INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

PRÁCTICA 4. Programación Dinámica

ALUMNOS: Ramírez Jiménez Itzel Guadalupe

Colín Ramiro Joel

GRUPO: 3CM3

PROFESORA: Sánchez García Luz María

MATERIA: Análisis y Diseño de Algoritmos

Planteamiento del problema

Existe una serie de problemas cuyas soluciones pueden ser expresadas recursivamente en términos matemáticos, y posiblemente la manera más natural de resolverlos es mediante un algoritmo recursivo. Sin embargo, el tiempo de ejecución de la solución recursiva, normalmente de orden exponencial y por tanto impracticable, puede mejorarse substancialmente mediante la Programación Dinámica.

En esta práctica se analizaron 4 algoritmos que se pueden resolver mediante la programación dinámica, se compararon y se llegó a una conclusión entre cual de los 4 es más eficiente y cuál el más ineficiente mediante esta estrategia.

Actividades

Fibonacci (Top down) y (Button-up)

Código

Top down

```
#include<stdio.h>
int Fibonacci (int N){
    if(N <= 1)
        return N;
    return Fibonacci(N-1) + fibonacci(N-2);
}
int main(){
    int n;
    printf("Ingrese la posición de la serie: ");
    scanf("%d",&n);
    printf("Fib(%d) = %d\n",n,fibonacci(n));

    return 0;
}
```

Button-up

```
#include <stdio.h>
int fibonacci(int n){

    int fib[n+1],i;
    fib[0] = 0;
    fib[1] = 1;

    for(i = 2; i <= n; i++){
            fib[i] = fib[i-1] + fib[i-2];
            }
    return fib[n];
}
```

```
int main(){
    int n;
    printf("Ingrese la posición de la serie: ");
    scanf("%d",&n);

    if(n <= 1)
        printf("fib(%d) = %d\n",n,n);
    else
        printf("fib(%d) = %d\n",n,fibonacci(n));

    return 0;
}
```

¿En qué consiste?

La serie de Fibonacci es una secuencia de números enteros donde el siguiente entero de la serie es la suma de los dos anteriores.

- ***Top down***

La idea aquí es similar al enfoque recursivo, pero la diferencia es que se deben guardar las soluciones a los subproblemas que sean encontrados.
De esta manera, si se topa con el mismo subproblema más de una vez, se puede usar la solución guardada en lugar de tener que volver a calcularla. Esto permite calcular cada subproblema exactamente una vez.

- ***Button-up***

En este enfoque, se debe reorganizar el orden en el que se resuelven los subproblemas. Esto a su vez, permitirá calcular la solución a cada problema solo una vez, y solo se necesitará guardar dos resultados intermedios a la vez.

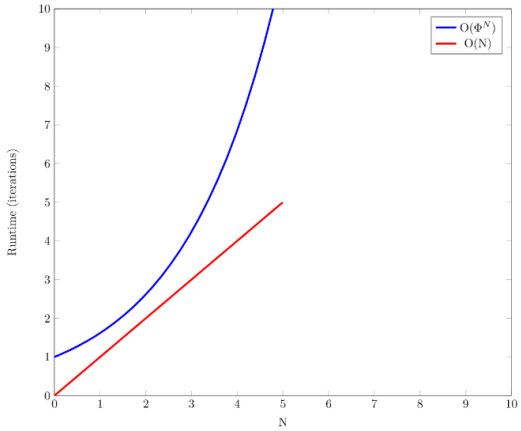
Ecuación de recurrencia

T(n) = n+2

T(n) = 2n+7

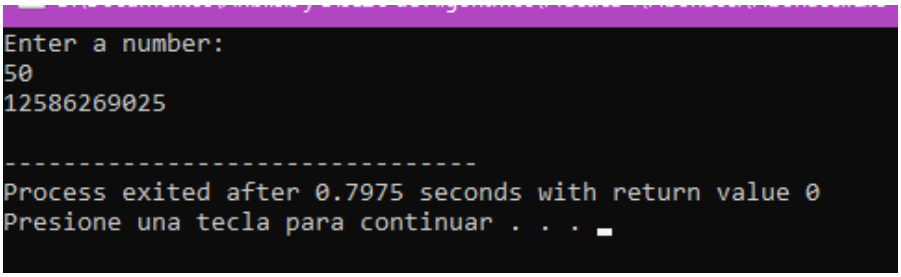
O(n) → Para ambos

Gráfica temporal

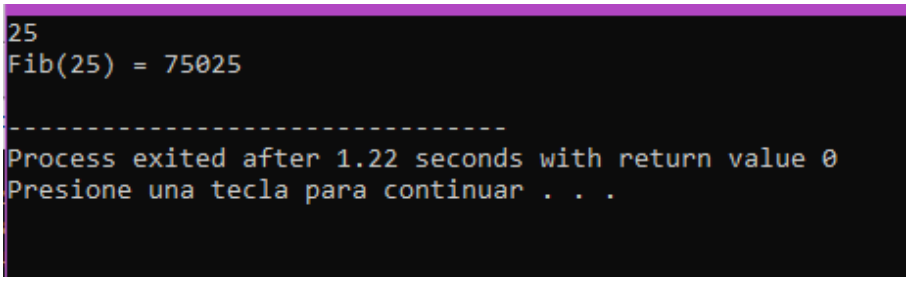


Valores de entrada y resultados

- *Top Down*



- *Bottom-up*



Coeficientes binomiales

Código

```
#include <stdio.h>
#include <iostream>
using namespace std;

int factorial (int);
int combinación (int, int);

int main(){
```

```
	int n;
	printf("Ingrese el valor de n: ");
	scanf("%d",&n);

	for(int i=0;i<=n;i++){
		for(int j=0;j<=i;j++){
			cout<<combinacion(i,j)<<" ";
		}
	}

	return 0;
}

int factorial (int n ){
	int resultado = 1;
	for(int i=1;i<=n;i++){
		resultado += 1;
	}
	return 0;
}

int combinación (int n, int k){
	return (factorial(n))/(factorial(k)*factorial(n-k));
}
```

¿En qué consiste?

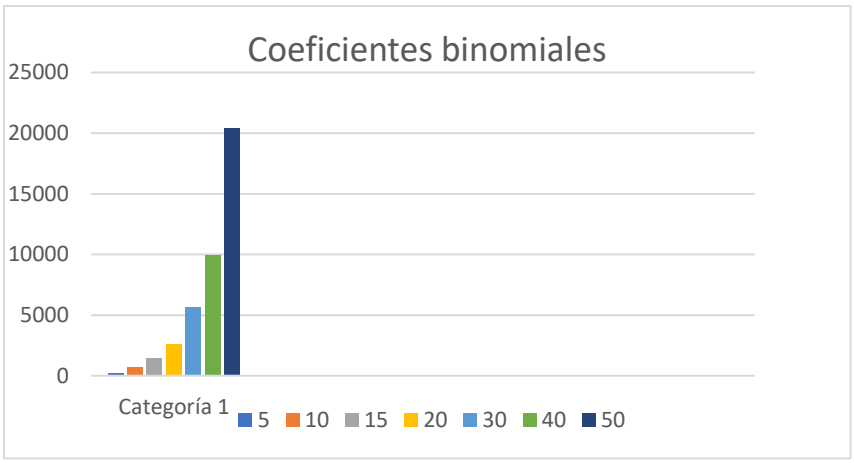
Los coeficientes binomiales se utilizan en combinatoria para calcular combinaciones de elementos, es decir, el número de formas es que se pueden extraer distintos subconjuntos a partir de un conjunto dado, sin importar su orden. El algoritmo que implementa directamente la ecuación recursiva anterior tiene complejidad exponencial, ya que repite números cálculos. Finalmente se conseguirá un algoritmo más eficiente almacenando los coeficientes que se van calculando.

Ecuación de recurrencia

$$T(n) = 6n^2 + 8n + 9$$

$$O(n^2)$$

Gráfica temporal



Valores de entrada y resultados

```
Valor de C(8, 2) es 28
--------------------------------
Process exited after 0.07489 seconds with return value 0
Presione una tecla para continuar . . .
```

Cuestionario

a) Para cada algoritmo ¿Existe alguna solución que no sea por programación dinámica y que sea óptima? ¿Cuál es?
Si, Fibonacci por Ciclos y Coeficiente Binomial por Programación numérica

b) ¿Cuál de los algoritmos es más difícil de implementar?

Fibonacci

c) ¿Cuál de los algoritmos es más fácil de implementar?

Coeficiente binomial

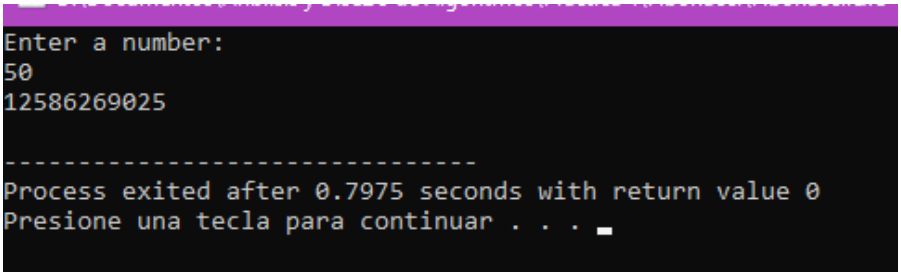
d) ¿El comportamiento experimental de los algoritmos era el esperado? ¿Por qué?
No. Antes de realizar esta práctica, se esperaba que el algoritmo de la mochila iba a ser más sencillo de ejecutarse, pero debido a algunos problemas con la máquina experimental se complicó un poco

e) ¿Qué recomendaciones adicionales propone para realizar esta práctica?
Comprender a fondo el tema de algoritmos que se pueden resolver por programación dinámica y primordialmente comprender como es que funciona esta técnica.

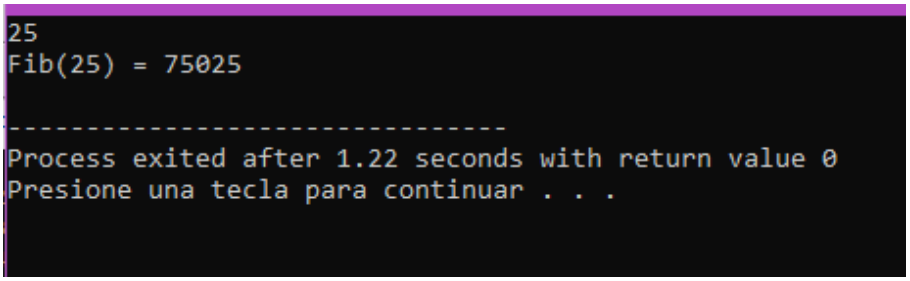
Pruebas

Fibonacci

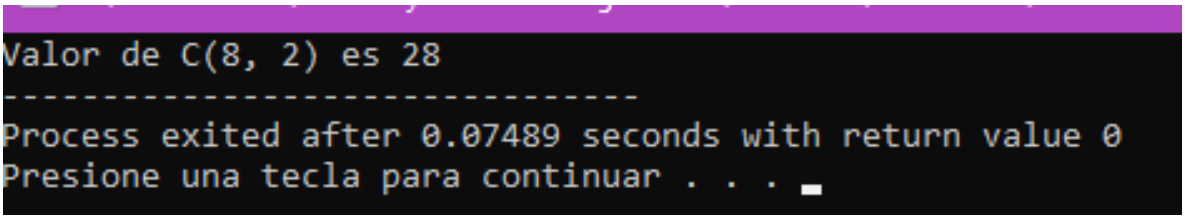
- *Top Down*



- *Button-up*



Coeficiente Binomial



Conclusiones

Frente a una serie de problemas cuyas soluciones pueden ser expresadas recursivamente en términos matemáticos, posiblemente la manera más natural de resolverlos es mediante un método recursivo. Sin embargo, el tiempo de ejecución de una solución, normalmente de orden exponencial, puede mejorarse mediante la programación dinámica.
Para resolver un problema se pueden hacer muchas divisiones y obtener subproblemas independientes, de esta manera se puede llegar más fácil a la solución del problema original, sin embargo, no todos los problemas se pueden resolver de esta forma, ya que cuando los subproblemas obtenidos no son independientes, sino que existe solapamiento entre ellos, la solución no resulta ser eficiente por la repetición de cálculos que conlleva. En estos casos es cuando la programación dinámica ofrece una solución aceptable.

Bibliografía

https://www.wextensible.com/temas/programacion-dinamica/

https://dis.unal.edu.co/~fgonza/courses/2003/pmge/progDinamica.pdf}

https://onlinejudge.inf.um.es/curso/leccion7.1.html

http://www.lcc.uma.es/~av/Libro/CAP5.pdf

https://aprendeyprogramablog.wordpress.com/2016/08/20/programacion-dinamica-coeficientes-binomiales/#text=Conseguiremos%20un%20algoritmo%20m%C3%A1s%20eficiente,de%20orden%20O(nk).