UNIVERSIDAD NACIONAL SAN AGUSTIN DE AREQUIPA
FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS
ESCUELA PROFESIONAL DE CIENCIA DE LA COMPUTACIÓN

LABORATORIO O9

Asignatura: Ciencia de la Computación II

Docente: Rolando Jesus Cardenas Talavera

Grupo: B

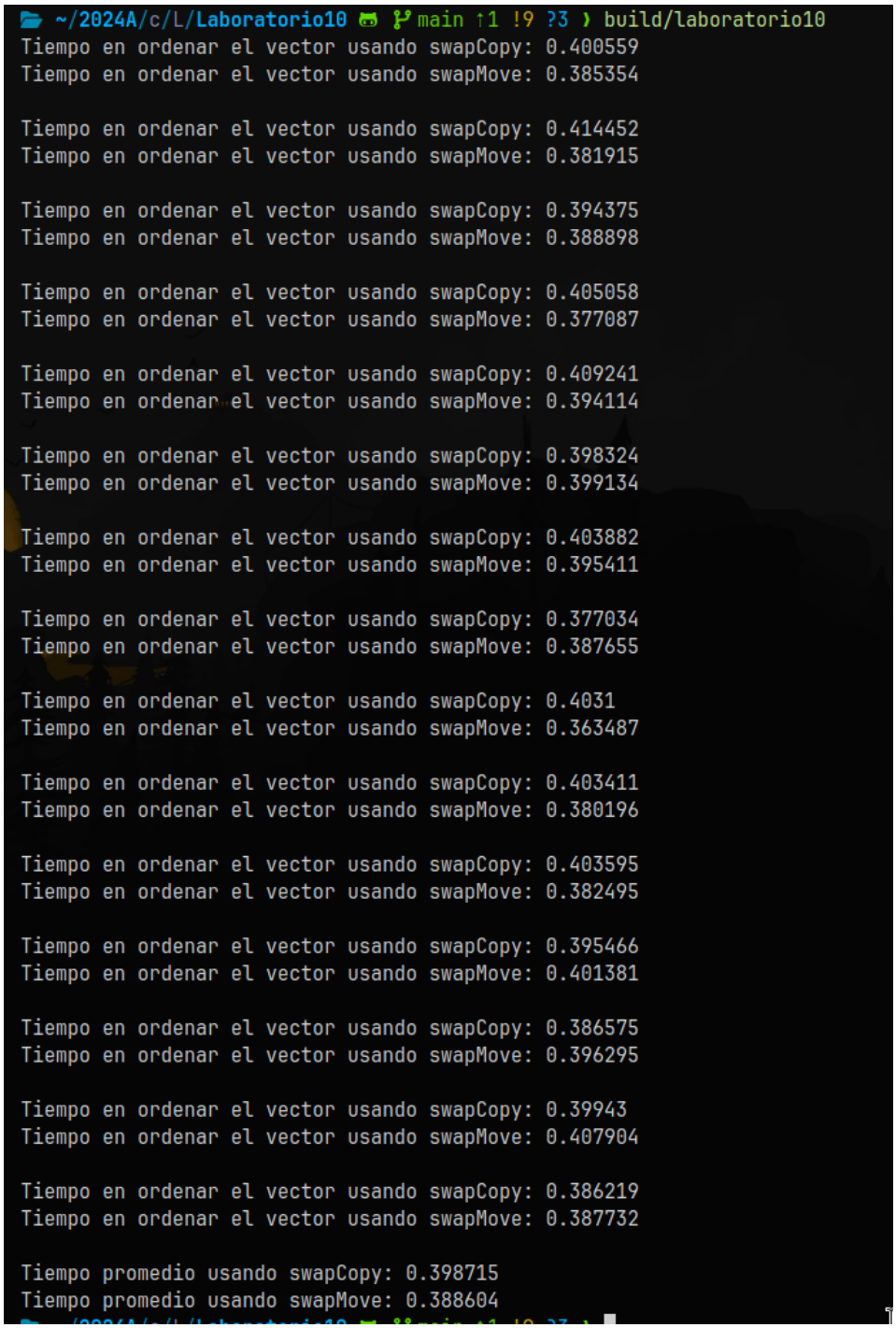
Estudiante: Roni Ezequiel Montañez Pacco

Arequipa-Perú

14 de Junio del 2024

EJERCICIO

CAPTURA



CÓDIGO

```
#include <iostream>
#include <vector>
#include <fstream>
#include <iomanip>
#include <chrono>
using namespace std;

class Coord{
    friend ostream& operator<<(ostream& output, const Coord& a);
  private:
    double x;
    double y;
  public:
    Coord(){};
    Coord(double x, double y) : x(x), y(y) {};
    Coord(Coord&& a){
      this->x = a.x;
      this->y = a.y;
    }
    Coord(const Coord& a){
      this->x = a.x;
      this->y = a.y;
    }
    Coord& operator=(const Coord& a){
      this->x = a.x;
      this->y = a.y;
      return *this;
    }
    Coord& operator=(Coord&& a){
      this->x = a.x;
      this->y = a.y;
      return *this;
    }
    bool operator<(const Coord& a) const{
      return this->x < a.x;
    }
};

ostream& operator<<(ostream& output, const Coord& a){
  output << fixed << setprecision(6) << a.x << ", " << a.y;
  return output;
}

template <typename T>
void swapCopy(T& a, T& b){
  T tmp = a;
  a = b;
  b = tmp;
}
```

```cpp
template <typename T>
void swapMove(T& a, T& b){
  T tmp = move(a);
  a = move(b);
  b = move(tmp);
}

template <typename T>
int partitionCopy(vector<T>& a, int ini, int fin) {
  T pivot = a[fin];
  int c = ini - 1;
  for(int j = ini; j < fin; j++){
    if(a[j] < pivot){
      c++;
      swapCopy(a[c], a[j]);
    }
  }
  swapCopy(a[c + 1], a[fin]);
  return c + 1;
}
template <typename T>
int partitionMove(vector<T>& a, int ini, int fin) {
  T pivot = a[fin];
  int c = ini - 1;
  for(int j = ini; j < fin; j++){
    if(a[j] < pivot){
      c++;
      swapMove(a[c], a[j]);
    }
  }
  swapMove(a[c + 1], a[fin]);
  return c + 1;
}
template <typename T>
void quicksortCopy(vector<T>& a, int ini, int fin){
  if(ini < fin){
    int pos = partitionCopy(a, ini, fin);
    quicksortCopy(a, ini, pos - 1);
    quicksortCopy(a, pos + 1, fin);
  }
}
template <typename T>
void quicksortMove(vector<T>& a, int ini, int fin){
  if(ini < fin){
    int pos = partitionMove(a, ini, fin);
    quicksortMove(a, ini, pos - 1);
    quicksortMove(a, pos + 1, fin);
  }
}

chrono::duration<double> medirTiempoCopy(const vector<Coord> coor){
  vector<Coord> coordenadas = coor;
  const auto start = chrono::high_resolution_clock::now();
```

```
  quicksortCopy(coordenadas,0,499999);
  const auto end = chrono::high_resolution_clock::now();
  const chrono::duration<double> diff = end - start;
  return diff;
}

chrono::duration<double> medirTiempoMove(const vector<Coord> coor){
  vector<Coord> coordenadas = coor;
  const auto start = chrono::high_resolution_clock::now();
  quicksortMove(coordenadas,0,499999);
  const auto end = chrono::high_resolution_clock::now();
  const chrono::duration<double> diff = end - start;
  return diff;
}

void ingresar(vector<Coord>& a, const char* nombre){
  ifstream file(nombre);
  if(!file){
    cout << "No se puede abrir el archivo" << endl;
  }
  double x,y;
  char comma;
  while(file >> x >> comma >> y){
    a.push_back(Coord(x,y));
  }
  file.close();
}

int main(){
  vector<Coord> coordenadas;
  ofstream output("output.txt");
  ingresar(coordenadas,"DataGen.txt");
  chrono::duration<double> countMove;
  chrono::duration<double> countCopy;
  for(int i=0; i<15; i++){
    const chrono::duration<double> diffCopy = medirTiempoCopy(coordenadas);
    cout << "Tiempo en ordenar el vector usando swapCopy: " << diffCopy.count() << endl;
    countCopy+=diffCopy;
    const chrono::duration<double> diffMove = medirTiempoMove(coordenadas);
    cout << "Tiempo en ordenar el vector usando swapMove: " << diffMove.count() << "\n\n";
    countMove+=diffMove;
  }
  cout << "Tiempo promedio usando swapCopy: " << countCopy.count()/15 << endl;
  cout << "Tiempo promedio usando swapMove: " << countMove.count()/15 << endl;

  quicksortMove(coordenadas,0,499999);
  for(const Coord& coord : coordenadas){
    output << coord << endl;
  }
  output.close();
  return 0;
}
```

CONCLUSIÓN

El tiempo que se emplea al ordenar usando std::move es ligeramente menor que al emplear una variable temporal copia, en este caso de 500 000 coordenadas. De 15 pruebas realizadas a ambos metodos (copy y move) se obtuvo una diferencia promedio de 0.010111 segundos.