

9-3-2019

Métodos de ordenamiento recursivos



CUCEI

david gutierrez alvarez
Estructura de datos I

RESUMEN PERSONAL Y FORMA DE ABORDAR EL PROBLEMA

En esta tarea lo que se me complico fue el echo de trabajar con números aleatorios, al principio no se generaban tan aleatorio y se repetina bastante hasta que investigando encotre lo que me ayudo a solucionar este error y pude continuar, otro error que tuve fue que se congelaba en el método de ordenamiento merge, según lo que investigue era un desbordamiento de datos, pero lamentablemente no lo pude solucionar por lo que se que no tendre todos los puntos de esta actividad :’c

Esta actividad me agrado y me enfado bastante, empecé a encontrar error que nunca me habían pasado y fui conociendo nuevas funciones de las bibliotecas :3

Main.cpp

```
#include <iostream>
#include "menu.h"

using namespace std;

int main() {
    Menu menu;
    menu.init();

    return 0;
}
```

Menu.h

```
#ifndef MENU_H
#define MENU_H

#include <iostream>
#include "Rand.h"
#include "order.h"
#include "order.cpp"
#include <ctime>

class Menu {
public:
    Menu();

    void init();
};

#endif // MENU_H
```

Menu.cpp

```
#include "menu.h"

using namespace std;

Menu::Menu() { }

void Menu::init() {
    char option;
    long start, finish;

    do{
        cout << "\t\t\t\t\t.:ORDENAMINETO DE DATOS ALEATORIOS:." << endl
             << "1.Ordenamiento Burbuja" << endl
             << "2.Ordenamiento Seleccion" << endl
             << "3.Ordenamiento Insercion" << endl
             << "4.Ordenamiento Shell" << endl
             << "5.Ordenamiento MergeSort "<< endl
             << "6.Ordenamiento QuickSort" << endl
             << "7.Salir" << endl
             <<"Elige una opcion : ";
        cin >> option;

        Order<Rand, 100000> numeros;/*genero los numeros aleatorios*/
        start = clock();/*inicializo el conteo del tiempo*/

        switch (option) {
            case '1': numeros.bubble();
            break;

            case '2': numeros.select();
```

```

        break;

        case '3': numeros.insert();
        break;

        case '4': numeros.shell();
        break;

        case '5': numeros.merge();
        break;

        case '6': numeros.quick();
        break;

        case '7': cout << "Fin del programa." << endl;
        break;

        default: cout << "introduce un dato valido" << endl;
    }

    if(option > '0' and option < '7') { /*asi solo me arroja el tiempo cuando uso
algun metodo de ordenamiento*/
        finish = clock(); /*despues de terminar el ordenamiento cuenta el tiempo de
nuevo*/

        numeros.print(); /*imprime los numero*/
        double time = (double(finish-start)/CLOCKS_PER_SEC);
        cout << "el tiempo consumido en segundos es: " << time << endl;
    }
} while (option != '7');
}

```

Order.h

```

#ifndef ORDER_H
#define ORDER_H

#include <iostream>

template< class Type, const unsigned long int MAX>
class Order {
private:
    Type data[MAX];
    void merge(unsigned long, unsigned long);
    void quick(unsigned long, unsigned long);

public:
    Order();

    void sort(Type &, Type &); //ordena dos datos
    void bubble();
    void shell();
    void select();
    void insert();
    void merge();
    void quick();
    void print();
    int index();
};

#endif // ORDER_H

```

Order.cpp

```

#include "order.h"

```

```

template<class Type, const unsigned long int MAX>
Order<Type, MAX>::Order() { }

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::sort(Type &a, Type &b) {
    Type aux;
    aux = a;
    a = b;
    b = aux;
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::bubble() {
    int i = MAX, j;
    bool band;

    do{
        band = false;
        j=0;

        while(j < i) {
            if(data[j] > data[j+1]) {
                sort(data[j], data[j+1]);
                band = true;
            }
            j++;
        }
        i--;
    }while(band);
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::shell() {
    float fact = 3.0/4.0;
    unsigned long dif = MAX*fact;

    while(dif > 0){
        for(unsigned long i=0; i <= MAX-dif; i++) {
            if(data[i] > data[i+dif]) {
                sort(data[i], data[i+dif]);
            }
        }
        dif*=fact;
    }
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::select() {
    unsigned long i = 0, j, menor;

    while(i < MAX-1) {
        menor = i;
        j = i+1;
        while(j < MAX) {
            if(data[j] < data[menor]){
                menor = j;
            }
            j++;
        }
        if(menor != i) {
            sort(data[i], data[menor]);
        }
        i++;
    }
}

```

```

    }
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::insert() {
    unsigned long i = 1, j;
    Type element;

    while(i < MAX) {
        element = data[i];
        j = i;
        while(j > 0 and element < data[j-1]) {
            data[j] = data[j-1];
            j--;
        }
        if(i != j) {
            data[j] = element;
        }
        i++;
    }
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::merge() {
    merge(0, MAX);
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::merge(unsigned long left, unsigned long right) {
    if(left >= right){
        return;
    }

    Type temp[MAX];
    for(unsigned long i(left); i <= right; i++){
        temp[i] = data[i];
    }

    unsigned long medio((left+right)/2);

    merge(left, medio);
    merge(medio+1, right);

    unsigned long i(left), j(medio+1), x(left);

    while(i<=medio and j<=right){
        while(i<=medio and temp[i] <= temp[j]){
            data[x++] = temp[i++];
        }
        if(i<=medio){
            while(j<=right and temp[j]<=temp[i]){
                data[x++] = temp[j++];
            }
        }
    }
    while(i<=medio){
        data[x++] = temp[i++];
    }
    while(j<=right){
        data[x++] = temp[j++];
    }
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::quick() {
    quick(0, MAX);
}

```

```

}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::quick(unsigned long left, unsigned long right) {
    if(left>=right){
        return;
    }

    unsigned long i(left), j(right);
    unsigned long pivote((left+right)/2);
    sort(data[pivote], data[right]);

    while(i < j) {
        while(i < j and data[i] <= data[right]) {
            i++;
        }
        while(i < j and data[j] >= data[right]) {
            j--;
        }
        if(i != j) {
            sort(data[i], data[j]);
        }
    }

    if(i != right) {
        sort(data[i], data[right]);
    }
    quick(left, i-1);
    quick(i+1, right);
}

template<class Type, const unsigned long int MAX>
void Order<Type, MAX>::print() {
    for(unsigned long i(0); i < MAX; i++) {
        std::cout<< data[i] << " ";
    }
}

template<class Type, const unsigned long int MAX>
int Order<Type, MAX>::index() {
    return MAX;
}

```

Rand.h

```

#ifndef DATA_H
#define DATA_H

#include <iostream>

class Rand {
private:
    long value;

public:
    Rand();
    Rand(const Rand &);

    long redo();

    Rand operator = (const Rand &elem);
    bool operator == (const Rand &elem) const;
    bool operator != (const Rand &elem) const;
    bool operator < (const Rand &elem) const;
    bool operator > (const Rand &elem) const;

```

```

    bool operator <= (const Rand &elem) const;
    bool operator >= (const Rand &elem) const;
    //Serializar el objeto
    friend std::ostream &operator << (std::ostream& os, Rand& elem);
};

#endif // DATA_H

```

Rand.cpp

```

#include "Rand.h"
#include <random>
#include <chrono>

Rand::Rand() : value(redo()) { }

Rand::Rand(const Rand &copy) : value(copy.value) { }

long Rand::redo() {
    std::default_random_engine
engine{std::chrono::steady_clock::now().time_since_epoch().count()};
    std::uniform_int_distribution<int> range{0, 1000000};

    long random_generated = range(engine);
    return random_generated;
}

Rand Rand::operator =(const Rand &element) {
    value = element.value;
    return *this;
}

bool Rand::operator ==(const Rand &element) const {
    return this->value ==element.value;
}

bool Rand::operator !=(const Rand &element) const {
    return this->value !=element.value;
}

bool Rand::operator <=(const Rand &element) const {
    return this->value <=element.value;
}

bool Rand::operator >=(const Rand &element) const {
    return this->value >=element.value;
}

bool Rand::operator <(const Rand &element) const {
    return this->value < element.value;
}

bool Rand::operator >(const Rand &element) const {
    return this->value > element.value;
}

std::ostream& operator << (std::ostream& os, Rand &element){
    os << element.value << " ";
    return os;
}

```


CAPTURAS DE PANTALLA

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
1 996821 996843 996843 996843 996850 996850 996880 996880 996917 996917 996917 996939 996939 996976 99697
6 996998 996998 997035 997035 997035 997065 997094 997094 997094 997123 997123 997123 997160 997160 99718
2 997182 997182 997189 997189 997189 997219 997219 997241 997241 997256 997256 997256 997278 997278 99731
5 997315 997337 997337 997374 997374 997374 997396 997396 997396 997433 997433 997433 997462 997462 99746
2 997499 997499 997499 997521 997521 997521 997558 997558 997580 997580 997580 997595 997595 997595 99761
7 997617 997639 997639 997654 997654 997654 997676 997676 997676 997713 997713 997735 997735 997735 99777
2 997772 997802 997802 997802 997831 997831 997831 997860 997860 997860 997897 997897 997897 997919 99791
9 997919 997956 997956 997956 997978 997978 997978 997993 997993 997993 998015 998015 998037 998037 99805
2 998052 998052 998074 998074 998074 998111 998111 998141 998141 998141 998170 998170 998199 998199 99819
9 998236 998236 998258 998258 998258 998295 998295 998317 998317 998354 998354 998376 998376 998376 99837
6 998413 998413 998413 998443 998443 998450 998450 998472 998472 998509 998509 998509 998538 998538 99853
8 998538 998575 998575 998575 998597 998597 998597 998634 998634 998634 998656 998656 998656 998693 99869
3 998693 998715 998715 998752 998752 998752 998782 998782 998782 998811 998811 998811 998841 998841 99884
8 998848 998848 998878 998878 998878 998899 998899 998907 998907 998936 998936 998936 998973 998973 99899
5 998995 998995 999032 999032 999054 999054 999054 999091 999091 999113 999113 999113 999150 999150 99918
0 999180 999180 999217 999217 999239 999239 999246 999246 999275 999275 999275 999297 999297 999297 99931
2 999312 999312 999334 999334 999334 999371 999371 999393 999393 999430 999430 999430 999452 999452 99945
2 999489 999489 999519 999519 999519 999548 999548 999548 999578 999578 999578 999615 999615 999615 99963
6 999636 999651 999651 999651 999673 999673 999695 999695 999710 999710 999710 999732 999732 999769 99976
9 999791 999791 999791 999828 999828 999828 999858 999858 999858 999887 999887 999887 999917 999917 99995
4 999954 999954 999975 999975 999975 el tiempo consumido en segundos es: 562.622
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Selecccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion :
```

el metodo burbuja tarda cas 6 minutos

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
767 996804 996804 996834 996834 996834 996863 996863 996863 996893 996893 996930 996930 996952 996952 996
952 996989 996989 996989 997010 997010 997010 997047 997047 997047 997069 997069 997084 997084 997106 997
106 997106 997136 997136 997143 997143 997143 997165 997165 997165 997202 997202 997232 997232 997232 997
269 997269 997269 997291 997291 997328 997328 997349 997349 997349 997386 997386 997408 997408 997445 997
445 997445 997475 997475 997482 997482 997504 997504 997504 997534 997534 997541 997541 997541 997541 997
571 997571 997600 997600 997600 997630 997630 997667 997667 997667 997689 997689 997689 997725 997725 997
725 997747 997747 997747 997784 997784 997814 997814 997814 997843 997843 997873 997873 997880 997880 997
910 997910 997910 997932 997932 997939 997939 997969 997969 997969 998006 998006 998006 998028 998028 998
065 998065 998065 998086 998086 998123 998123 998123 998145 998145 998145 998182 998182 998182 998212 998
212 998241 998241 998241 998271 998271 998278 998278 998278 998308 998308 998308 998330 998330 998330 998
345 998345 998345 998367 998367 998404 998404 998404 998426 998426 998426 998462 998462 998462 998484 998
484 998484 998521 998521 998521 998551 998551 998551 998580 998580 998610 998610 998647 998647 998669 998
669 998676 998676 998706 998706 998706 998728 998728 998728 998743 998743 998743 998765 998765 998765 998
802 998802 998823 998823 998823 998860 998860 998890 998890 998919 998919 998949 998949 998986 998986 999
008 999008 999008 999045 999045 999067 999067 999082 999082 999104 999104 999104 999126 999126 999126 999
126 999141 999141 999141 999162 999162 999199 999199 999199 999221 999221 999258 999258 999258 999288 999
288 999317 999317 999317 999347 999347 999347 999384 999384 999384 999406 999406 999443 999443 999443 999
465 999465 999480 999480 999502 999502 999531 999531 999538 999538 999538 999560 999560 999560 999597 999
597 999597 999627 999627 999656 999656 999686 999686 999723 999723 999723 999745 999745 999745 999782 999
782 999782 999804 999804 999804 999841 999841 999863 999863 999863 999878 999878 999899 999899 999929 999
929 999936 999936 999966 999966 999966 999995 999995 999995 el tiempo consumido en segundos es: 26.841
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Selecccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion :
```

El método de selección tarda un poco menos de medio minuto

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe

46 996946 996968 996968 996968 996968 997005 997005 997005 997005 997027 997027 997027 997027 997064 997064
64 997064 997086 997086 997086 997123 997123 997123 997189 997189 997189 997248 997248 997248 997248 997248
85 997285 997285 997307 997307 997307 997307 997344 997344 997344 997366 997366 997366 997366 997403 997403
03 997403 997425 997425 997425 997425 997462 997462 997462 997491 997491 997491 997520 997520 997520 997520
50 997550 997550 997550 997587 997587 997587 997646 997646 997646 997683 997683 997683 997683 997705 997705
05 997705 997742 997742 997742 997742 997764 997764 997764 997764 997801 997801 997801 997830 997830 997830
30 997860 997860 997860 997889 997889 997889 997889 997926 997926 997926 997948 997948 997948 997948 997948
85 997985 997985 998044 998044 998044 998081 998081 998081 998103 998103 998103 998140 998140 998140 998140
40 998162 998162 998162 998199 998199 998199 998228 998228 998228 998228 998265 998265 998265 998265 998287
87 998287 998324 998324 998324 998324 998346 998346 998346 998383 998383 998383 998383 998442 998442 998442
42 998479 998479 998479 998501 998501 998501 998501 998538 998538 998538 998567 998567 998567 998567 998567
97 998597 998597 998626 998626 998626 998626 998663 998663 998663 998663 998685 998685 998685 998722 998722
22 998722 998744 998744 998744 998781 998781 998781 998840 998840 998840 998840 998877 998877 998877 998877
06 998906 998906 998936 998936 998936 998965 998965 998965 998965 999002 999002 999002 999024 999024 999024
24 999061 999061 999061 999061 999061 999083 999083 999083 999120 999120 999120 999142 999142 999142 999142
79 999179 999179 999179 999238 999238 999238 999275 999275 999275 999304 999304 999304 999341 999341 999341
41 999363 999363 999363 999400 999400 999400 999422 999422 999422 999459 999459 999459 999481 999481 999481
81 999518 999518 999518 999518 999548 999548 999548 999548 999577 999577 999577 999673 999673 999673 999673
73 999702 999702 999702 999702 999739 999739 999739 999739 999761 999761 999761 999798 999798 999798 999798
98 999820 999820 999820 999857 999857 999857 999879 999879 999879 999879 999916 999916 999916 999946 999946
46 999982 999982 999982 el tiempo consumido en segundos es: 38.633
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Selecccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion :
```

Inserccion tarda mas de medio minuto

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe

996982 996982 997019 997040 997040 997040 997048 997048 997077 997077 997077 997099 997099 997099 997114
997114 997136 997136 997158 997158 997173 997173 997173 997195 997195 997195 997232 997232 997232 997254
997254 997254 997291 997291 997321 997321 997321 997350 997350 997350 997379 997379 997379 997416 997438
997438 997453 997453 997453 997475 997475 997497 997497 997497 997512 997512 997512 997534 997534 997556
997556 997556 997571 997571 997593 997593 997593 997630 997630 997660 997660 997660 997660 997689 997689
997719 997719 997755 997755 997755 997777 997777 997777 997814 997814 997836 997836 997836 997851 997851
997851 997873 997873 997873 997895 997895 997910 997910 997932 997932 997962 997962 997962 997962 997969
997969 997969 997991 997991 998028 998028 998028 998058 998058 998095 998095 998116 998116 998116 998153
998153 998153 998175 998175 998175 998212 998234 998234 998234 998271 998271 998301 998301 998301 998308
998308 998308 998330 998330 998330 998360 998360 998360 998367 998367 998397 998397 998426 998426 998426
998456 998456 998456 998492 998492 998492 998514 998514 998514 998551 998551 998573 998573 998573 998610
998632 998632 998632 998669 998669 998699 998699 998706 998706 998706 998736 998736 998736 998758 998758
998758 998765 998765 998765 998795 998795 998795 998832 998832 998853 998853 998853 998890 998890 998890
998912 998912 998912 998949 998949 998949 998971 998971 998971 999008 999008 999038 999038 999067 999067
999097 999097 999097 999104 999104 999104 999134 999134 999156 999156 999156 999171 999171 999171 999192
999192 999192 999229 999229 999229 999251 999251 999251 999288 999288 999310 999310 999310 999347 999377
999377 999377 999406 999406 999436 999436 999436 999473 999473 999495 999495 999495 999502 999502 999502
999532 999532 999532 999553 999553 999568 999568 999590 999590 999612 999612 999627 999627 999627 999649
999649 999649 999686 999686 999686 999708 999708 999708 999745 999745 999775 999775 999812 999812 999834
999834 999834 999871 999871 999893 999893 999893 999908 999908 999929 999929 999929 999951 999951 999951
999966 999966 999966 999988 999988 999988 el tiempo consumido en segundos es: 0.16
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Selecccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion :
```

Shell lo a echo en menos de un segundo

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
954 997054 997061 997061 997061 997091 997091 997091 997128 997128 997150 997150 997150 997187 997187 997
187 997209 997209 997246 997246 997246 997268 997268 997268 997304 997304 997304 997334 997334 997334 997
363 997363 997363 997393 997393 997393 997400 997400 997430 997430 997452 997452 997459 997459 997459 997
489 997489 997489 997511 997511 997511 997526 997526 997526 997548 997548 997548 997585 997585 997585 997
607 997607 997607 997644 997644 997644 997665 997665 997665 997702 997702 997702 997732 997732 997732 997
769 997769 997791 997791 997828 997828 997850 997850 997865 997865 997887 997887 997887 997909 997909 997
909 997924 997924 997924 997946 997946 997946 997983 997983 997983 998004 998004 998004 998041 998041 998
071 998071 998071 998100 998100 998100 998130 998130 998130 998167 998167 998167 998189 998189 998226 998
226 998248 998248 998248 998263 998263 998263 998285 998285 998285 998307 998307 998307 998322 998322 998
344 998344 998344 998380 998380 998380 998410 998410 998410 998439 998439 998439 998469 998469 998469 998
506 998506 998528 998528 998528 998565 998565 998565 998587 998587 998587 998624 998624 998624 998646 998646 998
646 998661 998661 998661 998683 998683 998712 998712 998712 998720 998720 998720 998741 998741 998741 998
778 998778 998808 998808 998808 998845 998845 998867 998867 998867 998904 998904 998904 998926 998926 998
963 998963 998985 998985 998985 999022 999051 999051 999051 999059 999059 999081 999081 999081 999110 999
110 999110 999117 999117 999117 999147 999147 999176 999176 999176 999206 999206 999243 999265 999243 999
265 999265 999302 999302 999324 999324 999361 999361 999361 999383 999383 999383 999420 999449 999
449 999449 999457 999457 999457 999486 999486 999486 999508 999508 999508 999515 999515 999515 999545 999
545 999582 999582 999582 999604 999604 999604 999641 999641 999641 999663 999663 999663 999700 999700 999
700 999722 999722 999722 999759 999759 999759 999788 999788 999817 999817 999817 999847 999847 999854 999
854 999884 999884 999906 999906 999906 999921 999921 999921 999943 999943 999943 999980 999980 999980 e1
tiempo consumido en segundos es: 0.151
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Seleccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion :
```

quickSort duro un poco menos

```
C:\Qt\Qt5.12.0\Tools\QtCreator\bin\qtcreator_process_stub.exe
187 997209 997209 997246 997246 997246 997268 997268 997268 997304 997304 997304 997334 997334 997334 997
363 997363 997363 997393 997393 997393 997400 997400 997430 997430 997452 997452 997459 997459 997459 997
489 997489 997489 997511 997511 997511 997526 997526 997526 997548 997548 997548 997585 997585 997585 997
607 997607 997607 997644 997644 997644 997665 997665 997665 997702 997702 997702 997732 997732 997732 997
769 997769 997791 997791 997828 997828 997850 997850 997865 997865 997887 997887 997887 997909 997909 997
909 997924 997924 997924 997946 997946 997946 997983 997983 997983 998004 998004 998004 998041 998041 998
071 998071 998071 998100 998100 998100 998130 998130 998130 998167 998167 998167 998189 998189 998226 998
226 998248 998248 998248 998263 998263 998263 998285 998285 998285 998307 998307 998307 998322 998322 998
344 998344 998344 998380 998380 998380 998410 998410 998410 998439 998439 998439 998469 998469 998469 998
506 998506 998528 998528 998528 998565 998565 998565 998587 998587 998587 998624 998624 998624 998646 998646 998
646 998661 998661 998661 998683 998683 998712 998712 998712 998720 998720 998720 998741 998741 998741 998
778 998778 998808 998808 998808 998845 998845 998867 998867 998867 998904 998904 998904 998926 998926 998
963 998963 998985 998985 998985 999022 999051 999051 999051 999059 999059 999081 999081 999081 999110 999
110 999110 999117 999117 999117 999147 999147 999176 999176 999176 999206 999206 999243 999265 999243 999
265 999265 999302 999302 999324 999324 999361 999361 999361 999383 999383 999383 999420 999449 999
449 999449 999457 999457 999457 999486 999486 999486 999508 999508 999508 999515 999515 999515 999545 999
545 999582 999582 999582 999604 999604 999604 999641 999641 999641 999663 999663 999663 999700 999700 999
700 999722 999722 999722 999759 999759 999759 999788 999788 999817 999817 999817 999847 999847 999854 999
854 999884 999884 999906 999906 999906 999921 999921 999921 999943 999943 999943 999980 999980 999980 e1
tiempo consumido en segundos es: 0.151
.:ORDENAMINETO DE DATOS ALEATORIOS:.
1.Ordenamiento Burbuja
2.Ordenamiento Seleccion
3.Ordenamiento Insercion
4.Ordenamiento Shell
5.Ordenamiento MergeSort
6.Ordenamiento QuickSort
7.Salir
Elige una opcion : 5
```

Y mergeSort no dejaba de explotar, solo funciono al ordenar una cantidad menor de datos