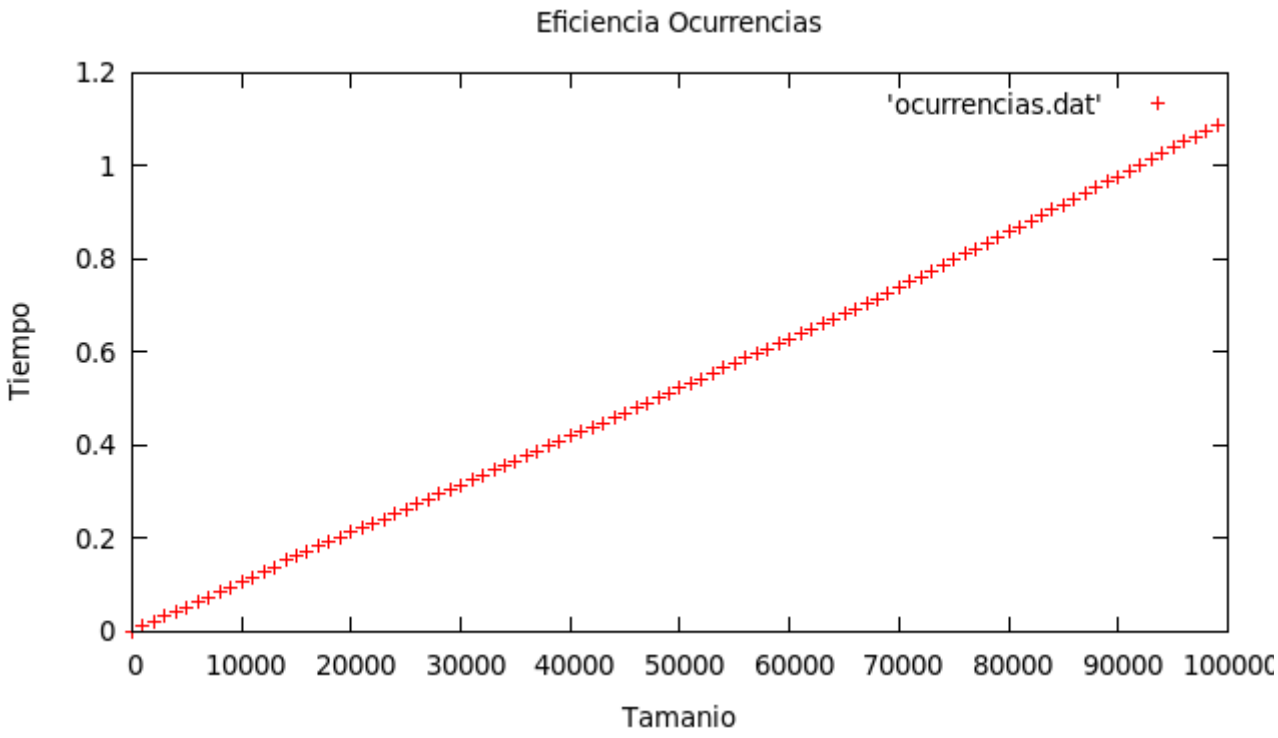
Adrián de la Torre Rodríguez 1ºA

PRACTICA 1

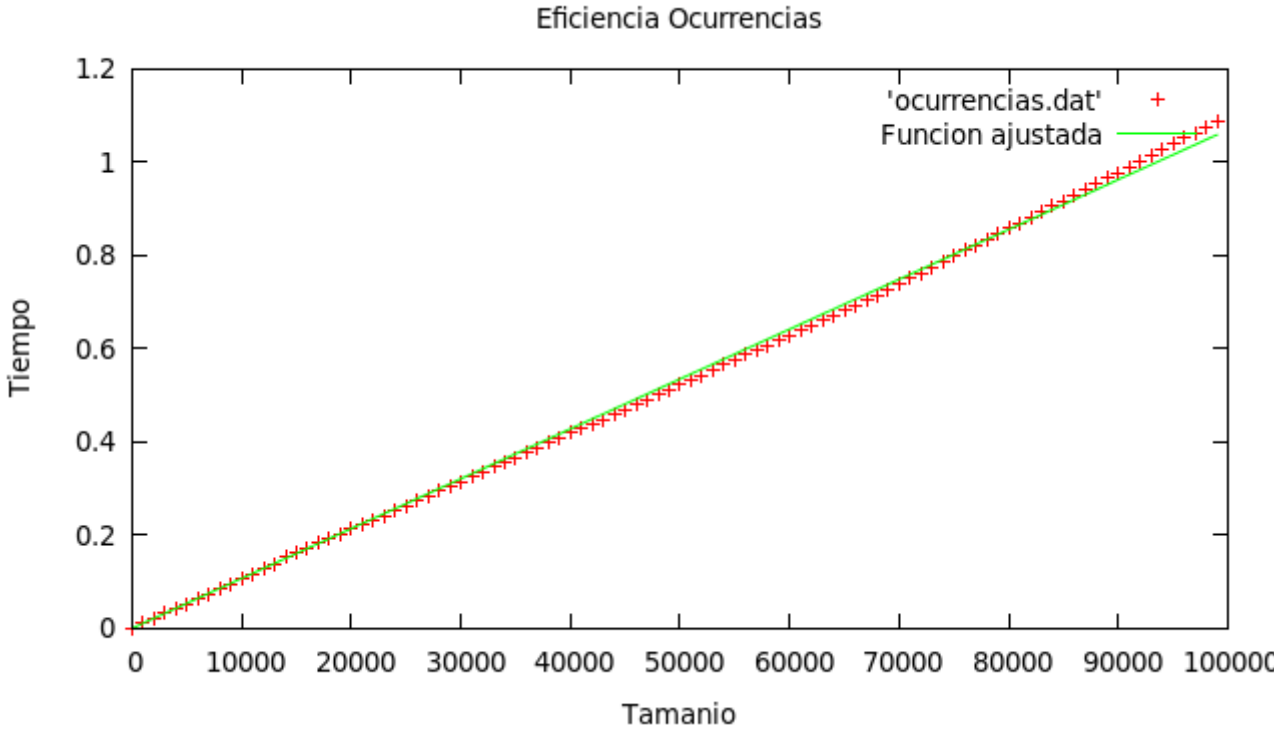
ANÁLISIS DE EFICIENCIA DE OCURRENCIAS.CPP

El algoritmo que aparece al final del pdf “ ocurrencias.cpp ” ha sido modificado para que la salida sea el tamaño y el tiempo separados por un espacio. Con el comando “ ./ocurrencias > ocurrencias.dat ” guardo los resultados en un archivo.

Con gnuplot represento la función tamaño-tiempo y obtengo esta gráfica:



Se puede observar a simple vista que es de orden n. Con gnuplot hago una regresión para ver que función se aproxima más a esta gráfica. Para ello primero defino una función f(x) = a*x y luego digo que ajuste el parámetro 'a': “ fit f(x) 'ocurrencias.dat' via a”. El resultado es que a=1.06953e-05. Voy a representar las dos funciones para ver si se ajusta:

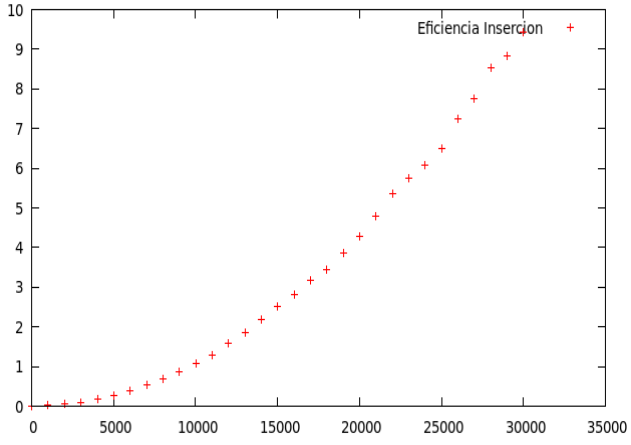


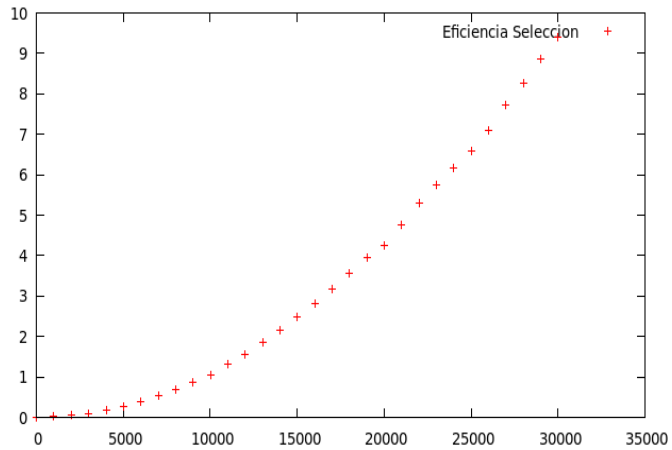
Vemos que se ajusta casi perfectamente, con lo cual confirmamos que es de orden x.

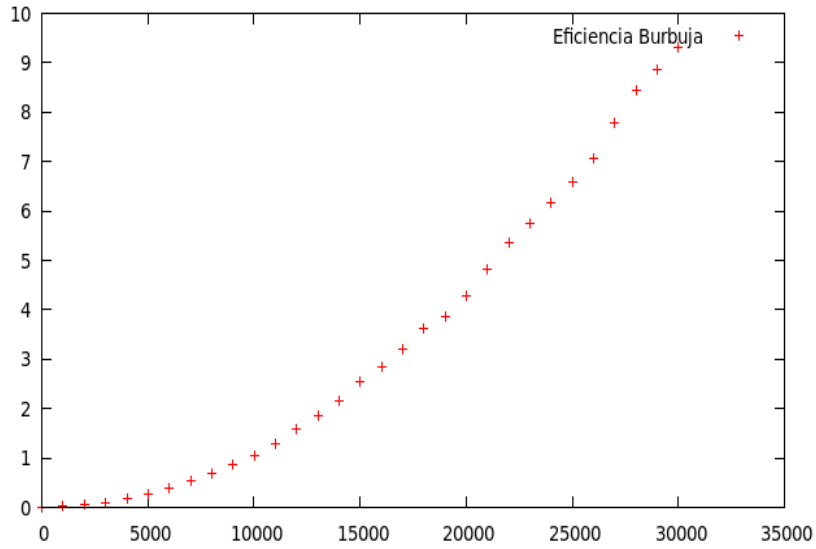
ANALISIS DE EFICIENCIA DE ALGORITMOS

Primero vamos a analizar teóricamente los algoritmos. Como podemos ver en el código al final del pdf "algoritmos.cpp" todos los algoritmos constan de dos bucles for anidados con operaciones de orden constante dentro. Los bucles for dependen del número de datos de entrada "n", como son dos bucles anidados entonces los algoritmos son de orden n^2 .

El análisis práctico lo hago de la misma manera que en el apartado anterior. Guardo los resultados en archivos con el mismo nombre del algoritmo que he usado. La gráficas obtenidas son:







Se puede observar el aumento cuadrático del tiempo en las tres gráficas de una manera similar. No hay diferencias significativas en el tiempo en los distintos algoritmos.

ALGORITMOS

OCURRENCIAS.CPP

/////// Las funciones son iguales que las que había en el archivo original//////

```
int main() {

	vector<string> Dicc;
	vector<string> Q;
	int pos;
	clock_t start,end;


	int contador =0;

	lee_fichero("lema.txt", Dicc);


	lee_fichero("quijote.txt", Q);


	for (int fin = 10; fin < 100000 ; fin+= 1000){

		string b="hidalgo";

		start = clock();

		for (int iteraciones = 0; iteraciones < 1000; iteraciones++)
			pos = contar_hasta(Q, 0,fin, b);

		end= clock();

		double dif = end-start;

		cout<< fin << " " << dif/CLOCKS_PER_SEC <<endl;  // Parte modificada
                                                                 //para adecuar la salida.

	 }




}
```

ALGORITMOS.CPP

```
//ALGORITMO DE ORDENACION BURBUJA

void burbuja(vector<string> & T, int inicial, int final) {
	int i, j;
	string aux;
	for (i = inicial; i < final - 1; i++)
		for (j = final - 1; j > i; j--)
			if (T[j] < T[j-1]) {
				aux = T[j];
				T[j]= T[j-1];
				T[j-1] = aux;
			}

}


//Función que se usa en el algoritmo de ordenación por serlección
void intercambiar(  string &a, string &b){

	string aux = a;
	a = b;
	b = aux;

}


//ALGORITMO DE ORDENACION POR SELECCION
void seleccion( vector<string> & T, int inicial, int final ){

	int minimo=0;

	for ( int i = inicial; i < final; i++){

		T[minimo] = T[i];

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

			if( T[j] < T[minimo])

				minimo = j;

		intercambiar( T[i] , T[minimo] );
	}
}
```

```
//ALGORITMO DE ORDENACION POR INSERCION
void insercion ( vector<string> & T, int inicial, int final ){

	string aux;
	int i,j;
	for (  i = inicial; i < final; i++){

		aux = T[i];

		for(  j = i-1; j >= 0 && aux < T[j]; j--)

			T[j+1] = T[j];

		T[j+1] = aux;
	}
}

//Función que lee el fichero que le pasamos y lo copia a un vector V
void lee_fichero( const char * nf, vector<string> & V) {

	ifstream fe;
	string s;
	fe.open(nf, ifstream::in);

	while ( !fe.eof() ){
		fe >> s;
		if (!fe.eof())
			V.push_back(s);
	}

	fe.close();
}



int main(){

	vector<string> Dicc;

	clock_t start,end;
	double dif;

	int fin=35000;

	lee_fichero("lema.txt", Dicc);




	for( int i=10; i < fin; i+=1000){
```

```
            start = clock();

            //seleccion(Dicc,0,i);                  //Descomentar el algoritmo que se desea usar
            //burbuja(Dicc,0,i);
            insercion(Dicc,0,i);

            end = clock();
            dif = end-start;
            cout<< i << " " << dif/CLOCKS_PER_SEC <<endl;
        }




}
```