

Міністерство освіти і науки України
Національний технічний університет України «Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра ІПІ

Звіт

з лабораторної роботи № 1 з дисципліни

«Алгоритми та структури даних 2. Структури даних»

„Проектування і аналіз алгоритмів внутрішнього сортування”

Виконав: ІП-13 Дейнега Владислав Миколайович

Перевірив: Сопов Олексій Олександрович

Київ 2022

Зміст

1. Мета лабораторної роботи

2. Завдання

3. Виконання

3.1. Аналіз алгоритму на відповідність властивостям.

3.2. Псевдокод алгоритму

3.3. Аналіз часової складності

3.4. Програмна реалізація алгоритму

3.4.1. Вихідний код

3.4.2. Приклад роботи

3.5. Тестування алгоритму

3.5.1. Часові характеристики оцінювання

3.5.2. Графіки залежності часових характеристик оцінювання
від розмірності масиву

Висновок

Критерії оцінювання

Мета лабораторної роботи

Мета роботи – вивчити основні методи аналізу обчислювальної складності алгоритмів внутрішнього сортування і оцінити поріг їх ефективності.

Завдання

Виконати аналіз алгоритму внутрішнього сортування на відповідність наступним властивостям (таблиця 2.1):

1. стійкість
2. «природність» поведінки (Adaptability);
3. базуються на порівняннях;
4. необхідність додаткової пам'яті (об'єму);
5. необхідність в знаннях про структуру даних.

Записати алгоритм внутрішнього сортування за допомогою псевдокоду (чи іншого способу по вибору).

Провести аналіз часової складності в гіршому, кращому і середньому випадках та записати часову складність в асимптотичних оцінках.

Виконати програмну реалізацію алгоритму на будь-якій мові програмування з фіксацією часових характеристик оцінювання (кількість порівнянь, кількість перестановок, глибина рекурсивного поглиблення та інше в залежності від алгоритму).

Провести ряд випробувань алгоритму на масивах різної розмірності (10, 100, 1000, 5000, 10000, 20000, 50000 елементів) і різних наборів вхідних даних (впорядкований масив, зворотно упорядкований масив, масив випадкових чисел) і побудувати графіки залежності часових характеристик оцінювання від розмірності масиву, нанести на графік асимптотичну оцінку гіршого і кращого випадків для порівняння.

Зробити порівняльний аналіз двох алгоритмів.

Зробити узагальнений висновок з лабораторної роботи.

Таблиця 2.1 – Варіанти алгоритмів

№	Алгоритм сортування
1	Сортування бульбашкою
2	Сортування гребінцем («розчіскою»)

Виконання

Аналіз алгоритму на відповідність властивостям

Аналіз алгоритму сортування бульбашкою та гребінцем на відповідність властивостям наведено в таблиці

Властивість	Сортування бульбашкою	Сортування гребінцем
Стійкість	Так	Ні
«Природність» поведінки (Adaptability)	Так	ні
Базуються на порівняннях	Так	Так
Необхідність в додатковій пам'яті (об'єм)	Ні	Ні
Необхідність в знаннях про структури даних	Так	Так

Псевдокод алгоритму

Булбашкою:

```
повторити для і від 0 до n-1 з кроком 1
    повторити для j від 0 до n-1 з кроком 1
        якщо mass[j] > mass[j+1]
            temp = mass[j]
            mass[j] = mass[j+1]
            mass[j+1] = temp
        все якщо
    все повторити
все повторити
```

Гребінцем:

функція newGap(int gap)

gap /= 1.247

якщо gap < 1

повернути 1

повернути gap

все функція

функція shellSort(int* mass, int size, unsigned int swapC, unsigned int compC)

gap = size

swap = true

поки gap > 1 **або** swap == true **повторити**

swap = false

gap = newGap(gap)

повторити для i від 0 до size - gap з кроком 1

якщо mass[i] > mass[i+gap]

temp = mass[i]

mass[i] = mass[i + gap]

mass[i + gap] = temp

swap = true

все якщо

все повторити

все повторити

все функція

Аналіз часової складності

	Бульбашкою	Гребінцем
Найкраща швидкодія	$O(n)$	$O(n)$
Середня швидкодія	$O(n^2)$	$\Omega(n \log n)$
Найгірша швидкодія	$O(n^2)$	$\Omega(n^2)$

Програмна реалізація алгоритму

Вихідний код

```
#include <iomanip>
#include <iostream>

using namespace std;

void matr_out(int*, int);
void fillRand(int*, int);
void fillTrue(int*, int);
void fillReverse(int*, int);
void bubbleSort(int*, int, unsigned int, unsigned int);
void shellSort(int* , int, unsigned int, unsigned int);
int newGap(int);

int main()
{
    int size, flag;
    unsigned int swapC = 0, compC = 0;

    cout << "Enter size of arrey:\n";
    cin >> size;

    int* mass = new int[size];

    cout << "How do you want to fill the array?\n 1 - random\n 2 - optimally\n 3 - revers\n";
    cin >> flag;
    switch (flag)
    {
        case 1:
            fillRand(mass, size);
            matr_out(mass, size);
            break;
        case 2:
            fillTrue(mass, size);
            matr_out(mass, size);
            break;
        case 3:
            fillReverse(mass, size);
            matr_out(mass, size);
            break;
        default:
            cout << "You enter invalid number!";
            return 0;
    }

    cout << "How do you want to sort the array?\n 1 - Bubble sort\n 2 - Shell sort\n";
```



```

    cin >> flag;
    switch (flag)
    {
    case 1:
        bubbleSort(mass, size, swapC, compC);
        break;
    case 2:
        shellSort(mass, size, swapC, compC);
        break;
    default:
        cout << "You enter invalid number!";
        return 0;
    }
}

void matr_out(int* mass, int size)
{
    for (int i = 0; i < size; i++)
    {
        cout << mass[i] << " ";
    }
    cout << endl;
}

void fillRand(int* mass, int size)
{
    srand(time(NULL));
    for (int i = 0; i < size; i++)
    {
        mass[i] = rand() % size;
    }
}

void fillTrue(int* mass, int size)
{
    srand(time(NULL));
    for (int i = 0; i < size; i++)
    {
        mass[i] = i+1;
    }
}

void fillReverse(int* mass, int size)
{
    srand(time(NULL));
    for (int i = 0; i < size; i++)
    {
        mass[i] = size - i;
    }
}

```

```

void bubbleSort(int* mass, int size, unsigned int swapC, unsigned int compC)
{
    int temp;
    for (int i = 0; i < size-1; i++)
    {
        for (int j = 0; j < size - 1; j++)
        {
            compC++;
            if (mass[j] > mass[j+1])
            {
                temp = mass[j];
                mass[j] = mass[j + 1];
                mass[j + 1] = temp;
                swapC++;
            }
        }
    }

    matr_out(mass, size);
    cout << "Count of swap = " << swapC << endl;
    cout << "Count of compare = " << compC << endl;
}

```

```

int newGap(int gap)
{
    gap /= 1.247;
    if (gap<1)
    {
        return 1;
    }
    return gap;
}

```

```

void shellSort(int* mass, int size, unsigned int swapC, unsigned int compC)
{
    int gap = size, temp;
    bool swap = true;

    while (gap >1 || swap == true)
    {
        swap = false;
        gap = newGap(gap);

        for (int i = 0; i < size - gap; i++)
        {
            compC++;
            if (mass[i] > mass[i+gap])
            {
                swapC++;
                temp = mass[i];
                mass[i] = mass[i + gap];
            }
        }
    }
}

```

```

        mass[i + gap] = temp;
        swap = true;
    }

}

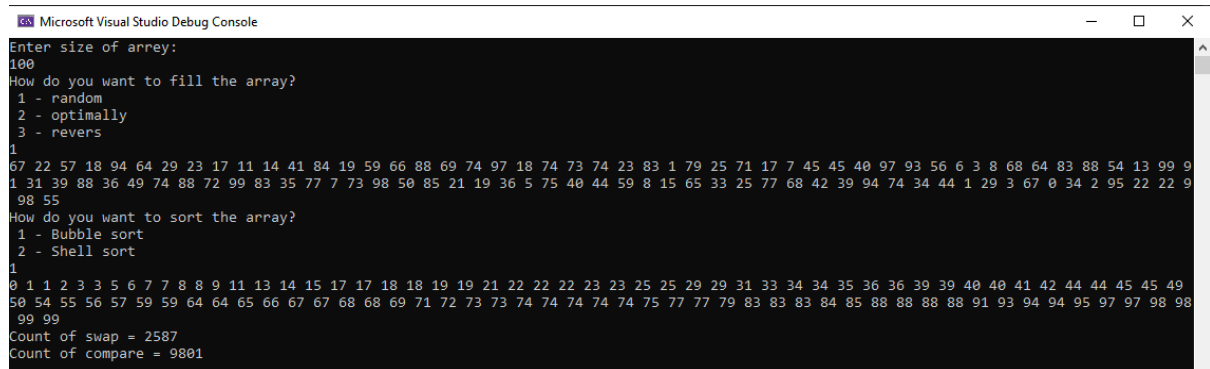
matr_out(mass, size);
cout << "Count of swap = " << swapC << endl;
cout << "Count of compare = " << compC << endl;
}

```

Приклад роботи

Бульбашкою:

Масив на 100 елементів

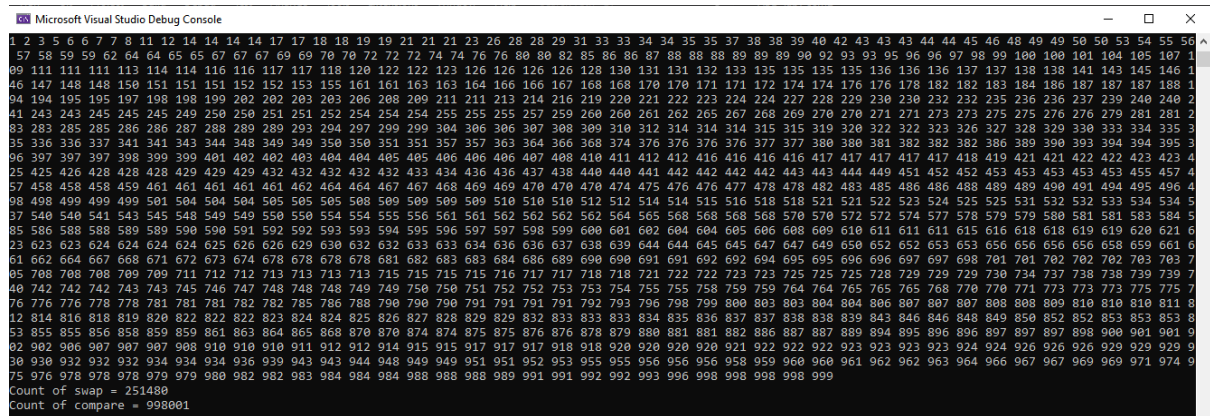


```

Microsoft Visual Studio Debug Console
Enter size of array:
100
How do you want to fill the array?
1 - random
2 - optimally
3 - revers
1
67 22 57 18 94 64 29 23 17 11 14 41 84 19 59 66 88 69 74 97 18 74 73 74 23 83 1 79 25 71 17 7 45 45 40 97 93 56 6 3 8 68 64 83 88 54 13 99 9
1 31 39 88 36 49 74 88 72 99 83 35 77 7 73 98 50 85 21 19 36 5 75 40 44 59 8 15 65 33 25 77 68 42 39 94 74 34 44 1 2 9 3 67 0 34 2 95 22 22 9
98 55
How do you want to sort the array?
1 - Bubble sort
2 - Shell sort
1
0 1 1 2 3 3 5 6 7 7 8 8 9 11 13 14 15 17 17 18 18 19 19 21 22 22 22 23 23 25 25 29 29 31 33 34 34 35 36 36 39 39 40 40 41 42 44 44 45 45 49
50 54 55 56 57 59 59 64 64 65 66 67 67 68 68 69 71 72 73 73 74 74 74 74 75 77 77 79 83 83 83 84 85 88 88 88 88 91 93 94 94 95 97 97 98 98
99 99
Count of swap = 2587
Count of compare = 9801

```

Масив на 1000 елементів



```

Microsoft Visual Studio Debug Console
1 2 3 5 6 7 7 8 11 12 14 14 14 14 14 17 17 18 18 19 19 21 21 21 23 26 28 28 29 31 33 33 34 34 35 35 37 38 38 39 40 42 43 43 43 44 44 45 46 48 49 49 50 50 53 54 55 56
57 58 59 59 62 64 64 65 65 67 67 67 69 69 70 70 72 72 72 74 74 76 76 80 80 82 85 86 86 87 88 88 88 89 89 89 90 92 93 93 95 96 96 97 98 99 100 100 101 104 105 107 1
99 111 111 111 113 114 114 116 116 117 117 118 120 122 122 123 126 126 126 126 128 130 131 131 132 133 135 135 135 135 136 136 136 137 137 138 138 141 143 145 146 1
46 147 148 148 150 151 151 151 152 152 153 155 161 161 163 163 164 166 166 167 168 168 170 170 171 171 172 174 174 176 176 178 182 182 183 184 186 187 187 187 188 1
94 194 195 195 197 198 198 199 202 202 203 203 206 208 209 211 211 213 214 216 219 220 221 222 223 224 224 227 228 229 230 230 232 232 235 236 236 237 239 240 240 2
41 243 243 245 245 249 250 250 251 251 252 254 254 254 255 255 255 257 259 260 260 261 262 265 267 268 269 270 270 271 271 273 273 275 275 276 276 279 281 281 2
83 283 285 285 286 286 287 288 289 289 293 294 297 299 299 304 306 306 307 308 309 310 312 314 314 314 315 315 319 320 322 322 323 326 327 328 329 330 333 334 335 3
35 336 336 337 341 341 343 344 348 349 349 350 351 351 357 357 363 364 366 368 374 376 376 376 377 377 380 380 381 382 382 382 386 389 390 393 394 394 395 3
96 397 397 398 399 399 401 402 402 403 404 404 405 405 406 406 406 408 408 410 411 412 412 416 416 416 416 417 417 417 417 418 419 421 421 422 422 423 423 4
25 425 426 428 428 429 429 429 432 432 432 432 432 433 434 436 436 437 438 440 440 441 442 442 442 442 443 443 444 449 451 452 452 453 453 453 455 457 4
57 458 458 458 459 461 461 461 461 462 464 464 467 467 468 469 469 470 470 470 474 475 476 476 477 477 478 478 482 483 485 486 486 488 489 489 490 491 494 495 496 4
98 498 499 499 499 501 504 504 504 505 505 505 508 508 509 509 509 510 510 510 512 512 514 514 515 516 518 518 521 521 522 523 524 525 525 531 532 532 533 534 534 5
37 540 540 541 543 545 548 549 549 550 550 554 554 555 556 561 561 562 562 562 564 565 568 568 568 568 570 570 572 572 574 577 578 579 579 580 581 581 583 584 5
85 586 588 588 589 589 590 591 592 592 593 593 594 595 596 597 597 598 599 600 601 602 604 604 605 606 608 609 610 611 611 611 615 616 618 618 619 619 620 621 6
23 623 623 624 624 624 624 625 626 626 629 630 632 632 633 633 634 636 636 637 638 639 644 644 645 645 647 647 649 650 652 652 653 653 656 656 656 656 658 659 661 6
61 662 664 667 668 671 672 673 674 678 678 678 681 682 683 683 684 686 689 690 690 691 691 692 692 694 695 695 696 696 697 697 698 701 701 702 702 702 703 703 7
85 708 708 708 709 709 711 712 712 713 713 713 713 715 715 715 715 716 717 717 718 718 721 722 722 723 723 725 725 725 728 729 729 729 730 734 737 738 738 739 739 7
40 742 742 742 743 743 745 746 747 748 748 748 749 749 750 750 751 752 752 753 753 754 755 755 758 759 759 764 764 765 765 765 768 770 770 771 773 773 773 775 775 7
76 776 776 778 778 781 781 782 782 785 786 788 790 790 790 791 791 791 792 793 796 798 799 800 800 803 803 804 804 806 807 807 807 808 808 809 810 810 810 811 8
12 814 816 818 819 820 822 822 822 824 824 825 826 827 828 829 829 832 833 833 833 834 835 836 837 837 838 838 839 843 846 846 848 849 850 852 852 853 853 853 8
53 855 855 856 858 859 859 861 863 864 865 868 870 870 874 874 875 875 876 876 878 879 880 881 881 882 886 887 887 889 894 895 896 896 897 897 897 898 900 901 901 9
92 902 906 907 907 907 908 910 910 910 911 912 912 914 915 915 917 917 917 918 918 920 920 920 920 921 922 922 922 923 923 923 924 924 926 926 926 926 929 929 9
30 930 932 932 932 934 934 936 936 939 943 943 944 948 949 949 951 951 952 953 955 955 956 956 956 958 959 960 960 961 962 962 963 964 966 967 967 969 969 971 974 9
75 976 978 978 978 979 979 980 982 982 983 984 984 984 988 988 988 989 991 991 992 992 993 996 998 998 998 998 999
Count of swap = 251480
Count of compare = 998001

```

Гребінцем:

Масив на 100 елементів

```
Microsoft Visual Studio Debug Console
Enter size of array:
100
How do you want to fill the array?
1 - random
2 - optimally
3 - revers
1
33 41 0 34 15 16 77 83 77 89 9 58 31 65 76 93 3 94 93 1 53 57 73 60 42 50 69 83 5 88 18 95 60 62 28 29 50 74 21 10 33 29
34 12 13 47 15 18 93 33 47 42 94 4 10 57 50 7 78 39 44 40 50 64 48 23 42 10 56 48 92 36 12 86 0 14 85 3 50 89 74 38 70
33 90 35 33 14 62 54 2 64 74 62 88 30 35 35 38 13
How do you want to sort the array?
1 - Bubble sort
2 - Shell sort
2
0 0 1 2 3 4 5 7 9 10 10 10 12 12 12 13 13 14 14 15 15 16 18 18 21 23 28 29 29 30 31 33 33 33 33 33 34 34 35 35 35 36 38 3
8 39 40 41 42 42 44 44 47 47 48 48 50 50 50 50 50 53 54 56 57 57 58 60 60 62 62 62 64 64 65 69 70 73 74 74 74 76 77 77 7
8 83 83 85 86 88 88 89 89 90 92 93 93 93 94 94 95
Count of swap = 211
Count of compare = 1328
```

Масив на 1000 елементів

```
Microsoft Visual Studio Debug Console
0 3 4 5 7 8 10 12 14 18 18 18 19 19 19 20 20 21 21 22 22 23 23 23 24 25 28 28 28 30 32 33 33 33 35 39 40 42 43 43 43 44 45 46 46 47 49 50
52 53 54 55 57 58 59 59 62 62 62 63 63 63 64 66 69 71 73 73 75 77 77 78 79 80 82 82 83 86 87 88 88 92 94 99 99 104 104 105 105 105 106 107
107 107 108 108 109 114 114 115 116 118 120 121 121 122 122 124 124 124 125 125 125 125 125 126 126 126 128 128 130 131 132 136 138 138 139
140 141 141 146 147 147 153 154 154 155 155 156 156 159 159 159 160 160 161 162 163 163 164 164 165 166 167 168 169 170 172 173 173 174 175
176 176 176 177 177 179 180 180 180 180 182 183 184 185 187 187 188 188 189 190 190 190 190 191 191 191 191 193 193 193 193 194 194 197 197
198 199 199 200 202 204 204 205 205 205 207 207 208 208 209 212 212 214 216 218 219 219 220 220 220 222 222 222 224 224 226 226 226 226 227
228 229 229 229 229 230 231 232 232 233 236 237 237 238 238 240 241 242 243 244 245 245 247 248 248 248 248 252 253 253 254 255 256 258 258
259 260 262 263 263 265 268 268 269 272 273 273 278 278 279 281 281 281 282 283 284 285 287 288 289 290 290 291 292 293 294 295 296 297 298
298 298 298 299 299 300 300 301 302 303 303 305 306 307 307 309 313 314 314 314 317 319 319 321 322 322 324 324 325 326 328 328 328 329 331
332 333 333 335 336 337 337 338 342 342 343 344 344 348 348 348 350 350 352 353 353 354 355 355 357 358 359 360 361 364 365 366 369 371 371
371 374 374 375 375 375 377 378 379 380 380 381 383 383 384 385 385 386 386 388 389 390 391 394 398 400 401 401 403 407 408 411 411 411 413
413 414 414 415 417 417 418 419 419 421 421 422 424 426 426 427 429 430 430 433 433 437 437 437 439 439 439 440 440 442 445 445 446 446 447
447 448 448 449 449 449 450 451 453 454 455 455 455 456 457 457 457 463 463 464 464 466 467 467 469 470 471 471 472 473 474 477 478 479 479
480 480 481 485 485 486 486 487 488 491 491 492 492 494 496 496 496 498 499 501 501 501 502 502 503 503 506 506 506 507 507 508 510 511 511 512
512 515 516 516 517 518 518 518 520 520 521 522 525 525 525 525 526 526 528 529 529 529 530 531 533 534 534 534 535 535 536 537 537 538 538
539 539 539 540 541 541 541 543 544 545 545 547 549 549 549 551 552 553 556 556 559 560 560 561 562 562 565 566 567 569 570 570 571 571 572
573 574 576 577 578 578 578 579 582 582 582 583 584 584 585 586 588 589 591 591 591 595 595 597 597 597 598 599 600 600 601 602 604 604 604 605
605 606 606 606 607 607 608 608 608 609 610 610 610 610 611 612 614 615 616 616 617 618 618 618 618 621 622 622 623 625 627 627 628 629 630
631 631 633 633 634 635 636 642 643 645 647 654 656 656 657 658 658 659 660 661 662 662 662 664 664 665 666 668 668 669 669 672 673 674 674
676 678 678 679 683 683 684 685 686 687 687 688 689 689 689 689 690 690 690 691 691 691 691 693 694 694 698 699 699 704 704 707 707 711 711
712 713 713 713 714 715 716 717 717 717 717 718 719 719 721 722 724 725 726 727 729 730 731 733 733 737 737 739 740 741 741 741 741 743 744
746 746 746 747 747 750 753 755 756 757 759 760 760 762 762 764 766 766 767 768 768 768 770 771 772 773 774 774 774 775 775 775 775 778
779 781 782 783 785 786 787 789 790 790 791 792 792 793 797 799 799 800 801 801 804 806 806 807 813 813 813 815 817 817 818 818 818 818 819
822 822 823 824 824 824 824 825 829 831 831 832 832 832 834 834 835 838 839 839 842 843 845 845 845 847 847 848 851 851 851 854 855 856 857 858
859 859 860 862 863 865 866 867 867 871 871 871 873 873 875 877 878 878 879 881 883 885 891 893 893 894 894 895 898 898 900 900 901 903 903
904 904 904 905 906 906 907 907 908 909 910 910 911 911 915 916 916 917 917 918 919 919 922 923 924 926 928 928 929 931 931 932 932 933 933
934 936 937 937 938 939 939 940 941 941 942 942 943 943 944 946 948 948 949 949 950 951 951 952 952 952 956 960 960 960 961 961 965 966 966
970 972 973 974 975 975 978 980 981 983 983 984 984 985 985 986 986 986 987 988 990 992 993 993 993 995 996 997 998 998 999
```

Тестування

Часові характеристики оцінювання

Характеристики оцінювання алгоритму сортування для упорядкованої послідовності елементів у масиві

Бульбаешкою:

Розмірність масиву	Число порівнянь	Число перестановок
10	81	0
100	9801	0

1000	998001	0
5000	24990001	0
10000	99980001	0
20000	399960001	0
50000	2499900001	0

Гребінцем:

Розмірність масиву	Число порівнянь	Число перестановок
10	36	0
100	1229	0
1000	22022	0
5000	144832	0
10000	329598	0
20000	719136	0
50000	1997680	0

Характеристики оцінювання алгоритму сортування для *зворотно упорядкованої послідовності елементів у масиві*

Бульбашкою:

Розмірність масиву	Число порівнянь	Число перестановок
10	81	45
100	9801	4950
1000	998001	499500
5000	24990001	12497500
10000	99980001	49995000
20000	399960001	199990000
50000	2499900001	1249975000

Гребінцем:

Розмірність масиву	Число порівнянь	Число перестановок
10	45	9
100	1328	110

1000	23021	1512
5000	149831	9154
10000	339597	19018
20000	739135	40730
50000	2047679	110332

Характеристика оцінювання алгоритму сортування для *випадкової послідовності елементів у масиві*

Бульбашкою:

Розмірність масиву	Число порівнянь	Число перестановок
10	81	23
100	9801	2678
1000	998001	240907
5000	24990001	6232038
10000	99980001	25100101
20000	399960001	99432057

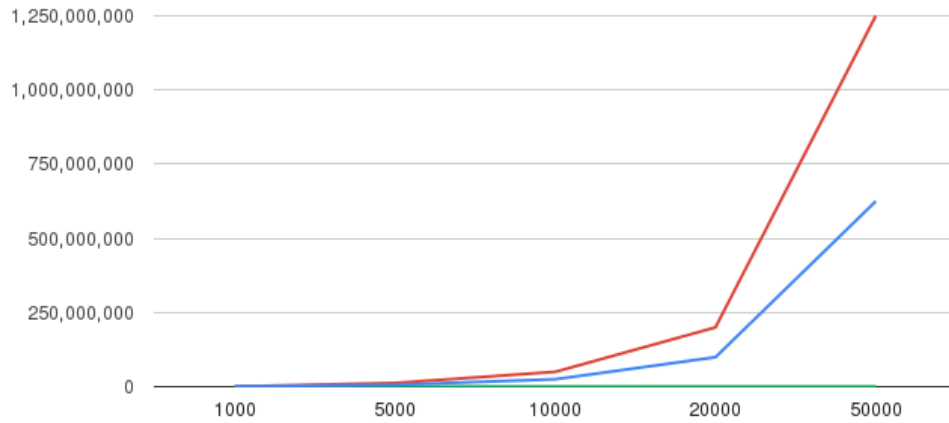
50000	2499900001	624946360
-------	------------	-----------

Гребінцем:

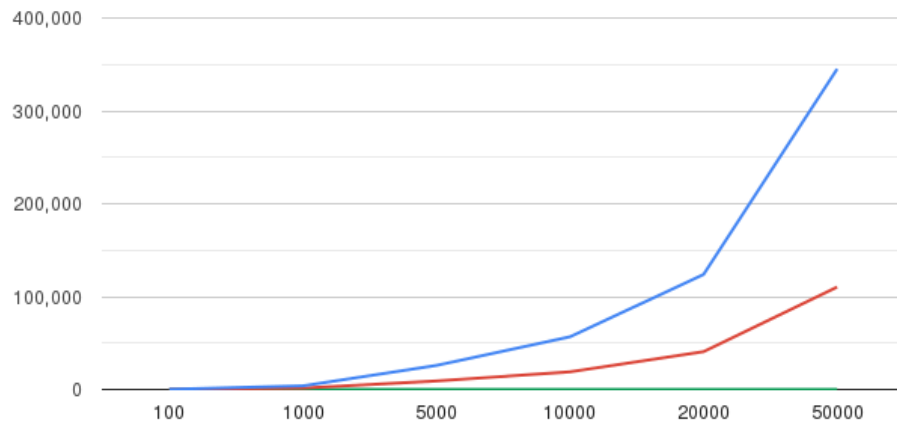
Розмірність масиву	Число порівнянь	Число перестановок
10	45	5
100	1328	215
1000	23021	3921
5000	154830	25922
10000	349596	56758
20000	759134	123878
50000	2097678	345125

Графіки залежності часових характеристик оцінювання

Бульбашка



Гребінцем



Висновок

Під час виконання лабораторної роботи я вивчив основні методи аналізу обчислювальної складності алгоритмів сортування бульбашкою та гребінцем і оцінив поріг їх ефективності.