Сотриворки

Сортиворки

 Сортировки

Цель: выяснить, за какое время разные сортировки сортируют разные массивы.

Сегодня у нас в гостях: bubble sort, selection sort, insertion sort и гвоздь программы – counting sort.

Мега главная таблица:

|  |  |  |  |
| --- | --- | --- | --- |
| t в мсек  | $$10^{3}$$ | $$10^{4}$$ | $$10^{5}$$ |
| Отсорт | 3 | 275 | 26642 |
| Обрат | 3 | 488 | 48786 |
| Random | 3 | 525 | 62087 |
| Отсорт | 2 | 265 | 25962 |
| Обрат | 2 | 256 | 25369 |
| Random | 3 | 261 | 25997 |
| Отсорт | 0 | 0 | 0 |
| Обрат | 7 | 470 | 50364 |
| Random | 4 | 232 | 23472 |

Голубой - bubble sort;

Жёлтый - selection sort;

Коричневый - insertion sort.

И ещё одна табличка для counting sort:

Вот это быстро! В чём подвох?! Да в том, что требуется куча памяти, а массив {1, 10000000000} вообще не отсортирует: потребуется создавать массив на кучу памяти.

Интересно, что insertion sort на отсортированном массиве всё делает моментально, на случайном обычно чуть быстрей других, а вот на обратном –капец как долго.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Maxэл. | $$10^{3}$$ | $$10^{4}$$ | $$10^{5}$$ | $$10^{6}$$ |
| $$10^{3}$$ | 0 | 0 | 3 | 15 |
| $$10^{4}$$ | 1 | 1 | 2 | 14 |
| $$10^{5}$$ | 1 | 1 | 3 | 18 |
| $$10^{6}$$ | 10 | 10 | 14 | 43 |

А вот и красивая диаграмма:

~~Вывод: если в массиве небольшие целые числа, а у вас есть память, используйте counting sort. Если массив почти отсортирован, используйте insertion sort. Если массив непонятный – selection sort, если важно сохранить порядок одинаковых элементов – bubble sort.~~

Вывод: используйте quicksort.

void csort(int a[], int n)

{

    int A = a[0];

    for (int i = 0; i < n; i ++)

    {

        if (a[i] > A)

        {

            A = a[i];

        }

    }

    Int c[A];

    for (int i = 0; i < A + 1; i ++)

    {

        c[i] = 0;

    }

    for (int i = 0; i < n; i ++)

    {

        c[a[i]] ++;

    }

    int t = 0;

    for (int i = 0; i < A + 1; i ++)

    {

        for (int j = 0; j < c[i]; j ++)

        {

            a[t] = i;

            t ++;

        }

    }

}

K

k

void ssort(int a[], int n)

{

    for (int i = 0; i < n - 1; i ++)

    {

        int M = i;

        for (int j = i + 1; j < n; j ++)

        {

            if (a[j] < a[M])

            {

                M = j;

            }

        }

        int A = a[i];

        a[i] = a[M];

        a[M] = A;

    }

}

Ну, а на последок тексты программ:

void isort(int a[], int n)

{

    for (int i = 1; i < n; i ++)

    {

        int j = i - 1;

        while (a[j] > a[j + 1] && j >= 0)

        {

            int Y = a[j];

            a[j] = a[j + 1];

            a[j + 1] = Y;

            j --;

        }

    }

}

void bsort(int a[], int n)

{

    for (int i = 0; i < n - 1; i ++)

     {

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

         {

             if (a[j] > a[j + 1])

             {

                 int M = a[j];

                 a[j] = a[j + 1];

                 a[j + 1] = M;

             }

         }

     }

}