Programacion Dinamica

17 de enero de 2023

Analisis y Diseño de Algoritmos

Herrera Guadarrama Juan Pablo

0. Programación dinámica

La programación dinámica es un método que consiste en simplificar un problema de programación matemática complejo en subproblemas más simples, de manera recursiva, de forma que, resolviendo estos últimos, podamos hallar una solución óptima para el problema original.

1. Análisis y diseño

Con programación dinámica la serie de Fibonacci aunque esta función parece más compleja que si se aplicara recursividad recursiva definida anteriormente, en realidad es mucho más óptima. En particular, sólo implica n-1 cálculos no triviales que corresponden al cálculo de los términos F_2, F_3, … , F_n, una vez cada uno. Esto contrasta con la implementación recursiva, que implica un número de cálculos que crece exponencialmente con *n*.

2. Código fuente

Usando solo programación dinámica

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LIMITE 100

long long int calcFiboButtomUp(long long int n, long long int* fibo, int* contador);
long long int calcFiboTopDown(long long int n, long long int* fibo, int* contador);
long long int* reservarMemoria(int tamanio);

int main(int argc, char *argv[])
      {

      long long int contadorTopDown, contadorButtomUp, resultadoTopDown,
resultadoButtomUp;
      long long int* arregloTopDown, arregloButtomUp;
      register int i;

       for(i = 1; i <= LIMITE; i++)
         {
            contadorButtomUp = 0;
             arregloButtomUp = reservarMemoria(i+1);
             resultadoButtomUp = calcFiboButtomUp((long long int)i, arregloButtomUp,
&contadorButtomUp);

              free(arregloButtomUp);
          }
      return 0;
}

long long int calcFiboButtomUp(long long int n, long long int* fibo, int* contador)
{
      FILE *archivoResultados;

         archivoResultados = fopen("fibonacci.dat","w");
```

```
      if(n<=1)
          {
            return 1;
      }
      else{
      fibo[0] = 0;
              fibo[1] = 1;
             register int i;
             for(i=0; i<2;i++)
             {
             fprintf(archivoResultados,"%d,", fibo[i]);
                  }
              for(i = 2; i <= n; i++)
              {
                fibo[i] = fibo[i-2] + fibo[i-1];
                 fprintf(archivoResultados,"%d,", fibo[i]);
               }
              return fibo[n];
    }
        fclose(archivoResultados);

}

long long int* reservarMemoria(int tamanio)
      {
    long long int* arreglo = (long long int*)malloc(sizeof(long long int)*tamanio);
    register int i;
    for(i = 0; i < tamanio; i++)
      {
            arreglo[i] = -1;
      }
      arreglo[0] = 0;
      arreglo[1] = 1;
      return arreglo;
}
```

Haciendo la comparación de top Down VS Buttom UP para ver el numero de pasos que conllevan

Main.c

```
#include <stdio.h>
#include <stdlib.h>

#define LIMITE 10
/* run this program using the console pauser or add your own getch, system("pause") or
input loop */

int main(int argc, char *argv[]) {
             //Declaracion de variables
      FILE *archivoResultados;
      long long int contadorTopDown, contadorButtomUp, resultadoTopDown,
resultadoButtomUp;
      long long int* arregloTopDown, arregloButtomUp;
      register int i;

          //Inicializacion de variables y apertura del archivo
          archivoResultados = fopen("fibonacci.dat","w");

          //Ciclo desde 1 hasta limite para obtener los resultados
          for(i = 1; i <= LIMITE; i++){
              //Re definir las variables que se ocuparan
              contadorTopDown = contadorButtomUp = 0;
              arregloTopDown = reservarMemoria(i+1);
```

```
            arregloButtomUp = reservarMemoria(i+1);

            //Calculo de resultados
            resultadoTopDown = calcFiboTopDown((long long int)i, arregloTopDown,
&contadorTopDown);
            resultadoButtomUp = calcFiboButtomUp((long long int)i, arregloButtomUp,
&contadorButtomUp);

            //Impresion y almacenaje
            fprintf(archivoResultados,"%d   %d    %d\n", i, contadorTopDown,
contadorButtomUp);//Se almacenan los resultados de la siguiente forma: n  top-down
buttom-up
            printf("Para %d, top-down optuvo %d en %d operaciones y buttom-up %d en %d
operaciones\n",i,resultadoTopDown,contadorTopDown,resultadoButtomUp,contadorButtomUp);
//Se realiza la impresion

            //Liberar los arreglos
            free(arregloButtomUp);
            free(arregloTopDown);
        }
        //Cerrar el archivo
        fclose(archivoResultados);
    return 0;
}
```

cabecera.h

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>


long long int calcFiboButtomUp(long long int n, long long int* fibo, int* contador);
long long int calcFiboTopDown(long long int n, long long int* fibo, int* contador);
long long int* reservarMemoria(int tamanio);
```

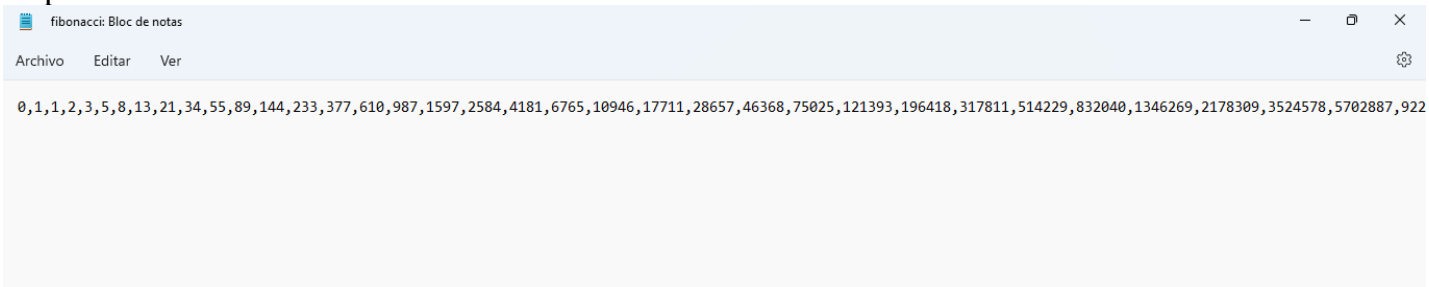
funciones.c

```
#include "cabecera.h"


long long int calcFiboButtomUp(long long int n, long long int* fibo, int* contador){
    if(n<=1){(*contador)++;
        return 1;(*contador)++;
    }
    else{
        (*contador)++;
        fibo[0] = 0;
          (*contador)++;
          fibo[1] = 1;(*contador)++;
         register int i;(*contador)++;
         for(i=0; i<2;i++)
                printf("valor de fibonaci  %d\n",fibo[i]);

          for(i = 2; i <= n; i++)
          {
              (*contador)++;
              fibo[i] = fibo[i-2] + fibo[i-1];
              printf("valor de fibonaci  %d\n",fibo[i]);
              (*contador)++;
          }
              (*contador)++;
              (*contador)++;
```

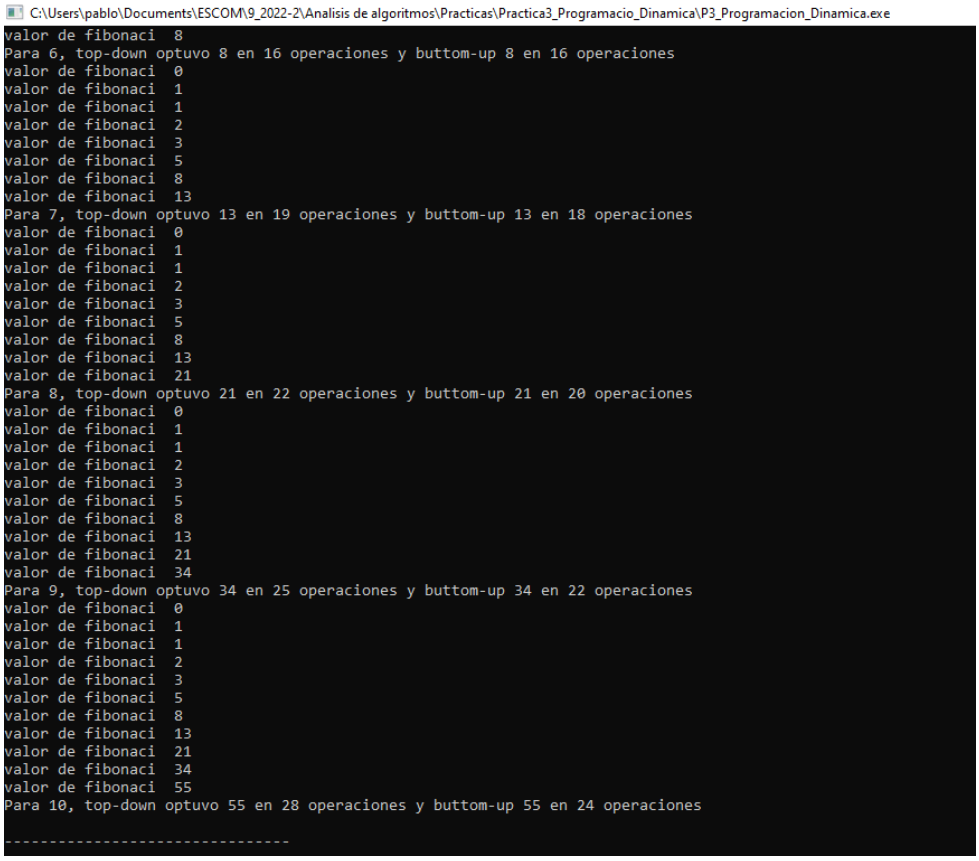
```
                return fibo[n];
    }
}
/**
 * Esta funcion realiza el calculo del termino con el enfoque top-down
 * long long int} [n] termino a encontrar
 * long long int*} [fibo] arreglo de memorizacion
 * int*) [contador] contador de operaciones
 * @return {int} retorna el n-simo termino de la serie de fibonacci
*/
long long int calcFiboTopDown(long long int n, long long int* fibo, int* contador)
{
    if(fibo[n] != -1)
    {
            (*contador)++;
            return fibo[n];(*contador)++;
    }
      else{(*contador)++;
        (*contador)++;
        return fibo[n] = calcFiboTopDown(n-2, fibo, contador)+calcFiboTopDown(n-1, fibo,
contador);
    }
}
/**
 * Esta funcion crea un arreglo con memoria dinamica y lo inicializa en -1
 *  {int} [tamanio] tamanio del arreglo a crear
 * @return {long long int*} retorna el arreglo creado
*/
long long int* reservarMemoria(int tamanio)
      {
    long long int* arreglo = (long long int*)malloc(sizeof(long long int)*tamanio);
    register int i;
    for(i = 0; i < tamanio; i++)
      {
            arreglo[i] = -1;
      }
      arreglo[0] = 0;
      arreglo[1] = 1;
      return arreglo;
}
```

3. Evidencias

Imprimimos la serie en un archivo txt



Evidencia del segundo código donde se ve una VS entre una recursiva y la programación dinámica donde que da lineal su complejidad



4. Referencias

5. BettaTech. (2020, 30 noviembre). *¿En qué consiste REALMENTE la PROGRAMACIÓN DINÁMICA?* [Vídeo]. YouTube. https://www.youtube.com/watch?v=C240g6_Dsl4