



CS111-Design and Analysis of Algorithms

Bicol University

Legazpi City, Albay

Programming Project

Members:

Renmar Balana

Raphael James Madrid

Joshua Navia

BSCS - II C

I. Structure of the Code

line (001 - 004)	header files
line (006 - 007)	constants definition
line (010 - 028)	function prototype declaration
line (031 - 226)	main function
(033 - 042)	variable declarations, initialize array to the heap
(046 - 051)	prompt user input for N
(052 - 054)	Generate random array, then print on screen
(057 - 062)	prompt user input for X
(063 - 065)	Generate sorted array, then print on screen
(071 - 080)	make 4 copies of the Random array
(083 - 084)	run and clock the algorithm for random input, then print the time: insertion sort;
(087 - 091)	merge sort;
(094 - 098)	quicksort ;
(101 - 106)	heap sort;
(111 - 120)	make 4 copies of the Sorted array
(123 - 124)	run and clock the algorithm for sorted input, then print the time: insertion sort;
(127 - 131)	merge sort;
(134 - 138)	quicksort ;
(141 - 145)	heap sort;
(150 - 172)	print to a text file named "Array_Values+Processor_Time", the time taken the algorithms, original random array, random array after sorting, and the array with already sorted values.
(174-217)	Test case to see if the respective arrays are actually sorted inside and Generate output file for insertion,merge,quick,and heap sort to double check if it is truly sorted. (Optional)
(220 - 223)	freeing of all the allocated memory used earlier.
(225 - 226)	end of main
line (230 - 483)	Function Definitions
(230 - 249)	Function to generate random array
(250 - 256)	Function to generate an array sorted in ascending order.
(258 - 277)	Function for insertion sort
(279 - 289)	Function for merge sort
(290 - 335)	Function to merge subarrays of mergesort
(337 - 349)	Function for Quicksort
(350 - 373)	Function to find partition position for Quicksort
(375 - 386)	Function for Heapsort
(387 - 403)	Function to heapify
(406 - 410)	function to swap the value of two integers
(412 - 437)	function to print an array to a screen
(438 - 463)	function to print an array to a text file
(465 - 472)	median function, finds median between 3 integers (Unused code)
(474 - 483)	delay function, delays clock by the entered parameter in milliseconds

II. Screenshots of actual execution

1. Compute the average (over five different runs) of the actual running time T(N) for five different N values, N=10 , 100 ,1000 ,10000,100000 .

The pictures below show the results of our last attempt running the program with five different N values.

Random N=10

Insertion Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 10
Generated array of random numbers:
| 223895 872466 465225 537261 52874 768776 755362 968470 767463 425987 |

PROGRAM RESULTS:
Time insertion Sort took for N =10, Random Input : 0.0000000000 seconds

Process exited after 4.491 seconds with return value 0
Press any key to continue . . .


```

Merge Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 10
Generated array of random numbers:
| 582251 251956 599366 761461 178078 659493 |

RAM RESULTS:
Time merge Sort took for N =10, Random Input : 0.0000000000 seconds

Process exited after 6.501 seconds with return value 0
Press any key to continue . . .


```

Quick Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms):
Generated array of random numbers:
| 711551 834798 869491 625981 347855 332959 |

PROGRAM RESULTS:
Time Quick Sort took for N =10, Random Input : 0.0000000000 seconds

Process exited after 5.061 seconds with return value 0
Press any key to continue . . .


```

Heap Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 10
array of random numbers:
| 807800 145545 204086 542594 481847 475991 475991 475991 475991 475991 |

Sort took for N =10, Random Input : 0.0000000000 seconds


```

Random N=100

Insertion Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 100
Generated array of random numbers:
| 766286 665470 689280 835874 689713 996520 746147 399392 286887 86753 |
| 363377 863719 145585 865182 249497 51265 718454 968147 814312 285797 |
| 559850 622886 626448 367983 347804 865396 968147 814312 159426 159426 |
| 159395 398744 580744 727249 580744 626448 769818 634654 159426 159426 |
| 386478 441103 299899 415607 992551 743392 508623 768784 497726 497726 |
| 486132 467713 467713 467713 467713 467713 508623 768784 497726 497726 |
| 822398 127754 436713 626878 647363 626248 596275 788347 608898 608898 |
| 87787 666620 148113 589877 339541 339556 466184 832795 382081 518461 |
| 566265 398893 398893 566168 566168 566168 566168 566168 566168 566168 |
| 581787 438032 175566 620473 175804 566168 566168 688752 886690 798268 |
| 289930 488598 18714 861182 819490 259728 427940 347187 711764 795987 |

PROGRAM RESULTS:
Time insertion Sort took for N =100, Random Input : 0.0000000000 seconds

Process exited after 5.681 seconds with return value 0
Press any key to continue . . .


```

Merge Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 100
Generated array of random numbers:
| 180212 463354 719897 939172 161017 154148 676735 433462 981890 |
| 559850 622886 626448 367983 347804 865396 968147 814312 159426 159426 |
| 159395 398744 580744 727249 580744 626448 769818 634654 159426 159426 |
| 597851 825294 291602 594726 225818 721492 486961 744997 453146 453146 |
| 232193 927785 35163 268031 648645 837757 839035 372725 874970 874970 |
| 88311 4141486 35116 980645 808645 918645 886645 886645 811544 811544 |
| 993638 970498 23378 487448 324173 467460 1524494 82046 682374 682374 |
| 943662 895235 693256 852907 230385 230385 169590 141933 783743 665765 |
| 186738 868494 471602 675305 229653 697583 745520 419932 306375 306375 |
| 736836 208475 432525 959641 749764 587015 955839 752681 244130 244130 |

TS:
merge Sort took for N =100, Random Input : 0.0000000000 seconds


```

Quick Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 100
Generated array of random numbers:
| 53789 814599 112142 812000 466636 345125 684266 197429 334821 |
| 497118 2763 296065 165530 977497 66669 822885 145764 992890 |
| 326626 157649 404395 332308 751665 463288 716621 452683 102497 |
| 935205 947745 337745 562160 76613 208136 650406 399256 802456 |
| 162532 436049 436049 436049 138344 731349 731349 138344 439968 439968 |
| 249773 211889 627086 817221 741975 516819 958501 509509 514031 |
| 119502 473402 775855 131713 349347 415536 985729 796558 531996 |
| 238213 238213 238213 238213 542937 608203 531996 531996 531996 531996 |
| 462676 231344 893168 198978 694837 433386 327658 715157 396914 396914 |
| 203343 246378 116487 174341 646765 940892 600859 198498 7277 7277 |

PROGRAM RESULTS:
Time Quick Sort took for N =100, Random Input : 0.0000000000 seconds


```

Heap Sort

```
Please enter the value of N (no. of integers as input to Sorting Algorithms): 100
Generated array of random numbers:
| 790115 466525 495748 524835 939316 724984 113791 252593 93855 |
| 958667 406125 270162 840953 840953 626605 664649 159426 159426 159426 |
| 643515 531159 531159 745266 531159 531159 531159 700346 352322 50941 |
| 346121 810748 531179 63282 716252 16269 165728 580537 86920 86920 |
| 797966 774357 885173 787573 77801 113898 350971 934821 63175 63175 |
| 684638 339996 597126 440748 863253 673576 37132 138898 83155 83155 |
| 652332 652332 652332 548159 548159 548159 548159 806352 459292 310292 |
| 347419 762388 134290 441593 916009 181113 309583 326463 24294 24294 |
| 821548 749582 156368 367021 798757 851970 236289 63348 61776 61776 |
| 276171 262078 94962 195078 78398 320676 844916 340764 94936 94936 |

TS:
Sort took for N =100, Random Input : 0.0000000000 seconds


```

Random N=1000

Insertion Sort

775354	68043	959903	63393	371135	308394	328721	304924	63141	846694
754404	698567	327166	415988	380832	175394	521598	285899	773623	397282
5662	765005	747965	374778	500455	237833	959796	188476	33338	327548
183490	754296	814728	182812	600042	101749	236172	54628	681854	889948
793294	754296	804749	593205	151793	151793	151793	151793	151793	151793
92869	777855	84114	945153	739211	57964	143876	135821	615621	85378
149889	385552	142718	808679	320281	540473	507645	657481	642770	383597
451835	951831	69244	281196	162159	218774	134259	339911	659194	448593
133549	231432	126303	695005	620261	880251	338383	623345	623345	623345
801581	566268	473885	386522	876844	741191	376599	412444	863137	
752475	752164	184300	228732	486131	629643	977954	921286	523444	183868
854346	456757	499775	886474	485951	665179	2258	6979	6979	6979
133997	623934	456675	738205	4422810	538620	847449	778389	835593	
567722	574751	996515	542321	315995	447263	62555	276786	567727	
35828	447882	192476	788620	542679	639698	363308	234197	182057	328276
524638	366565	789634	922992	162852	542974	916833	390454	109611	875772

PROGRAM RESULTS:
Time insertion Sort took for N =10000, Random Input : 0.1300000000 seconds

Merge Sort

582813	96831	520589	796240	296684	657429	618064	78013	460911
106252	542273	785651	663954	34883	639766	178724	252294	397921
65297	835664	866648	978496	367372	150039	321915	331760	100005
92687	886743	189352	32164	139565	297437	248669	756669	222379
918	136392	28517	69668	888289	745485	666555	987186	712996
9151	307659	655245	756872	678254	823202	101875	474738	550868
52322	698061	313715	699561	122215	441075	870383	922592	
581558	681111	654296	712341	724245	576914	226208	582679	606169
776432	881927	377893	537484	657193	137586	167549	44825	930187
75066	957637	35826	168298	746582	622487	951042	967624	794416
921133	949895	399835	65328	349834	280279	522128	731468	831685
747479	551808	455419	384047	423647	601514	653564	534223	631694

S: Merge Sort took for N =1000, Random Input : 0.0000000000 seconds

Quick Sort

48919	707365	600215	773954	626710	110251	498639	394253	7910	878203
19689	653366	880108	540778	530833	602061	455245	489664	540855	218649
74088	288552	134662	865448	545330	534814	607585	286983	65912	252902
107046	575496	99540	464733	214712	267711	499226	699388	663994	
44137	621409	870514	421874	530212	530212	337159	152769	152769	152769
742738	456657	185644	677466	900416	588108	467731	31728	341322	
50543	567343	916755	804306	39129	121780	366597	976649	763638	835883
960279	839066	761314	79106	385701	603320	989153	555206	860938	
133502	42238	407134	739518	632054	5279	51242	12526	935153	
319451	434671	597188	265863	737835	84388	950048	569159	897930	
640996	307373	17611	204880	589532	48562	691668	313095	33919	601243
582768	337885	37987	747999	462891	664685	141870	692689	53704	186997
878567	887677	32039	633151	206055	322777	337155	337075	337066	
725181	706003	956617	269429	309543	343224	144587	367912	324875	867388
723704	163696	268889	605482	38979	327276	749835	38734	145462	282126
103366	180369	180369	29529	59749	30749	29729	75253	180369	180369
583964	669783	262537	93829	21968	892648	982883	187947	#12370	847955
485001	712932	602358	738100	430716	268287	485000	217813	525258	702466

PROGRAM RESULTS:
Time Quick Sort took for N =1000, Random Input : 0.0000000000 seconds

Heap Sort

642819	924352	56133	310284	721884	136404	720537	744356	411642	936237
57428	835451	811394	338564	220808	520248	232795	639671	861910	787634
60845	924352	56133	310284	721884	136404	720537	744356	411642	936237
788705	737654	848952	774557	533664	533664	520728	418260	365085	11269
612944	35592	50868	39159	594518	424566	527715	737634	3499	353216
478530	728208	865183	125795	86994	478855	695655	480664	269892	956480
91111	787520	824249	31434	598108	463666	388181	941123	508256	465759
917398	394874	233085	645693	90750	376252	105515	891694	140904	154868
364783	859795	494837	703178	731238	941741	822258	943989	726608	
161340	121845	79795	104795	274718	782768	969881	859596	572779	75991
70329	266265	50893	545272	889269	467133	467133	32603	32603	32603
8970	587931	884549	524717	612118	276645	276645	257410	61231	
967921	574548	587201	248528	343535	953935	369672	686081	486529	11432
802056	777520	820801	580891	580891	13820	953935	296982	549885	549885
916524	774959	788663	989334	989334	526769	288873	418924	324572	178465
628885	808736	545404	607388	985268	27385	406133	751511	625851	751161

PROGRAM RESULTS:
Time merge Sort took for N =1000, Random Input : 0.0000000000 seconds

Quick Sort

249696	637634	212352	976255	199567	619400	557716	577454	767544	422558
826662	543623	957581	667067	865957	73586	338004	723746	595983	437529
41790	485356	815525	387578	668902	336789	948997	983665	420662	86869
44137	733069	138795	986178	986178	986178	987341	533961	733069	
865656	809453	206107	263196	395749	544871	306164	832465	544871	
439515	126633	946742	231997	105842	359252	86176	419814	398776	594258
514199	273698	542525	606827	598347	997744	416253	603798	349905	
133546	777527	946742	231997	105842	359252	86176	419814	398776	594258
985340	178047	286047	731787	921518	21761	219491	442266	32329	
730682	978628	358450	1237	576588	223257	928734	31282	485623	935631
318489	984895	1842	573949	296947	522579	215471	328246	295552	
139168	178047	286047	817159	630478	465678	465678	465678	465678	465678
579348	759324	215842	453448	87383	232913	167643	935462	205760	
537540	552738	819428	4804	330669	315349	212102	413511	838920	518789
133546	292055	8311	696109	212102	413511	838920			

Random N=100000

Insertion Sort

589221	87964	676555	488810	64214	782184	669974	47637	273536	678935
183438	81416	391188	934247	465957	398877	57925	748936	774973	232797
754879	312696	986837	757795	272794	252578	702962	252467	532868	356977
716669	244226	18578	970588	858543	174156	35277	294174	804889	309212
458860	780015	873923	621861	75159	562926	70776	277751	324413	832392
43395	562149	73451	70821	455159	941151	335139	700251	530151	83151
481337	62149	151132	29872	885541	674898	188718	687665	962651	230751
225682	47688	675247	62698	25111	615784	192324	826115	721438	170767
16818	74485	48982	521756	270526	715271	322777	12543	778726	824246
796818	199528	422861	275293	445843	72479	77776	517438	128374	832884
455049	486249	98185	368543	518579	438543	138579	32586	703159	10011
21136	313256	114163	51297	52872	552818	589562	386114	688555	114163
561483	295817	861198	891713	254519	986188	907537	141856	964783	956527
618467	319689	244770	968195	414883	551181	767288	485216	692149	267388
364892	193806	509121	791918	358283	88925	64769	728787	612698	19218
35733	25337	338253	38815	516871	871651	57598	310593	408950	286510

PROGRAM RESULTS:
Time insertion Sort took for N =100000, Random Input : 15.8540000000 seconds

Merge Sort

662892	159678	166011	777638	464952	137688	179657	487008	505739	789258
857759	414772	283178	710875	825681	158888	144765	459999	529435	488892
623655	85322	162688	616545	156730	769533	978466	491759	316651	376772
778908	671291	209486	406159	406159	83997	138333	537079	147159	147159
40851	644416	328497	818351	546819	7171	187314	697466	668581	171132
725654	936769	911498	722862	927048	248647	881660	433952	681779	924968
62677	557244	38882	915554	394483	528403	764982	196475	796208	866435
71844	312753	961325	277726	281888	385331	447247	169863	379299	786257
269814	240271	634079	730947	697222	159741	473293	379299	379299	379299
18820	973171	358030	485206	91288	59888	466572	152644	747179	747179
187366	996351	707591	208994	655849	635788	48766	596665	582419	266408
468905	153675	842675	145995	107724	386108	497686	389998	865558	865558
68686	121521	88035	620795	703759	777081	877081	673144	673144	673144
339108	888775	966872	482304	761937	830149	308072	886336	664065	706000
857883	807545	932072	297297	799478	517282	939328	432459	567714	250403
854601	604185	166493	426986	344263	722531	72581	835267	2094	961520
567997	448529	756991	888301	448451	672556	493595	205426	394135	795297

PROGRAM RESULTS:
Time merge Sort took for N =100000, Random Input : 0.0480000000 seconds

Quick Sort

309343	717049	784219	811910	657729	993893	598438	736795	983288	823436
525753	477871	473031	469205	98500	882066	929413	382532	125374	984181
45727	365778	111480	207797	157146	988353	773156	428979	458620	584113
455329	439216	636202	407261	303696	703159	703159	703159	703159	703159
165166	779795	542494	877914	559542	714552	845212	788837	778593	661813
717758	659298	787482	882297	360635	511483	542256	686151	484816	38667
293961	812162	71018	762764	109497	332591	782437	638869	751822	481448
408529	4562372	317876	864479	971141	674221	238996	980079	800079	800079
717758	664184	471156	114031	185377	239366	593196	602161	240769	834331
608358	694448	178937	215376	246598	481951	65882	247975	12313	53399
757776	44898	458947	249723	873532	204563	856985	89893	364494	78902
142109	459567	241771	75219	18039	13193	449161	572581	575861	324619
58897	659593	58787	899507	538669	786509	441794	14993	99466	711888
104543	337110	337114	337116	337116	337116	337116	337116	337116	337116
444996	876205	778265	331195	925800	888358	487828	969578	359017	129627
5026	762342	389227	435528	217738	106489	270487	121485	394759	268826
556909	948811	527478	95939	233699	506483	870013	35748	173985	939363
18456	847566	932188	272612	487960	907093	28844	315455	866617	1

PROGRAM RESULTS:
Time Quick Sort took for N =100000, Random Input : 0.0160000000 seconds

Heap Sort

162697	162930	162930	162930	162930	162930	162930	162930	162930	162930
881193	881253	963766	644894	561503	555818	609557	298941	687860	433113
147128	291868	67891	417850	17552	259684	735561	556761	433113	112655
226957	525885	756553	569176	145179	719028	101849	249843	392722	604887
763416	843395	381216	260810	815922	911579	73988	313464	788984	720758
897503	958172	209649	640119	541597	527549	35749	845374	133119	848681
976289	333171	275980	756211	177321	815865	106041	580799	106041	580799
595943	221965	556033	32127	298421	452469	741567	196463	652691	85769
419463	744984	276945	436855	586087	525181	71591	449170	88664	872819
85467	737862	615248	394863	448952	798839	367125	749784	30393	746926
15803	703116	5488	859108	859108	39898	38598	757589	251231	251231
807349	493957	757577	566415	873011	781969	457192	193387	526453	526453
581013	117614	876119	570624	831317	992563	478785	361456	996106	82899
668056	685141	651886	942687	243022	430282	665304	509526	216783	1
854258	154566	675179	982602	218867	33739	479865	253731	583568	852905
19960	279763	357513	43811	83811	98654	37886	43981	894941	1
734668	372965	645658	645658	645644	320000	336709	760664	493821	16673
55451	463754	645754	774448	925734	522292	68326	518713	326789	986669

PROGRAM RESULTS:
Time Heap Sort took for N =100000, Random Input : 0.0420000000 seconds

Quick Sort

value of N (no. of integers as input to Sorting Algorithms): 10
positive integer X(interval for sorted array): 1
array of already sorted numbers:

11 12 13 14 15 16

LTS: insertion Sort took for N =10, Sorted Input : 0.0000000000 seconds

ed after 6.432 seconds with return value 0 y to continue . . .

Quick Sort took for N =10, Sorted Input : 0.0000000000 seconds

ter 5.926 seconds with return value 0 continue . . .

Merge Sort

no. of integers as input to Sorting Algorithms): 10
er X(interval for sorted array): 2
eady sorted numbers:

14 16 18 20 22

for N

Sorted N=100

Insertion Sort

```
the value of N (no. of integers as input to Sorting Algorithms): 100
positive integer X(interval for sorted array): 5
ed array of already sorted numbers:
105 110 115 120 125 130 135 140 145 150
155 160 165 170 175 180 185 190 195
205 210 215 220 225 230 235 240 245
255 260 265 270 275 280 285 290 295
305 310 315 320 325 330 335 340 345
355 360 365 370 375 380 385 390 395
405 410 415 420 425 430 435 440 445
455 460 465 470 475 480 485 490 495
505 510 515 520 525 530 535 540 545
555 560 565 570 575 580 585 590 595
S:
insertion Sort took for N =100, Sorted Input : 0.0000000000 seconds
```

Merge Sort

```
the value of N (no. of integers as input to Sorting Algorithms): 100
er a positive integer X(interval for sorted array): 3
enerated array of already sorted numbers:
103 106 109 112 115 118 121 124
131 136 139 142 145 148 151 154
163 166 169 175 178 181 184
193 196 199 202 205 208 211 214
223 226 229 232 235 238 241 244
253 256 259 262 265 268 271 274
283 286 289 292 295 298 301 304
313 316 319 322 325 328 331 334
343 346 349 352 355 358 361 364
373 376 379 382 385 388 391 394
GULTS:
me merge Sort took for N =100, Sorted Input : 0.0000000000 seconds
```

Quick Sort

```
the value of N (no. of integers as input to Sorting Algorithms): 100
positive integer X(interval for sorted array): 7
ed array of already sorted numbers:
107 114 121 128 135 142 149 156 163
177 184 191 198 205 212 219 226 233
247 254 261 268 275 282 289 295 306
317 324 331 338 345 352 359 366 377
387 394 401 408 415 422 429 436 444
457 464 471 478 485 492 499 506 513
527 534 541 548 555 562 569 576 583
597 604 611 618 625 632 639 646 653
667 674 681 688 695 702 709 716 723
737 744 751 758 765 772 779 786 793
S:
quick Sort took for N =100, Sorted Input : 0.0000000000 seconds
```

Heap Sort

```
value of N (no. of integers as input to Sorting Algorithms): 100
positive integer X(interval for sorted array): 34
d array of already sorted numbers:
134 168 202 236 270 304 338 372 406
474 508 542 576 610 644 678 712 746
814 848 882 916 950 984 1018 1052 1086
114 118 122 126 130 134 138 142 146
1494 1528 1562 1596 1630 1664 1698 1732 1766
1834 1868 1902 1936 1970 2004 2038 2072 2106
2174 2208 2242 2276 2310 2344 2378 2412 2446
2514 2548 2582 2616 2650 2684 2718 2752 2786
2854 2888 2922 2956 2990 3024 3058 3092 3126
3194 3228 3262 3296 3330 3364 3398 3432 3464
S:
p Sort took for N =100, Sorted Input : 0.0000000000 seconds
```

Sorted N=1000

Insertion Sort

```
7568 7576 7584 7592 7600 7608 7616 7624 7632 7640
7648 7656 7664 7672 7680 7688 7696 7704 7712 7720
7728 7736 7744 7752 7760 7768 7776 7784 7792 7800
7888 7816 7824 7832 7840 7848 7856 7864 7872 7880
7968 7976 7984 7992 8000 8008 8016 8024 8032 8040
8088 8096 8104 8112 8120 8128 8136 8144 8152 8160
8128 8136 8144 8152 8160 8168 8176 8184 8192 8200
8288 8216 8224 8232 8240 8248 8256 8264 8272 8280
8288 8304 8312 8320 8328 8336 8344 8352 8360 8368
8368 8376 8384 8392 8400 8408 8416 8424 8432 8440
8448 8504 8561 8569 8576 8584 8592 8600 8608
8518 8536 8544 8552 8560 8568 8576 8584 8592 8600
8688 8616 8624 8632 8640 8648 8656 8664 8672 8680
8688 8696 8704 8712 8720 8728 8736 8744 8752 8760
8768 8776 8784 8792 8800 8808 8816 8824 8832 8840
8848 8856 8864 8872 8880 8888 8896 8904 8912 8920
8928 8936 8944 8952 8960 8968 8976 8984 8992 9000
RAM RESULTS:
Time insertion Sort took for N =1000, Sorted Input : 0.0000000000 seconds
```

Merge Sort

```
4935 4960 4985 5010 5015 5020 5025 5030 5035 5040 5045 5050 5055 5060 5065 5070 5075 5080 5085 5090 5095 5100
5095 5100 5105 5110 5115 5120 5125 5130 5135 5140 5145 5150 5155 5160 5165 5170 5175 5180 5185 5190 5195 5200
5205 5210 5215 5220 5225 5230 5235 5240 5245 5250 5255 5260 5265 5270 5275 5280 5285 5290 5295 5300 5305 5310 5315 5320 5325 5330 5335 5340 5345 5350 5355 5360 5365 5370 5375 5380 5385 5390 5395 5400
5405 5410 5415 5420 5425 5430 5435 5440 5445 5450 5455 5460 5465 5470 5475 5480 5485 5490 5495 5500
5495 5500 5505 5510 5515 5520 5525 5530 5535 5540 5545 5550 5555 5560 5565 5570 5575 5580 5585 5590 5595 5600
5595 5600 5605 5610 5615 5620 5625 5630 5635 5640 5645 5650 5655 5660 5665 5670 5675 5680 5685 5690 5695 5700
5705 5710 5715 5720 5725 5730 5735 5740 5745 5750 5755 5760 5765 5770 5775 5780 5785 5790 5795 5800
5805 5810 5815 5820 5825 5830 5835 5840 5845 5850 5855 5860 5865 5870 5875 5880 5885 5890 5895 5900
5905 5910 5915 5920 5925 5930 5935 5940 5945 5950 5955 5960 5965 5970 5975 5980 5985 5990 5995 6000
PROGRAM RESULTS:
Time merge Sort took for N =1000, Sorted Input : 0.0010000000 seconds
```

Quick Sort

```
27634 27720 27701 27710 27720 27730 27740 27750 27760 27770
28234 28268 28282 28296 28310 28324 28338 28352 28366 28380
25790 28648 28662 28676 28690 28704 28718 28732 28746 28760
28914 28948 28982 29006 29030 29054 29078 29092 29116 29130
29254 29288 29322 29356 29390 29424 29458 29492 29526 29560
29594 29628 29662 29696 29730 29764 29798 29832 29866 29900
29968 29988 30012 30036 30060 30084 30108 30132 30156 30180
30274 30342 30366 30410 30444 30478 30512 30546 30580
30614 30648 30682 30716 30750 30784 30818 30852 30886 30920
30954 30988 31022 31056 31090 31124 31158 31192 31226 31260
31294 31318 31363 31396 31430 31464 31498 31532 31566 31600
31974 32008 32042 32076 32110 32144 32178 32212 32246 32280
32314 32348 32382 32416 32450 32484 32518 32552 32586 32620
32654 32688 32722 32756 32790 32824 32858 32892 32926 32960
33090 33124 33158 33192 33226 33260 33294 33328 33362 33396
33334 33368 33402 33436 33470 33504 33538 33572 33606 33640
33674 33708 33742 33776 33810 33844 33878 33912 33946 33980
34014 34048 34082 34116 34150 34184 34218 34252 34286 34320
34156 34188 34242 34456 34496 34530 34562 34608 34640
34694 34728 34762 34796 34830 34864 34898 34932 34966 35000
PROGRAM RESULTS:
Time quick Sort took for N =1000, Sorted Input : 0.0120000000 seconds
```

Heap Sort

```
274686 275352 275718 275744 276070 276416 276762 277108 277454 277800
278146 278492 278838 279184 279538 279876 280222 280568 280914 281268
281696 281952 282298 282644 282996 283336 283682 284028 284374 284718
284856 285202 285548 285894 286240 286586 286932 287278 287624 288060
288516 288872 289218 289564 289919 290256 290602 291048 291494 291940
291986 292332 292678 293024 293370 293716 294062 294408 294754 295100
295446 295792 296138 296484 296838 297176 297522 297868 298214 298568
298906 299252 299598 299944 300290 300636 300982 301328 301674 302020
302364 302712 303058 303404 303750 304141 304488 304834 305180 305526
305816 306172 306518 306864 307210 307556 307902 308248 308594 308940
309286 309632 309978 310324 310676 311015 311362 311788 312054
312746 313091 313438 313784 314138 314474 314822 315160 315514 315868
316206 316552 316898 317244 317590 317936 318282 318638 319084 319438
319666 320012 320358 320704 321050 321396 321742 322088 322434 322788
323116 323472 323818 324164 324518 324855 325202 325548 325894 326248
326586 326932 327278 327624 327970 328316 328662 329098 329354 329700
329944 330290 330636 331082 331428 331774 332120 332466 332812 333158
333506 333852 334198 334544 334898 335256 335582 335928 336274 336628
336966 337312 337568 338004 338350 338695 339042 339384 340000
340416 340772 341118 341464 341810 342156 342502 342848 343194 343548
343896 344232 344678 345024 345370 345716 346062 346408 346754 347100
PROGRAM RESULTS:
Time heap Sort took for N =1000, Sorted Input : 0.0000000000 seconds
```

Sorted N=10000

Insertion Sort

49104	49108	49112	49116	49120	49125	49128	49132	49136	49140
49284	49288	49322	49216	49220	49224	49228	49232	49236	49240
49324	49328	49332	49336	49340	49344	49348	49352	49356	49360
49324	49328	49332	49296	49300	49304	49308	49312	49316	49320
49324	49328	49332	49336	49340	49344	49348	49352	49356	49360
49364	49368	49372	49376	49380	49384	49388	49392	49396	49400
49404	49408	49412	49416	49420	49424	49428	49432	49436	49440
49444	49448	49452	49456	49460	49464	49468	49472	49476	49480
49484	49488	49492	49496	49500	49504	49508	49512	49516	49520
49524	49528	49532	49536	49540	49544	49548	49552	49556	49560
49564	49568	49572	49576	49580	49584	49588	49592	49596	49600
49604	49608	49612	49616	49620	49624	49628	49632	49636	49640
49644	49648	49652	49656	49660	49664	49668	49672	49676	49680
49684	49688	49692	49696	49700	49704	49708	49712	49716	49720
49724	49728	49732	49736	49740	49744	49748	49752	49756	49760
49764	49768	49772	49776	49780	49784	49788	49792	49796	49800
49804	49808	49812	49816	49820	49824	49828	49832	49836	49840
49844	49848	49852	49856	49860	49864	49868	49872	49876	49880
49884	49888	49892	49896	49900	49904	49908	49912	49916	49920
49924	49928	49932	49936	49940	49944	49948	49952	49956	49960
49964	49968	49972	49976	49980	49984	49988	49992	49996	50000

PROGRAM RESULTS:
Time insertion Sort took for N =10000, Sorted Input : 0.0000000000 seconds

Merge Sort

77425	77471	77474	77475	77476	77477	77478	77479	77480	77481	77482	77483	77484	77485	77486	77487	77488	77489	77490	77491	77492	77493	77494	77495	77496	77497	77498	77499	77500	77501	77502	77503	77504	77505	77506	77507	77508	77509	77510	77511	77512	77513	77514	77515	77516	77517	77518																																																																																																																																																						
77448	77455	77463	77471	77479	77486	77494	77502	77510	77518	77526	77534	77542	77550	77558	77566	77574	77582	77590	77598	77606	77614	77622	77630	77638	77646	77654	77662	77670	77678	77686	77694	77702	77710	77718	77726	77734	77742	77750	77758	77766	77774	77782	77790	77798	77806	77814	77822	77830	77838	77846	77854	77862	77870	77878	77886	77894	77902	77910	77918	77926	77934	77942	77950	77958	77966	77974	77982	77990	77998	78006	78014	78022	78030	78038	78046	78054	78062	78070	78078	78086	78094	78102	78110	78118	78126	78134	78142	78150	78158	78166	78174	78182	78190	78198	78206	78214	78222	78230	78238	78246	78254	78262	78270	78278	78286	78294	78302	78310	78318	78326	78334	78342	78350	78358	78366	78374	78382	78390	78398	78406	78414	78422	78430	78438	78446	78454	78462	78470	78478	78486	78494	78502	78510	78518	78526	78534	78542	78550	78558	78566	78574	78582	78590	78598	78606	78614	78622	78630	78638	78646	78654	78662	78670	78678	78686	78694	78702	78710	78718	78726	78734	78742	78750	78758	78766	78774	78782	78790	78798	78806	78814	78822	78830	78838	78846	78854	78862	78870	78878	78886	78894	78898	78906	78914	78922	78930	78938	78946	78954	78962	78970	78978	78986	78994	78998	79006
77603	77611	77619	77627	77635	77643	77651	77659	77667	77675	77683	77691	77699	77707	77715	77723	77731	77739	77747	77755	77763	77771	77779	77787	77795	777	7783	77841	77849	77857	77865	77873	77881	77889	77897	77905	77913	77921	77929	77937	77945	77953	77961	77969	77977	77985	77993	779	7801	78019	78027	78035	78043	78051	78059	78067	78075	78083	78091	78099	78107	78115	78123	78131	78139	78147	78155	78163	78171	78179	78187	78195	78203	78211	78219	78227	78235	78243	78251	78259	78267	78274	78282	78290	78298	78306	78314	78322	78330	78338	78346	78354	78362	78370	78378	78386	78394	78402	78410	78418	78426	78434	78442	78450	78458	78466	78474	78482	78490	78498	78506	78514	78522	78530	78538	78546	78554	78562	78570	78578	78586	78594	78602	78610	78618	78626	78634	78642	78650	78658	78666	78674	78682	78690	78698	78706	78714	78722	78730	78738	78746	78754	78762	78770	78778	78786	78794	78802	78810	78818	78826	78834	78842	78850	78858	78866	78874	78882	78890	78898	78906	78914	78922	78930	78938	78946	78954	78962	78970	78978	78986	78994	78998	79006																							
77818	77896	77914	77932	77950	77968	77986	77994	78012	78030	78048	78066	78084	78102	78120	78138	78156	78174	78192	78210	78228	78246	78264	78282	78300	78318	78336	78354	78372	78390	78408	78426	78444	78462	78480	78498	78516	78534	78552	78570	78588	78606	78624	78642	78660	78678	78696	78714	78732	78750	78768	78786	78804	78822	78840	78858	78876	78894	78912	78930	78948	78966	78984	78992	79000																																																																																																																																				
77915	77932	77950	77968	77986	77994	78012	78030	78048	78066	78084	78102	78120	78138	78156	78174	78192	78210	78228	78246	78264	78282	78300	78318	78336	78354	78372	78390	78408	78426	78444	78462	78480	78498	78516	78534	78552	78570	78588	78606	78624	78642	78660	78678	78696	78714	78732	78750	78768	78786	78804	78822	78840	78858	78876	78894	78912	78930	78948	78966	78984	78992	79000																																																																																																																																						
78018	78036	78054	78072	78090	78108	78126	78144	78162	78180	78198	78216	78234	78252	78270	78288	78306	78324	78342	78360	78378	78396	78414	78432	78450	78468	78486	78504	78522	78540	78558	78576	78594	78612	78630	78648	78666	78684	78702	78720	78738	78756	78774	78792	78810	78828	78846	78864	78882	78890	78898	78906	78914	78922	78930	78938	78946	78954	78962	78970	78978	78986	78994	78998	79006																																																																																																																																				
78125	78143	78161	78179	78197	78215	78233	78251	78269	78287	78305	78323	78341	78359	78377	78395	78413	78431	78449	78467	78485	78503	78521	78539	78557	78575	78593	78611	78629	78647	78665	78683	78701	78719	78737	78755	78773	78791	78809	78827	78845	78863	78881	78899	78907	78915	78923	78931	78939	78947	78955	78963	78971	78979	78987	78995	78998	79006																																																																																																																																											
78225	78243	78261	78279	78297	78315	78333	78351	78369	78387	78405	78423	78441	78459	78477	78495	78513	78531	78549	78567	78585	78603	78621	78639	78657	78675	78693	78711	78729	78747	78765	78783	78801	78819	78837	78855	78873	78891	78909	78927	78945	78963	78981	78999	79007	79015	79023	79031	79039	79047	79055	79063	79071	79079	79087	79095	79103	79111	79119	79127	79135	79143	79151	79159	79167	79175	79183	79191	79199	79207	79215	79223	79231	79239	79247	79255	79263	79271	79279	79287	79295	79303	79311	79319	79327	79335	79343	79351	79359	79367	79375	79383	79391	79399	79407	79415	79423	79431	79439	79447	79455	79463	79471	79479	79487	79495	79503	79511	79519	79527	79535	79543	79551	79559	79567	79575	79583	79591	79599	79607	79615	79623	79631	79639	79647	79655	79663	79671	79679	79687	79695	79703	79711	79719	79727	79735	79743	79751	79759	79767	79775	79783	79791	79799	79807	79815	79823	79831	79839	79847	79855	79863	79871	79879	79887	79895	79903	79911	79919	79927	79935	79943	79951	79959	79967	79975	79983	79991	79999	79998																											
79018	79036	79054	79072	79090	79108	79126	79144	79162	79180	79198	79216	79234	79252	79270	79288	79306	79324	79342	79360	79378	79396	79414	79432	79450	79468	79486	79504	79522	79540	79558	79576	79594	79612	79630	79648	79666	79684	79702	79720	79738	79756	79774	79792	79810	79828	79846	79864	79882	79890	79898	79906	79914	79922	79930	79938	79946	79954	79962	79970	79978	79986	79994	79998	79999																																																																																																																																				

PROGRAM RESULTS:
Time merge Sort took for N =10000, Sorted Input : 0.0040000000 seconds

Quick Sort

PROGRAM RESULTS:
Time quick Sort took for N =10000, Sorted Input : 0.5310000000 second

Heap Sort

801399	881458	881577	881608	881755	881804	881933	882022	882111	882201
882289	882378	882467	882556	882645	882734	882823	882912	883001	883090
883179	883268	883357	883446	883535	883624	883713	883802	883891	883980
884069	884158	884247	884336	884425	884514	884603	884692	884781	884870
884959	885048	885137	885226	885315	885404	885493	885582	885671	885760
885849	885938	886027	886116	886205	886294	886383	886472	886561	886650
886739	886828	886917	887006	887095	887184	887273	887362	887451	887540
887629	887718	887807	887896	887985	888074	888163	888252	888341	888430
888519	888608	888697	888786	888875	888964	889053	889142	889231	889320
889499	889588	889677	889766	889854	889943	890032	890121	890210	890309
890289	890378	890467	890556	890645	890734	890823	890912	891001	891090
891179	891268	891357	891446	891534	891623	891712	891801	891890	891980
892079	892168	892257	892346	892435	892524	892613	892702	892791	892880
892969	893058	893147	893236	893325	893414	893503	893592	893681	893770
893859	893948	894037	894126	894215	894304	894393	894482	894571	894660
894749	894838	894927	895016	895105	895194	895283	895372	895461	895550
895639	895728	895817	895906	895995	896084	896173	896262	896351	896440
896519	896608	896697	896786	896875	896964	897053	897142	897231	897320
897419	897508	897597	897686	897775	897864	897953	898042	898131	898220
898399	898388	898478	898567	898656	898745	898834	898923	899012	899101
899199	899288	899377	899466	899555	899644	899733	899822	899911	900000

PROGRAM RESULTS:
Time heap Sort took for N =10000, Sorted Input : 0.0030000000 seconds

Insertion Sort

Merge Sort

4690846	4690892	4690938	4690984	4691038	4691076	4691122	4691168	4691214	4691260
4691306	4691352	4691398	4691444	4691490	4691536	4691582	4691628	4691674	4691720
4691767	4691812	4691858	4691904	4691950	4691995	4692042	4692088	4692134	4692180
4692227	4692272	4692318	4692364	4692410	4692456	4692502	4692548	4692594	4692640
4692687	4692733	4692779	4692825	4692871	4692917	4692963	4693009	4693054	4693100
4693146	4693192	4693238	4693284	4693330	4693376	4693422	4693468	4693514	4693560
4693606	4693652	4693698	4693744	4693790	4693836	4693882	4693928	4693974	4694020
4694665	4694712	4694758	4694804	4694850	4694896	4694942	4694988	4695443	4694480
4694526	4694572	4694618	4694664	4694710	4694756	4694802	4694848	4694894	4694940
4694936	4695082	4695097	4695143	4695178	4695215	4695262	4695308	4695354	4695400
4695506	4695552	4695598	4695644	4695690	4695736	4695782	4695828	4695874	4695920
4696366	4696412	4696458	4696504	4696550	4696596	4696642	4696688	4696734	4696780
4696826	4696872	4696918	4696964	4697020	4697065	4697120	4697174	4697194	4697240
4697278	4697323	4697379	4697424	4697478	4697515	4697562	4697608	4697654	4697700
4697747	4697793	4697839	4697885	4697939	4697997	4698025	4698068	4698114	4698160
4698244	4698290	4698336	4698382	4698438	4698484	4698530	4698576	4698622	4698668
4698644	4698712	4698758	4698804	4698850	4698896	4698942	4698988	4699034	4699080
4699126	4699172	4699218	4699264	4699318	4699356	4699402	4699448	4699494	4699540
4699586	4699632	4699678	4699724	4699770	4699816	4699862	4699908	4699954	4700000

PROGRAM RESULTS:
Time insertion Sort took for N =1000000. Sorted Input : A 0010000000 seconds

23553236	23553479	23553705	23553940	23554175	23554410	23554644	23554880	23555115	23555359
23555587	23555820	23556095	23556290	23556473	23556695	23556996	23557230	23557523	23557746
23557935	23558170	23558445	23558610	23558875	23559110	23559345	23559580	23559815	23560050
23560285	23560528	23560775	23560990	23561225	23561460	23561693	23561930	23562165	23562408
23562035	23562278	23562503	23562738	23563043	23563268	23563503	23563740	23564045	23564276
23563375	23563610	23563845	23564070	23564305	23564540	23564775	23565010	23565245	23565476
23567335	23567570	23567805	23568040	23568275	23568510	23568745	23568980	23569215	23569450
23569885	23570095	23570310	23570590	23570865	23571095	23571330	23571565	23571800	
23572839	23572727	23572740	23572745	23572747	23572748	23572749	23572750	23572751	23572752
23574385	23574620	23574855	23575096	23575325	23575560	23575795	23576030	23576265	23576508
23576739	23576979	23577205	23577440	23577675	23577910	23578145	23578380	23578615	23578850
23578185	23578420	23578655	23578890	23579125	23579360	23579595	23579830	23580065	23580300
23581435	23581670	23581905	23582140	23582375	23582610	23582845	23583080	23583315	23583550
23583785	23584020	23584255	23584490	23584725	23584960	23585195	23585430	23585665	23585900
23586135	23586370	23586605	23586840	23587075	23587310	23587545	23587780	23588015	23588250
23588485	23588720	23588955	23589190	23589425	23589660	23589895	23590130	23590365	23590600
23590185	23590420	23590655	23590890	23591125	23591360	23591595	23591830	23592065	23592300
23591385	23591620	23591855	23592090	23592325	23592560	23592795	23593030	23593265	23593500
23595535	23595770	23596005	23596240	23596475	23596710	23596945	23597180	23597415	23597650
23597885	23598120	23598355	23598590	23598825	23599060	23599295	23599530	23599765	23600000

PROGRAM RESULTS:
Time merge Sort took for N =100000, Sorted Input : 0.0450000000 seconds

Quick Sort

PROGRAM RESULTS: The following table summarizes the results of the program.

Heap Sort

PROGRAM RESULTS:
Time heap Sort took for N =1000000, Sorted Input : 0.0410000000 seconds

III. Analysis of the outputs

3. Present the results in Table format, where rows of the tables will indicate different N values and columns will indicate the T(N) for the different sorting algorithm.

Average Running Time for an Input Array that is Random

N	Insertionsort	Mergesort	Quicksort	Heapsort
10	0	0	0	0
100	0	0	0	0
1000	0	0	0	0
10000	0.1472	0.0082	0	0.0032
100000	13.5048	0.0532	0.0192	0.0412

Average Running Time for an Input Array that is Sorted

N	Insertionsort	Mergesort	QuickSort	Heapsort
10	0	0	0	0
100	0	0	0	0
1000	0	0.0024	0.0112	0.0002
10000	0	0.0042	0.5360	0.0030
100000	0.0008	0.044	61.7186	0.0346

for n = 10 and 100, the time is too fast to be clocked, so data is only zero and inconclusive.

- Insertion sort progressively gets slower as the input size gets larger for random input. It's also the slowest algorithm for a random array. In the worst case, this algorithm would perform $O(n^2)$ time. For an input array that is already sorted, insertion sort is always fast no matter the size of N. This is the best case for insertion sort as it takes only $O(n)$ time.
- Time for Merge sort is consistently fast, and grows longer as N gets larger. And it doesn't matter if the array is random or already sorted. This is because the time complexity of merge sort for all cases, is always $O(n * \log n)$. Its downside is that it takes more memory space, because we need to also declare temporary arrays that will store the divided subarrays.
- Quicksort is the fastest performing algorithm for a randomized array, so we can say its name is well deserved. Like other divide-and-conquer algorithms, its average time is $O(n * \log n)$. However, the case is the complete opposite if it analyzes an array that is already sorted. Quicksort is notably the slowest across all values of N, if the input array is already sorted. This is because the chosen pivot for quicksort is always the last element, which will always be the largest because the array is sorted in ascending order. Therefore, partitioning will take $O(n)$ time (no swapping will be done anyways), for all the n-number of elements. So for worst case, time will be $O(n^2)$
- Heapsort, although not a divide-and-conquer algo, is similar to how merge sort behaves, where it doesn't matter very much if the input array is random or already sorted.

Conclusion:

As expected, the time it takes for all the sorting algorithms to process the data also increases directly proportional as N grows larger.

Quick sort is the best sorting algorithm to use if the array to be sorted is randomly spread. However, we need to be careful in using it, because feeding an already sorted array into quicksort will take a lot of time to process.

Inversely, Insertion sort works best in an array that is already sorted, but struggles in practical use, because we usually arrange unsorted data in real world scenarios.

When all of the facts and experiments are taken into consideration, it becomes clear that each sorting algorithm has its own set of strengths and weaknesses when dealing with the values of arrays. This is strongly supported by pictures of our executions per input of N and the presentation of the results in table format, which enable the user to easily see the differences between each sorting algorithm and conclude what is efficient and less efficient in producing an output,taking in consideration what the end user is asking,for example randomized or Sorted output of arrays.

IV. Challenges encountered

- Figuring out a function to print the generated array into a neat and organized table, so we can confirm the functions that generate the random and sorted array are behaving as expected.

We chose 10 columns, as the base of the grid, the 11th element would be printed on the next row and so on...

- Naïve implementation of array, ex: int array[SIZE]; can only store a limited number of elements. We later found out this is because this kind of array is stored on the computer's stack memory, which is quite limited and not designed to handle very large amounts of data.

A better implementation would be to use malloc to allocate an adequate amount of memory, to exactly fit the size of the array we will need. By using malloc, the array is stored in the heap; we were even able to reach 1 million integers as input.

- Quicksort gave us a problem, because testing an already sorted array through a quicksort would result in the worst possible case, that is $T(n) = n^2$. This is because our initial trials in partitioning used the last element as its pivot. Since the last element will always be the largest for an already sorted array, this makes the recursion depth for large amounts of input to crash the program by stack overflow. We were only able to reach 25,000 as the value of N, before the program would crash.

1st attempt to fix: pivot was chosen randomly. reached 55,000 as the value of N. Still not enough.

2nd attempt to fix: pivot chosen was the median between first, second and last elements. Only 65,000 was feasible.

3rd attempt to fix: Quicksort function was changed into a half-recursive algorithm. Instead of recursing both sub partitions, only the smaller side used recursion, while the larger side was iterated. This way, the recursion depth won't reach dangerously high levels that would crash the program. Instead of having one very deep recursion, we end up with multiple "shallow" recursions.

After having solved the stack overflow problem, we later changed back the program to pick the last element as the pivot, as we believe this would best simulate the worst case for quicksort given an array with already sorted elements.

- `rand()` function can only generate integers from 0 until 32767, which is equivalent to the binary value 1111111111111111: 15 bits that are all flipped. Because the MAXRANGE to be used is 1 million, with binary value 11110100001001000000, we need a minimum of 20 bits to be represented.

Decimal Value of X	Binary Value	Operation
32,767	1111111111111111	Shift to the left: $x \ll= 15$
1,073,709,056	11111111111110000000000000000000	XOR x with another random number (10,922) $x \wedge= \text{rand}()$ X 11111111111110000000000000000000 rand 010101010101010 Result: 1111111111111010101010101010
1,073,719,978	11111111111110101010101010	Modulo by 1M+1 $x \%= \text{MAXRANGE} + 1$ $\begin{array}{r} 1,073,719,978 \\ \% \quad \underline{1,000,001} \\ 718,905 \end{array}$
718,905	1010111100000111001	

However, this solution is not perfect. Because of the process we used, shifting the random number then performing XOR with another random number can yield 1,073,741,823 at most (XOR of two 32,767).

11111111111111111111111111111111 (30 1's will yield:) 1,073,741,823 as maximum possible value generated, before modulo operation.

Because the actual sample size of 1 billion+ is not perfectly divisible by 1 million and 1,

range (0 – 740,749) appears 1074 times, when 1,073,741,823 % MAXRANGE+1. (740,750)

range (740,750 – 1,000,000) appears 1073 times only. (259,251)

$$740,750 * (1074) + 259,251 * (1073) = 1,073,741,823$$

$$\begin{array}{rcl} & 1,073,741,823 & 1,073,741,823 \\ \% & \underline{1,000,001} & / \quad \underline{1,000,001} \\ = & 740,750 & = \quad 1073.74074926 \end{array}$$

Probability = (number of possible outcomes)/(total number of outcomes)

ideal probability: (1/1,000,001) = 0.000000999999

probability (0-740,749): $1074/1,073,741,823 = 0.00000100024$

probability (740,750 – 1,000,000): $1073/1,073,741,823 = 0.00000099930$

Sample set is very, very, very slightly more biased in the first 74% integers of 1 million. But this bias is so small that it is practically negligible.