IPS 5

ITA5002

Problem Solving with Data Structures and Algorithms

Submitted by: -

Moeenul Islam

Submitted to: -

Dr Vijayan E

School of Computer Science and Engineering

Vellore Institute of Technology, Vellore

06 January, 2022

Insertion Sort

First, we have executed inserted sort with an array of 1000 size , by using the rand function we have imputed the data in the array..Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00104689 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=1000;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0 && a[j]>key)
        {
        a[j+1]=a[j];
        j=j-1;
        }
    a[j+1]=key;
   }
   for (int i=0; i<n; ++i) std::cout << a[i]<<" ";
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
```

```
  return 0;
}
```

OUTPUT

```
0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569
569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657
657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726
728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770
771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808
808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828
829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856
856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873
873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898
```

898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955 955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977 980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
It took me 0.00104689 seconds.

Now, we have executed inserted sort with an array of size 800 , by using the rand function we have imputed the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be `0.000685773 seconds.`

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0 && a[j]>key)
        {
        a[j+1]=a[j];
        j=j-1;
        }
    a[j+1]=key;
   }
   for (int i=0; i<n; ++i) std::cout << a[i]<<" ";
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

```
2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34
36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59
60 62 63 67 69 71 72 73 74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91
93 94 95 97 97 98 100 100 105 107 115 117 117 120 121 123 124 124 125 126
127 127 127 128 130 131 133 134 135 137 139 139 142 143 144 149 150 151 152
153 155 157 157 157 159 159 163 167 168 170 172 172 172 173 176 177 178 179
179 183 188 189 189 190 190 190 193 195 196 197 198 199 202 202 204 205 205
205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226 227 228
228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275
276 278 280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298
299 299 301 301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320
321 321 324 324 324 325 326 327 328 333 335 335 336 336 336 338 338 339 340
340 340 340 342 342 343 343 346 346 347 348 348 350 353 355 355 355 358 358
360 362 363 363 364 365 367 367 367 368 368 368 368 369 370 372 373 375 376
377 378 379 379 379 379 379 383 385 386 386 388 390 390 393 395 396 398 399
403 403 404 404 407 412 412 413 414 416 417 417 418 421 421 422 422 425 426
427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441 443 443 444
444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474 478
479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499
500 500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522
524 524 526 528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541
542 542 543 545 550 551 552 555 556 560 560 563 566 567 567 567 567 568 569
569 570 571 573 573 574 574 578 579 580 581 582 584 586 586 587 589 590 590
593 593 596 599 600 601 604 605 606 606 607 610 611 613 613 614 614 618 618
619 620 621 622 622 624 624 625 626 626 627 629 630 631 633 637 637 639 640
640 641 644 647 647 648 649 651 652 658 659 660 661 661 667 668 669 672 675
676 677 677 681 681 682 682 683 683 684 685 685 686 688 689 690 690 691 692
697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721 723 723
725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768
769 770 770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791
793 793 794 794 795 795 796 796 797 801 802 804 805 805 805 808 808 810 811
812 813 814 814 818 818 818 819 819 822 826 828 829 829 830 836 839 840 841
842 846 846 847 848 849 850 850 851 856 856 856 857 857 857 858 858 859 860
862 862 862 865 865 868 868 871 872 873 873 873 878 881 882 884 886 886 887
888 890 892 894 895 898 898 899 899 900 902 902 904 904 904 904 908 911 914
915 916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 929
930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 954
954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975 977
980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.000685773 seconds.
```

Now, we have executed inserted sort with an array of size 600 , by using the rand function we have imputed the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000457649 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=600;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0 && a[j]>key)
        {
        a[j+1]=a[j];
        j=j-1;
        }
    a[j+1]=key;
   }
   for (int i=0; i<n; ++i) std::cout << a[i]<<" ";
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000457649 seconds.

Now, we have executed inserted sort with an array of size 400 , by using the rand function we have imputed the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000257412 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=400;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0 && a[j]>key)
        {
        a[j+1]=a[j];
        j=j-1;
        }
    a[j+1]=key;
   }
   for (int i=0; i<n; ++i) std::cout << a[i]<<" ";
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000257412 seconds.

Now, we have executed inserted sort with an array of size 200 , by using the rand function we have imputed the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000107346 seconds .

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=200;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    int i,j,key;
    for(i=1;i<n;i++)
    {
        key=a[i];
        j=i-1;
        while(j>=0 && a[j]>key)
        {
        a[j+1]=a[j];
        j=j-1;
        }
    a[j+1]=key;
   }
   for (int i=0; i<n; ++i) std::cout << a[i]<<" ";
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

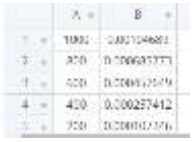
OUTPUT

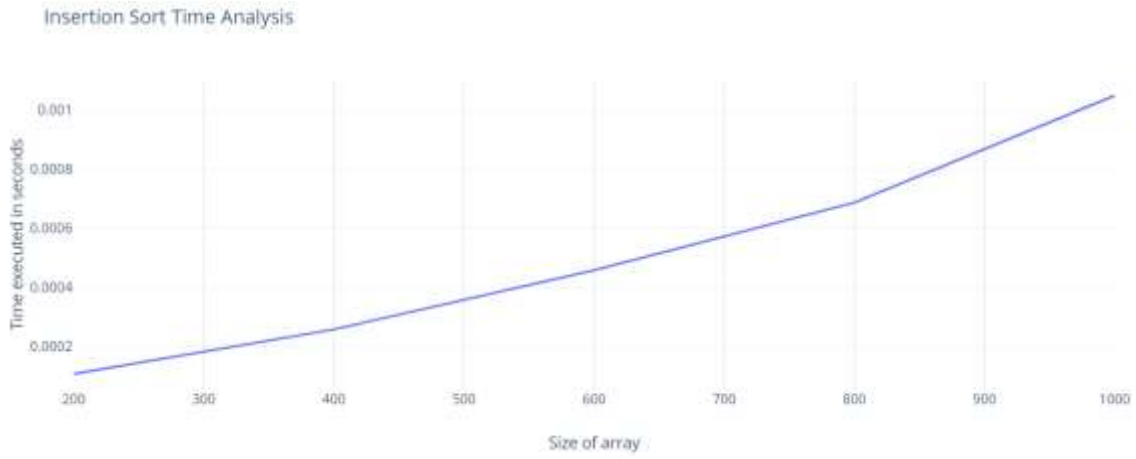
11 11 12 19 22 27 29 31 34 42 43 58 59 60 67 69 84 87 91 94 97 97 117 121 123 124 124 135 143 149 167 170 172 178 193 198 211 219 226 227 228 228 229 235 237 245 270 275

276 280 281 286 301 305 306 313 315 317 324 327 335 336 340 350 353 362 364 365 367 368 368 368 370 373 378 379 383 386 393 395 399 403 407 413 421 421 426 428 429 432 434 434 437 440 441 444 451 456 467 474 481 488 491 492 492 497 500 503 505 526 528 529 530 537 539 540 545 551 555 567 567 570 582 584 586 586 601 618 619 624 649 651 652 675 676 683 689 690 708 709 715 723 729 729 732 736 739 743 750 754 756 763 764 764 771 777 782 784 788 793 793 795 796 802 808 814 818 829 841 846 846 856 856 857 858 859 862 862 865 871 873 886 895 902 914 915 919 921 925 926 927 928 929 932 936 956 965 980 987 996
It took me 0.000107346 seconds.

Now, we have executed inserted sort with an array of size 10 , by using the rand function we have imputed the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000107346 seconds .

1000	0.00104689
800	0.000685773
600	0.000457649
400	0.000257412
200	0.000107346





As of conclusion we can conclude that insertion sort have slight variation when we executed on array size>800 ,so there is almost no effect and the graph on the libe is coming straight.

SELECTION SORT

Now, we have executed selection sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00139567 seconds .

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

void selectionSort(int arr[], int n)
{
	int i, j, min_idx;


	for (i = 0; i < n-1; i++)
	{

		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;
		swap(&arr[min_idx], &arr[i]);
	}
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=1000;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    selectionSort(a, n);
```

```
for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569
569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657

657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726
728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770
771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808
808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828
829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856
856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873
873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898
898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919
919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933
934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955
955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977
980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996
996 996 997 999
It took me 0.00139567 seconds.

Now, we have executed selection sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00115807 seconds .

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

void selectionSort(int arr[], int n)
{
	int i, j, min_idx;


	for (i = 0; i < n-1; i++)
	{

		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;
		swap(&arr[min_idx], &arr[i]);
	}
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    selectionSort(a, n);

for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794
795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850
851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925
926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950

950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.00115807 seconds.

Now, we have executed selection sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000654049 seconds .

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

void selectionSort(int arr[], int n)
{
	int i, j, min_idx;


	for (i = 0; i < n-1; i++)
	{

		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;
		swap(&arr[min_idx], &arr[i]);
	}
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=600;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    selectionSort(a, n);

for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000654049 seconds.
```

Now, we have executed selection sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000366082 seconds .

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

void selectionSort(int arr[], int n)
{
	int i, j, min_idx;


	for (i = 0; i < n-1; i++)
	{

		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;
		swap(&arr[min_idx], &arr[i]);
	}
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=400;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    selectionSort(a, n);

for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000366082 seconds.
```

Now, we have executed selection sort with an array of size 200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000148071 seconds

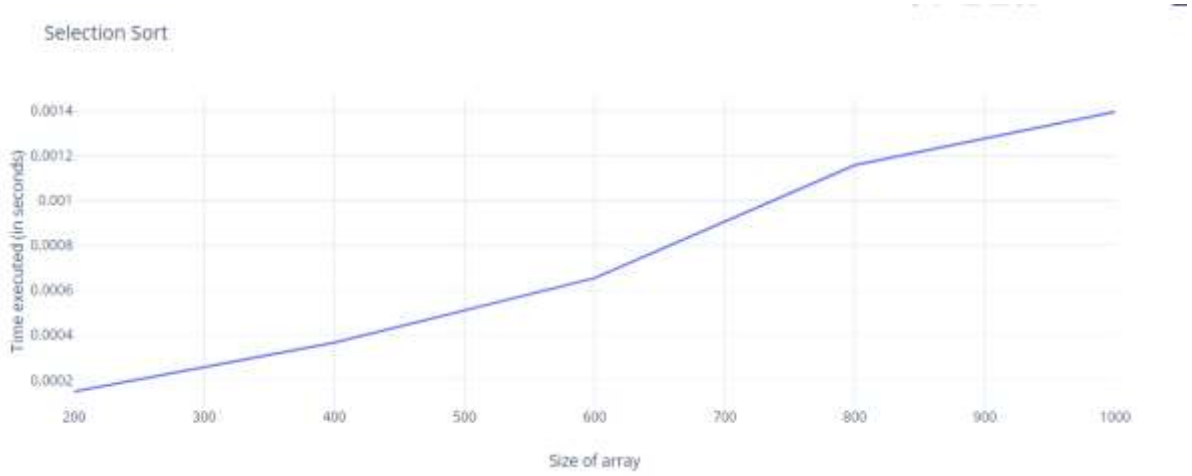
```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
	int temp = *xp;
	*xp = *yp;
	*yp = temp;
}

void selectionSort(int arr[], int n)
{
	int i, j, min_idx;


	for (i = 0; i < n-1; i++)
	{

		min_idx = i;
		for (j = i+1; j < n; j++)
		if (arr[j] < arr[min_idx])
			min_idx = j;
		swap(&arr[min_idx], &arr[i]);
	}
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

    int n=200;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
    selectionSort(a, n);

for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

OUTPUT

```
11 11 12 19 22 27 29 31 34 42 43 58 59 60 67 69 84 87 91 94 97 97 117 121 123 124 124
135 143 149 167 170 172 178 193 198 211 219 226 227 228 228 229 235 237 245 270 275
276 280 281 286 301 305 306 313 315 317 324 327 335 336 340 350 353 362 364 365 367
368 368 368 370 373 378 379 383 386 393 395 399 403 407 413 421 421 426 428 429 432
434 434 437 440 441 444 451 456 467 474 481 488 491 492 492 497 500 503 505 526 528
529 530 537 539 540 545 551 555 567 567 570 582 584 586 586 601 618 619 624 649 651
652 675 676 683 689 690 708 709 715 723 729 729 732 736 739 743 750 754 756 763 764
764 771 777 782 784 788 793 793 795 796 802 808 814 818 829 841 846 846 856 856 857
858 859 862 862 865 871 873 886 895 902 914 915 919 921 925 926 927 928 929 932 936
956 965 980 987 996
It took me 0.000148071 seconds.
```

	A	B
1	1000	0.00139567
2	800	0.00115807
3	600	0.000654049
4	400	0.000366082
5	200	0.000148071



From this we also conclude that the by increasing the size of inputs the timing also increases i.e. after 600 inputs we can see graph having abnormal behaviour.

Shell Sort

Now, we have executed shell sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000344695 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

 int shellSort(int arr[], int n)
{
        for (int gap = n/2; gap > 0; gap /= 2)
        {

                for (int i = gap; i < n; i += 1)
                {

                        int temp = arr[i];


                        int j;
                        for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                                arr[j] = arr[j - gap];


                        arr[j] = temp;
                }
        }
        return 0;
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=1000;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
        shellSort(a, n);


for(int i=0;i<n;i++)
```

```
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569
569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657
657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726

728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770
771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808
808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828
829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856
856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873
873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898
898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919
919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933
934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955
955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977
980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996
996 996 997 999
It took me 0.000344695 seconds.

Now, we have executed shell sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000277468 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

 int shellSort(int arr[], int n)
{
        for (int gap = n/2; gap > 0; gap /= 2)
        {

                for (int i = gap; i < n; i += 1)
                {

                        int temp = arr[i];


                        int j;
                        for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                                arr[j] = arr[j - gap];


                        arr[j] = temp;
                }
        }
        return 0;
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
        shellSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";
```

```
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794
795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850
851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925

926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950
950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.000277468 seconds.

Now, we have executed shell sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000216706 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

 int shellSort(int arr[], int n)
{
        for (int gap = n/2; gap > 0; gap /= 2)
        {

                for (int i = gap; i < n; i += 1)
                {

                        int temp = arr[i];


                        int j;
                        for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                                arr[j] = arr[j - gap];


                        arr[j] = temp;
                }
        }
        return 0;
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=600;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
        shellSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";
```

```
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000216706 seconds.
```

Now, we have executed shell sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000155435 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

 int shellSort(int arr[], int n)
{
        for (int gap = n/2; gap > 0; gap /= 2)
        {

                for (int i = gap; i < n; i += 1)
                {

                        int temp = arr[i];


                        int j;
                        for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                                arr[j] = arr[j - gap];


                        arr[j] = temp;
                }
        }
        return 0;
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=400;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
        shellSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";
```

```
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000155435 seconds.
```

Now, we have executed shell sort with an array of size 1200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.0004257 seconds

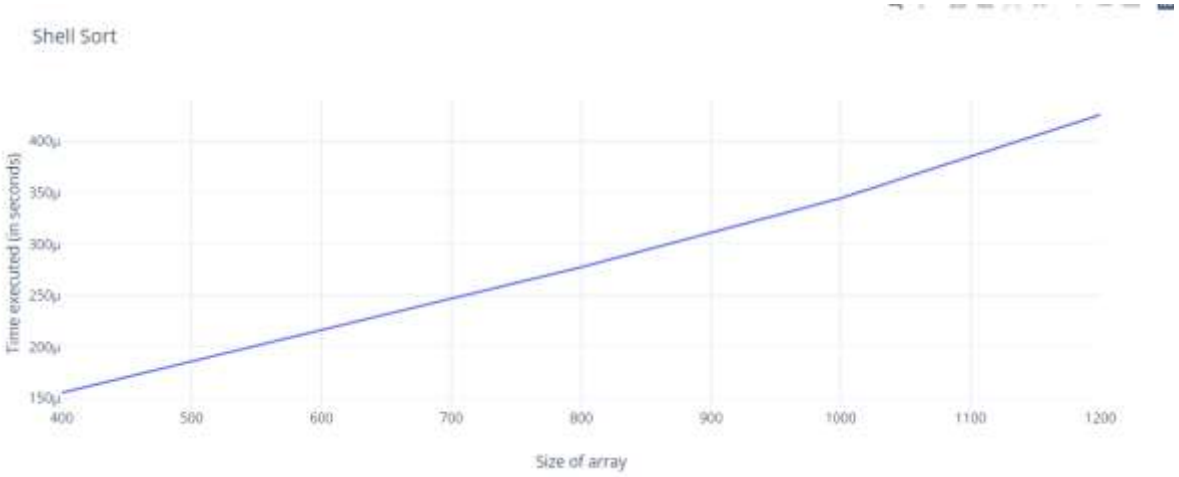
```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

 int shellSort(int arr[], int n)
{
        for (int gap = n/2; gap > 0; gap /= 2)
        {

                for (int i = gap; i < n; i += 1)
                {

                        int temp = arr[i];


                        int j;
                        for (j = i; j >= gap && arr[j - gap] > temp; j -= gap)
                                arr[j] = arr[j - gap];


                        arr[j] = temp;
                }
        }
        return 0;
}
int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=1200;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
        shellSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";
```

```
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

0 0 2 2 2 4 6 8 9 10 10 11 11 11 12 15 16 17 17 18 19 19 21 21 21 21 22 23 25 27 27 28 29
29 30 30 30 30 31 32 32 33 34 36 36 36 37 39 39 40 41 42 42 42 42 43 43 46 47 49 49 50
50 51 52 53 54 55 57 57 57 58 59 59 59 60 62 62 63 67 67 69 69 71 71 72 73 74 74 75 80
81 81 81 81 81 82 82 83 84 84 84 84 86 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100
100 100 104 105 107 109 109 109 112 114 115 116 117 117 120 121 122 122 123 124 124
125 125 126 126 127 127 127 127 128 130 131 131 131 133 133 134 134 135 136 137 139
139 140 140 141 141 142 143 144 147 149 149 150 151 152 152 153 153 154 154 154 154
155 155 156 157 157 157 159 159 159 160 163 167 168 170 170 171 171 171 172 172 172
172 173 173 174 176 177 178 179 179 179 181 181 181 181 183 183 183 184 188 189 189
189 190 190 190 192 193 195 195 196 197 197 198 198 199 199 202 202 204 205 205 205
206 207 207 209 210 211 211 211 214 215 216 217 217 217 218 218 219 220 222 223 224
224 225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237
237 238 238 244 245 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 263
264 266 266 269 269 269 270 270 270 272 273 274 275 276 278 278 280 280 280 281 281
282 283 283 284 285 285 286 286 289 289 289 289 290 291 291 292 292 292 294 295 296
297 297 298 298 299 299 300 301 301 303 303 304 304 305 305 305 306 308 308 309 309
310 311 312 312 313 313 314 314 314 315 317 320 321 321 323 324 324 324 325 325 326
326 327 328 333 334 334 334 335 335 335 336 336 336 337 338 338 338 339 340 340 340
340 341 342 342 343 343 343 346 346 347 348 348 350 350 350 351 352 353 354 355 355
355 355 358 358 358 360 360 361 362 363 363 364 364 364 364 365 365 367 367 367 368
368 368 368 368 369 370 372 373 375 376 376 377 377 378 378 379 379 379 379 379 382
382 383 383 385 386 386 388 390 390 393 393 393 393 395 396 398 398 399 399 403 403
404 404 404 407 412 412 412 412 413 414 415 415 416 417 417 418 418 421 421 421 421
422 422 424 425 426 426 427 428 428 428 429 429 431 431 432 433 433 434 434 435 436
437 437 438 439 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 465 466 467 468 468 469 470 471 472 474 474 475 476 478 479 480 481 482 483
483 483 485 485 486 486 487 487 488 489 490 490 491 491 491 492 492 493 493 494 496
497 497 498 499 499 500 500 500 503 503 504 504 505 505 506 506 506 506 506 507 511
512 516 516 517 518 518 521 522 522 524 524 525 525 526 528 528 528 528 528 529 529
529 530 532 532 535 535 536 537 537 537 538 538 538 539 539 540 541 541 542 542 543
545 548 550 551 551 551 552 552 555 555 556 557 559 559 560 560 562 563 563 566 566
567 567 567 567 568 569 569 570 570 571 573 573 573 574 574 577 578 579 580 581 582
582 583 583 584 586 586 587 587 589 590 590 590 590 593 593 593 595 596 599 600 601

603 604 605 606 606 607 610 610 611 613 613 614 614 618 618 618 619 620 621 621 621
622 622 624 624 624 625 626 626 627 629 630 631 631 633 636 637 637 638 639 639 640
640 641 641 641 643 644 647 647 648 649 649 651 651 652 653 657 657 658 659 659 660
661 661 662 665 666 667 667 668 669 672 672 673 675 676 677 677 677 679 681 681 682
682 682 683 683 684 685 685 685 686 688 689 690 690 690 691 692 693 697 698 699 699
701 703 705 708 708 708 708 709 710 710 710 711 713 713 713 714 715 715 716 717 718
719 720 720 721 722 722 723 723 723 725 726 726 726 728 729 729 729 729 729 730 732
732 732 735 735 736 736 739 740 743 743 743 744 746 746 747 748 750 753 754 754 754
756 756 757 759 760 761 761 761 762 762 763 763 763 764 764 768 769 770 770 771 772
773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788 789 789
790 790 791 793 793 793 794 794 795 795 796 796 797 797 797 801 802 804 805 805 805
805 806 808 808 810 811 812 813 813 814 814 814 815 818 818 818 819 819 820 821 822
823 825 826 827 828 828 829 829 830 833 836 839 839 840 841 842 844 844 846 846 847
848 849 850 850 850 850 851 851 852 853 856 856 856 856 857 857 857 857 858 858 859
859 860 860 862 862 862 865 865 868 868 868 871 871 872 873 873 873 875 875 877 878
881 882 884 886 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898 898 898
899 899 900 902 902 902 902 904 904 904 904 905 907 908 910 911 912 914 915 916 917
917 918 919 919 919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931
932 932 933 933 933 933 934 936 936 938 939 940 943 944 945 946 947 948 949 949 949
950 950 951 952 954 954 954 954 955 955 955 956 956 956 957 957 958 959 959 960 961
962 963 964 965 967 967 969 969 969 970 970 970 971 972 972 973 975 976 976 977 977
980 980 981 981 981 982 982 984 984 985 987 987 987 988 989 990 990 990 991 993 993
994 994 996 996 996 996 996 997 999
It took me 0.0004257 seconds.

	A	B
1	1200	0.0004257
2	1000	0.000344695
3	800	0.000277468
4	600	0.000216706
5	400	0.000155435



Now from this graph we can conclude that shell sort is unaffected by the size of array.

Now, we have executed bubble sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00275965 seconds

BUBBLE SORT

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>


void swap(int *xp, int *yp)
{
          int temp = *xp;
          *xp = *yp;
          *yp = temp;
}

void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   for (j = 0; j < n-i-1; j++)
       if (arr[j] > arr[j+1])
           swap(&arr[j], &arr[j+1]);
}

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=1000;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;
          bubbleSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";
```

```
  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569
569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657
657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726
728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770

771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808
808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828
829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856
856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873
873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898
898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919
919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933
934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955
955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977
980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996
996 996 997 999
It took me 0.00275965 seconds.

Now, we have executed bubble sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00224845 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void swap(int *xp, int *yp)
{
          int temp = *xp;
          *xp = *yp;
          *yp = temp;
}

void bubbleSort(int arr[], int n)
{
    int i, j;
    for (i = 0; i < n-1; i++)

    for (j = 0; j < n-i-1; j++)
        if (arr[j] > arr[j+1])
            swap(&arr[j], &arr[j+1]);
}

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();

     int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
          bubbleSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();
```

```
  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794
795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850
851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925
926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950
950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.00224845 seconds.

Now, we have executed bubble sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.00129407 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>


void swap(int *xp, int *yp)
{
          int temp = *xp;
          *xp = *yp;
          *yp = temp;
}

void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   for (j = 0; j < n-i-1; j++)
       if (arr[j] > arr[j+1])
           swap(&arr[j], &arr[j+1]);
}

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=600;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;
          bubbleSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.00129407 seconds.
```

Now, we have executed bubble sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000590582 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>


void swap(int *xp, int *yp)
{
        int temp = *xp;
        *xp = *yp;
        *yp = temp;
}

void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   for (j = 0; j < n-i-1; j++)
      if (arr[j] > arr[j+1])
         swap(&arr[j], &arr[j+1]);
}

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=400;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;
        bubbleSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000590582 seconds.
```

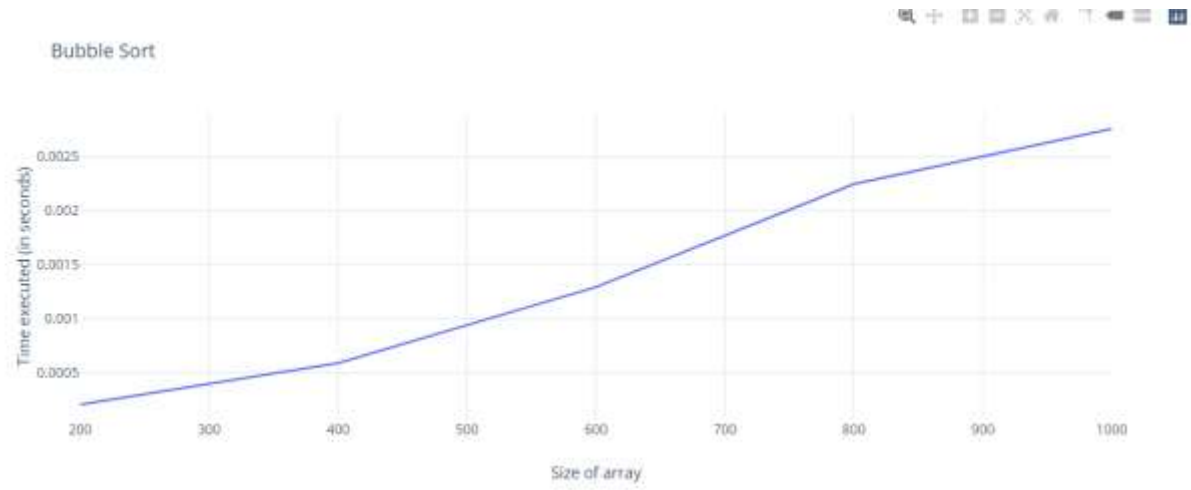
Now, we have executed bubble sort with an array of size 200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000207051 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>


void swap(int *xp, int *yp)
{
        int temp = *xp;
        *xp = *yp;
        *yp = temp;
}

void bubbleSort(int arr[], int n)
{
   int i, j;
   for (i = 0; i < n-1; i++)

   for (j = 0; j < n-i-1; j++)
      if (arr[j] > arr[j+1])
         swap(&arr[j], &arr[j+1]);
}

int main ()
{
  using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=200;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;
        bubbleSort(a, n);


for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

11 11 12 19 22 27 29 31 34 42 43 58 59 60 67 69 84 87 91 94 97 97 117 121 123 124 124 135 143 149 167 170 172 178 193 198 211 219 226 227 228 228 229 235 237 245 270 275 276 280 281 286 301 305 306 313 315 317 324 327 335 336 340 350 353 362 364 365 367 368 368 368 370 373 378 379 383 386 393 395 399 403 407 413 421 421 426 428 429 432 434 434 437 440 441 444 451 456 467 474 481 488 491 492 492 497 500 503 505 526 528 529 530 537 539 540 545 551 555 567 567 570 582 584 586 586 601 618 619 624 649 651 652 675 676 683 689 690 708 709 715 723 729 729 732 736 739 743 750 754 756 763 764 764 771 777 782 784 788 793 793 795 796 802 808 814 818 829 841 846 846 856 856 857 858 859 862 862 865 871 873 886 895 902 914 915 919 921 925 926 927 928 929 932 936 956 965 980 987 996
It took me 0.000207051 seconds.

	A	B
1	1000	0.00275965
2	800	0.00224845
3	600	0.00129407
4	400	0.000590582
5	200	0.000207051



As of conclusion we can conclude that bubble sort have variation when we executed on array size>400 ,sao graph is affected by input size.

Merge Sort

Now, we have executed merge sort with an array of size 200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000126149 seconds

```
// C++ program for Merge Sort
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


void merge(int array[], int const left, int const mid, int const right)
{
	auto const subArrayOne = mid - left + 1;
	auto const subArrayTwo = right - mid;


	auto *leftArray = new int[subArrayOne],
		*rightArray = new int[subArrayTwo];


	for (auto i = 0; i < subArrayOne; i++)
		leftArray[i] = array[left + i];
	for (auto j = 0; j < subArrayTwo; j++)
		rightArray[j] = array[mid + 1 + j];

	auto indexOfSubArrayOne = 0, // Initial index of first sub-array
		indexOfSubArrayTwo = 0; // Initial index of second sub-array
	int indexOfMergedArray = left; // Initial index of merged array


	while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo <
subArrayTwo) {
		if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
			array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
			indexOfSubArrayOne++;
		}
		else {
			array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
			indexOfSubArrayTwo++;
		}
		indexOfMergedArray++;
	}
```

```
        while (indexOfSubArrayOne < subArrayOne) {
                array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
                indexOfSubArrayOne++;
                indexOfMergedArray++;
        }

        while (indexOfSubArrayTwo < subArrayTwo) {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
}


void mergeSort(int array[], int const begin, int const end)
{
        if (begin >= end)
                return; // Returns recursively

        auto mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end);
        merge(array, begin, mid, end);
}


int main()
{
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=200;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
            mergeSort(a, 0, n - 1);
          for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();
```

```
  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
11 11 12 19 22 27 29 31 34 42 43 58 59 60 67 69 84 87 91 94 97 97 117 121 123 124 124
135 143 149 167 170 172 178 193 198 211 219 226 227 228 228 229 235 237 245 270 275
276 280 281 286 301 305 306 313 315 317 324 327 335 336 340 350 353 362 364 365 367
368 368 368 370 373 378 379 383 386 393 395 399 403 407 413 421 421 426 428 429 432
434 434 437 440 441 444 451 456 467 474 481 488 491 492 492 497 500 503 505 526 528
529 530 537 539 540 545 551 555 567 567 570 582 584 586 586 601 618 619 624 649 651
652 675 676 683 689 690 708 709 715 723 729 729 732 736 739 743 750 754 756 763 764
764 771 777 782 784 788 793 793 795 796 802 808 814 818 829 841 846 846 856 856 857
858 859 862 862 865 871 873 886 895 902 914 915 919 921 925 926 927 928 929 932 936
956 965 980 987 996
It took me 0.000126149 seconds.
```

Now, we have executed merge sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000225189 seconds

```
// C++ program for Merge Sort
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void merge(int array[], int const left, int const mid, int const right)
{
	auto const subArrayOne = mid - left + 1;
	auto const subArrayTwo = right - mid;


	auto *leftArray = new int[subArrayOne],
		*rightArray = new int[subArrayTwo];


	for (auto i = 0; i < subArrayOne; i++)
		leftArray[i] = array[left + i];
	for (auto j = 0; j < subArrayTwo; j++)
		rightArray[j] = array[mid + 1 + j];

	auto indexOfSubArrayOne = 0,
		indexOfSubArrayTwo = 0; // Initial index of second sub-array
	int indexOfMergedArray = left;


	while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo <
subArrayTwo) {
		if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
			array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
			indexOfSubArrayOne++;
		}
		else {
			array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
			indexOfSubArrayTwo++;
		}
		indexOfMergedArray++;
	}
	ny
	while (indexOfSubArrayOne < subArrayOne) {
		array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
		indexOfSubArrayOne++;
		indexOfMergedArray++;
	}
```

```
        while (indexOfSubArrayTwo < subArrayTwo) {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
}


void mergeSort(int array[], int const begin, int const end)
{
        if (begin >= end)
                return; // Returns recursively

        auto mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end);
        merge(array, begin, mid, end);
}


void printArray(int A[], int size)
{
        for (auto i = 0; i < size; i++)
                cout << A[i] << " ";
}

// Driver code
int main()
{
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=400;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
            mergeSort(a, 0, n - 1);
          for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();
```

```
  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000225189 seconds.
```

Now, we have executed merge sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000359487 seconds

```
// C++ program for Merge Sort
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>

void merge(int array[], int const left, int const mid, int const right)
{
	auto const subArrayOne = mid - left + 1;
	auto const subArrayTwo = right - mid;


	auto *leftArray = new int[subArrayOne],
		*rightArray = new int[subArrayTwo];


	for (auto i = 0; i < subArrayOne; i++)
		leftArray[i] = array[left + i];
	for (auto j = 0; j < subArrayTwo; j++)
		rightArray[j] = array[mid + 1 + j];

	auto indexOfSubArrayOne = 0,
		indexOfSubArrayTwo = 0; // Initial index of second sub-array
	int indexOfMergedArray = left;


	while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo <
subArrayTwo) {
		if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
			array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
			indexOfSubArrayOne++;
		}
		else {
			array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
			indexOfSubArrayTwo++;
		}
		indexOfMergedArray++;
	}
	ny
	while (indexOfSubArrayOne < subArrayOne) {
		array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
		indexOfSubArrayOne++;
		indexOfMergedArray++;
	}
```

```
        while (indexOfSubArrayTwo < subArrayTwo) {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
}


void mergeSort(int array[], int const begin, int const end)
{
        if (begin >= end)
                return; // Returns recursively

        auto mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end);
        merge(array, begin, mid, end);
}


void printArray(int A[], int size)
{
        for (auto i = 0; i < size; i++)
                cout << A[i] << " ";
}

// Driver code
int main()
{
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=600;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
            mergeSort(a, 0, n - 1);
          for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;


  high_resolution_clock::time_point t2 = high_resolution_clock::now();
```

```
  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000359487 seconds.
```

Now, we have executed merge sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000418488 seconds

```
// C++ program for Merge Sort
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


void merge(int array[], int const left, int const mid, int const right)
{
	auto const subArrayOne = mid - left + 1;
	auto const subArrayTwo = right - mid;


	auto *leftArray = new int[subArrayOne],
		*rightArray = new int[subArrayTwo];


	for (auto i = 0; i < subArrayOne; i++)
		leftArray[i] = array[left + i];
	for (auto j = 0; j < subArrayTwo; j++)
		rightArray[j] = array[mid + 1 + j];

	auto indexOfSubArrayOne = 0,
		indexOfSubArrayTwo = 0; // Initial index of second sub-array
	int indexOfMergedArray = left;


	while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo <
subArrayTwo) {
		if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
			array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
			indexOfSubArrayOne++;
		}
		else {
			array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
			indexOfSubArrayTwo++;
		}
		indexOfMergedArray++;
	}

	while (indexOfSubArrayOne < subArrayOne) {
		array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
```

```
                indexOfSubArrayOne++;
                indexOfMergedArray++;
        }

        while (indexOfSubArrayTwo < subArrayTwo) {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
}


void mergeSort(int array[], int const begin, int const end)
{
        if (begin >= end)
                return; // Returns recursively

        auto mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end);
        merge(array, begin, mid, end);
}


void printArray(int A[], int size)
{
        for (auto i = 0; i < size; i++)
                cout << A[i] << " ";
}

// Driver code
int main()
{
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
            mergeSort(a, 0, n - 1);
          for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794
795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850
851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925
926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950

950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.000418488 seconds.

Now, we have executed merge sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000509705 seconds

```
// C++ program for Merge Sort
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


void merge(int array[], int const left, int const mid, int const right)
{
	auto const subArrayOne = mid - left + 1;
	auto const subArrayTwo = right - mid;


	auto *leftArray = new int[subArrayOne],
		*rightArray = new int[subArrayTwo];


	for (auto i = 0; i < subArrayOne; i++)
		leftArray[i] = array[left + i];
	for (auto j = 0; j < subArrayTwo; j++)
		rightArray[j] = array[mid + 1 + j];

	auto indexOfSubArrayOne = 0,
		indexOfSubArrayTwo = 0; // Initial index of second sub-array
	int indexOfMergedArray = left;


	while (indexOfSubArrayOne < subArrayOne && indexOfSubArrayTwo <
subArrayTwo) {
		if (leftArray[indexOfSubArrayOne] <= rightArray[indexOfSubArrayTwo]) {
			array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
			indexOfSubArrayOne++;
		}
		else {
			array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
			indexOfSubArrayTwo++;
		}
		indexOfMergedArray++;
	}

	while (indexOfSubArrayOne < subArrayOne) {
		array[indexOfMergedArray] = leftArray[indexOfSubArrayOne];
		indexOfSubArrayOne++;
```

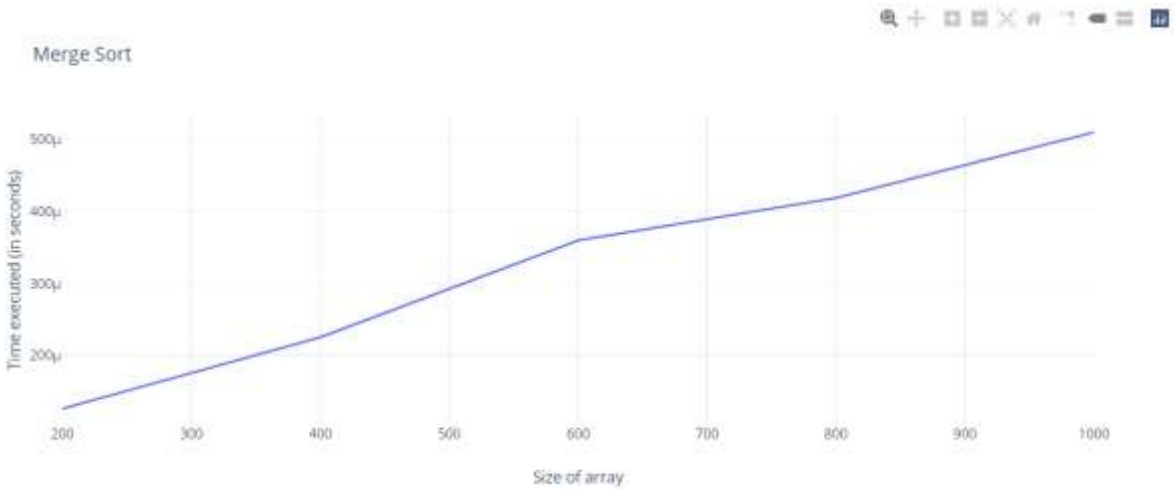
```
                indexOfMergedArray++;
        }

        while (indexOfSubArrayTwo < subArrayTwo) {
                array[indexOfMergedArray] = rightArray[indexOfSubArrayTwo];
                indexOfSubArrayTwo++;
                indexOfMergedArray++;
        }
}


void mergeSort(int array[], int const begin, int const end)
{
        if (begin >= end)
                return; // Returns recursively

        auto mid = begin + (end - begin) / 2;
        mergeSort(array, begin, mid);
        mergeSort(array, mid + 1, end);
        merge(array, begin, mid, end);
}


void printArray(int A[], int size)
{
        for (auto i = 0; i < size; i++)
                cout << A[i] << " ";
}

// Driver code
int main()
{
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=1000;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
            mergeSort(a, 0, n - 1);
          for(int i=0;i<n;i++)
std::cout << a[i] << " ";

  std::cout << std::endl;
```

```
  high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;

  return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569
569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657
657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726
728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770
771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808

808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828 829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856 856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873 873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898 898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955 955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977 980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
It took me 0.000509705 seconds.

	A	B
1	1000	0.000509705
2	800	0.000418488
3	600	0.000359487
4	400	0.000225189
5	200	0.000126149



We can conclude that the merge is also slight affected by increasing the input size.

BUCKET SORT

Now, we have executed bucket sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000220722 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


int largest(int arr[], int n)
{
    int i;

    int max = arr[0];


    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
int main() {
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=1000;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
           int m,j;
           m=largest(a,n);
           int aux[100];
           for(int i=0;i<m;i++)
           aux[i]=0;

           for(int i=0;i<n;i++)
           aux[a[i]]++;
```

```
          for(int i=0,j=0;i<=m;i++)
          {
            for(;aux[i]>0;aux[i]--)
            {
               a[j]=i;
               j++;
            }
          }
                    for(int i=0;i<n;i++)
        std::cout << a[i] << " ";
       std::cout << std::endl;

       high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
          return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30 31
32 33 34 36 36 36 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57 57 57 58 59
59 59 60 62 63 67 67 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 84 84 84 84 86 87 87
88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 105 107 109 109 114 115 116 117 117
120 121 122 122 123 124 124 125 126 126 127 127 127 127 128 130 131 131 133 134 134
135 136 137 139 139 142 143 144 147 149 149 150 151 152 153 154 154 155 157 157 157
159 159 163 167 168 170 171 171 172 172 172 173 176 177 178 179 179 179 181 181 181
183 184 188 189 189 189 190 190 190 193 195 195 196 197 197 198 199 199 202 202 204
205 205 205 206 207 209 210 211 211 211 214 215 217 217 218 218 219 222 223 224 224
225 226 226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237
238 244 245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 281 281 282 283 283 284 285 285 286
289 289 290 291 291 292 292 292 294 296 297 297 298 298 299 299 301 301 303 303 304
305 305 306 308 308 309 309 310 311 312 312 313 314 314 315 317 320 321 321 324 324
324 325 325 326 327 328 333 334 335 335 335 336 336 336 337 338 338 338 339 340 340
340 340 342 342 343 343 346 346 347 348 348 350 350 350 353 355 355 355 358 358 360
360 362 363 363 364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376
376 377 378 378 379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 395
396 398 398 399 403 403 404 404 404 407 412 412 412 413 414 415 415 416 417 417 418
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460 462
464 465 466 467 468 470 471 474 476 478 479 481 483 483 483 485 486 486 487 488 490
490 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503 503 504 504 505 505
506 506 506 507 512 516 516 518 518 521 522 522 524 524 525 525 526 528 528 528 528
528 529 529 529 530 532 532 535 536 537 537 538 538 539 539 540 541 542 542 543 545
550 551 551 552 552 555 555 556 559 560 560 563 563 566 566 567 567 567 567 568 569

569 570 570 571 573 573 574 574 577 578 579 580 581 582 582 583 584 586 586 587 587
589 590 590 590 593 593 593 595 596 599 600 601 603 604 605 606 606 607 610 610 611
613 613 614 614 618 618 618 619 620 621 621 622 622 624 624 625 626 626 627 629 630
631 633 637 637 639 639 640 640 641 641 643 644 647 647 648 649 651 651 652 653 657
657 658 659 660 661 661 667 668 669 672 672 675 676 677 677 681 681 682 682 682 683
683 684 685 685 685 686 688 689 690 690 691 692 697 698 699 699 701 703 705 708 708
708 709 710 710 711 713 713 713 714 715 715 717 720 720 721 722 722 723 723 725 726
728 729 729 729 729 730 732 732 732 735 736 736 739 740 743 743 743 744 746 746 747
748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770 770
771 772 773 774 775 776 776 776 776 776 777 777 782 783 783 784 784 786 787 788 788
789 791 793 793 794 794 795 795 796 796 797 797 801 802 804 805 805 805 805 806 808
808 810 811 812 813 813 814 814 815 818 818 818 819 819 820 821 822 825 826 827 828
829 829 830 836 839 839 840 841 842 846 846 847 848 849 850 850 850 851 853 856 856
856 857 857 857 857 858 858 859 860 860 862 862 862 865 865 868 868 871 872 873 873
873 875 878 881 882 884 886 886 887 888 888 888 890 890 892 892 892 894 895 898 898
898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915 916 917 917 918 919
919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930 931 932 932 933 933
934 936 936 939 940 943 944 945 946 947 949 949 950 950 951 952 954 954 954 955 955
955 956 958 959 959 961 962 963 964 965 967 969 970 970 971 972 973 975 976 977 977
980 981 981 982 982 984 984 987 987 988 989 990 990 991 993 993 994 994 996 996 996
996 996 997 999
It took me 0.000220722 seconds.

Now, we have executed bucket sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000184788 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


int largest(int arr[], int n)
{
    int i;

    int max = arr[0];


    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
int main() {
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=800;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
           int m,j;
           m=largest(a,n);
           int aux[100];
           for(int i=0;i<m;i++)
           aux[i]=0;

           for(int i=0;i<n;i++)
           aux[a[i]]++;

           for(int i=0,j=0;i<=m;i++)
           {
             for(;aux[i]>0;aux[i]--)
             {
```

```
                a[j]=i;
                j++;
              }
            }
                    for(int i=0;i<n;i++)
          std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
           return 0;
}
```

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794
795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850

851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925
926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950
950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.000184788 seconds.

Now, we have executed bucket sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000161775 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


int largest(int arr[], int n)
{
    int i;

    int max = arr[0];


    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
int main() {
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=600;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
           int m,j;
           m=largest(a,n);
           int aux[100];
           for(int i=0;i<m;i++)
           aux[i]=0;

           for(int i=0;i<n;i++)
           aux[a[i]]++;

           for(int i=0,j=0;i<=m;i++)
           {
             for(;aux[i]>0;aux[i]--)
             {
```

```
                a[j]=i;
                j++;
              }
            }
                    for(int i=0;i<n;i++)
          std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
           return 0;
}
```

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000161775 seconds.
```

Now, we have executed bucket sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 8.3016e-05 seconds

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


int largest(int arr[], int n)
{
    int i;

    int max = arr[0];


    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
int main() {
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=400;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
           int m,j;
           m=largest(a,n);
           int aux[100];
           for(int i=0;i<m;i++)
           aux[i]=0;

           for(int i=0;i<n;i++)
           aux[a[i]]++;

           for(int i=0,j=0;i<=m;i++)
           {
             for(;aux[i]>0;aux[i]--)
             {
```

```
                a[j]=i;
                j++;
              }
            }
                    for(int i=0;i<n;i++)
          std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
           return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 8.3016e-05 seconds.
```

Now, we have executed bucket sort with an array of size 1200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000247402 seconds

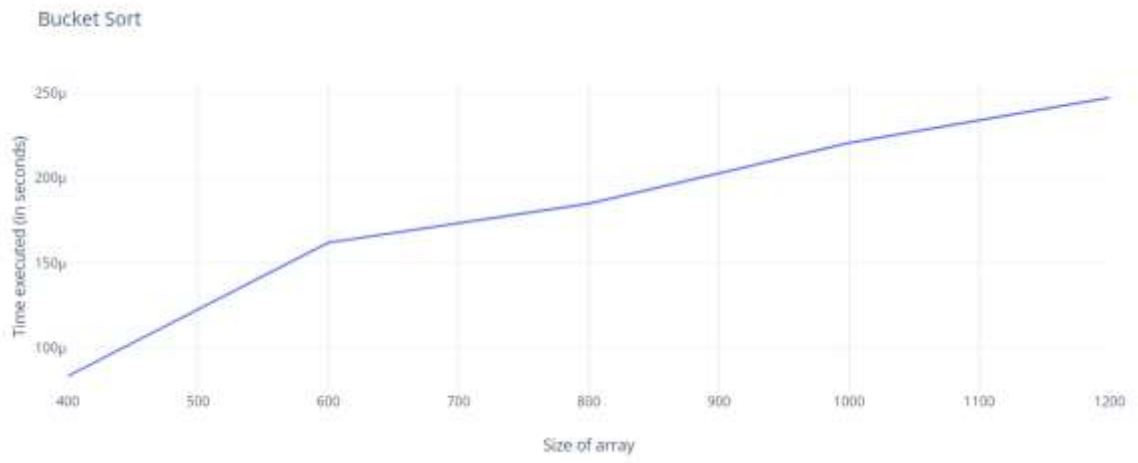
```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;


int largest(int arr[], int n)
{
    int i;

    int max = arr[0];


    for (i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];

    return max;
}
int main() {
    using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



     int n=1100;
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;
           int m,j;
           m=largest(a,n);
           int aux[100];
           for(int i=0;i<m;i++)
           aux[i]=0;

           for(int i=0;i<n;i++)
           aux[a[i]]++;

           for(int i=0,j=0;i<=m;i++)
           {
             for(;aux[i]>0;aux[i]--)
             {
```

```
                a[j]=i;
                j++;
              }
            }
                    for(int i=0;i<n;i++)
          std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
           return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 15 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30
31 32 32 33 34 36 36 36 37 39 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57
57 57 58 59 59 59 60 62 63 67 67 69 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 83 84
84 84 84 86 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 104 105 107 109 109
109 112 114 115 116 117 117 120 121 122 122 123 124 124 125 126 126 127 127 127 127
128 130 131 131 133 134 134 135 136 137 139 139 141 142 143 144 147 149 149 150 151
152 152 153 154 154 154 154 155 155 157 157 157 159 159 159 163 167 168 170 170 171
171 172 172 172 172 173 173 176 177 178 179 179 179 181 181 181 183 183 184 188 189
189 189 190 190 190 193 195 195 196 197 197 198 198 199 199 202 202 204 205 205 205
206 207 209 210 211 211 211 214 215 217 217 217 218 218 219 222 223 224 224 225 226
226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237 238 244
245 248 249 250 253 253 254 255 255 258 259 260 261 262 262 263 264 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 280 281 281 282 283 283 284 285 285
286 289 289 289 290 291 291 292 292 292 294 295 296 297 297 298 298 299 299 300 301
301 303 303 304 305 305 306 308 308 309 309 310 311 312 312 313 313 314 314 314 315
317 320 321 321 324 324 324 325 325 326 326 327 328 333 334 334 334 335 335 335 336
336 336 337 338 338 338 339 340 340 340 340 341 342 342 343 343 346 346 347 348 348
350 350 350 351 352 353 354 355 355 355 355 358 358 360 360 361 362 363 363 364 364
364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376 376 377 378 378
379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 393 395 396 398 398
399 399 403 403 404 404 404 407 412 412 412 412 413 414 415 415 416 417 417 418 421
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 439 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460
462 464 465 465 466 467 468 470 471 474 476 478 479 481 482 483 483 483 485 486 486
487 488 489 490 490 491 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503
503 504 504 505 505 506 506 506 506 506 507 511 512 516 516 517 518 518 521 522 522
524 524 525 525 526 528 528 528 528 528 529 529 529 530 532 532 535 535 536 537 537
537 538 538 538 539 539 540 541 541 542 542 543 545 548 550 551 551 552 552 555 555
556 559 560 560 563 563 566 566 567 567 567 567 568 569 569 570 570 571 573 573 574

574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 590 590 590 593 593
593 595 596 599 600 601 603 604 605 606 606 607 610 610 611 613 613 614 614 618 618
618 619 620 621 621 621 622 622 624 624 625 626 626 627 629 630 631 631 633 636 637
637 638 639 639 640 640 641 641 643 644 647 647 648 649 649 651 651 652 653 657 657
658 659 660 661 661 662 665 667 667 668 669 672 672 673 675 676 677 677 679 681 681
682 682 682 683 683 684 685 685 685 686 688 689 690 690 691 692 693 697 698 699 699
701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 716 717 720 720 721
722 722 723 723 725 726 728 729 729 729 729 730 732 732 732 735 735 736 736 739 740
743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757 759 760 761 761 762
762 763 763 763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 776 776 777
777 782 783 783 784 784 786 787 788 788 789 790 790 791 793 793 793 794 794 795 795
796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814
814 815 818 818 818 819 819 820 821 822 825 826 827 828 828 829 829 830 833 836 839
839 840 841 842 844 846 846 847 848 849 850 850 850 850 851 851 852 853 856 856 856
856 857 857 857 857 858 858 859 859 860 860 862 862 862 865 865 868 868 871 871 872
873 873 873 875 875 877 878 881 882 884 886 886 886 887 888 888 888 890 890 892 892
892 894 895 898 898 898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915
916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930
931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 949 950 950 951
952 954 954 954 955 955 955 956 956 957 958 959 959 961 962 963 964 965 967 969 969
969 970 970 970 971 972 972 973 975 976 977 977 980 981 981 982 982 984 984 985 987
987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
It took me 0.000247402 seconds.

	A	B
1	1200	0.000247402
2	1000	0.000220722
3	800	0.000184788
4	600	0.000161775
5	400	8.3016e-05



We can conclude that the merge is also st affected by increasing the input size.

Quick Sort

Now, we have executed quick sort with an array of size 1000 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000349351 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;

int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {

      if (a[j] < pivot)
      {
         i++;
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
void quick(int a[], int start, int end)
{
   if (start < end)
   {
      int p = partition(a, start, end);
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}
int main() {
   using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();
```

```
    int n=1000;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;

        quick(a, 0, n - 1);
                  for(int i=0;i<n;i++)
        std::cout << a[i] << " ";
       std::cout << std::endl;

      high_resolution_clock::time_point t2 = high_resolution_clock::now();

 duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

 std::cout << "It took me " << time_span.count() << " seconds.";
 std::cout << std::endl;
         return 0;
}
```

0 0 2 2 4 6 8 9 10 11 11 12 15 16 17 17 18 19 21 21 21 21 22 23 25 27 27 28 29 30 30 30
31 32 32 33 34 36 36 36 37 39 39 40 41 42 42 42 43 43 46 47 49 49 50 51 52 53 54 55 57
57 57 58 59 59 59 60 62 63 67 67 69 69 71 71 72 73 74 74 80 81 81 81 81 81 82 82 83 84
84 84 84 86 87 87 88 90 90 90 91 93 94 95 97 97 97 98 99 100 100 104 105 107 109 109
109 112 114 115 116 117 117 120 121 122 122 123 124 124 125 126 126 127 127 127 127
128 130 131 131 133 134 134 135 136 137 139 139 141 142 143 144 147 149 149 150 151
152 152 153 154 154 154 154 155 155 157 157 157 159 159 159 163 167 168 170 170 171
171 172 172 172 172 173 173 176 177 178 179 179 179 181 181 181 183 183 184 188 189
189 189 190 190 190 193 195 195 196 197 197 198 198 199 199 202 202 204 205 205 205
206 207 209 210 211 211 211 214 215 217 217 217 218 218 219 222 223 224 224 225 226
226 227 228 228 228 229 231 231 231 232 232 233 234 234 235 235 236 237 237 238 244
245 248 249 250 253 253 254 255 255 255 258 259 260 261 262 262 263 264 266 266 269 269
269 270 270 270 272 273 274 275 276 278 278 280 280 281 281 282 283 283 284 285 285
286 289 289 289 290 291 291 292 292 292 294 295 296 297 297 298 298 299 299 300 301
301 303 303 304 305 305 306 308 308 309 309 310 311 312 312 313 313 314 314 314 315
317 320 321 321 324 324 324 325 325 326 326 327 328 333 334 334 334 335 335 335 336
336 336 337 338 338 338 339 340 340 340 340 341 342 342 343 343 346 346 347 348 348
350 350 350 351 352 353 354 355 355 355 355 358 358 360 360 361 362 363 363 364 364
364 365 365 367 367 367 368 368 368 368 368 369 370 372 373 375 376 376 377 378 378
379 379 379 379 379 382 382 383 385 386 386 388 390 390 393 393 393 395 396 398 398
399 399 403 403 404 404 404 407 412 412 412 412 413 414 415 415 416 417 417 418 421
421 421 421 422 422 425 426 426 427 428 428 428 429 429 431 432 433 433 434 434 435
436 437 437 438 439 440 441 443 443 444 444 445 445 451 452 456 457 458 459 460 460
462 464 465 465 466 467 468 470 471 474 476 478 479 481 482 483 483 483 485 486 486
487 488 489 490 490 491 491 491 492 492 493 494 497 497 498 499 499 500 500 500 503
503 504 504 505 505 506 506 506 506 506 507 511 512 516 516 517 518 518 521 522 522
524 524 525 525 526 528 528 528 528 528 529 529 529 530 532 532 535 535 536 537 537

537 538 538 538 539 539 540 541 541 542 542 543 545 548 550 551 551 552 552 555 555
556 559 560 560 563 563 566 566 567 567 567 567 568 569 569 570 570 571 573 573 574
574 577 578 579 580 581 582 582 583 584 586 586 587 587 589 590 590 590 590 593 593
593 595 596 599 600 601 603 604 605 606 606 607 610 610 611 613 613 614 614 618 618
618 619 620 621 621 621 622 622 624 624 625 626 626 627 629 630 631 631 633 636 637
637 638 639 639 640 640 641 641 643 644 647 647 648 649 649 651 651 652 653 657 657
658 659 660 661 661 662 665 667 667 668 669 672 672 673 675 676 677 677 679 681 681
682 682 682 683 683 684 685 685 685 686 688 689 690 690 691 692 693 697 698 699 699
701 703 705 708 708 708 709 710 710 711 713 713 713 714 715 715 716 717 720 720 721
722 722 723 723 725 726 728 729 729 729 729 730 732 732 732 735 735 736 736 739 740
743 743 743 744 746 746 747 748 750 753 754 754 754 756 756 757 759 760 761 761 762
762 763 763 763 764 764 768 769 770 770 771 772 773 774 775 776 776 776 776 776 777
777 782 783 783 784 784 786 787 788 788 789 790 790 791 793 793 793 794 794 795 795
796 796 797 797 801 802 804 805 805 805 805 806 808 808 810 811 812 813 813 814 814
814 815 818 818 818 819 819 820 821 822 825 826 827 828 828 829 829 830 833 836 839
839 840 841 842 844 846 846 847 848 849 850 850 850 850 851 851 852 853 856 856 856
856 857 857 857 857 858 858 859 859 860 860 862 862 862 865 865 868 868 871 871 872
873 873 873 875 875 877 878 881 882 884 886 886 886 887 888 888 888 890 890 892 892
892 894 895 898 898 898 899 899 900 902 902 904 904 904 904 907 908 910 911 914 915
916 917 917 918 919 919 920 921 921 924 924 925 925 926 926 927 928 928 929 930 930
931 932 932 933 933 934 936 936 939 940 943 944 945 946 947 949 949 949 950 950 951
952 954 954 954 955 955 955 956 956 957 958 959 959 961 962 963 964 965 967 969 969
969 970 970 970 971 972 972 973 975 976 977 977 980 981 981 982 982 984 984 985 987
987 987 988 989 990 990 991 993 993 994 994 996 996 996 996 996 997 999
It took me 0.000349351 seconds.

Now, we have executed quick sort with an array of size 800 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000233157 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;

int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {

      if (a[j] < pivot)
      {
         i++;
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
void quick(int a[], int start, int end)
{
   if (start < end)
   {
      int p = partition(a, start, end);
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}
int main() {
   using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();
```

```
    int n=800;
  int a[n];
  for(int i=0;i<n;i++)
    a[i]=rand()%1000;

        quick(a, 0, n - 1);
                for(int i=0;i<n;i++)
        std::cout << a[i] << " ";
      std::cout << std::endl;

      high_resolution_clock::time_point t2 = high_resolution_clock::now();

 duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

 std::cout << "It took me " << time_span.count() << " seconds.";
 std::cout << std::endl;
         return 0;
}
```

2 2 4 6 9 11 11 12 17 18 19 21 21 21 22 23 25 27 27 28 29 30 30 30 31 33 34 36 36 36 39
40 41 42 42 42 43 43 46 47 49 49 50 51 54 55 57 57 58 59 59 59 60 62 63 67 69 71 72 73
74 74 80 81 81 81 82 82 84 84 84 87 88 90 90 90 91 93 94 95 97 97 98 100 100 105 107
115 117 117 120 121 123 124 124 125 126 127 127 127 128 130 131 133 134 135 137 139
139 142 143 144 149 150 151 152 153 155 157 157 157 159 159 163 167 168 170 172 172
172 173 176 177 178 179 179 183 188 189 189 190 190 190 193 195 196 197 198 199 202
202 204 205 205 205 206 207 209 210 211 211 214 215 217 217 218 219 222 224 226 226
227 228 228 228 229 231 231 232 232 233 234 234 235 235 236 237 237 238 244 245 248
249 250 253 253 255 255 258 259 260 261 262 262 269 270 270 270 273 274 275 276 278
280 281 282 283 284 285 286 289 289 290 291 291 292 292 292 294 298 299 299 301 301
303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 321 324 324 324 325
326 327 328 333 335 335 336 336 336 338 338 339 340 340 340 340 342 342 343 343 346
346 347 348 348 350 353 355 355 355 358 358 360 362 363 363 364 365 367 367 367 368
368 368 368 369 370 372 373 375 376 377 378 379 379 379 379 379 383 385 386 386 388
390 390 393 395 396 398 399 403 403 404 404 407 412 412 413 414 416 417 417 418 421
421 422 422 425 426 427 428 428 429 429 431 432 433 434 434 435 436 437 438 440 441
443 443 444 444 445 451 452 456 458 459 460 460 462 464 465 466 467 468 470 471 474
478 479 481 483 483 485 486 486 488 490 491 491 492 492 493 497 497 498 499 499 500
500 500 503 503 504 504 505 505 506 507 512 516 516 518 518 521 522 522 524 524 526
528 528 529 529 529 530 532 535 536 537 537 538 538 539 540 541 542 542 543 545 550
551 552 555 556 560 560 563 566 567 567 567 567 568 569 569 570 571 573 573 574 574
578 579 580 581 582 584 586 586 587 589 590 590 593 593 596 599 600 601 604 605 606
606 607 610 611 613 613 614 614 618 618 619 620 621 622 622 624 624 625 626 626 627
629 630 631 633 637 637 639 640 640 641 644 647 647 648 649 651 652 658 659 660 661
661 667 668 669 672 675 676 677 677 681 681 682 682 683 683 684 685 685 686 688 689
690 690 691 692 697 699 699 705 708 708 709 710 710 711 713 713 713 714 715 717 721
723 723 725 726 729 729 729 729 730 732 732 732 736 736 739 740 743 743 743 744 746
746 748 750 753 754 754 754 756 756 757 759 761 762 763 763 763 764 764 768 769 770
770 771 772 773 775 776 776 777 782 783 784 784 786 788 788 789 791 793 793 794 794

795 795 796 796 797 801 802 804 805 805 805 808 808 810 811 812 813 814 814 818 818
818 819 819 822 826 828 829 829 830 836 839 840 841 842 846 846 847 848 849 850 850
851 856 856 856 857 857 857 858 858 859 860 862 862 862 865 865 868 868 871 872 873
873 873 878 881 882 884 886 886 887 888 890 892 894 895 898 898 899 899 900 902 902
904 904 904 904 908 911 914 915 916 917 917 918 919 919 920 921 921 924 924 925 925
926 926 927 928 929 930 932 933 934 936 936 939 940 943 944 945 946 947 949 949 950
950 951 954 954 955 955 956 958 959 961 963 964 965 967 969 970 970 971 972 973 975
977 980 981 981 982 984 984 987 987 988 990 993 993 994 994 996 996 996 996 996
It took me 0.000233157 seconds.

Now, we have executed quick sort with an array of size 600 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000208242 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;

int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {

      if (a[j] < pivot)
      {
         i++;
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
void quick(int a[], int start, int end)
{
   if (start < end)
   {
      int p = partition(a, start, end);
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}
int main() {
   using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=600;
```

```
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;

         quick(a, 0, n - 1);
                  for(int i=0;i<n;i++)
         std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
          return 0;
}
```

```
2 4 9 11 11 12 19 21 22 23 25 27 27 28 29 31 33 34 36 36 39 42 42 42 43 43 46 49 50 51
57 58 59 59 60 62 67 69 71 74 80 81 81 82 84 84 84 87 90 90 91 94 95 97 97 100 100 107
117 117 121 123 124 124 125 127 127 128 131 133 134 135 139 142 143 144 149 150 151
152 155 157 157 159 163 167 168 170 172 172 177 178 179 183 188 189 189 190 190 190
193 197 198 199 202 202 205 205 205 206 209 210 211 214 215 217 219 222 224 226 227
228 228 228 229 232 233 234 235 237 237 238 245 248 249 253 255 255 258 259 260 261
269 270 273 274 275 276 280 281 282 284 285 286 289 289 290 292 292 298 299 299 301
301 303 303 304 305 305 306 309 311 312 312 313 314 315 317 320 321 324 324 326 327
333 335 335 336 336 336 338 339 340 340 340 340 342 343 343 346 348 348 350 353 355
355 355 358 358 360 362 363 364 365 367 367 367 368 368 368 368 369 370 372 373 376
378 379 379 379 383 385 386 386 388 390 393 395 396 399 403 403 404 407 412 412 413
414 416 417 418 421 421 422 422 425 426 428 428 429 429 432 433 434 434 436 437 438
440 441 443 444 444 445 451 452 456 458 459 460 460 464 465 466 467 468 470 471 474
478 479 481 483 488 490 491 492 492 493 497 497 498 499 500 500 500 503 504 504 505
505 506 507 512 518 522 524 524 526 528 528 529 529 529 530 535 537 538 538 539 540
541 542 542 545 550 551 552 555 556 566 567 567 567 567 568 569 570 578 579 581 582
584 586 586 587 590 590 593 599 600 601 605 606 606 607 610 611 613 613 614 618 618
619 620 621 622 624 624 625 626 626 627 629 630 631 637 637 640 641 644 647 647 648
649 651 652 658 659 660 661 667 668 669 672 675 676 677 681 681 682 683 684 686 688
689 690 690 692 697 699 705 708 708 709 713 713 715 721 723 725 726 729 729 729 729
730 732 732 736 736 739 743 743 743 746 746 748 750 753 754 754 754 756 756 757 761
763 763 764 764 769 770 771 772 773 776 776 777 782 783 784 784 786 788 788 793 793
794 795 795 796 797 801 802 804 808 808 810 811 813 814 818 818 819 819 826 828 829
829 840 841 842 846 846 849 850 850 856 856 856 857 857 858 859 860 862 862 865 868
871 872 873 873 873 878 881 884 886 886 887 888 890 892 894 895 898 899 899 900 902
902 904 904 908 914 915 916 917 917 919 920 921 921 924 925 926 926 927 928 929 930
932 933 936 936 939 940 944 945 946 949 954 956 958 959 961 964 965 969 970 971 972
973 975 980 981 984 987 987 988 990 993 994 994 996 996 996 996 996
It took me 0.000208242 seconds.
```

Now, we have executed quick sort with an array of size 400 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 0.000145518 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;

int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {

      if (a[j] < pivot)
      {
         i++;
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
void quick(int a[], int start, int end)
{
   if (start < end)
   {
      int p = partition(a, start, end);
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}
int main() {
   using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();
```

```
    int n=400;
  int a[n];
  for(int i=0;i<n;i++)
     a[i]=rand()%1000;

        quick(a, 0, n - 1);
                for(int i=0;i<n;i++)
        std::cout << a[i] << " ";
       std::cout << std::endl;

      high_resolution_clock::time_point t2 = high_resolution_clock::now();

 duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

 std::cout << "It took me " << time_span.count() << " seconds.";
 std::cout << std::endl;
         return 0;
}
```

```
4 9 11 11 12 19 21 22 27 29 31 34 36 42 42 43 58 59 60 62 67 69 71 81 82 84 84 84 87 90
91 94 95 97 97 107 117 121 123 124 124 127 128 131 135 139 142 143 149 150 155 163
167 170 172 172 178 179 183 190 190 193 197 198 199 202 202 205 206 209 210 211 219
222 224 226 227 228 228 228 229 235 237 245 255 255 258 259 269 270 273 275 276 280
281 282 286 292 298 299 299 301 301 305 305 306 311 313 315 317 320 321 324 324 326
327 335 336 336 336 340 340 340 342 343 348 348 350 353 355 362 364 365 367 368 368
368 369 370 372 373 376 378 379 379 379 383 385 386 393 395 396 399 403 404 407 412
413 416 418 421 421 422 422 426 428 429 432 434 434 437 440 441 443 444 444 445 451
452 456 464 465 466 467 468 470 474 481 488 490 491 492 492 497 499 500 503 504 505
505 506 507 522 524 526 528 529 530 537 538 539 540 542 542 545 550 551 555 567 567
567 568 570 579 581 582 584 586 586 587 590 590 599 600 601 605 606 611 613 613 618
619 621 622 624 624 625 626 630 637 640 644 649 651 652 658 659 660 661 667 668 669
672 675 676 683 684 688 689 690 705 708 708 709 713 713 715 721 723 725 729 729 730
732 732 736 736 739 743 743 746 750 754 754 756 756 763 763 764 764 769 771 772 773
776 776 777 782 784 784 786 788 793 793 795 796 801 802 804 808 810 811 813 814 818
818 819 819 828 829 840 841 842 846 846 850 850 856 856 857 858 859 860 862 862 865
868 871 873 878 881 884 886 887 894 895 898 899 899 902 904 904 914 915 917 917 919
920 921 924 925 926 926 927 928 929 930 932 933 936 936 939 940 944 954 956 959 961
965 972 973 980 981 984 987 993 994 996 996 996
It took me 0.000145518 seconds.
```

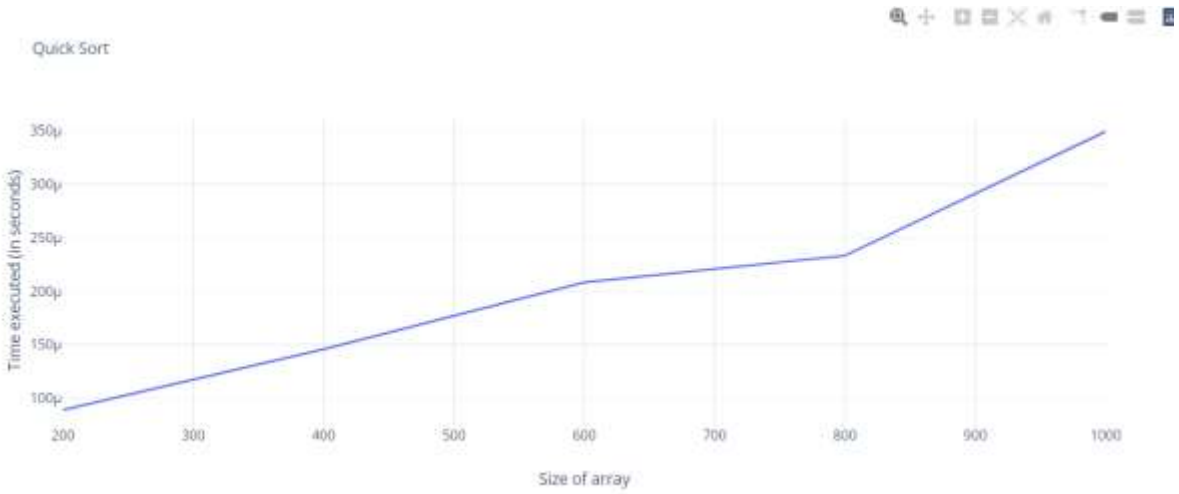
Now, we have executed quick sort with an array of size 200 , by using the rand function we have input the data in the array. Using the inbuilt library chrono we have used high resolution clock we have measured the time of the insertion sort which came out to be 8.869e-05 seconds.

```
#include <iostream>
#include <ctime>
#include <ratio>
#include <chrono>
using namespace std;

int partition (int a[], int start, int end)
{
   int pivot = a[end]; // pivot element
   int i = (start - 1);

   for (int j = start; j <= end - 1; j++)
   {

      if (a[j] < pivot)
      {
         i++;
         int t = a[i];
         a[i] = a[j];
         a[j] = t;
      }
   }
   int t = a[i+1];
   a[i+1] = a[end];
   a[end] = t;
   return (i + 1);
}
void quick(int a[], int start, int end)
{
   if (start < end)
   {
      int p = partition(a, start, end);
      quick(a, start, p - 1);
      quick(a, p + 1, end);
   }
}
int main() {
   using namespace std::chrono;

  high_resolution_clock::time_point t1 = high_resolution_clock::now();



    int n=200;
```

```
   int a[n];
   for(int i=0;i<n;i++)
      a[i]=rand()%1000;

          quick(a, 0, n - 1);
                    for(int i=0;i<n;i++)
          std::cout << a[i] << " ";
        std::cout << std::endl;

        high_resolution_clock::time_point t2 = high_resolution_clock::now();

  duration<double> time_span = duration_cast<duration<double>>(t2 - t1);

  std::cout << "It took me " << time_span.count() << " seconds.";
  std::cout << std::endl;
           return 0;
}
```

11 11 12 19 22 27 29 31 34 42 43 58 59 60 67 69 84 87 91 94 97 97 117 121 123 124 124 135 143 149 167 170 172 178 193 198 211 219 226 227 228 228 229 235 237 245 270 275 276 280 281 286 301 305 306 313 315 317 324 327 335 336 340 350 353 362 364 365 367 368 368 368 370 373 378 379 383 386 393 395 399 403 407 413 421 421 426 428 429 432 434 434 437 440 441 444 451 456 467 474 481 488 491 492 492 497 500 503 505 526 528 529 530 537 539 540 545 551 555 567 567 570 582 584 586 586 601 618 619 624 649 651 652 675 676 683 689 690 708 709 715 723 729 729 732 736 739 743 750 754 756 763 764 764 771 777 782 784 788 793 793 795 796 802 808 814 818 829 841 846 846 856 856 857 858 859 862 862 865 871 873 886 895 902 914 915 919 921 925 926 927 928 929 932 936 956 965 980 987 996
It took me 8.869e-05 seconds.

	A	B
1	1000	0.000349351
2	800	0.000233157
3	600	0.000208242
4	400	0.000145518
5	200	8.869e-05



We can conclude that the Quick sort is also affected by increasing the input size.

Conclusion

Apart from sheel sort , all other algorithm are getting affected by changing the input sie of array.