

NAMA : ARTHA NUGRAHA FAJAR SIDIQ

NIM : 1117000439

1. PENJABARAN FIBONACCI

Bilangan Fibonacci adalah suatu deret bilangan bulat (*integer*) yang tak berhingga dimana bilangan tersebut secara berurutan dapat didefinisikan sebagai berikut ini :

1 1 2 3 5 8 13 21 34 55 89.....n

Deret bilangan Fibonacci di atas, maka dapat dipahami bahwa untuk mencari nilai suku ke-n pada deret bilangan Fibonacci dengan cara menjumlahkan nilai deret yang saling berdekatan.

2. ALGORITMA ITERASI

Saya menggunakan algoritma iterasi dimana cara pencariannya adalah dengan menjumlahkan nilai data terbaru (n) dengan nilai sebelum data terbaru (n-1).

Berikut algoritma yang saya gunakan

1. Start
2. Inisialisasi/menentukan nilai min (batas bawah) = 0, max (batas atas) = 1, dan value (nilai data awal) = 0.
3. Masukkan suku ke-n bilangan fibonacci yang ingin dicari.
4. Jika kondisi nilai variabel i tidak sama dengan nilai n (batas), maka proses yang dijalankan sebagai berikut :
 - a. Nilai variabel i dinaikkan sebesar 1.
 - b. Nilai variabel value adalah hasil penambahan nilai dari variabel min dan max.
 - c. Nilai variabel min sama dengan nilai variabel max.
 - d. Nilai variabel max sama dengan nilai variabel value.
5. Jika kondisi nilai variabel i sama dengan nilai n, maka proses dihentikan.
6. Hasil pencarian nilai bilangan Fibonacci pada suku ke-n ditampilkan dalam bentuk deret
7. Selesai

3. SOURCE CODE

```
package tugasdaa;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.text.DecimalFormat;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;

public class TugasDAA {
    public static void main(String[] args) {
        int jum = 6;
        int[] n = {0, 1, 2, 10, 50, 100};
        int[] result = new int[n.length];
        String[] waktu = new String[n.length];
        BufferedReader numberinput, suminput;
        long timeStart = System.currentTimeMillis();
        if (jum <= 0) {
            System.out.print("Close Program");
        } else {
            System.out.println("(Search Fibonacci Numbers)\n");
            for (int j = 0; j < jum; j++) {
                System.out.print("Fibonacci Number " + n[j] + "\n");
                int min = 0;
                int max = 1;
                int value = 0;
                for (int i = 0; i <= n[j]; i++) {
                    result[j] = min;
                    System.out.print(min + " ");
                    value = min + max;
                    min = max;
                    max = value;
                }
                long timeFinish = System.currentTimeMillis();
                double newRunTime = (double) (timeFinish - timeStart) /
1000;

                DecimalFormat runtimeDF = new DecimalFormat("##0.000000");
                System.out.println("\nDuration of execution : " +
runtimeDF.format(newRunTime) + " seconds\n");
                waktu[j] = runtimeDF.format(newRunTime) + " seconds";
            }
            System.out.println("(Result Fibonacci Numbers)\n");
            for (int j = 0; j < jum; j++) {
                System.out.println("Fibonacci Number " + n[j] + " = " +
result[j] + "( " + waktu[j] + ")");
            }
        }
    }
}
```

```
(Search Fibonacci Numbers)

Fibonacci Number 0
0
Duration of execution : 0.001000 seconds

Fibonacci Number 1
0 1
Duration of execution : 0.036000 seconds

Fibonacci Number 2
0 1 1
Duration of execution : 0.036000 seconds

Fibonacci Number 10
0 1 1 2 3 5 8 13 21 34 55
Duration of execution : 0.036000 seconds

Fibonacci Number 50
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 1346269 2178309
Duration of execution : 0.038000 seconds

Fibonacci Number 100
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 1346269 2178309 3570133 5777876 9227465 14700494 23447881 37178226 59128682 93812141 147632244 231477977 365580442 581349663 914125573 1434890684 2247115717 3542245401 5597066678 8771528693 13690505602 21213700320 33196247985 51422976703 78657621471 121393091226 187881196609 289340941726 444730762897 684031766623 1049131415140 1605908049057 2453139684000 3732214714967 5674556060000 8630958643090 13146867791460 20196134623649 30619531601480 46368033896640 70140873360160 106485363646080 161805321741729 244145170266889 366380536963690 554686077125600 841256634475329 1264511585421458 1903520439184687 2869381177260246 4342892616445935 6558239514097123 9909216679762480 14891359049907104 22431566484274560 34015566544181683 51422976703537792 77117667090834176 115935035168640000 174431436824480000 264246261717760000 398739064464640000 598694437696000000 895678969676800000 1351621470028800000 2030300438720000000 3022971808409600000 4514593278092800000 6771214714880000000 10144808093440000000 15216391872000000000 22561199974400000000 33977611968000000000 50994011852800000000 75810403840000000000 112577615808000000000 169388027648000000000 253200643584000000000 378588671360000000000 563976700224000000000 844364728800000000000 1260253347200000000000 1874617975040000000000 2798982692800000000000 4173600668160000000000 6242582643200000000000 9316563328000000000000 13950166016000000000000 20792748672000000000000 30842914649600000000000 45735663321600000000000 68578580070400000000000 102421243776000000000000 152163918720000000000000 225611999744000000000000 339776119680000000000000 509940118528000000000000 758104038400000000000000 1125776158080000000000000 1693880276480000000000000 2532006435840000000000000 3785886713600000000000000 5639767002240000000000000 8443647288000000000000000 12602533472000000000000000 18746179750400000000000000 27989826928000000000000000 41736006681600000000000000 62425826432000000000000000 93165633280000000000000000 139501660160000000000000000 207927486720000000000000000 308429146496000000000000000 457356633216000000000000000 685785800704000000000000000 1024212437760000000000000000 1521639187200000000000000000 2256119997440000000000000000 3397761196800000000000000000 5099401185280000000000000000 7581040384000000000000000000 11257761580800000000000000000 16938802764800000000000000000 25320064358400000000000000000 37858867136000000000000000000 56397670022400000000000000000 84436472880000000000000000000 126025334720000000000000000000 187461797504000000000000000000 279898269280000000000000000000 417360066816000000000000000000 624258264320000000000000000000 931656332800000000000000000000 1395016601600000000000000000000 2079274867200000000000000000000 3084291464960000000000000000000 4573566332160000000000000000000 6857858007040000000000000000000 10242124377600000000000000000000 15216391872000000000000000000000 22561199974400000000000000000000 33977611968000000000000000000000 50994011852800000000000000000000 75810403840000000000000000000000 112577615808000000000000000000000 169388027648000000000000000000000 253200643584000000000000000000000 378588671360000000000000000000000 563976700224000000000000000000000 844364728800000000000000000000000 1260253347200000000000000000000000 1874617975040000000000000000000000 2798982692800000000000000000000000 4173600668160000000000000000000000 6242582643200000000000000000000000 9316563328000000000000000000000000 13950166016000000000000000000000000 20792748672000000000000000000000000 30842914649600000000000000000000000 45735663321600000000000000000000000 68578580070400000000000000000000000 102421243776000000000000000000000000 15216391872000000
```

6. HASIL AKHIR

Berdasarkan dari percobaan diatas, dapat saya simpulkan bahwa pencarian nilai suku ke-n pada bilangan Fibonacci lebih cepat dan lebih efisien menggunakan fungsi iterasi daripada menggunakan fungsi rekursif.