



Instituto Politécnico Nacional

Escuela Superior de Computo



Ejercicio 04

"Análisis de Casos"

Mora Ayala José Antonio

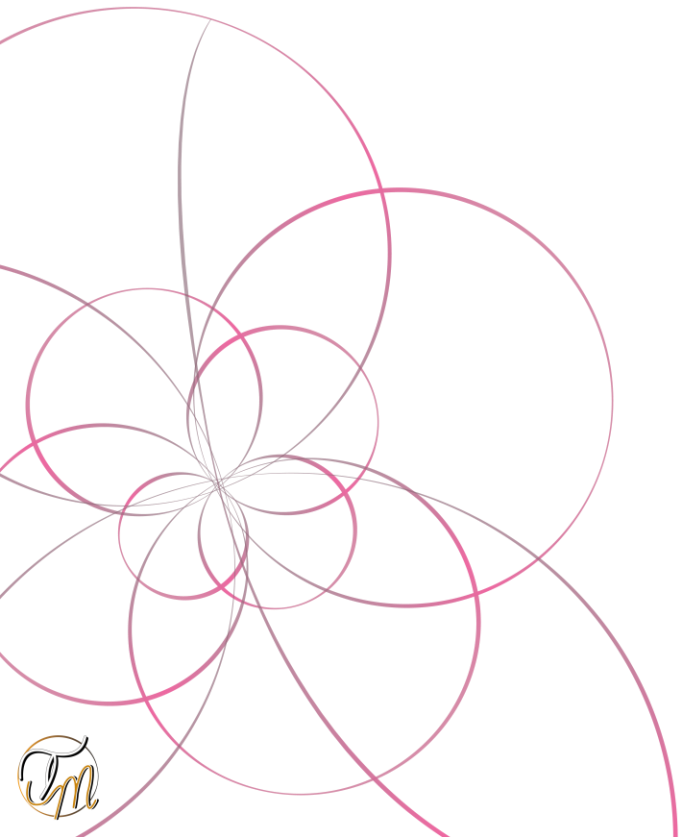
Análisis de Algoritmos



Instrucciones

Para los siguientes algoritmos determine las funciones de complejidad temporal, para el mejor caso, peor caso y caso medio de cada algoritmo. Indique cual(es) son las condiciones (instancia de entrada) del peor caso y cual(es) la del mejor caso en términos del problema que resuelven.

Señale las operaciones básicas que considerará para el análisis de cada algoritmo (Si no logra distinguirlas considere todas las asignaciones, aritméticas y condicionales)



CÓDIGO 1

```
void c1CasoMedio(){
    scanf("%d", &n);
    A = malloc(sizeof(int)*n);
    contador = 0;

    for (j = 0; j < n; j++){
        A[j]= rand()%(32000);
    }

    if (A[1] > A[2] && A[1] > A[3]){
        contador++;

        m1 = A[1];      contador++;

        if (A[2] > A[3]){
            contador++;

            m2 = A[2];  contador++;
            m3 = A[3];  contador++;
        }
        else{
            contador++;

            m2 = A[3];  contador++;
            m3 = A[2];  contador++;
        }
    }
    else if (A[2] > A[1] && A[2] > A[3]){
        contador+=2;

        m1 = A[2];      contador++;

        if (A[1] > A[3]){
            contador++;

            m2 = A[1];  contador++;
            m3 = A[3];  contador++;
        }
        else{
            contador++;

            m2 = A[3];  contador++;
            m3 = A[1];  contador++;
        }
    }
    else{
```

```
    contador+=2;

    m1 = A[3];  contador++;

    if (A[1] > A[2]){
        contador++;

        m2 = A[1];  contador++;
        m3 = A[2];  contador++;
    }
    else{

        contador++;

        m2 = A[2];  contador++;
        m3 = A[1];  contador++;
    }
}

i = 4;

while( i <= n ){

    if (A[i] > m1){

        contador++;

        m3 = m2;  contador++;
        m2 = m1;  contador++;
        m1 = A[i];  contador++;
    }
    else if (A[i] > m2){

        contador+=2;

        m3 = m2;  contador++;
        m2 = A[i];  contador++;
    }
    else if (A[i] > m3){

        contador+=3;
        m3 = A[i];  contador++;
    }
    else{

        contador+=3;
    }

    i++;
}

promedio+=contador;
}
```

SELECCIÓN DE OPERACIONES BÁSICAS

Como se ha visto durante la clase la selección de operaciones básicas se han definido en relación a que tanto se ven afectadas por el tamaño del problema, contabilizando solo aquellas que son necesarias para el mismo, por lo que usaremos aquellas que emplean el arreglo de tamaño n, estas son presentadas a continuación mediante comentarios en el código anterior

```
void c1CasoMedio(){
    scanf("%d", &n);
    A = malloc(sizeof(int)*n);
    contador = 0;
    for (j = 0; j < n; j++){
        A[j]= rand()%(32000);
    }
    if (A[1] > A[2] && A[1] > A[3]){
        contador++; //Contador comp

        m1 = A[1];    contador++; //contador Asig

        if (A[2] > A[3]){
            contador++; //Contador comp

            m2 = A[2]; contador++; //Contador Asig
            m3 = A[3]; contador++; //Contador Asig
        }
        else{
            contador++; //Contador comp

            m2 = A[3]; contador++; //Contador asig
            m3 = A[2]; contador++; //Contador asig
        }
    }
    else if (A[2] > A[1] && A[2] > A[3]){
        contador+=2; //Contador comp
        m1 = A[2];    contador++; //Contador Asig
        if (A[1] > A[3]){
            contador++; //Contador comp

            m2 = A[1]; contador++; //Contador Asig
            m3 = A[3]; contador++; //Contador Asig
        }
        else{
            contador++; //Contador comp
        }
    }
}
```

```
        m2 = A[3]; contador++; //Contador Asig
        m3 = A[1]; contador++; //Contador Asig
    }
}
else{

    contador+=2; //Contador comp

    m1 = A[3]; contador++; //Contador Asig

    if (A[1] > A[2]){

        contador++; //Contador comp

        m2 = A[1]; contador++; //Contador Asig
        m3 = A[2]; contador++; //Contador Asig
    }
    else{

        contador++; //Contador comp

        m2 = A[2]; contador++; //Contador comp
        m3 = A[1]; contador++; //Contador comp
    }
}

i = 4;

while( i <= n ){

    if (A[i] > m1){

        contador++; //Contador comp

        m3 = m2; contador++; //Contador Asig
        m2 = m1; contador++; //Contador Asig
        m1 = A[i]; contador++; //Contador Asig
    }
    else if (A[i] > m2){

        contador+=2; //Contador comp

        m3 = m2; contador++; //Contador Asig
        m2 = A[i]; contador++; //Contador Asig
    }
    else if (A[i] > m3){

        contador+=3; //Contador comp

        m3 = A[i]; contador++; //Contador Asig
    }
}
```



```
        else{  
            contador+=3;           //Contador comp  
        }  
        i++;  
    }  
    promedio+=contador;  
}
```

Ya que hemos definido las operaciones básicas podemos proceder a realizar nuestro análisis de casos, donde establecemos la formula correspondiente para cada una de las situaciones:

mejor, peor, caso medio:

MEJOR CASO

Para el mejor caso tendremos en cuenta que los primeros 3 números del arreglo son los 3 números mas grandes, por lo que al momento de entrar en el ciclo WHILE únicamente se efectuaran las comparaciones, sin la necesidad de entrar en ellas.

A continuación, se presentan las imágenes representativas de cada uno de los segmentos del código para la representación del caso correspondiente:

```
173  
174 > if (A[1] > A[2] && A[1] > A[3]){  
175  
176 >     contador++; //Mejor Caso  
177  
178 >     m1 = A[1]; contador++; //Mejor Caso  
179  
180 >     if (A[2] > A[3]){  
181  
182 >         contador++; //Mejor Caso  
183  
184 >         m2 = A[2]; contador++; //Mejor Caso  
185 >         m3 = A[3]; contador++; //Mejor Caso  
186 >     }  
187 > } else{ ...  
194 > }  
195 > else if (A[2] > A[1] && A[2] > A[3]){ ...  
216 > } else{ ...  
237  
238 > i = 4;  
239  
240 > while(i <= n){  
241  
242 >     if (A[i] > m1){ //Mejor Caso ...  
250 >     } else if (A[i] > m2){ //Mejor Caso ...  
257 >     } else if (A[i] > m3){ //Mejor Caso ...  
263 >     } else{ ...  
267  
268 >     i++;  
269 > }
```

Función de Complejidad Temporal
 $f_t(n) = 1 \text{ comparación} +$
 $1 \text{ asignación} + 1 \text{ comparación} +$
 $2 \text{ asignaciones} + 3(n -$
 $3) \text{ comparaciones}$

$$f_t(n) = 1 + 1 + 1 + 2 + 3n - 9$$
$$= 3n - 4$$

PEOR CASO

```

173
174 > if (A[1] > A[2] && A[1] > A[3]){ //1operacion-PC+
175 + else if (A[2] > A[1] && A[2] > A[3]){ //1operacion-PC
176
177 + contador+=2; + + + + +
178
179 + m1 = A[2]; + + + + + //1operacion-PC
180
181 > if (A[1] > A[3]){ + + + + +
182 + else{
183
184 + contador+=2; + + + + +
185
186 + m1 = A[3]; + + + + +
187
188 + if (A[1] > A[2]){ + + + + + //1operacion-PC
189 +
190 + m2 = A[1]; + + + + + //1operacion-PC
191 + m3 = A[2]; + + + + + //1operacion-PC
192 +
193 + }
194 + else{
195 +
196 + }
197
198 + i = 4;
199
200 + while( i <= n ){
201 +
202 + if (A[i] > m1){ + + + + + //1operacion-PC
203 +
204 + m3 = m2; + + + + + //1operacion-PC
205 + m2 = m1; + + + + + //1operacion-PC
206 + m1 = A[i]; + + + + + //1operacion-PC
207 +
208 + }
209 +
210 + }
211
212 + }
213
214 + }
215
216 + }
217
218 + }
219
220 + }
221
222 + }
223
224 + }
225
226 + }
227
228 + }
229
230 + }
231
232 + }
233
234 + }
235
236 + }
237
238 + }
239
240 + }
241
242 + }
243
244 + }
245
246 + }
247
248 + }
249
250 + }
251
252 + }
253
254 + }
255
256 + }
257
258 + }
259
260 + }
261
262 + }
263
264 + }
265
266 + }
267
268 + }
269
270 + }
271
272 + }
273
274 + }
275
276 + }
277
278 + }
279
280 + }
281
282 + }
283
284 + }
285
286 + }
287
288 + }
289
290 + }
291
292 + }
293
294 + }
295
296 + }
297
298 + }
299
300 + }
301
302 + }
303
304 + }
305
306 + }
307
308 + }
309
310 + }
311
312 + }
313
314 + }
315
316 + }
317
318 + }
319
320 + }
321
322 + }
323
324 + }
325
326 + }
327
328 + }
329
330 + }
331
332 + }
333
334 + }
335
336 + }
337
338 + }
339
340 + }
341
342 + }
343
344 + }
345
346 + }
347
348 + }
349
350 + }
351
352 + }
353
354 + }
355
356 + }
357
358 + }
359
360 + }
361
362 + }
363
364 + }
365
366 + }
367
368 + }
369
370 + }
371
372 + }
373
374 + }
375
376 + }
377
378 + }
379
380 + }
381
382 + }
383
384 + }
385
386 + }
387
388 + }
389
390 + }
391
392 + }
393
394 + }
395
396 + }
397
398 + }
399
400 + }
401
402 + }
403
404 + }
405
406 + }
407
408 + }
409
410 + }
411
412 + }
413
414 + }
415
416 + }
417
418 + }
419
420 + }
421
422 + }
423
424 + }
425
426 + }
427
428 + }
429
430 + }
431
432 + }
433
434 + }
435
436 + }
437
438 + }
439
440 + }
441
442 + }
443
444 + }
445
446 + }
447
448 + }
449
450 + }
451
452 + }
453
454 + }
455
456 + }
457
458 + }
459
460 + }
461
462 + }
463
464 + }
465
466 + }
467
468 + }
469
470 + }
471
472 + }
473
474 + }
475
476 + }
477
478 + }
479
480 + }
481
482 + }
483
484 + }
485
486 + }
487
488 + }
489
490 + }
491
492 + }
493
494 + }
495
496 + }
497
498 + }
499
500 + }
501
502 + }
503
504 + }
505
506 + }
507
508 + }
509
510 + }
511
512 + }
513
514 + }
515
516 + }
517
518 + }
519
520 + }
521
522 + }
523
524 + }
525
526 + }
527
528 + }
529
530 + }
531
532 + }
533
534 + }
535
536 + }
537
538 + }
539
540 + }
541
542 + }
543
544 + }
545
546 + }
547
548 + }
549
550 + }
551
552 + }
553
554 + }
555
556 + }
557
558 + }
559
560 + }
561
562 + }
563
564 + }
565
566 + }
567
568 + }
569
570 + }
571
572 + }
573
574 + }
575
576 + }
577
578 + }
579
580 + }
581
582 + }
583
584 + }
585
586 + }
587
588 + }
589
590 + }
591
592 + }
593
594 + }
595
596 + }
597
598 + }
599
600 + }
601
602 + }
603
604 + }
605
606 + }
607
608 + }
609
610 + }
611
612 + }
613
614 + }
615
616 + }
617
618 + }
619
620 + }
621
622 + }
623
624 + }
625
626 + }
627
628 + }
629
630 + }
631
632 + }
633
634 + }
635
636 + }
637
638 + }
639
640 + }
641
642 + }
643
644 + }
645
646 + }
647
648 + }
649
650 + }
651
652 + }
653
654 + }
655
656 + }
657
658 + }
659
660 + }
661
662 + }
663
664 + }
665
666 + }
667
668 + }
669
670 + }
671
672 + }
673
674 + }
675
676 + }
677
678 + }
679
680 + }
681
682 + }
683
684 + }
685
686 + }
687
688 + }
689
690 + }
691
692 + }
693
694 + }
695
696 + }
697
698 + }
699
700 + }
701
702 + }
703
704 + }
705
706 + }
707
708 + }
709
710 + }
711
712 + }
713
714 + }
715
716 + }
717
718 + }
719
720 + }
721
722 + }
723
724 + }
725
726 + }
727
728 + }
729
730 + }
731
732 + }
733
734 + }
735
736 + }
737
738 + }
739
740 + }
741
742 + }
743
744 + }
745
746 + }
747
748 + }
749
750 + }
751
752 + }
753
754 + }
755
756 + }
757
758 + }
759
760 + }
761
762 + }
763
764 + }
765
766 + }
767
768 + }
769
770 + }
771
772 + }
773
774 + }
775
776 + }
777
778 + }
779
780 + }
781
782 + }
783
784 + }
785
786 + }
787
788 + }
789
790 + }
791
792 + }
793
794 + }
795
796 + }
797
798 + }
799
800 + }
801
802 + }
803
804 + }
805
806 + }
807
808 + }
809
810 + }
811
812 + }
813
814 + }
815
816 + }
817
818 + }
819
820 + }
821
822 + }
823
824 + }
825
826 + }
827
828 + }
829
830 + }
831
832 + }
833
834 + }
835
836 + }
837
838 + }
839
840 + }
841
842 + }
843
844 + }
845
846 + }
847
848 + }
849
850 + }
851
852 + }
853
854 + }
855
856 + }
857
858 + }
859
860 + }
861
862 + }
863
864 + }
865
866 + }
867
868 + }
869
870 + }
871
872 + }
873
874 + }
875
876 + }
877
878 + }
879
880 + }
881
882 + }
883
884 + }
885
886 + }
887
888 + }
889
890 + }
891
892 + }
893
894 + }
895
896 + }
897
898 + }
899
900 + }
901
902 + }
903
904 + }
905
906 + }
907
908 + }
909
910 + }
911
912 + }
913
914 + }
915
916 + }
917
918 + }
919
920 + }
921
922 + }
923
924 + }
925
926 + }
927
928 + }
929
930 + }
931
932 + }
933
934 + }
935
936 + }
937
938 + }
939
940 + }
941
942 + }
943
944 + }
945
946 + }
947
948 + }
949
950 + }
951
952 + }
953
954 + }
955
956 + }
957
958 + }
959
960 + }
961
962 + }
963
964 + }
965
966 + }
967
968 + }
969
970 + }
971
972 + }
973
974 + }
975
976 + }
977
978 + }
979
980 + }
981
982 + }
983
984 + }
985
986 + }
987
988 + }
989
990 + }
991
992 + }
993
994 + }
995
996 + }
997
998 + }
999
1000 + }

```

Como ya sabemos, el peor caso será aquel que tome la ruta más larga para poder concluir el algoritmo con respecto a las operaciones básicas que hemos establecido, la cual consiste en que los números más grandes del arreglo se encuentren precisamente al final del mismo, obligando al algoritmo a pasar y realizar la comparación por todos los elementos, dando como resultado la siguiente función

$$\begin{aligned}
 f_t(n) = & 2 \text{ comparaciones} + 1 \text{ asignación} \\
 & + 1 \text{ comparación} \\
 & + 2 \text{ asignaciones} \\
 & + (n - 3) \text{ comparaciones} \\
 & + 3(n - 3) \text{ Asignaciones}
 \end{aligned}$$

$$f_t(n) = 6 + 4n - 12 - 1 = 4n - 7$$

Podemos observar como se realiza la extracción en una unidad a la última operación para obtener la función final, debido a que nuestro ciclo While incluye la iteración con igualdad a n y sabemos que un arreglo es de tamaño 0---n-1, por lo cual en ese lugar no hay existencia de elemento.

Caso Medio

```
176  
177 + if (A[1] > A[2] && A[1] > A[3]){ //Operacion-CM  
178 +   m1 = A[1]; //Operacion-CM  
179 +  
180 +   if (A[2] > A[3]){ //Operacion-CM  
181 +     m2 = A[2]; //Operacion-CM  
182 +     m3 = A[3]; //Operacion-CM  
183 +   }  
184 + } else{ ...  
191 + }  
192 + else if (A[2] > A[1] && A[2] > A[3]){ ...  
213 + } else{ ...  
234  
235 + i = 4;  
236  
237 + while( i <= n ){  
238 +  
239 +   if (A[i] > m1){ //Operacion-CM  
240 +     m3 = m2; //Operacion-CM  
241 +     m2 = m1; //Operacion-CM  
242 +     m1 = A[i]; //Operacion-CM  
243 +   }  
244 +   else if (A[i] > m2){ //Operacion-CM  
245 +     m3 = m2; //Operacion-CM  
246 +     m2 = A[i]; //Operacion-CM  
247 +   }  
248 +   else if (A[i] > m3){ //Operacion-CM  
249 +     m3 = A[i]; //Operacion-CM  
250 +   }  
251 + } else{ ...  
252 + }
```

Para sacar una media debemos considerar la mayor cantidad de rutas posibles, desde la perspectiva que hemos tomado y con la contabilización de las operaciones básicas establecidas en un inicio, he llegado a la siguiente función que nos determinara un resultado satisfactorio

1) Todos son false (Mejor Caso)

$$f_t(n) = 3n - 4$$

2. La primera comparación es true lo demás false

$$f_t(n) = 4n - 8$$

Pues contamos con las siguientes operaciones:

- 1 comparación
- 1 asignacion
- 1 comparacion
- 2 asignaciones
- (n-3) comparaciones
- 3(n-3) Asignaciones -1

3. Segunda Condición true el resto falso

$$f_t(n) = 4n - 8$$

- 1 comparacion
- 1 Asignacion
- 1 comparacion
- 2 asiganaciones
- 2(n-3) comparaciones
- 2(n-3) asignaciones -1

4. Tercera condición true, el resto falso

$$f_t(n) = 4n - 8$$

- 1 comparacion
- 1 Asignacion
- 1 comparacion
- 2 asignaciones
- 3(n-3) comparaciones
- (n-3) asignaciones -1

```
182  
183 > if (A[1] > A[2] && A[1] > A[3]){ //Operacion_CM  
188 else if (A[2] > A[1] && A[2] > A[3]){ //Operacion_CM  
199     m1 = A[2]; //Operacion_CM  
200  
201     if (A[1] > A[3]){ //Operacion_CM  
202  
203         m2 = A[1]; //Operacion_CM  
204         m3 = A[3]; //Operacion_CM  
205     }  
206 > else{  
213 }  
214 > else{  
235     i = 4;  
236  
237     while( i <= n ){  
238  
239         if (A[i] > m1){ //Operacion_CM  
240  
241             m3 = m2; //Operacion_CM  
242             m2 = m1; //Operacion_CM  
243             m1 = A[i]; //Operacion_CM  
244         }  
245         else if (A[i] > m2){ //Operacion_CM  
246             m3 = m2; //Operacion_CM  
247             m2 = A[i]; //Operacion_CM  
248         }  
249         else if (A[i] > m3){ //Operacion_CM  
250             m3 = A[i]; //Operacion_CM  
251         }  
252     }
```

Si nos ponemos a observar el segundo camino posible será muy similar al primero, solo que en lugar de entrar en la primera comparativa como verdadero, será falso por lo que en la segunda también, debido a que son dependientes por lo que entrará a nuestro caso ELSE

1. Todos son FALSE (Peor Caso)

$$f_t(n) = 4n - 7$$

2. Primera comparativa true, el resto false.

$$f_n(n) = 4n - 7$$

Ya que contamos con:

- 2 Comparaciones
- 1 Asignación
- 1 Comparación
- 2 Asignaciones
- (n-3) Comparaciones
- 3(n-3) Asignaciones

3. Segunda condición true

$$f_t(n) = 4n - 7$$

- 2 Comparaciones
- 1 Asignación

- 1 Comparación
- 2 Asignaciones
- $2(n-3)$ Comparaciones
- $2(n-3)$ Asignaciones -1

4. Tercera condición true

$$f_t(n) = 4n - 7$$

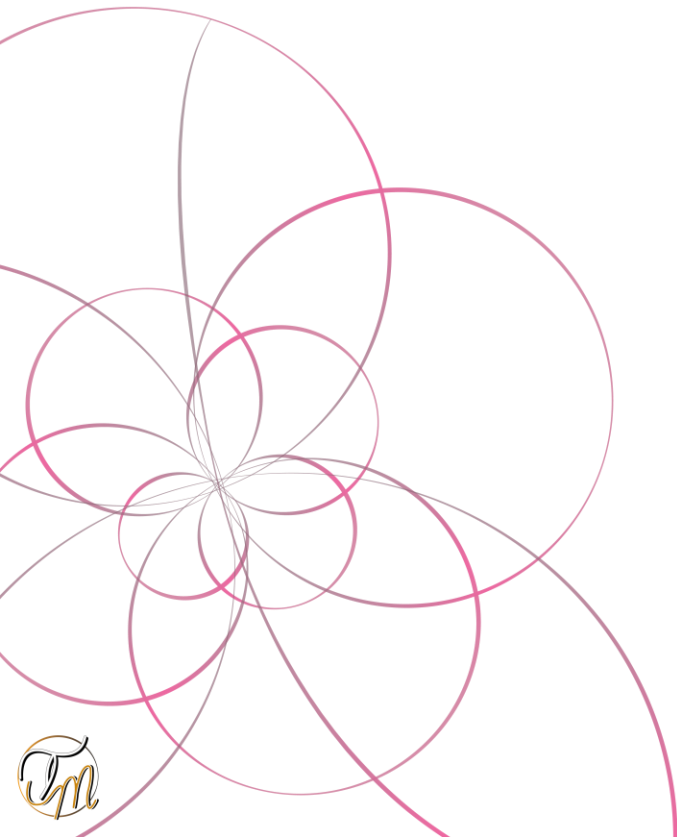
- 2 Comparaciones
- 1 Asignación
- 1 Comparación
- 2 Asignaciones
- $3(n-3)$ Comparaciones
- $(n-3)$ Asignaciones -1

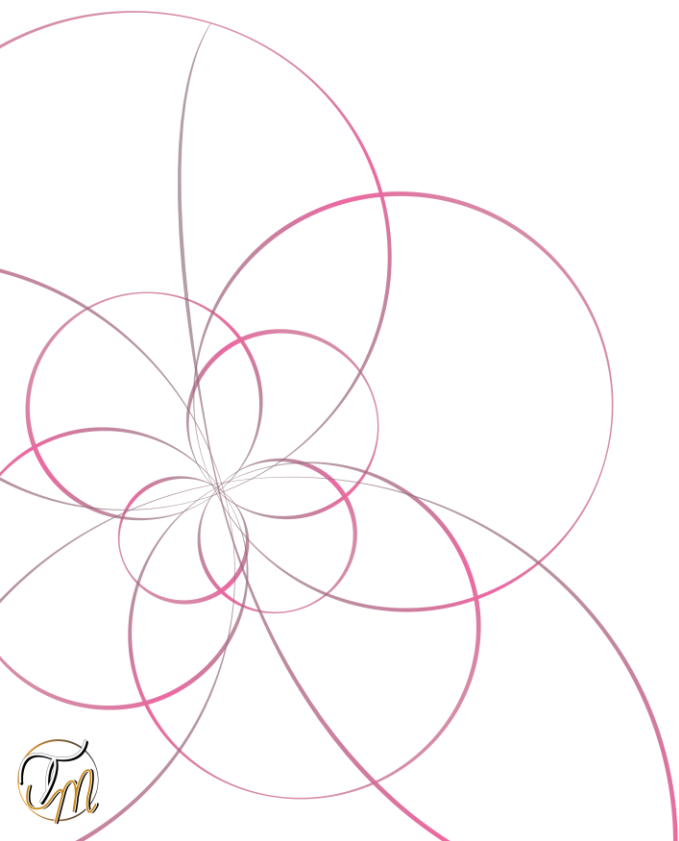
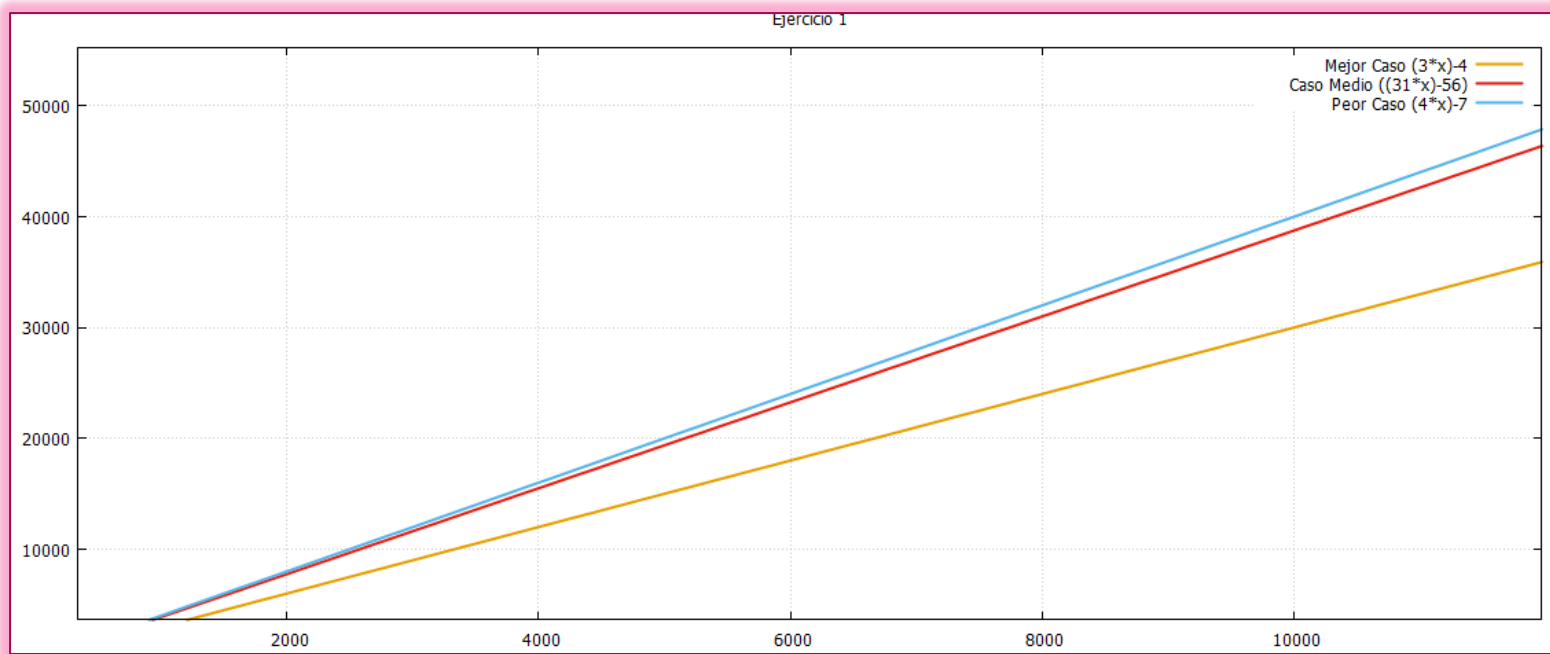
Por último, recordemos que cada camino que pudiese tomar es considerado como equiprobable por lo que nuestra función quedaría expresada de la siguiente forma:

$$f(n) = \frac{1}{8}(3n - 4 + 3(4n - 8) + 4(4n - 7))$$

$$f(n) = \frac{1}{8}(3n - 4 + 12n - 24 + 16n - 28)$$

$$f(n) = \frac{31n - 56}{8}$$





CÓDIGO 2

```
int MaximoComunDivisor(m, n){  
    a=max(n,m);  
    b=min(n,m);  
    residuo=1;  
    while (residuo>0)  
    {  
        residuo = a%b;  
        a=b;  
        b=residuo;  
    }  
    MaximoComunDivisor=a;  
    return MaximoComunDivisor;  
}
```

Para este caso en particular, la operación básica asignada a seguir fue aquella donde estuviera involucrada la operación de modulo entre a y b, la cual esta asignada a la variable de residuo, establecido esto, podemos proceder a realizar el análisis de los casos

MEJOR CASO

El mejor caso será aquel donde encontremos que las dos entradas dadas para el algoritmo sean múltiplos, por ejemplo: 5 y 10 , pues el modulo siempre será 0 y solo se realizaría una vez dicha operación

$f_t(n, m) = 1$ si y solo si $MCD(n, m) = n$ Donde n es múltiplo de m y a su vez menor que m

PEOR CASO

El peor caso será aquel donde los números ingresados, sean 2 números consecutivos pertenecientes a la serie de Fibonacci, mientras más grandes sean los valores de esta serie mayor será el numero de operaciones a realizar

Gracias a las fuentes consultas en internet acerca del algoritmo en cuestión (Algoritmo de Euclides) estará establecida aproximadamente por la siguiente formula:

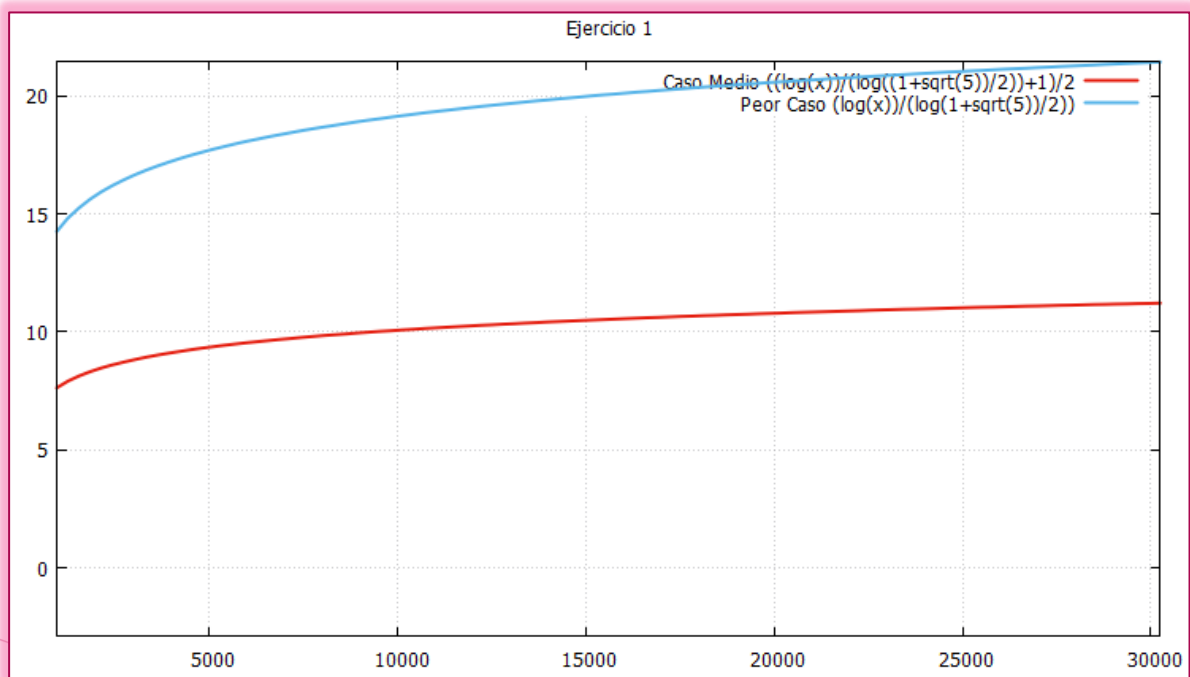
$$f_t(a) = \frac{\log a}{\log \Phi} \text{ donde } a \geq b \text{ y } \Phi = \frac{1+\sqrt{5}}{2}$$

CASO MEDIO

Dado el análisis anterior pude notar que solo tendríamos la posibilidad de que el MCD únicamente se haga en una operación o que se haga x cantidad de veces, lo cual sería la expresión del peor caso, por lo que propongo la siguiente expresión para esta cuestión:

$$f(n, m) = \frac{1}{2} \left(\frac{\log(a)}{\log(\phi)} + 1 \right) \text{ donde } a = \min(n, m) \text{ [1] [2] [3]}$$

GRÁFICA



CÓDIGO 3

```
int SumaCuadratica3MayoresV2(){
    scanf("%d", &n);
    A = malloc(sizeof(int)*n);
    for (j = 0; j < n; j++){

        A[j]= rand()%(32000);
    }
    for (i = 0; i < 3; i++)
    {
        for (j = 0; j < n-1-i ; j++)
        {
            if(A[j]>A[j+1]){
                aux=A[j];
                A[j]=A[j+1];
                A[j+1]=aux;
            }
        }
    }
    r=A[n-1]+A[n-2]+A[n-3];
    return pow(r,2);
}
```

MEJOR CASO

$$f_t(n) = 3n - 5$$

Dado que tendríamos la entrada al ciclo de i y la entrada al ciclo de j, si la condición no se cumple tendremos como consecuencia lo siguiente:

- (n-1) Comparaciones
- (n-2) Comparaciones
- (n-3) Comparaciones
- 1 Asignación

PEOR CASO

$$f_t(n) = 12n - 23$$

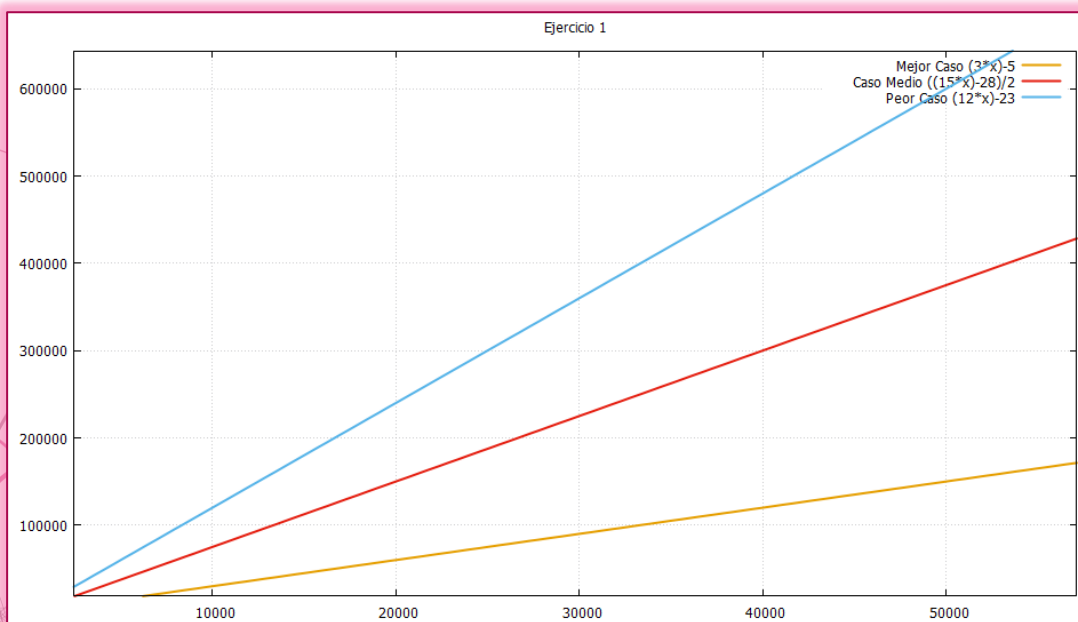
El peor caso estará dado cuando los números estén ordenados de forma descendente, por lo que se encontraran en las ultimas posiciones, dando como consecuencia que la condicional sea efectiva y las asignaciones realizadas teniendo como resultado las siguientes operaciones:

- (n-1) Comparaciones
 - 3(n-1) Asignaciones
- (n-2) Comparaciones
 - 3(n-2) Asignaciones
- (n-3) Comparaciones
 - 3(n-3) Asignaciones
- 1 Asignación

CASO MEDIO

$$f_t(n) = \frac{15n - 28}{2}$$

Para el caso medios consideraron las rutas analizadas anteriormente, tomando aquellas donde sucede en el mejor caso junto con el peor caso, realizando una división entre 2.



CÓDIGO 4

```
int Codigo4(){
int f,i,j,num,ftemp,ntemp,n,A[n];
scanf("%d",&n);
i=1;
while (i<=n)
{
    scanf("%d",A[i]);
    i+=1;
}
f=0;
i=1;
while (i<=n)
{
    ntemp=A[i];
    j=1;
    ftemp=0;
    while (j<=n)
    {
        if (ntemp=A[j])
        {
            ftemp=ftemp+1;
        }
        j=j+1;
    }

    if (f<ftemp)
    {
        f=ftemp;
        num=ntemp;
    }
    i=i+1;
}
printf("d",num);
}
```

Para este código se tendrá en consideración las siguientes operaciones:

- Aquellas que tienen que ver con el tamaño del problema (ósea que están relacionadas directamente con el arreglo)
- Variables que determinan la cantidad de operaciones que se van a realizar, las cuales están dadas por **ftemp** y **ntemp**

MEJOR CASO

Para este caso tendremos un arreglo donde los números que este contiene nunca se repitan, cada valor es único dentro del arreglo de tamaño n, de tal forma que se evitaría la realización



de operaciones a la variable ftemp, además de que no siempre se realizaría la asignación de num en el IF que esta fuera del ciclo WHILE de j por lo que tendremos:

$$f_t(n) = n^2 + 4n + 2$$

- 2 Asignaciones
- (n) Asignaciones + (n) Asignaciones
- (n^2) Comparaciones + (n) Asignaciones +
- (n) Comparaciones + 2 Asignaciones

PEOR CASO

Para este caso tendremos la situación de un arreglo lleno del mismo numero una n cantidad de veces, dada esta situación la condicional dentro del while siempre será cumplida

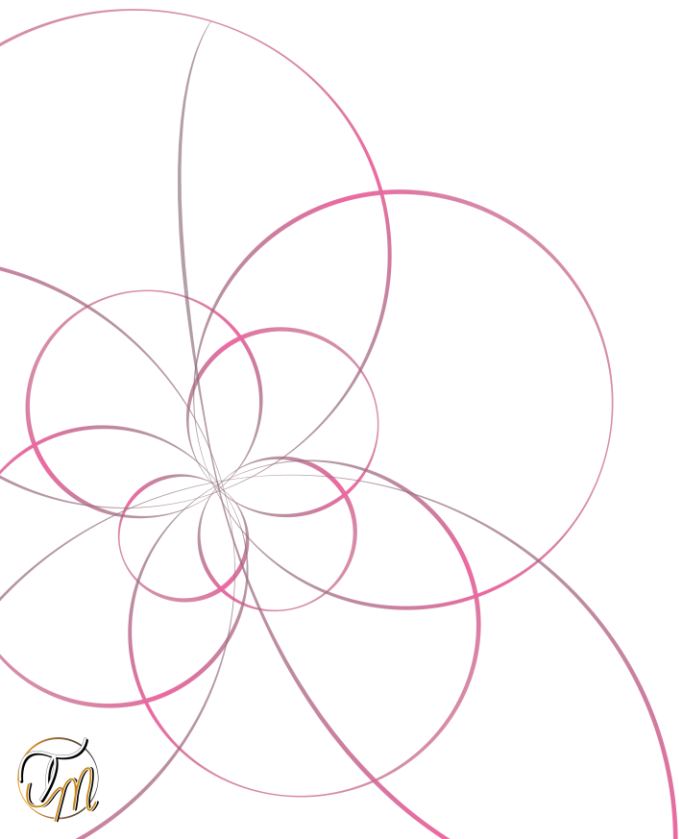
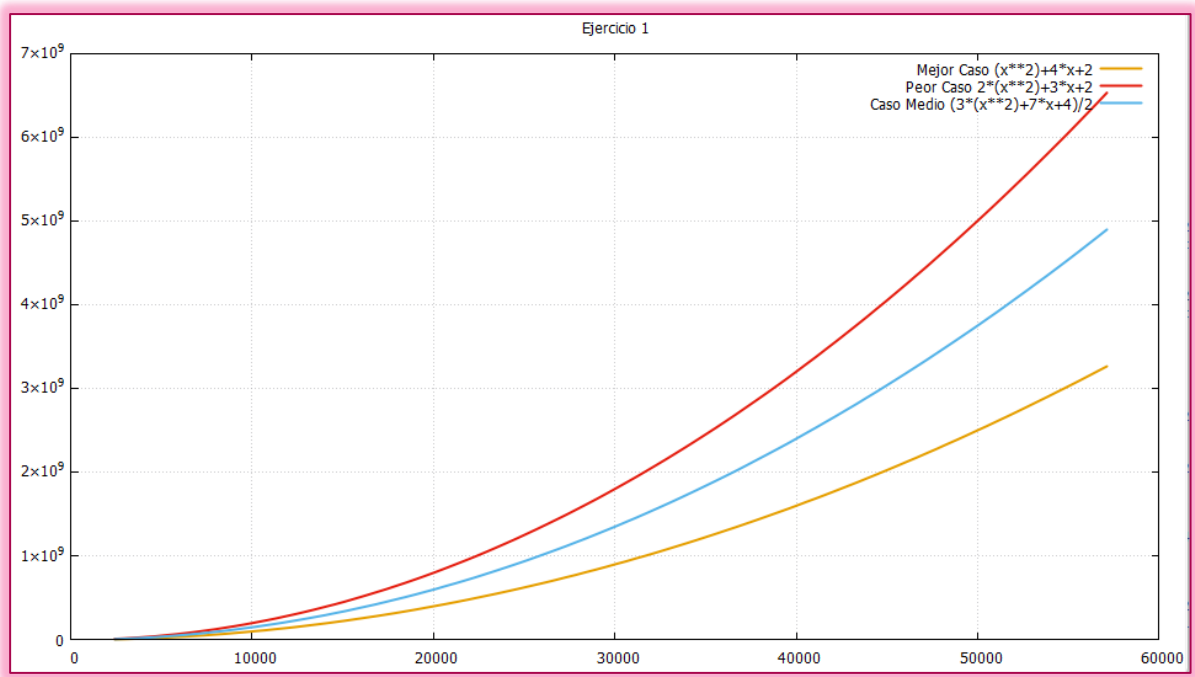
$$f_t(n) = 2n^3 + 3n + 2$$

- n Asignaciones + (n) Asignaciones + (n^2) Comparaciones
- (n^2) Comparaciones + (n) Asignaciones + 2 Asignaciones

CASO MEDIO

Para este caso, puede llegar a ser algo complejo determinar un caso medio debido a la naturaleza que posee el objetivo del código y realmente no existen muchas alternativas posibles que seguir o considerar, por lo que realizaremos la unión del mejor y peor caso, dando una división entre 2, estoy consciente de que probablemente no es la mejor opción, pero me parece una buena alternativa para encontrar una buena solución con resultados muy cercanos a la realidad de ejecución en pruebas a posteriori.

$$f_t(n) = \frac{3n^2 + 7n + 4}{2}$$



CICLO 5

```
polinomio=0;

for (i = 0; i <= n; i++)
{
    polinomio=polinomio*z + A[n-i];
    cont+=1;
}
printf("%d\n",polinomio);
printf("%d",cont);
```

MEJOR CASO

$$f_t(n) = 1 \rightarrow \text{Cuando } n = 0$$

$$f_t(n) = n + 1$$

Con respecto a las operaciones previamente establecidas, he determinado que el mejor caso será aquel cuando $n=0$ o $n=1$ ya que si le damos un arreglo vacío la operación será realizada únicamente en una iteración, aunque son muy poco probables aquellas situaciones en las que un arreglo será usado de forma vacía precisamente con la intención de operar, por lo que lo mejor sería poder mandar un arreglo con al menos un elemento.

PEOR CASO

El peor caso desde mi perspectiva sería aquel donde el tamaño del arreglo sea mayor que 1, pues las operaciones que se irían realizando serían cada vez más y más, por lo que el tamaño máximo que aguantaría sería aquellas donde sea del tamaño máximo del tipo de variable con la cual el arreglo en cuestión fue declarado.

CASO MEDIO

Con respecto a lo anteriormente expuesto el caso medio como tal no podría estar determinado, pues, dependemos directamente del valor de "n".

BIBLIOGRAFÍA

- [1] M. R. Castañeda, «Super Prof Material Didactico,» 18 08 2017. [En línea]. Available:
<https://www.superprof.es/apuntes/escolar/matematicas/aritmetica/divisibilidad/algoritmo-de-euclides.html>. [Último acceso: 17 Septiembre 2021].
- [2] UNAM, «Instituto de Matemáticas UNAM,» 21 Febrero 2009. [En línea]. Available:
] <https://paginas.matem.unam.mx/cprieto/biografias-de-matematicos-a-e/198-euclides>.
[Último acceso: 17 09 2021].
- [3] G. Brassard y P. Bratley, Fundamentos de Algoritmia., Madrid: PrenticeHall, 1997.
]

