Sequência de Fibonacci.

A sequência de Fibonacci na matematica, é uma sequência de numeros inteiros que começa por 0 e 1, sendo cada termo subsequente correspondente a soma dos dois termos anteriores.

Sequencia:

0,1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597, 2584, ...

A definição matetica desta sequência é dada por:

$$F_n = F_{n-1} + F_{n-2},$$

Onde os termos inicias são,

$$F_1 = 1,\ F_2 = 1.$$

Recursividade

Inicialmente o problema pode ser facilmente implementado utilizando recurssao com o metodo de divisao e conquista, porém neste caso, o algoritmo recalcula diversas vezes valores já obtidos anteriomente, tornando assim muito custoso, computacionalmente, o calculo de sequencias muito altas, como por exemplo Fib(100).

Observe na figura 1 um simples exemplo onde é recalculado valores já obtidos:

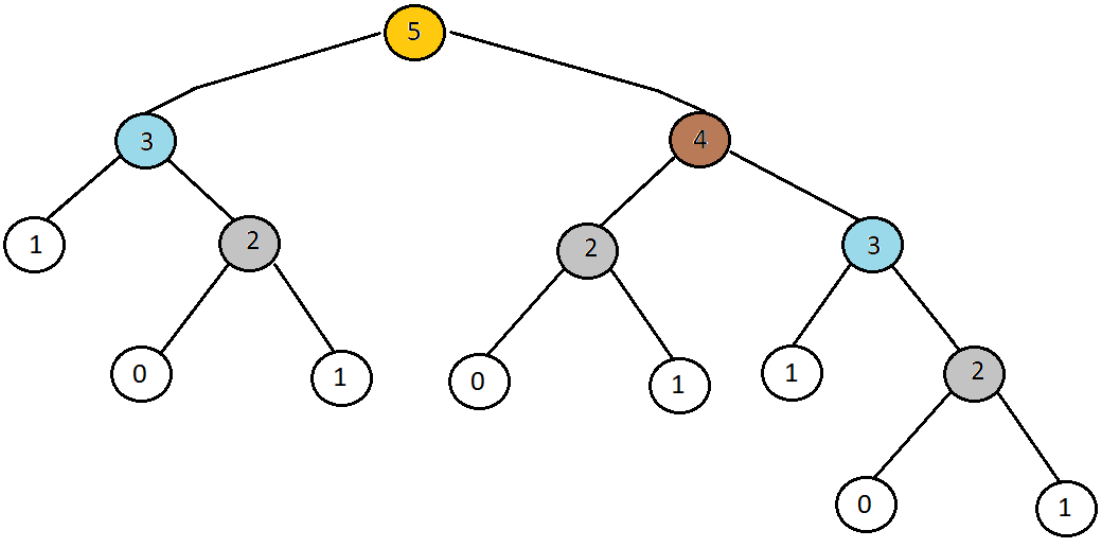
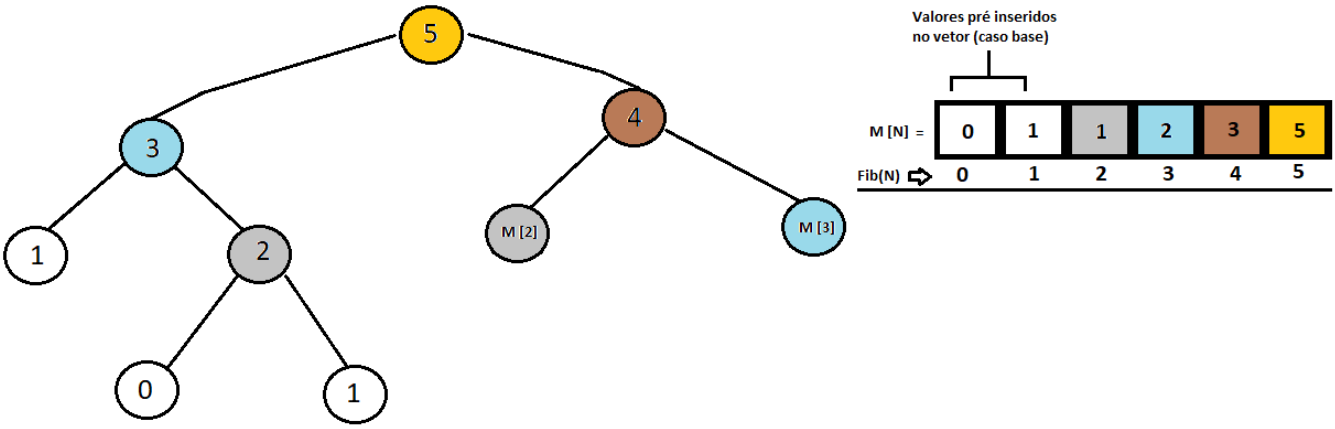


Figura 1

Recursividade com Memorização.

Para resolvermos o problema de recalculo do algoritmo recursivo podemos implementar em conjunto uma espécie de memorização, fazendo com que o algoritmo guarde em um vetor todas funções já calculadas. Observe a figura N um exemplo da recursividade com memorização:



Note que neste caso, os valores de Fib(2) e Fib(3) não são recalculados, sendo utilizado o vetor para obter o resultado.

Dinâmico (Convolução)

Esta implementação é a mais próxima de programão dinâmica e não faz a utilização de recursividade. O algoritmo utiliza uma estratégia de convolução onde são utilizados três índices, sendo dois deles são a "representação" de Fib(n-1) e Fib(n-2), e o terceiro índice é a resposta da soma dos outros dois. Nas próximas iterações do algoritmos ele iguala o primeiro incide e com segundo, e o segundo índice é igualado com o terceiro (que na verdade é a resposta da iteração anterior), e assim é feito o cálculo da próxima resposta. Observe na figura 2 um exemplo da convolução:

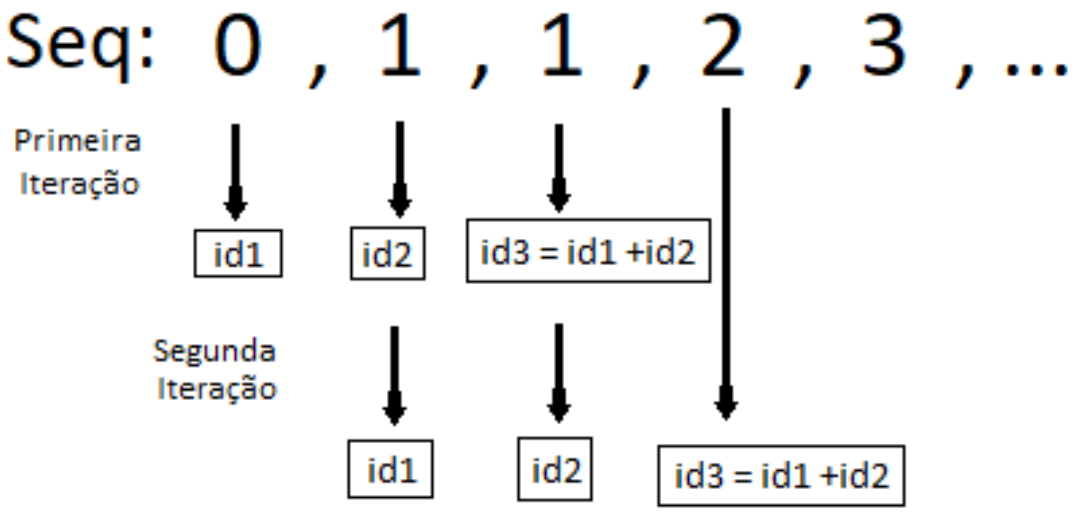


Figura 2

Implementação em C

Foi feita a implementação em C/C++ dos três algoritmos onde o recursivo sem memorização faz o cálculo da sequência de Fibonacci de 1 a 44 e os algoritmos recursivo com memorização e dinâmico fazem o cálculo da sequência de Fibonacci de 1 a 500.

Foi cronometrado o tempo de execução de cada um deles em nanosegundos(ns) utilizando a biblioteca Chrono, ao terminar o algoritmo gera três arquivos .txt onde é salvo os tempos de cada execução e é gerado também três scripts .plt para plotar os gráficos.

O algoritmo exibe três gráficos, um dos tempos de execução do algoritmo recursivo (Figura 3) o segundo dos tempos execução do algoritmo recursivo com memorização (Figura 4) e o terceiro dos tempos de execução do algoritmo Convolução Dinâmica (Figura 5)

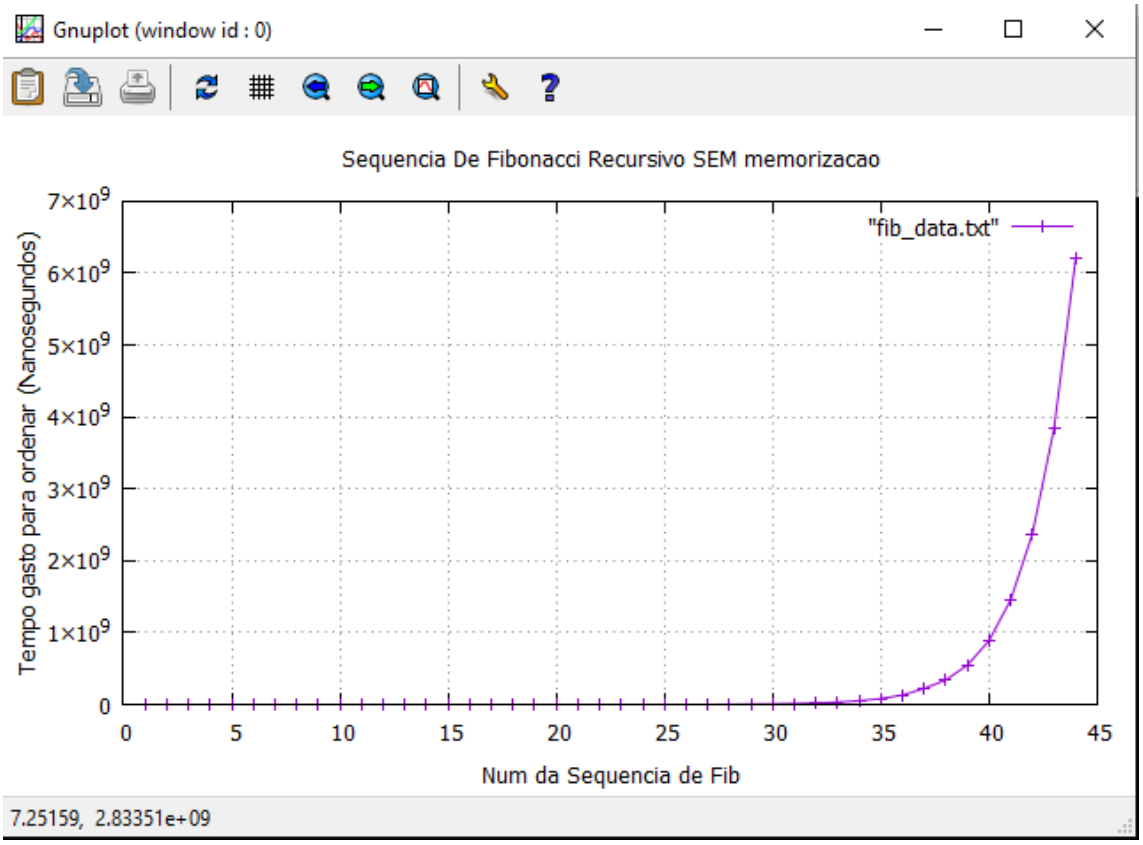


Figura 3

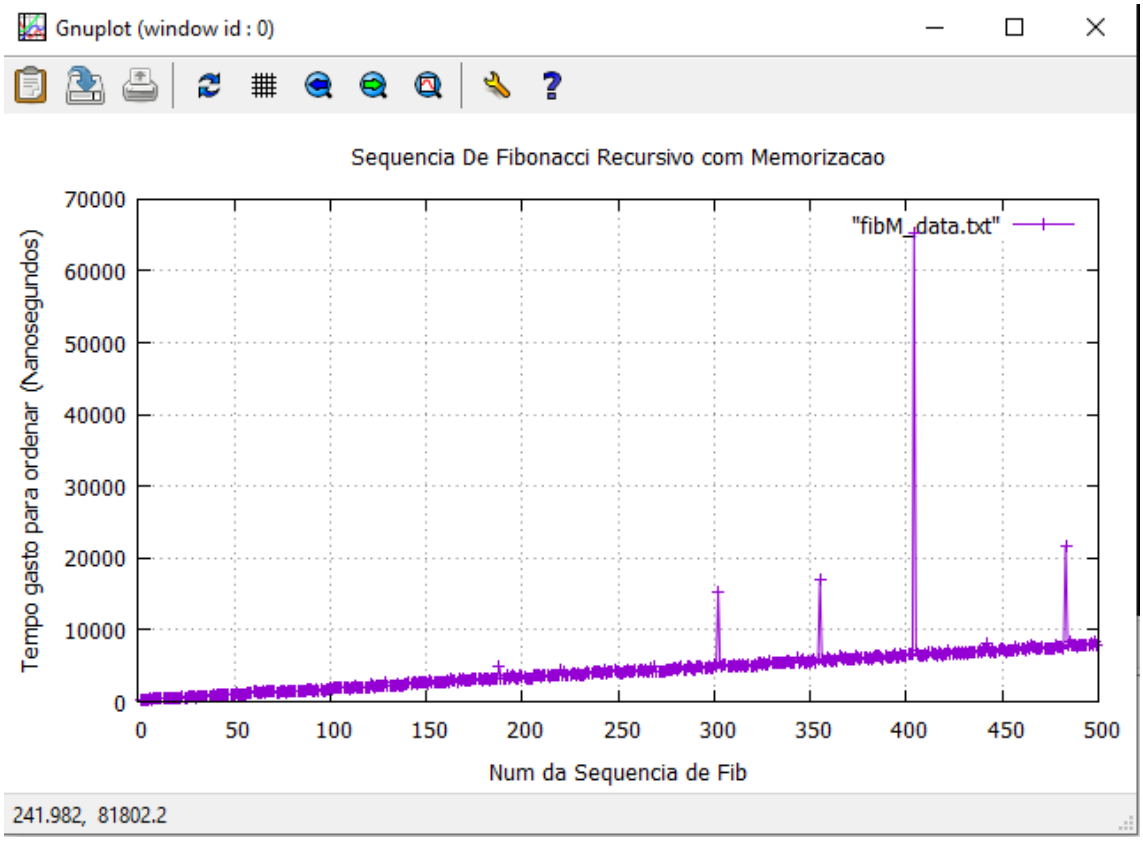


Figura 4

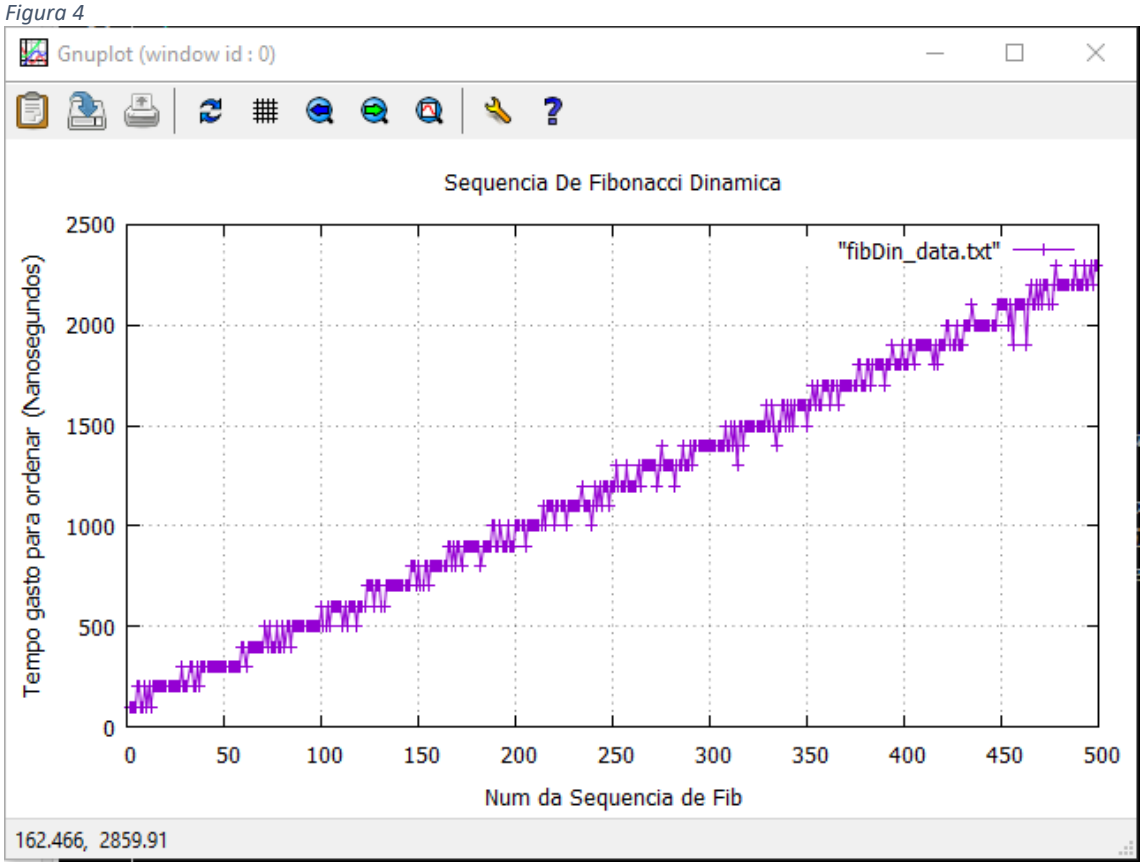


Figura 5

No algoritmo recursivo sem memorização, até o Fib(30) não é notável o tempo decorrido para o cálculo, porem ao chegar no Fib(44) o algoritmo levou um pouco mais de 6 Segundos (6x10^9 nanosegundos) para calcular.

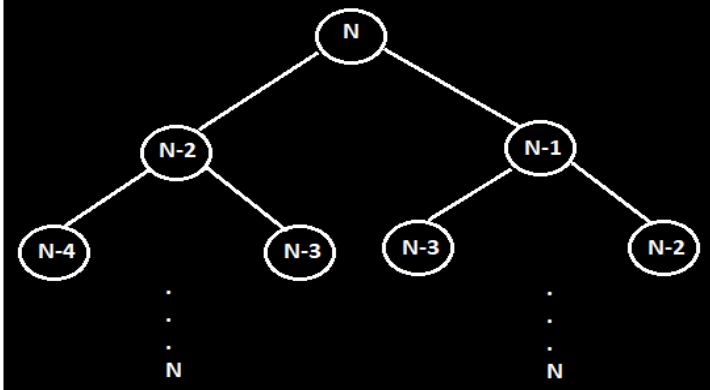
No algoritmo recursivo com memorização vemos que ao calcular o Fib(500) ele levou aproximadamente 0,01 milissegundos(10000 nanosegundos). Observe o quanto a performance do algoritmo melhorou, sem trocar a técnica de recursividade (divisão e conquista), apenas adicionando um vetor que memoriza os valores calculados.

No algoritmo dinâmico o desempenho foi de longe o melhor dos três, ele calculou o Fib(500) em aproximadamente 0,0025 milissegundos(2500 nano segundos) sendo até 4 vezes mais rápido do que o algoritmo recursivo com memorização. Isso se deve a estratégia de convolução do algoritmo, tornando o mais performático entre os três estudados aqui.

Análise Assintótica dos algoritmos.

Analise Assintotica Fibonacci Recursivo sem Memorização

```
//Funcao de fibonacci recursivo
int fib (int n){
   if(n==0  || n==1){
      return n;
    }
   else{
      return fib(n - 2) + fib(n - 1);
    }
}
```

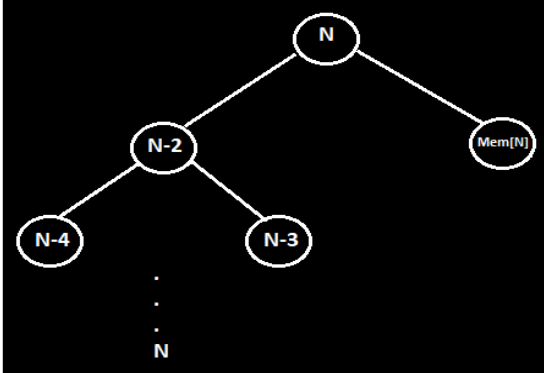


Altura da arvore: N-1

Note: A Arvore cresce mais do lado direito onde é feito (N-1)

Analise Assintotica Fibonacci Recursivo com Memorização

```
//Funcao de fibonacci recursivo com memorização
long long int mem[500];
long long int fibMem (int n){
   if(n==0 || n==1){
      return n;
    }
   else{
       if(mem[n] != 99){
           return mem[n];
          }
       else if(mem[n] == 99){
           mem[n-2]= fibMem(n-2);
           mem[n-1]= fibMem(n-1);
           mem[n] = mem[n-2] + mem[n-1];
           return mem[n];
       }
    }
}
```



Altura da arvore = N-2

Note: A arvore neste caso cresce apenas na primeira recursividade (N-2), a segunda utiliza a memoria

Analise Assintotica Fibonacci Dinamico

```
//Funcao de fibonacci utilizando programacao dinamica
long long int fibDin(int n){

    long long int st = 0;
    long long int nd = 1;
    if(n <= 1){
        return n;
    }

    long long int ans = st + nd;
    for(int i = 2; i <= n ; i++){
        ans = st + nd;
        st = nd;
        nd = ans;
    }

    return ans;
}
```

c

N-2

Tempo = N - 2 + C

O(n)