

MEMORIA PRÁCTICA DIVIDE Y VENCERÁS

Grupo: PCEO

Curso: 2022-2023

Autores
Jose Guillén Riquelme
Gonzalo Lucena Vázquez

ÍNDICE

Introducción.	3
Pseudocódigo.	4
Breve explicación del algoritmo	6
Estudio teórico del tiempo de ejecución.	7
Validación del código.	8
Estudio experimental.	8
Contraste entre estudios.	16
Conclusiones.	16

Introducción.

En esta documentación de la práctica 1 se muestran los pasos realizados para llegar a completar el ejercicio 12. Se comentará pues: el pseudocódigo que refleja el esqueleto del código, el estudio teórico del tiempo de ejecución, los experimentos llevados a cabo para validar el código, el estudio experimental con diversos tamaños, el contraste entre ambos estudios y, finalmente, las conclusiones.

En cuanto al código, este vendrá adjunto en la propia entrega y que constará de los siguientes archivos:

- **divideVenceras(struct).cpp:** El cual contiene tanto el generador como la función de divide y vencerás, aparte del main que proporciona las soluciones finales de la entrega.
- **generador.cpp:** Contiene los distintos generadores utilizados para tenerlos de formas más aisladas y poder comprobar las salidas antes de hacer los procedimientos oportunos. Este código pide por pantalla el tamaño de la cadena y a continuación cuántos casos se quieren del caso peor, mejor y aleatorio. Cada caso tiene un fichero asociado en el cual se escriben directamente las entradas.
- **entradasPeor.cpp, entradasMejor.cpp, entradasAleatorias:** Ficheros enlazados en el código de generador.cpp. En cada uno aparecen las entradas exigidas. (Estos archivos son experimentales y no son utilizados en el main de divide y vencerás, solo las funciones generadoras).
- **tabla1.csv, tabla2.csv, tabla3.csv:** Son archivos CSV utilizados en el código del main de divideVenceras(struct).cpp y que guardan el tamaño de la entrada (n), el tiempo mejor (tabla2.csv), el tiempo promedio(tabla3.csv) y el tiempo peor(tabla1.csv) de manera automática al ser ejecutado.
- **solucionesPeores, solucionesMejores, solucionesAleatorias:** En ellos se archivan las respuestas de todos los problemas que se analizan en divide y vencerás.
- **ficheros Excel empleados**

Pseudocódigo.

(En el main se definirían tanto n como m)

estructura longPos :

int longitud;

int posicion;

estructura longPos solucionDirecta(int p, int q, char* cadena){

int maximo = -1;

int indiceMax = 0;

int contador = 1;

estructura longPos solucion;

for(i = p; i < q; i++){

si (cadena[i+1] != '\0' && abs(cadena[i+1] - cadena[i]) <= 2) {

contador++;

}

sino {

si (contador > maximo) {

```

        maximo = contador;

        indiceMax = i+1-contador;

    }

    contador = 1;

}

}

    solucion.longitud = maximo;

    solucion.posicion = indiceMax;

    return solucion;

}

int dividir(int p, int q){

    int valorMedio = (int)((p + q) DIV 2);

    return valorMedio;

}

estructura longPos combinar(struct longPos a, struct longPos b, int p, int q,, int m, char * cadena ){

    estructura longPos solucionFrontera;

    int indiceMedio = 0;

    int longitud = q-p+1;

    indiceMedio = (int)(p+q DIV 2);

    solucionFrontera = solucionDirecta(max(p, indiceMedio-(m-2)), min(q, indiceMedio+(m-1)),
cadena);

    si (a.longitud >= b.longitud && a.longitud >= solucionFrontera.longitud) return a;

    sino si (b.longitud > a.longitud && b.longitud > solucionFrontera.longitud) return b;

    sino si (solucionFrontera.longitud > a.longitud && solucionFrontera.longitud >= b.longitud)
return solucionFrontera;

}

```

```

estructura longPos divideVenceras(int p, int q, int m, char* cadena){

    si (q-p +1 <= m) return solucionDirecta(p, q, cadena);    //si es PEQUEÑO --> solDirecta

    sino {                //si no, sigue dividiendo y con la recursión

        int medio = dividir(p, q);

        estructura longPos subcadena1 = divideVenceras(p, medio, m, cadena);

        si (subcadena1.longitud == m) return subcadena1;

        estructura longPos subcadena1 = divideVenceras(medio+1, q, m, cadena);

        si (subcadena2.longitud == m) return subcadena2;

        sino {

            return combinar(subcadena1, subcadena2, p, q, m, cadena);

        }

    }

}

```

Breve explicación del algoritmo

Lo primero de todo es comentar que decidimos utilizar un struct para almacenar la posición y solución que exigía el problema para tenerlo todo más cómodo y a muy rápido acceso en caso de necesitarlo.

Dentro del algoritmo de divide y vencerás hemos analizado si se trata del caso directo (viendo si la longitud era menor o igual que el m con el que se trabaja) y si no llamando a la recursividad. En este caso llamamos primero a la solución de la primera rama y comprobamos si se cumple o no la condición necesaria para terminar el algoritmo. En el caso de que se cumpla se acaba, pero si no se analiza la segunda rama. Procediendo de igual manera si no se ha cumplido tampoco se pasa a la función combinar.

La función combinar procede de la siguiente manera:

- Recibe como parámetros la solución de la primera subcadena analizada, la de la segunda, p, q, m y el puntero de la cadena.

- Además de las dos soluciones disponibles que se disponen, se calcula una tercera (la de la frontera). Para ello aplicamos solución directa al máximo entre p e índice medio - m -2 y el mínimo entre q y el índice medio + m -1. Esto es así ya que con

estos valores se cubren todas las opciones posibles de frontera entre el bloque de la primera subcadena y el segundo.

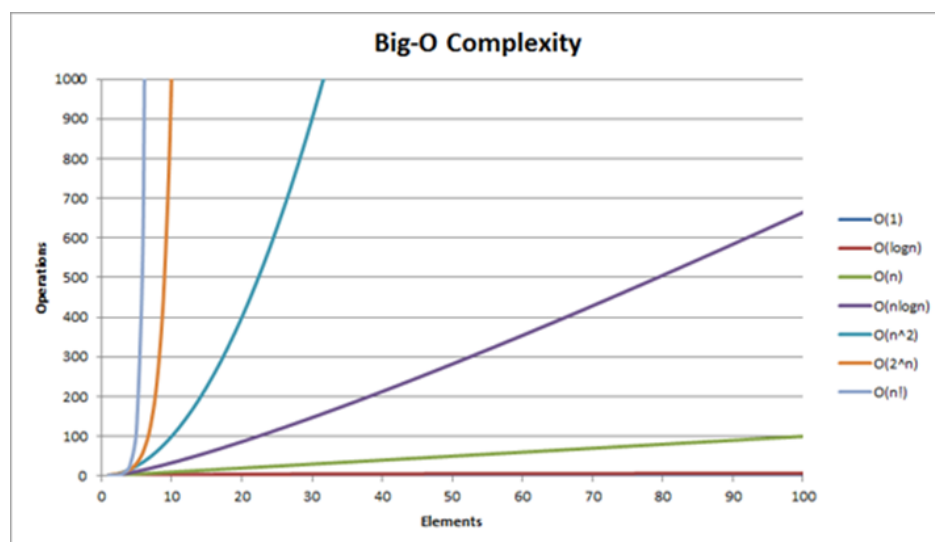
- Por último se analiza cuál de las tres soluciones es la máxima y en caso de empate nos quedamos con la que está más a la izquierda.

Estudio teórico del tiempo de ejecución.

Inicialmente ya nos percatamos de que, siendo Divide y vencerás, un esquema algorítmico basado en la división del problema, el orden de complejidad de este acabaría resultando ser logarítmico.

Y, efectivamente, tras realizar el estudio pertinente, obtuvimos que el tiempo para el caso peor pertenecería a $O(n \log n)$ mientras que el del caso mejor pertenecería a $\Omega(\log n)$. Posteriormente, se ejecutaría el código un gran número de veces y así poder comparar los tiempos, esto es lo que se comenta en el estudio experimental del tiempo de ejecución.

Según esta imagen de referencia, nuestro tiempo peor se correspondería con la curva morada mientras que el mejor sería la roja. A continuación lo comprobaremos.



Validación del código.

A la hora de validar el código se han realizado varios pasos. Primero, obviamente, se compilaba y generaba el ejecutable. Tras ello se le pasaban entradas, pero no valía cualquier entrada. Al pasarle entradas generadas aleatoriamente realmente no podíamos darnos cuenta de los posibles errores, ya que no sabríamos si el resultado obtenido es erróneo o correcto. Por tanto, optamos por pasarle 2 tipos de entradas en las que sería más fácil localizar errores: entradas para los casos mejor y peor.

Identificamos que el caso peor era el que, para todas las letras de la cadena menos sus $m-1$ últimas posiciones la diferencia con sus adyacentes sería mayor que 2, y para esas $m-1$ últimas letras si se cumple la condición.

Ahora después comentaremos más en profundidad que este no se trata del caso peor realmente ni tampoco otros dos casos más que llegamos a probar también.

Mientras que en el mejor, la cadena resultado de tamaño m se encontraría al principio. Y basándonos en si iba dando esos resultados, o no, para esas entradas, fuimos validando y corrigiendo errores.

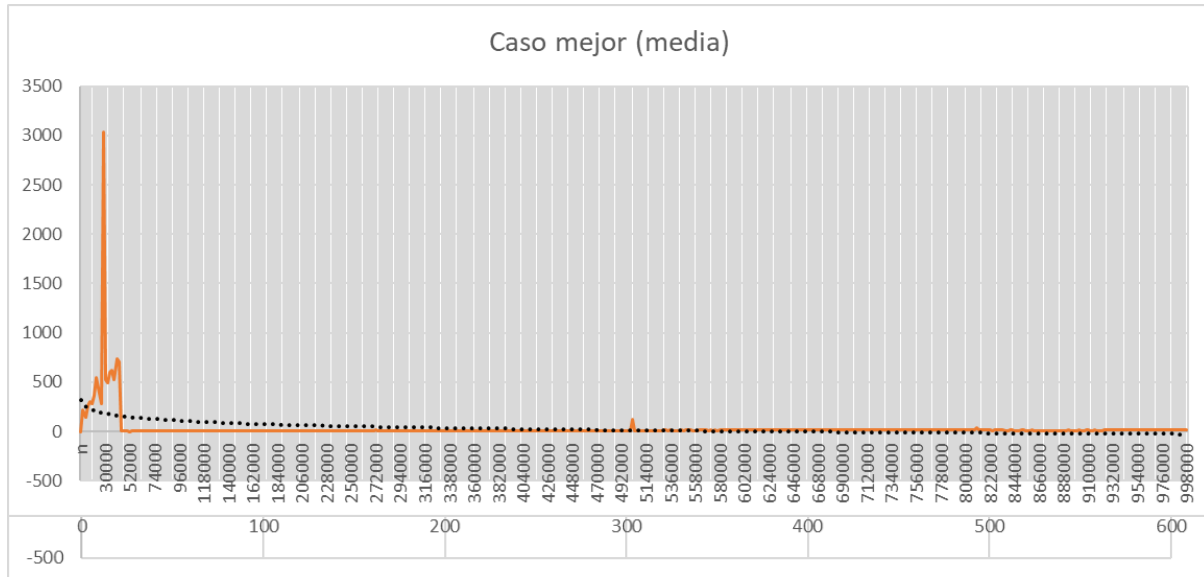
Las salidas que obteníamos se pueden ver en los archivos que adjuntaremos llamados solucionesMejores, solucionesPeores y solucionesAleatorias. En ellos aparecen las salidas que se esperaban.

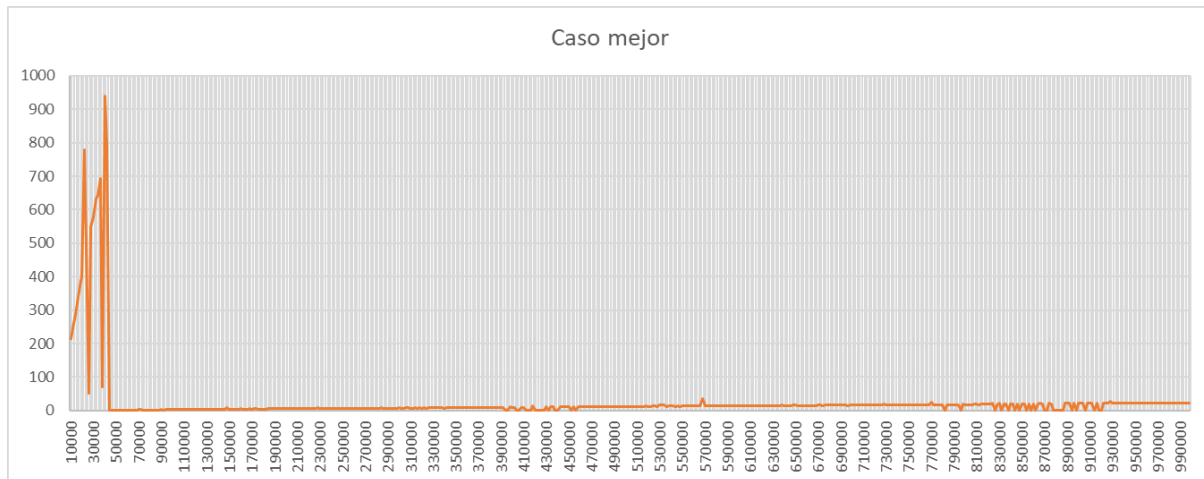
Estudio experimental.

Para el estudio experimental hemos optado por ejecutar el código un total de 500 veces, incrementando de 2000 en 2000 desde $n=10.000$ hasta 1.000.000. Los tiempos obtenidos, para los casos mejor, peor y aleatorios, los hemos almacenado en un fichero CSV, el cual nos ha permitido trabajar con ellos en excel y ver así sus gráficas. Además hemos realizado 5 experimentos con cada n y tomado la media. Esto nos permite centralizar y quitar resultados demasiado grandes o pequeños. Aún haciendo esto se puede observar mucho mejor usando un único experimento. Por eso vamos a adjuntar ambos casos (el de la media y el caso concreto). Además cada uno de ellos tendrá dos gráficas distintas (una de dispersión y otra lineal para poder ver mejor lo que pasa)

En el eje vertical aparece el tiempo en milisegundos y en el horizontal la n.

CASO MEJOR





Observando la gráfica obtenida para el tiempo mejor nos podemos dar cuenta que se asemeja bastante a la función logarítmica. De hecho la línea de tendencia ofrecida por el propio excel ayuda a verificarlo. Se puede ver que los tiempos son muy pequeños.

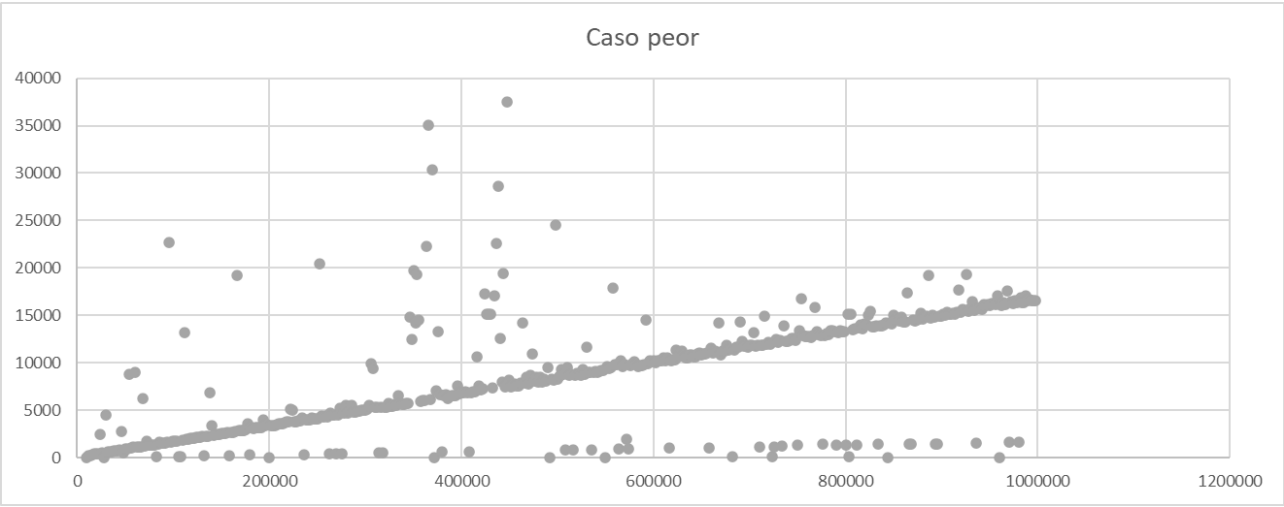
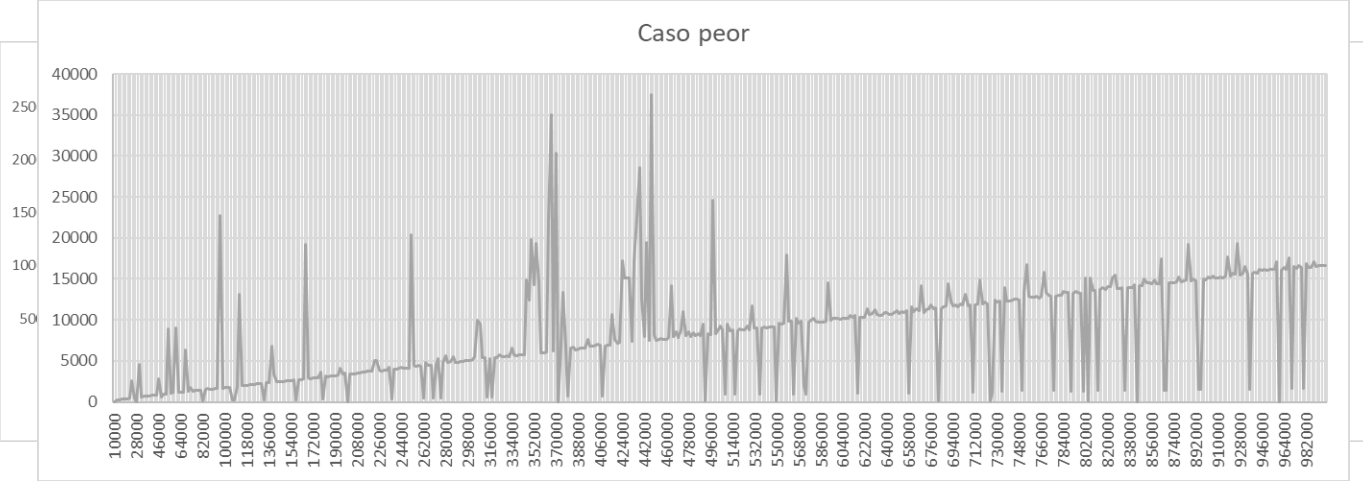
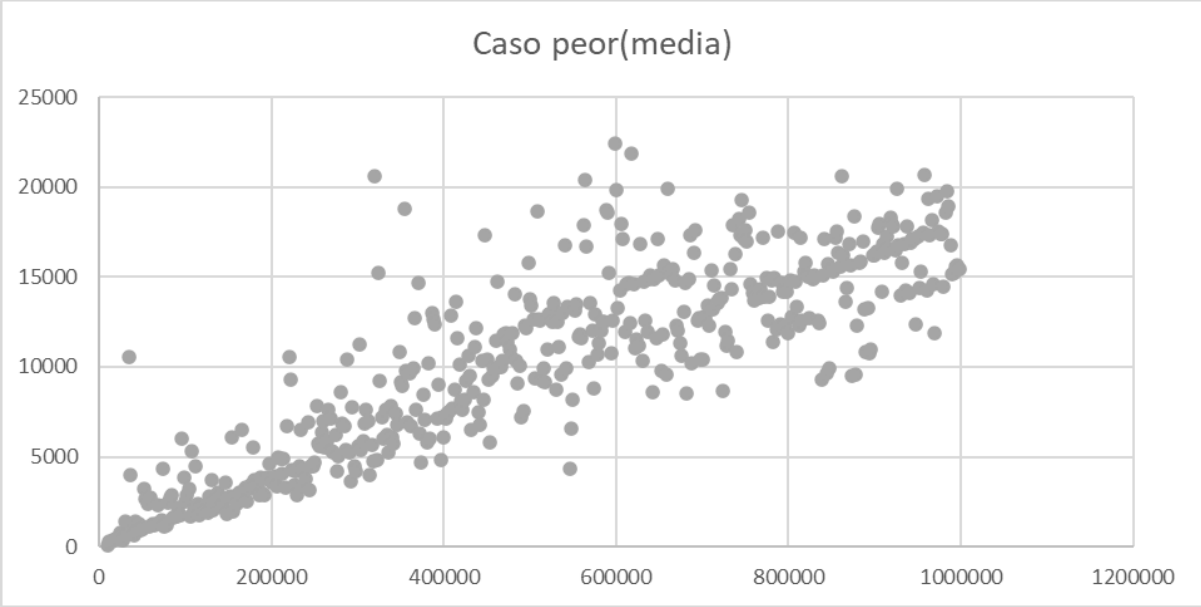
CASO PEOR

Para el caso peor habíamos generado un caso que creíamos que era el “peor” pero no lo es realmente.

Además los datos de las tablas nos han permitido ver que el caso promedio se acerca mucho a un posible caso peor. Realmente si nos paramos a pensarlo tiene sentido pues con una entrada de caracteres aleatoria la probabilidad de que dos caracteres consecutivos cumplan la condición exigida es de $5/26$ en la mayoría de casos (sólo se cumpliría con sus dos anteriores, él mismo y sus dos posteriores), de $4/26$ si se trata de la b y la y, y de $3/26$ si es la a o la z.

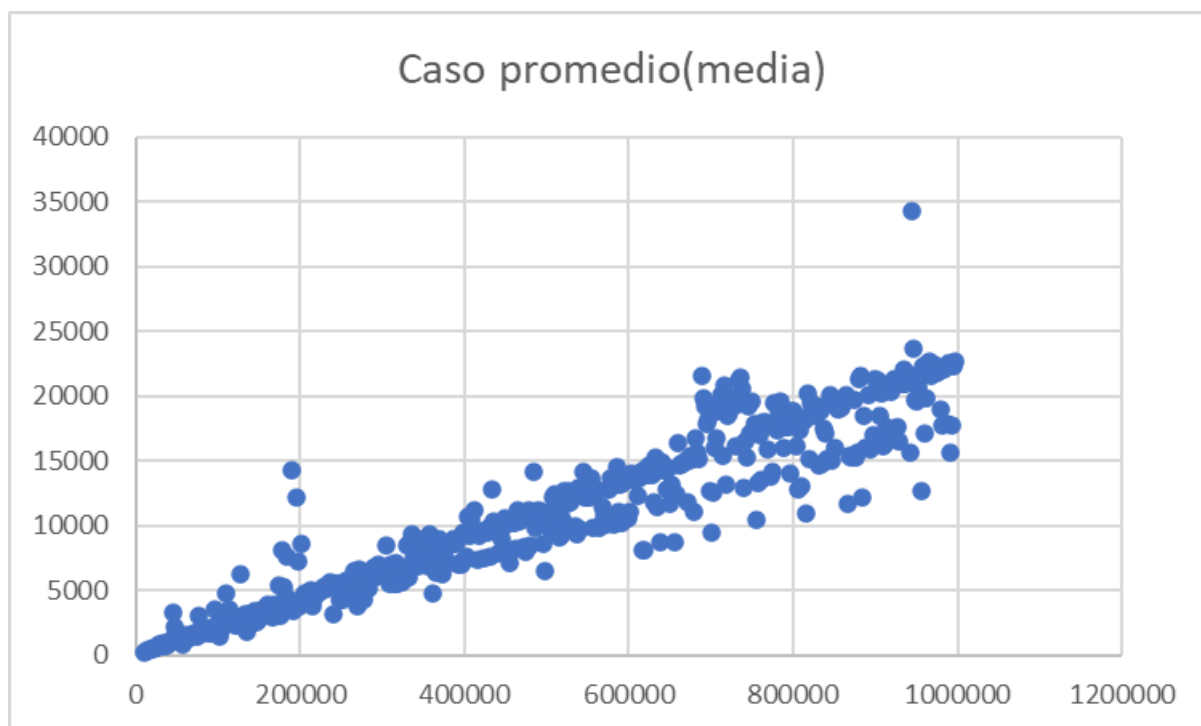
Posiblemente exista un caso peor que el aleatorio pero nosotros no hemos sido capaces de generarlo.

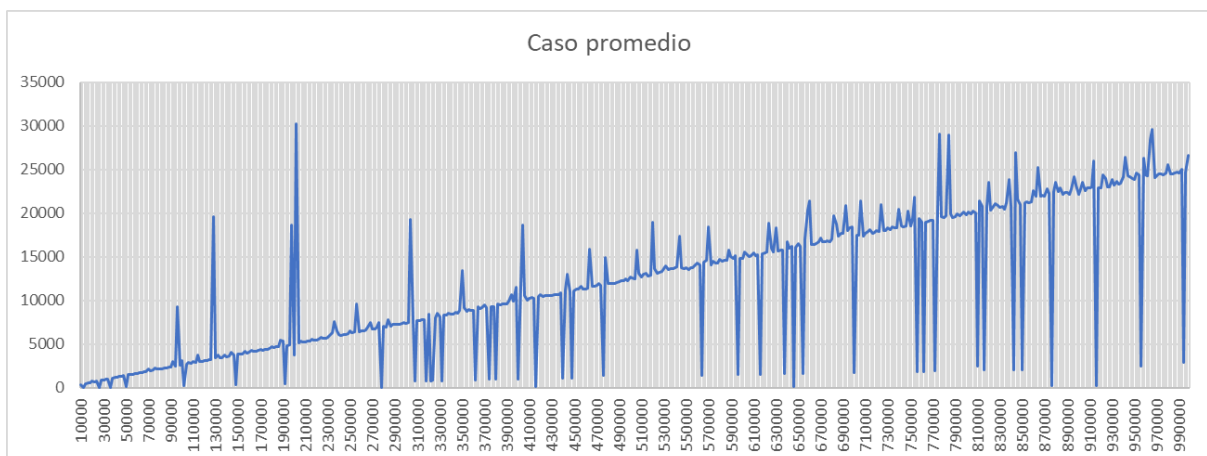
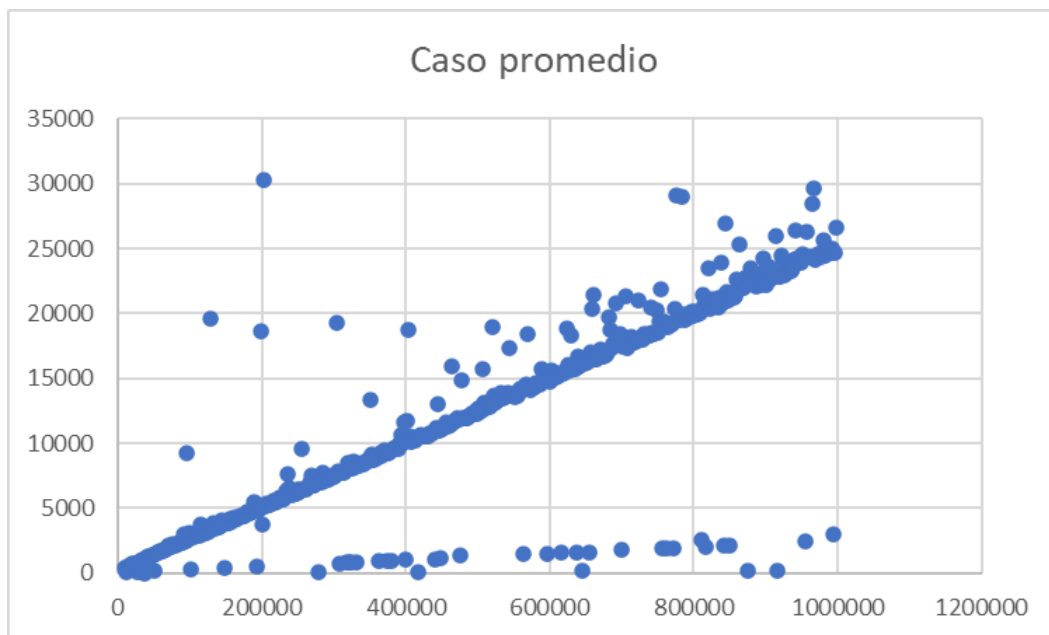
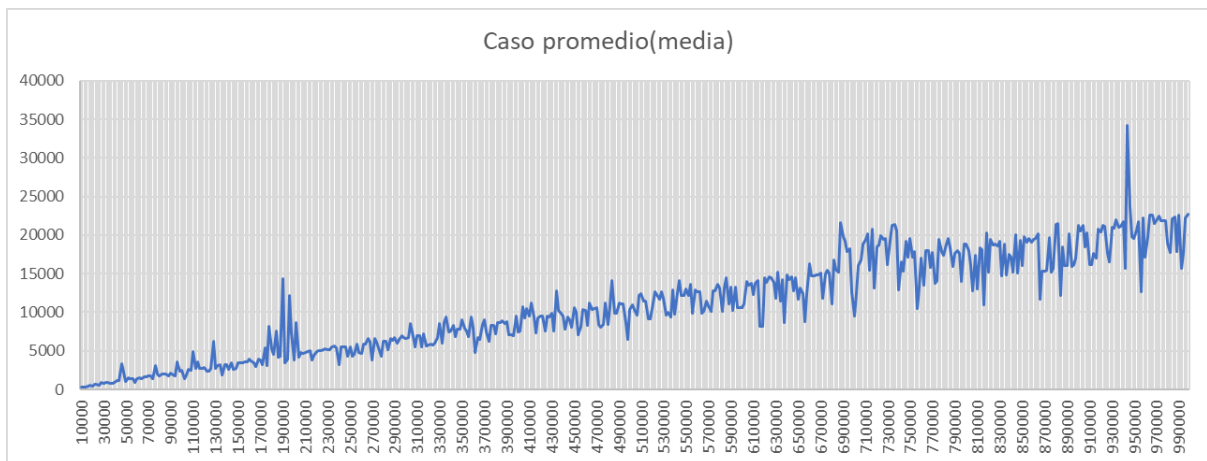
Aún así vamos a proporcionar las gráficas que nos han salido ya que nos han servido para darnos cuenta de nuestro error.



Después de ver el caso aleatorio veremos en una perspectiva general como estos puntos quedan por debajo del caso promedio verificando que no son el caso peor.

CASO PROMEDIO



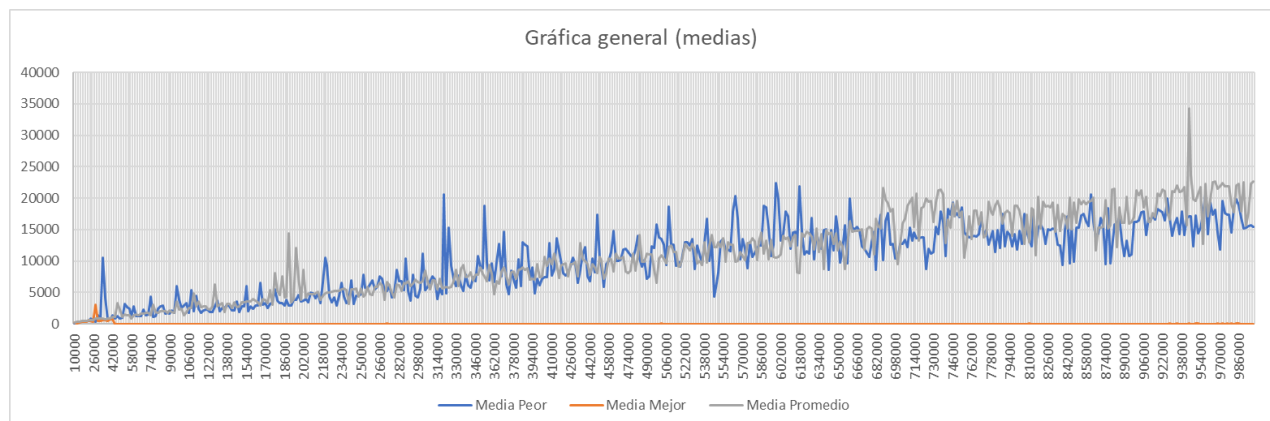
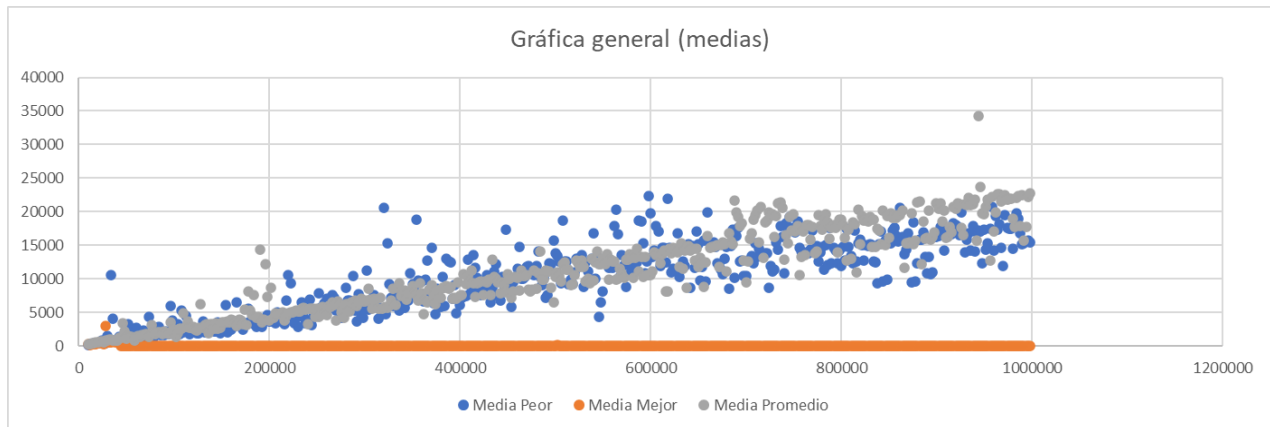


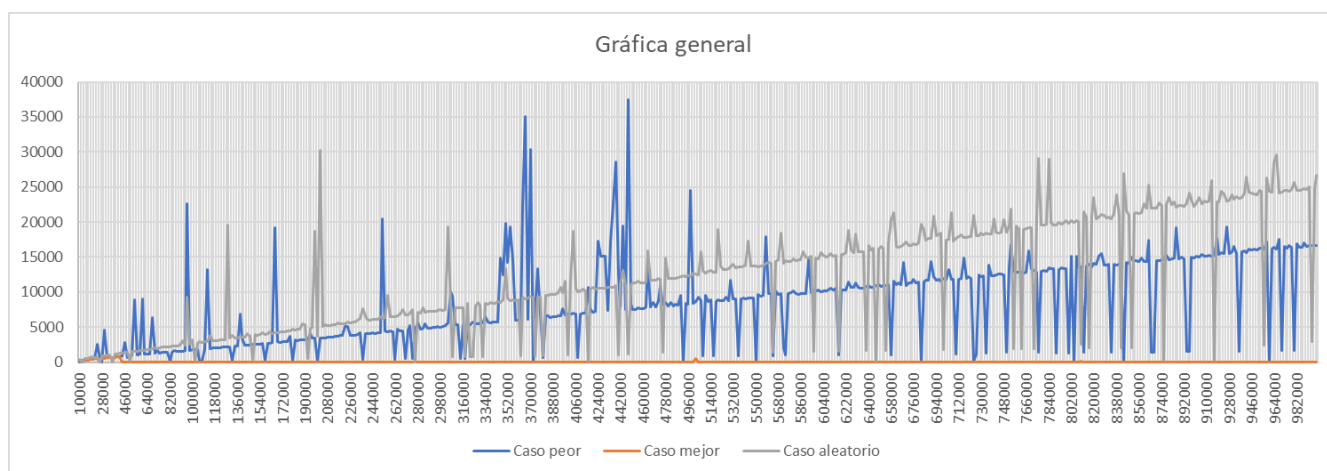
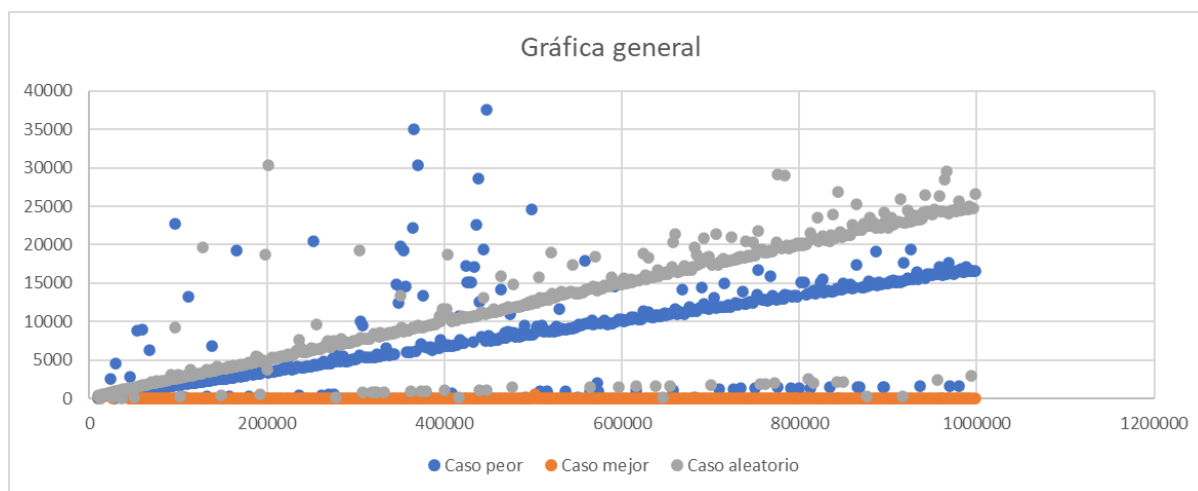
Para el caso aleatorio tenemos una gráfica muy parecida a la del tiempo peor e incluso quedando por encima como hemos comentado anteriormente. Gracias a esto, hemos replanteado si teníamos bien el caso peor.

La gráfica de dispersión sigue una clara tendencia y, al estudiarla también en Geogebra hemos concluido que la función ha de ser $x \cdot n \log n$ siendo x un coeficiente racional, haciendo así que tenga menor pendiente que $n \log n$.

Por tanto la gráfica obtenida pertenece a $O(n \log n)$ y a $\Omega(\log n)$.

PERSPECTIVA GENERAL





Aquí está la gráfica general que incluye todas las anteriores para poder tener una mejor visión del resultado. Se puede observar que el caso aleatorio supera al nuestro “caso peor”.

Contraste entre estudios.

La mayor adversidad que teníamos a la hora de comparar resultados residía sobre todo en la gráfica generada por el caso peor. Gracias al estudio teórico, sabíamos que tendría forma similar a $n \log n$, pero finalmente no hemos sido capaces de generarlo. Hemos probado con tres “supuestos” casos peores pero ninguno de ellos conseguía tener un tiempo mayor a los del caso aleatorio. Por tanto solo podemos decir que no hemos conseguido obtener un generador que lo hiciera y que el caso aleatorio tiene que estar muy cercano al caso peor que buscamos.

La gráfica del tiempo mejor nos coincidía perfectamente con $\log n$ mientras que, en los casos aleatorios, ya contábamos a priori con que su gráfica saldría bastante similar a la del caso peor debido a la probabilidad ya comentada. Lo que no esperábamos era que la superara.

Así que, en definitiva, se podría decir que, para el caso mejor y promedio hemos obtenido los resultados esperados y para el caso peor no.

Conclusiones.

Gonzalo: Cuando comenzamos a trabajar con la práctica constantemente pensaba que era algo bastante ineficiente, teniendo en cuenta que se podría solventar con un algoritmo directo y sin necesidad de dividir ni nada. A día de hoy lo sigo pensando aunque de manera menos tajante ya que, a pesar de que no haya visto este esquema con muy buenos ojos, todas las horas de trabajo, infinidad de correcciones y el estudio posterior, han hecho que con el tiempo mi perspectiva haya ido cambiando un poco. Así que, en definitiva, tras los resultados obtenidos diría que estoy bastante satisfecho con cómo ha ido.

Jose: Al principio pensaba que la práctica iba a resultar más dinámica y entretenida pero a medida que hemos ido avanzando se nos complicaba en cosas muy absurdas perdiendo en la mayoría de ocasiones tiempo tontamente.

Por lo tanto diría que no me ha gustado mucho la actividad; sin embargo, como método para aplicar divide y vencerás ha estado bastante bien ya que permite practicar lo visto en las clases teóricas.