8104 8151 8215 8574 8586 8831
9479 9513 9586 10122 10270 10296 10401 10408 10629 10722 10904 10942 10981 10984 11072
11142 11142 11222 11365 11375 11382 11695 11800 11893 12035 12290 12300 12301 12361 12694
12908 13068 13111 13124 13132 13403 13447 13527 13637 13651 13670 13702 13784 13870 14107
14341 14406 14414 14465 14472 14504 14529 14683 14765 14781 14945 15005 15124 15131 15166
15192 15254 15275 15302 15395 15467 15515 15662 15690 15808 15878 16034 16146 16289 16329
```

Ryan Gunawan
CS331-01
January 23, 2017

16467 16492 16492 16533 16599 16707 16710 16738 16989 17257 17357 17361 17363 17421 17429
17475 17475 17527 17579 17598 17658 17870 17922 18141 18141 18332 18447 18534 18622 18668
18672 18687 18858 19097 19323 19490 19490 19580 19800 19839 19933 19958 19993 19993 20396
20472 20664 20720 20943 20971 20999 21122 21223 21274 21274 21332 21337 21450 21546 21566
21640 21640 21658 21705 21758 21790 21934 21955 22061 22147 22181 22407 22592 22624 22663
22672 22683 22845 22887 22911 22957 23280 23284 23457 23575 23575 23620 23873 23932 24029
24500 24503 24505 24505 24521 24720 24943 25012 25017 25229 25229 25379 25514 25623 25673
25842 25842 25903 26015 26015 26088 26243 26250 26253 26253 26254 26259 26368 26402 26667
26697 26745 26750 27012 27033 27043 27225 27324 27723 27726 27764 27933 27962 27968 28242
28512 28517 28808 29175 29514 29646 29868 29954 30003 30103 30103 30193 30193 30211 30257
30308 30322 30440 30481 30627 30732 30801 30820 30840 31087 31163 31565 31795 32087 32204
32220 32368 32429 32643 32947 32947 33047 33468 33473 33546 33569 33577 33721 33792 33839
33882 33945 34732 34972 35237 35295 35314 35351 35358 35608 35652 35717 35722 35961 35961
36307 36332 36366 36498 36534 36584 36809 37291 37628 37641 37651 37810 38008 38132 38175
38193 38332 38427 38483 38488 38514 38520 38642 38890 39068 39080 39227 39227 39254 39306
39338 39746 39826 40052 40184 40184 40225 40228 40250 40303 40328 40383 40383 40445 40529
40539 40796 40940 41151 41215 41393 41429 41525 41668 41673 41833 41958 42192 42209 42410
42468 42496 42718 42926 42931 42931 42943 43217 43280 43497 43614 43697 43841 44126 44363
44363 44370 44686 44705 44886 44922 44923 44936 44948 45047 45131 45163 45242 45306 45337
45441 45458 45476 45534 45638 45700 46000 46166 46240 46283 46338 46345 46427 46429 46515
46530 46532 46532 46639 46745 46789 46860 46908 46908 46975 47015 47039 47095 47160 47299
47359 47559 47710 47710 47785 47799 47929 47991 48010 48013 48087 48107 48141 48295 48378
48541 48610 48740 49014 49172 49296 49310 49310 49363 49384 49486 49544 49607 49607 49651
49778 49864 50048 50113 50254 50526 50665 50743 50792 50803 50804 50974 51020 51066 51095
51247 51288 51387 51539 51543 51672 51754 51797 51856 51898 52274 52403 52505 52515 52590
52630 52699 52715 52887 53130 53130 53334 53334 53336 53665 53709 53899 53918 54086 54397
54422 54667 54858 54864 55238 55398 55462 55496 55540 55555 55562 55623 55625 55722 55957
56185 56258 56274 56356 56643 56750 56772 56790 56883 56929 56929 57086 57094 57393 57447
57480 57483 57528 57588 57712 57767 57783 57888 57889 57932 57951 58306 58358 58423 58474
58659 58659 58726 58804 59055 59114 59204 59311 59470 59906 60572 60582 60685 60791 60856
60856 60869 60944 60962 61013 61133 61451 61451 61662 61779 61781 62013 62077 62077 62251
62344 62453 62466 62691 62699 62800 62818 62821 63067 63181 63402 63525 63527 63527 63665
63761 63865 64033 64095 64105 64122 64122 64132 64270 64458 64458 64627 64680 64769 64949
64973 65080 65110 65151 65313 65340 65455 65550 65648 65779 65793 65810 65920 66283 66357
66483 66483 66865 67103 67266 67357 67357 67406 67510 67656 67794 68106 68413 68480 68537
68537 68574 68600 68600 68825 68866 68875 68893 68986 69111 69149 69220 69579 69603 69642
69738 69771 70082 70151 70236 70350 70407 70480 70567 70608 70614 70620 70688 70733 70956
71051 71281 71377 71542 71666 71843 71880 71928 72026 72069 72069 72243 72263 72366 72448
72767 72875 72986 72987 73055 73166 73206 73310 73443 73446 73829 73893 73934 74507 74585
74728 74780 74794 74868 75005 75028 75127 75230 75502 76193 76221 76378 76424 76497 76547
76557 76670 76687 76759 76804 76804 77066 77067 77295 77373 77486 78011 78157 78467 78475
78507 78714 78748 78838 78923 79044 79069 79146 79185 79222 79286 79399 79654 79654 80072
80073 80082 80107 80262 80341 80370 80384 80621 80729 81130 81176 81212 81220 81220 81239
81795 82048 82061 82067 82164 82276 82449 82633 82658 82662 83162 83208 83302 83569 83595
83695 83765 83802 83809 84042 84051 84078 84083 84159 84376 84770 84771 84782 84785 84794
84836 84915 85049 85076 85207 85274 85407 85523 85541 85549 85552 85707 85811 85827 85884
85915 85915 86042 86088 86111 86250 86385 86450 86463 86660 86727 86736 86874 86991 87155
87227 87551 87551 87587 87826 87851 87852 88242 88328 88379 88386 88469 88477 88866 89042
89055 89109 89260 89454 89475 89621 89623 89651 89767 89849 90073 90109 90872 90928 91164
91209 91267 91443 91459 91596 91746 92050 92143 92175 92192 92241 92457 92495 92699 92718
92749 93054 93081 93081 93172 93176 93245 93256 93328 93476 94061 94137 94184 94331 94379
94448 94584 94584 94611 94627 94669 94670 94710 94890 95038 95127 95214 95326 95464 95464
95466 95566 95567 95583 95730 95753 95777 95877 95890 95892 95893 96159 96199 96296 96377
96393 96493 96723 96729 97175 97274 97285 97422 97909 97909 98177 98208 98234 98245 98301
98495 98590 98692 98742 98780 98824 99041 99114 99114 99131 99152 99166 99399 99432 99580
99597 99646 99664 99664 99734 99745 99818 99891 99973 0 …