



**INSTITUTO POLITÉCNICO NACIONAL  
ESCUELA SUPERIOR DE CÓMPUTO**

**ANÁLISIS DE ALGORITMOS**

**PROFESORA: LUZ MARÍA SÁNCHEZ GARCÍA**

**INTEGRANTES:**

**VÁZQUEZ MORENO MARCOS OSWALDO 2016601777**

**PRÁCTICA 7 ANÁLISIS DE ALGORITMOS DIVIDE Y VENCERÁS  
FIBONACCI BOTTOM UP AND TOP DOWN**

**3CM2**

**10 DE ABRIL DE 2019**

## Introducción

En cuanto a la programación dinámica al igual que los métodos divide-and-conquer resuelve problemas combinando las soluciones de subproblemas. Fue creada por Richard Bellman en 1953. Retomando, los métodos divide-and-conquer dividen el problema en subproblemas independientes, resuelven el problema de manera recursiva y combinan sus soluciones.

La programación dinámica se aplica cuando los subproblemas no son independientes: cuando los subproblemas comparten subsubproblemas (donde hay traslape). Los métodos divide-and-conquer en este caso hacen más trabajo del necesario, un algoritmo de programación dinámica resuelve cada subproblema una sola vez y guarda la solución en una tabla.

El problema se puede dividir en subproblemas cuya solución puede ser utilizada varias veces. Esto se relaciona estrechamente con la recursión.

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ \prod_{k=1}^n k & \text{if } n > 0 \end{cases}$$

Programación dinámica bottom-up: resuelve todas las instancias a partir de las más chicas hasta la que se desea calcular.

```
F[0]=0;
F[1]=1;
for( int i=2; i<=n; i++)
    F[i] = F[i-1] + F[i-2];
```

O(n)

Programación dinámica top-down: esquema recursivo que expresa el problema como subproblemas y memoriza los resultados para reutilizarlos más tarde. (Demaine, 2011)

```
int F(int i){
    static int knownF[maxN];
    if(knownF[i] != 0 return knownF[i];
    int t=i;
    if (i<0) return 0;
    if (i>1) t = F(i-1)+F(i-2);
    return knownF[i] = t;
}
```

O(n)

## Planteamiento del problema

Entender e implementar los algoritmos basados en programación dinámica de Fibonacci top down y bottom up.

Descibir de manera detallada los algoritmos y su implementación.

Encontrar para cada algoritmo que refleje de mejor manera su comportamiento temporal y espacial.

Indicar cotas  $O$ .

Mostrar gráficamente la comparativa de las aproximaciones de cada algoritmo.

## Diseño de la solución

A continuación, se muestran los diagramas de flujo de nuestra propuesta de solución para los algoritmos de Fibonacci bottom-up y top down

Primeramente, se muestra en el diagrama 1.1 el fibo\_bottom.c de nuestra propuesta de solución para bottom-up.

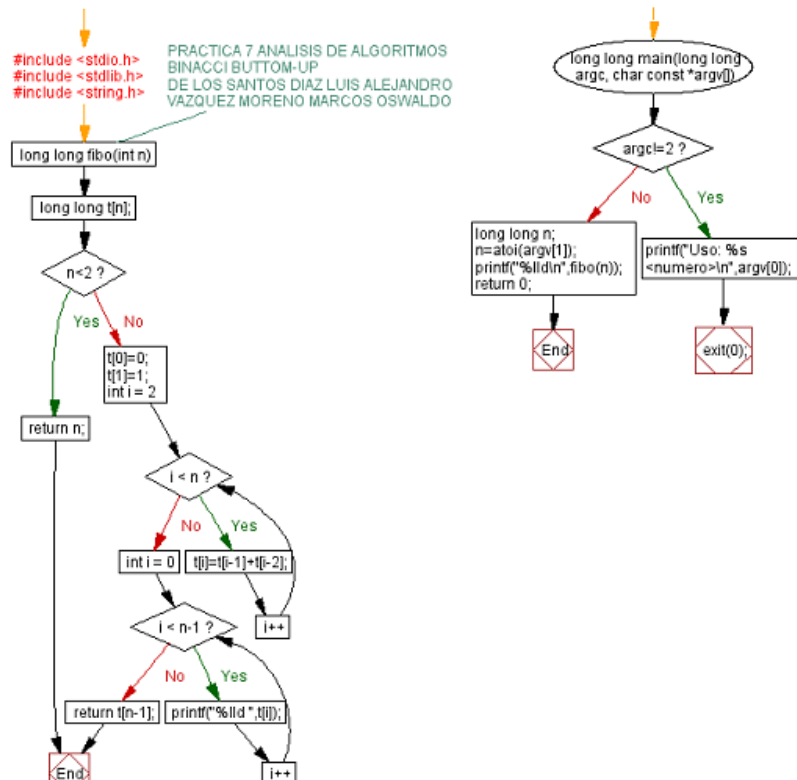


Diagrama 1.1 Fibo\_bottom.c

Después, se muestra en el diagrama 1.2 el fibo\_topdown.c de nuestra propuesta de solución para top down.

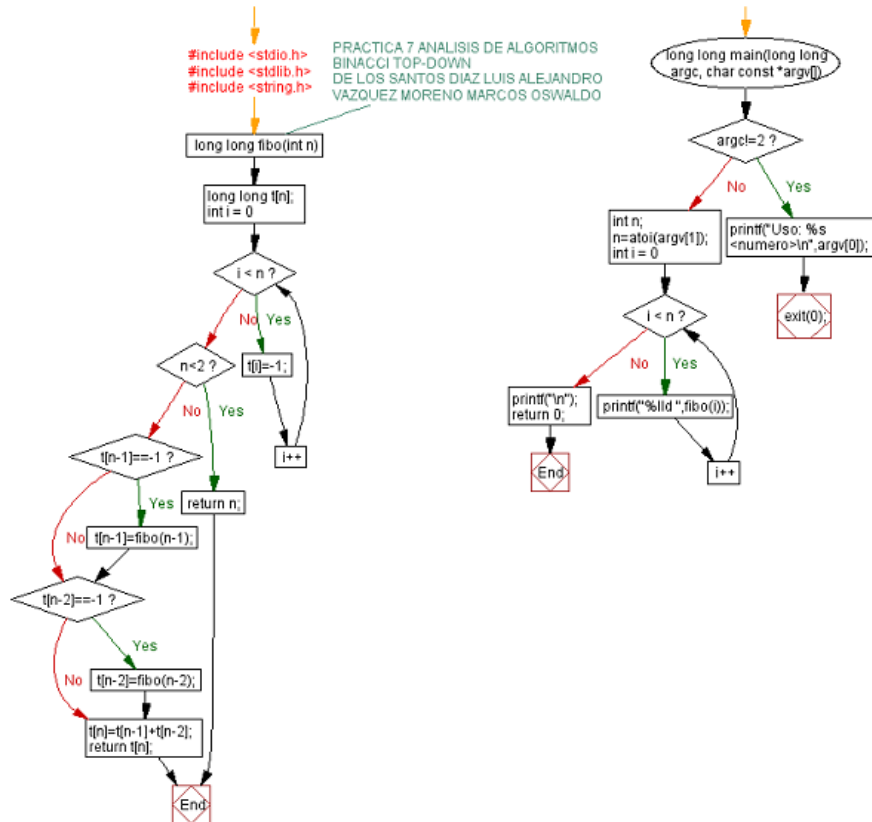


Diagrama 1.2 Fibo\_topdown.c

## Implementación de la solución

### Fibo\_bottom.c

```

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. /*
6. PRACTICA 7 ANALISIS DE ALGORITMOS
7. BINACCI BOTTOM-UP
8.
9. DE LOS SANTOS DIAZ LUIS ALEJANDRO
10. VAZQUEZ MORENO MARCOS OSWALDO
11.
12. */
13.
14.
15. long long fibo(int n){
16.     long long t[n];
17.
18.     if (n<2) {
19.         return n;
20.     }else{

```

```

21.     t[0]=0;
22.     t[1]=1;
23.     for(int i = 2; i < n; i++)
24.     {
25.         t[i]=t[i-1]+t[i-2];
26.     }
27.     for(int i = 0; i < n-1; i++)
28.     {
29.         printf("%lld ",t[i]);
30.     }
31.
32. }
33.
34.     return t[n-1];
35.
36. }
37.
38. long long main(long long argc, char const *argv[]){
39.     if (argc!=2) {
40.         printf("Uso: %s <numero>\n",argv[0]);
41.         exit(0);
42.     }
43.
44.     long long n;
45.     n=atoi(argv[1]);
46.
47.     printf("%lld\n",fibonacci(n));
48.     return 0;
49. }

```

### Fibo\_topdown.c

```

1. #include <stdio.h>
2. #include <stdlib.h>
3. #include <string.h>
4.
5. /*
6. PRACTICA 7 ANALISIS DE ALGORITMOS
7. BINACCI TOP-DOWN
8.
9. DE LOS SANTOS DIAZ LUIS ALEJANDRO
10. VAZQUEZ MORENO MARCOS OSWALDO
11.
12.
13. */
14. long long fibo(int n){
15.     long long t[n];
16.     for(int i = 0; i < n; i++)
17.     {
18.         t[i]=-1;
19.     }
20.
21.     if (n<2) {
22.         return n;
23.     }
24.
25.     if (t[n-1]==-1) {
26.         t[n-1]=fibo(n-1);
27.     }

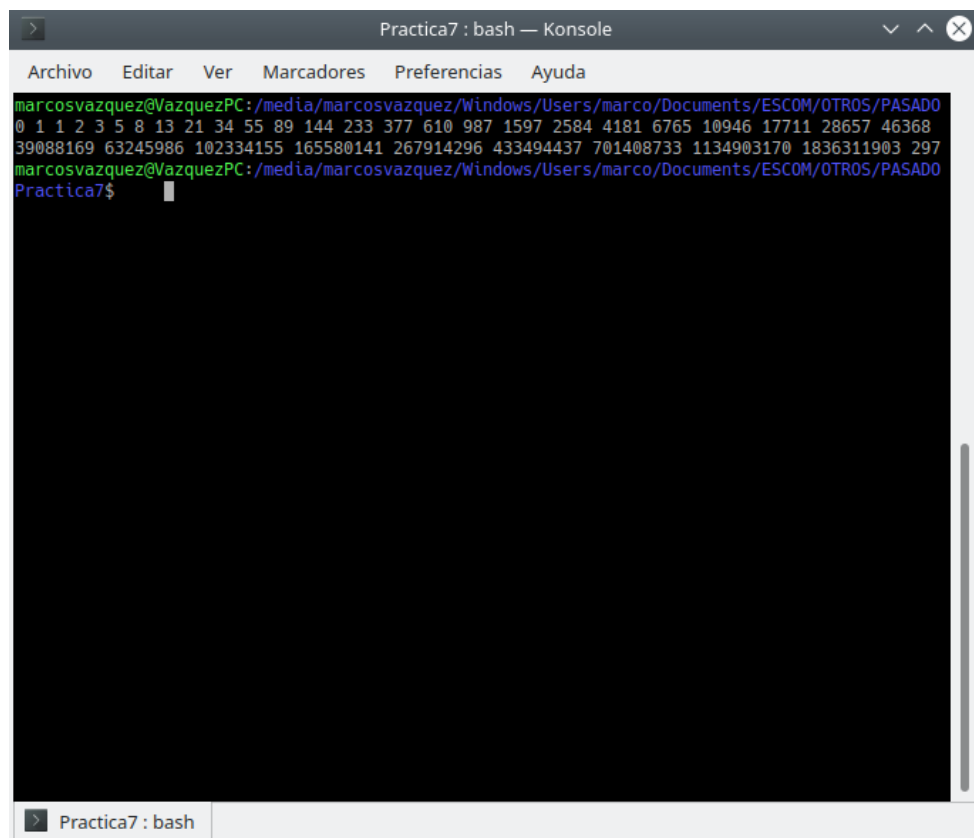
```

```

28.
29.     if (t[n-2]==-1) {
30.         t[n-2]=fibonacci(n-2);
31.     }
32.
33.     t[n]=t[n-1]+t[n-2];
34.
35.     return t[n];
36.
37. }
38.
39. long long main(long long argc, char const *argv[]){
40.     if (argc!=2) {
41.         printf("Uso: %s <numero>\n",argv[0]);
42.         exit(0);
43.     }
44.
45.     int n;
46.     n=atoi(argv[1]);
47.     for(int i = 0; i < n; i++)
48.     {
49.         printf("%lld ",fibonacci(i));
50.     }
51.     printf("\n");
52.
53.     return 0;
54. }

```

## Funcionamiento

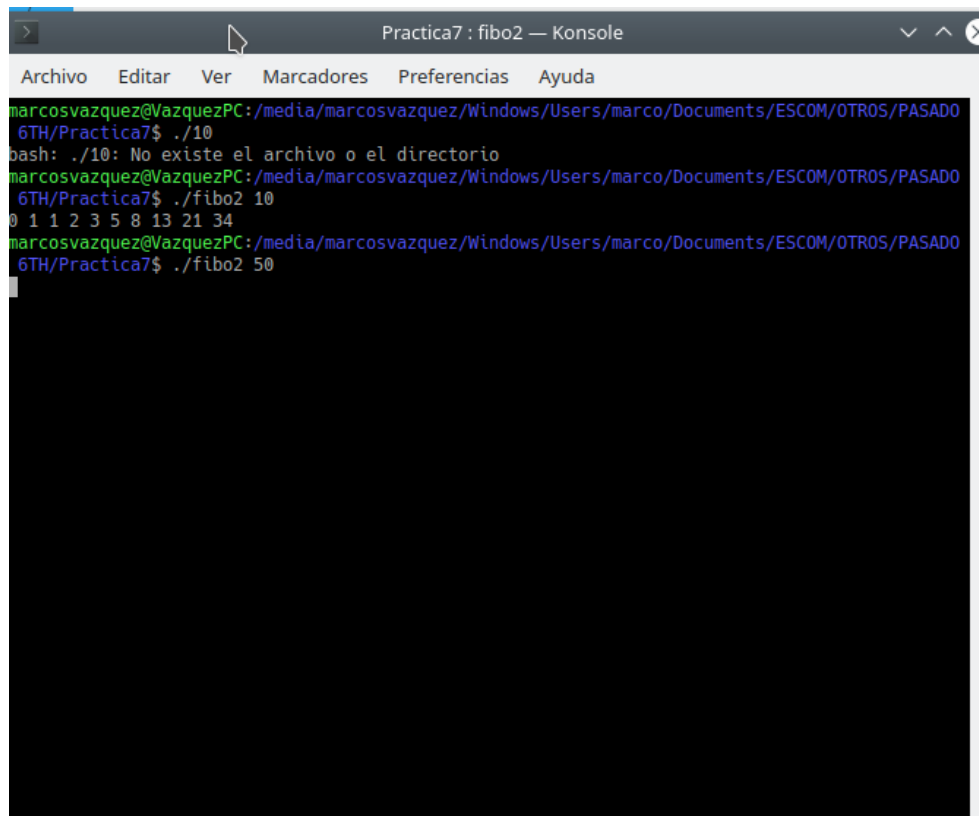


```

Practica7 : bash — Konsole
Archivo  Editar  Ver  Marcadores  Preferencias  Ayuda
marcosvazquez@VazquezPC: /media/marcosvazquez/Windows/Users/marco/Documents/ESCOM/OTROS/PASADO
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
75025 121393 196418 317811 514229 832040
marcosvazquez@VazquezPC: /media/marcosvazquez/Windows/Users/marco/Documents/ESCOM/OTROS/PASADO
Practica7$

```

Imagen 1.1 Fibo\_bottom.c



```
marcosvazquez@VazquezPC:/media/marcosvazquez/Windows/Users/marco/Documents/ESCOM/OTROS/PASADO
6TH/Practica7$ ./10
bash: ./10: No existe el archivo o el directorio
marcosvazquez@VazquezPC:/media/marcosvazquez/Windows/Users/marco/Documents/ESCOM/OTROS/PASADO
6TH/Practica7$ ./fibo2 10
0 1 1 2 3 5 8 13 21 34
marcosvazquez@VazquezPC:/media/marcosvazquez/Windows/Users/marco/Documents/ESCOM/OTROS/PASADO
6TH/Practica7$ ./fibo2 50
```

Imagen 1.2 Fibo\_topdown.c

## Plataforma experimental

La ejecución de los algoritmos anteriores se llevó a cabo en una computadora personal que se describe en la siguiente imagen.

Especificaciones del dispositivo	
HP Laptop 15-bs0xx	
Nombre del dispositivo	LAPTOP-13V7QO38
Procesador	Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz 2.00 GHz
RAM instalada	8.00 GB
Id. del dispositivo	ACEA8FAF-1F85-49D4-BC5D-A7059ECE254D
Id. del producto	00327-30000-00000-AAOEM
Tipo de sistema	Sistema operativo de 64 bits, procesador x64
Lápiz y entrada táctil	La entrada táctil o manuscrita no está disponible para esta pantalla

Imagen 3.1 Plataforma Experimental

El compilador utilizado fue gcc integrado dentro del IDE DevC en un sistema operativo de 64 bits Windows 10.

## Gráfica del comportamiento temporal.

En cuanto al Bottom-up la función Temporal y la Espacial son:

Temporal

$$T(n) = 9n - 1$$

Espacial

$$F(n) = 1$$

En cuanto al Top-Down la función Temporal y la Espacial son:

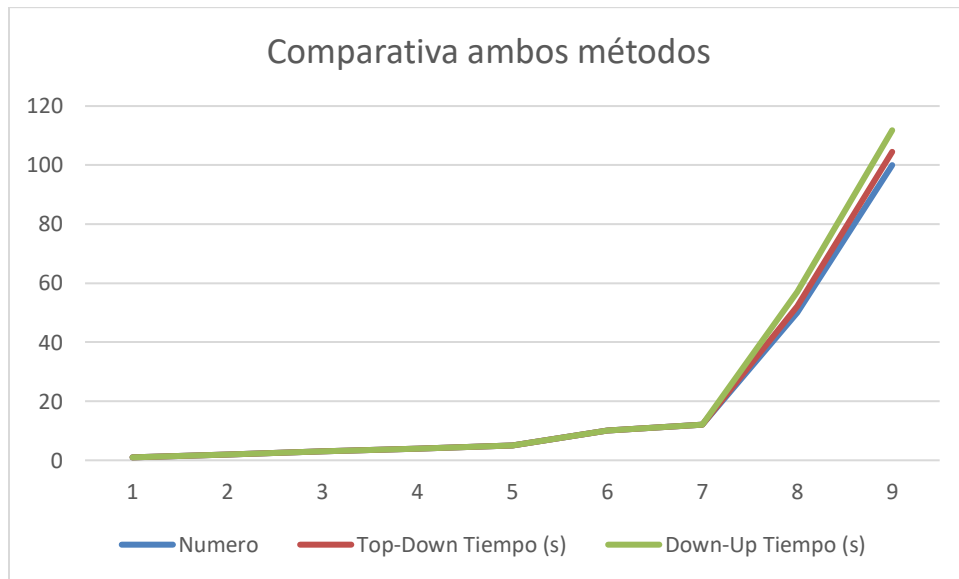
Temporal

$$T(n) = 3n + 21 = n + 7$$

Espacial

$$F(n) = 2$$

Las cotas mayores se muestran en la parte de la introducción donde menciona que es  $O(n)$ .



Gráfica 1.1

## Cuestionario

1. ¿Cuál de los 2 algoritmos es más fácil de implementar? Top-Down debido a que la forma de solución del algoritmo es más sencilla.
2. ¿Cuál de los 2 algoritmos es el más difícil de implementar? Bottom-Down



3. **¿Cuál algoritmo tiene menor complejidad temporal?** Top-Down
4. **¿Cuál algoritmo tiene mayor complejidad temporal?** Bottom-down
5. **¿El comportamiento experimental de los algoritmos era el esperado? ¿Por qué?** No, el documento que leí dice que el Bottom-Down es de menor complejidad.
6. **¿Sus resultados experimentales difieren mucho de los análisis teóricos que realizo? ¿A qué se debe?** Sí, es necesario un análisis teórico y práctico para tener un poco en cuenta cómo es que se va a comportar
7. **¿Existió un entorno controlado para realizar las pruebas experimentales? ¿Cuál fue?** Pruebas de escritorio y pruebas de fuerza bruta.
8. **¿Si solo se realizará el análisis teórico de un algoritmo antes de implementarlo, podrías asegurar cual es el mejor?** No, siempre es necesario un análisis práctico.
9. **¿Qué tan difícil fue realizar el análisis teórico de cada algoritmo?** No tan complicado, tomando en cuenta que tenemos un soporte de lectura el cuál apoya en el desarrollo del Algoritmo
10. **¿Qué recomendaciones darían a nuevos equipos para realizar esta práctica?** Hacer una buena búsqueda de información para tener en cuenta qué tan complejo puede ser la implementación del algoritmo

## **Conclusiones**

En conclusión, la programación dinámica encargada de problemas de optimización, mismos que pueden tener varias soluciones resultó bastante efectiva para esta práctica, sin embargo, no sabía que con programación dinámica se podía resolver, solo sabía de recursividad, claro que se debe ocupar para estos casos, pero bueno cada vez se aprende algo nuevo, algo que debemos de tomar en cuenta es que el hecho de implementar la solución de abajo para arriba es mejor ya que se tarda menos tiempo.

## Bibliografía

### Bibliografía

- Academy, K. (15 de 6 de 2006). *Algoritmo Divide y Vencerás* . Obtenido de <https://es.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms>
- Campos, J. (07 de Abril de 2019). *Algoritmia básica*. Obtenido de Algoritmia básica (Universidad de Zaragoza): <http://webdiis.unizar.es/asignaturas/AB/material/3-Divide%20y%20venceras.pdf>
- Demaine, P. E. (15 de 08 de 2011). *Introductions to Algorithms*. Obtenido de Lecture 19: Dynamic Programming: <http://home.iitk.ac.in/~swarnam/cs300/5.pdf>
- García, L. M. (15 de 03 de 2019). *Classrom*. Obtenido de Práctica 7 Ánlisis de Algoritmos: <https://classroom.google.com/c/NzEzNTUxMDU1NVpa/a/NzgxNTE2NTA3MVpa/de tails>