Programas de Sat y Fibonacci
Tiempos de ejecución Lic. Anel R.A.

SATISFACTIBILIDAD

Programa realizado en prolog con recursión. Código:

```
:- op(610, fy, -). % negación    DECLARACION  DE OPERACIONES POR SIMBOLO

:- op(620, xfy, &). % conjuncion

:- op(630, xfy, v). % disyuncion

:- op(640, xfy, =>). % condicional

:- op(650, xfy, <=>). % equivalencia


valor_de_verdad(0). % valores de verdad

valor_de_verdad(1).


función_de_verdad(v, 0, 0, 0) :- !.   %Declaración de resultados para las implicaciones, disyun…

función_de_verdad(v, _, _, 1).

función_de_verdad(&, 1, 1, 1) :- !.

función_de_verdad(&, _, _, 0).

función_de_verdad(=>, 1, 0, 0) :- !.

función_de_verdad(=>, _, _, 1).

función_de_verdad(<=>, X, X, 1) :- !.

función_de_verdad(<=>, _, _, 0).

función_de_verdad(-, 1, 0).

función_de_verdad(-, 0, 1).


valor(F, I, V) :- memberchk((F,V), I).                  %  valor((p v q) & (-q v r),[(p,1),(q,0),(r,1)],V).

valor(-A, I, V) :- valor(A, I, VA), función_de_verdad(-, VA, V).

valor(F, I, V) :- F =..[Op,A,B], valor(A, I, VA), valor(B, I, VB),

función_de_verdad(Op, VA, VB, V).    % interpretaciones_fórmula((p v q) & (-q v r),L).


interpretaciones_fórmula(F,U) :- findall(I,interpretación_fórmula(I,F),U).

                                                % interpretación_fórmula(I,(p v q) & (-q v r)).
```

```
interpretación_fórmula(I,F) :- símbolos_fórmula(F,U),
interpretación_símbolos(U,I).                    %  símbolos_fórmula((p v q) & (-q v r), U).

símbolos_fórmula(F,U) :- símbolos_fórmula_aux(F,U1), sort(U1,U).
símbolos_fórmula_aux(F,[F]) :- atom(F).
símbolos_fórmula_aux(-F,U) :- símbolos_fórmula_aux(F,U).
símbolos_fórmula_aux(F,U) :- F =..[_Op,A,B], símbolos_fórmula_aux(A,UA),
símbolos_fórmula_aux(B,UB), union(UA,UB,U).
interpretación_símbolos([],[]).                  %  interpretación_símbolos([p,q,r],I).
interpretación_símbolos([A|L],[(A,V)|IL]) :- valor_de_verdad(V),
interpretación_símbolos(L,IL).
es_modelo_fórmula(I,F) :- valor(F,I,V),     %es_modelo_fórmula([(p,1),(q,0),(r,1)], (p v q) & (-q v r)).
V = 1.                                      % es_modelo_fórmula([(p,0),(q,0),(r,1)], (p v q) & (-q v r)).
                                            % modelo_fórmula(I,(p v q) & (-q v r)).
modelo_fórmula(I,F) :- interpretación_fórmula(I,F),
es_modelo_fórmula(I,F).
modelos_fórmula(F,L) :- findall(I,modelo_fórmula(I,F),L).    % modelos_fórmula((p v q) & (-q v r),L).
es_satisfacible(F) :- interpretación_fórmula(I,F),   % es_satisfacible((p & q) & (x => s) & (y => -t) & (n v u) & (m v -w) & (xf => s)).
es_modelo_fórmula(I,F).
% Para calcular tiempo de ejecución :  time(es_satisfacible((p & q) & (x => s) & (y => -t) & (n v u) & (m v -w) & (xf => s))).
```

EJEMPLOS DE COMPILCIÓN:

es_satisfacible((p & q) & (x => s) & (y => -t) & (n v u) & (m v -w) & (xf => s)).

true.

es_satisfacible((p & q) & (x => s) & (y => -t)).

true

es_satisfacible((p & q) & (p => r) & (q => -r)).

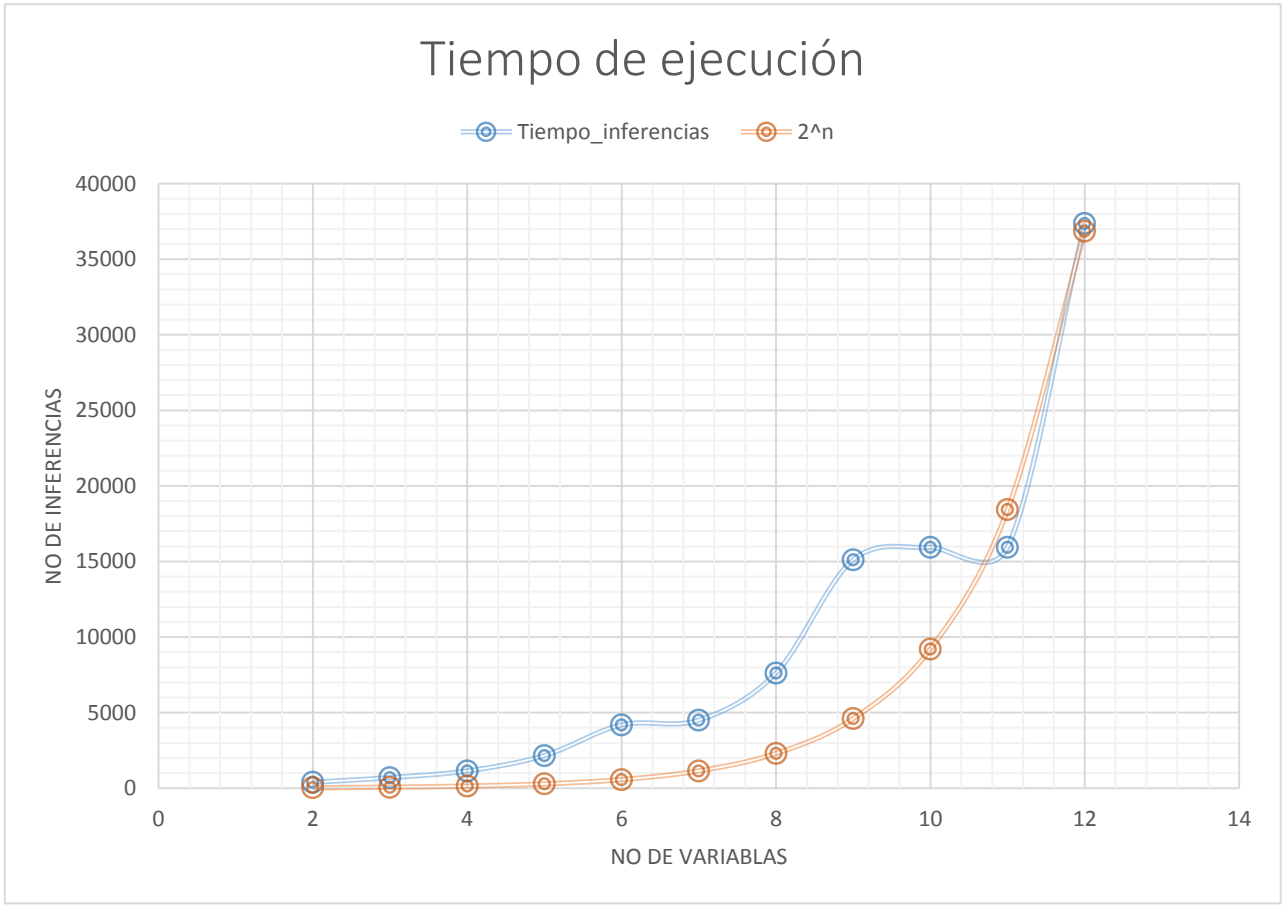
false.

TIEMPOS DE EJECUCIÓN

En función del número de inferencias:

No variables = n	Tiempo(num inf)	2^n
2	392	36
3	704	72
4	1147	144
5	2163	288
6	4193	576
7	4505	1152
8	7629	2304
9	15119	4608
10	15944	9216
11	15946	18432
12	37374	36864

Tabla 1. Datos de respuesta en funció del núm. de variables en el programa SAT realizado en prolog, respecto al número de inferencias.

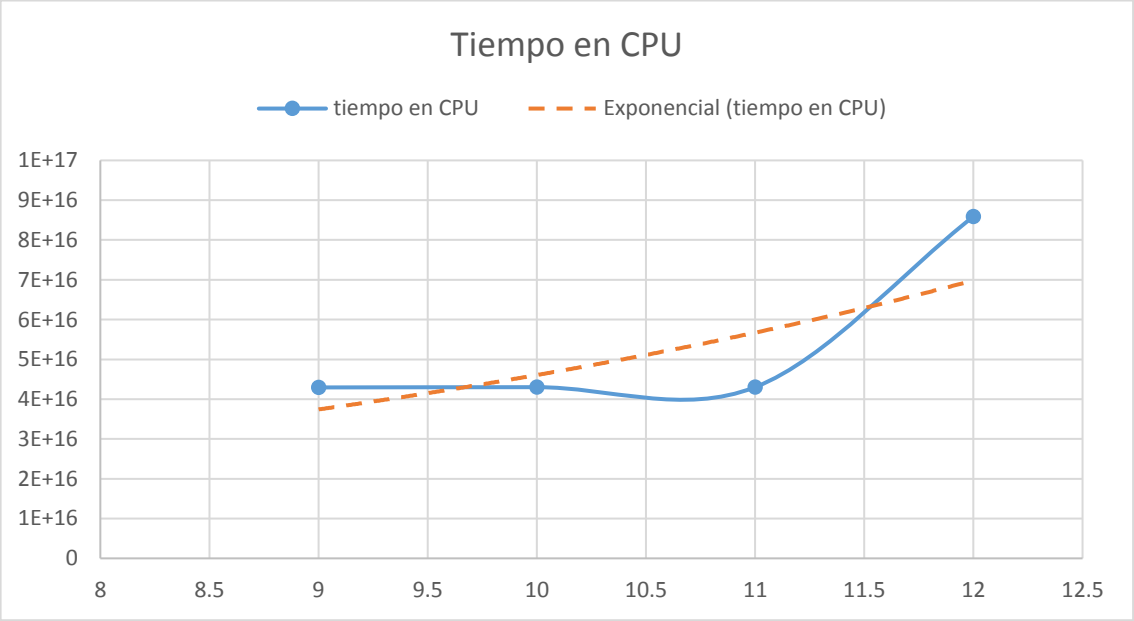


Gráfica 1. Gráficas de Tabla 1.

En función del tiempo en CPU:

No variables	tiempo en CPU
9	4.2953E+16
10	4.3021E+16
11	4.3027E+16
12	8.5894E+16

Tabla 2. Datos de respuesta en funció del num. de variables en el programa SAT realizado en prolog, respecto al tiempo en CPU.



Gráfica 2. Gráficas de Tabla 2.

OBSERVACIONES

En la Gráfica 1 podemos ver el comportamiento exponencial (línea naranja) del problema de satis factibilidad en función del número de variables del problema,,esto en función de las inferencias (línea azul) que son los llamados de recursión que realizo el programa para resolver la función de satisfactibilidad. En la Gráfica 2 tambien podemos ver este comportamiendo reflejado en la línea azul y donde la línea punteada naranja marca la tendencia exponencial.

Programas de Sat y Fibonacci
Tiempos de ejecución Lic. Anel R.A.

FIBONACCI

Programa realizado en Python. Basado en la primera forma y más común, donde fn es el nth número de Fibonacci y f0 = 0 y f1 = 1 , sea:

$$f_n = f_{n-1} + f_{n-2}$$

Y una segunda forma; algebraica basada en el número de Oro:

$$f_n = \frac{\varphi^n - (1-\varphi)^n}{\sqrt{5}}$$

Donde: $\varphi = \frac{1+\sqrt{5}}{2}$

Código:

```
from timeit import default_timer #importamos la función time para capturar tiempos
from math import sqrt

        ### FIBONACCI POR RECURSION
def fib(n):
   a, b = 0,1
   while a < n:
      print(a, end=' ')
      a, b = b, a+b
   print()
fib(9000)

   ### FIBONACCI POR EL NUM DE ORO (algebraico)
fi = 1.618033988749895
def fibNumOro(n):
   return (fi ** n - (1 - fi) ** n) / sqrt(5)

for i in range(30):
        print(int(fibNumOro(i)))#f=fibNumOro(i)
```

Código para calcular el tiempo de ejecución:

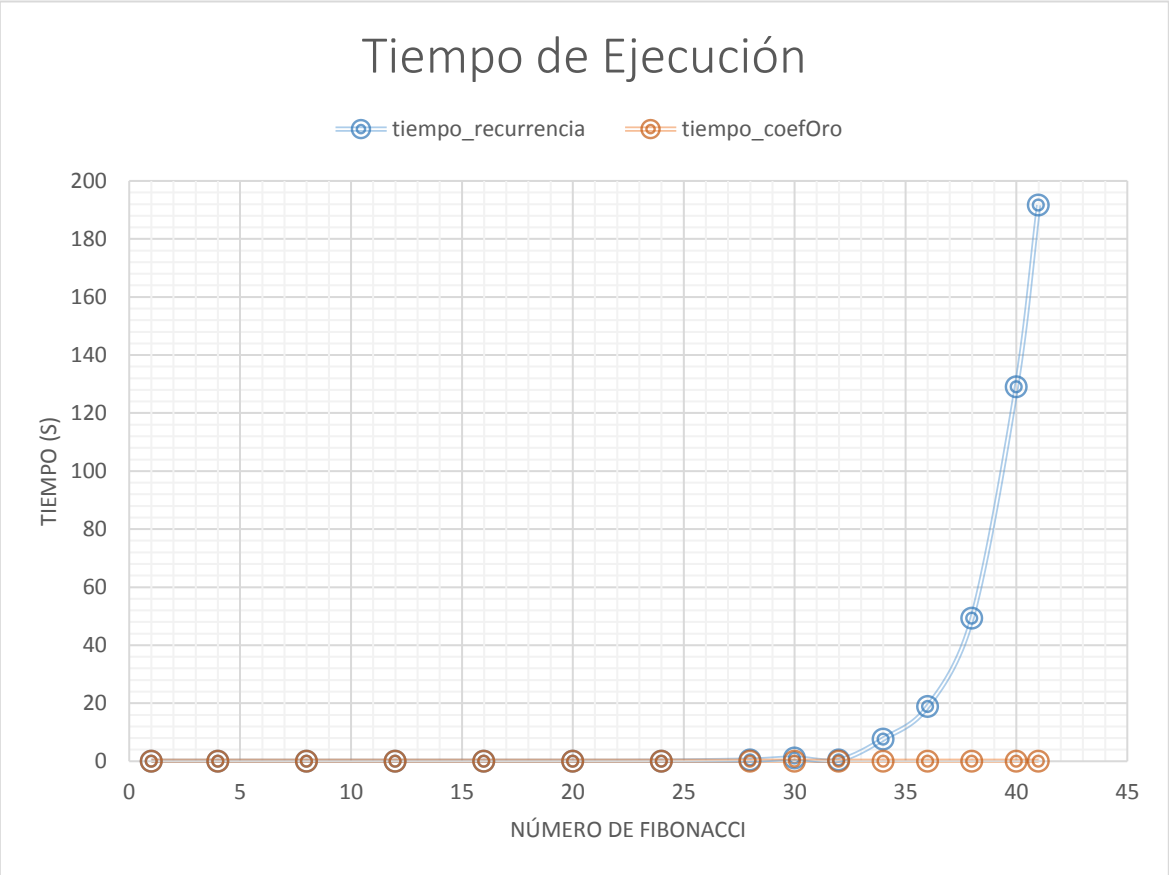
```
from timeit import default_timer #importamos la función time para capturar tiempos
from math import sqrt

def fibonacci(n):
   if n == 0 or n == 1:
        return n
   else:
        return fibonacci(n-1) + fibonacci(n-2)
inicio = default_timer()
fibonacci(41)
fin = default_timer()
print(fin - inicio)
```

TIEMPOS DE EJECUCIÓN

Número de Fibonacci= n	tiempo_recurrencia	tiempo_coefOro
1	1.539E-06	0.000103105
4	6.155E-06	0.000189282
8	3.3342E-05	0.000325217
12	0.000211852	0.00047141
16	0.001299841	0.000491416
20	0.009186091	0.000749435
24	0.059736779	0.000708398
28	0.418846363	0.000775596
30	1.093020715	0.000903323
32	0.418846363	0.001576326
34	7.676913047	0.001356267
36	18.86145715	0.001268038
38	49.32673718	0.001396791
40	129.0955332	0.00150759
41	191.6552825	0.001597

Tabla 3. Datos de respuesta en función del núm. de variables = núm. de Fibonacci.



Gráfica 3. Gráficas de Tabla 3.

Gráfica 4. Gráficas de Tabla 3.

OBSERVACIONES

En la gráfica 3, línea azul podemos ver la tendencia exponencial, en función del número de Fibonacci. Para la forma recursiva con el uso de la formuna fn =fn-1 + fn- 2; donde para el num. de Fibonacci de 41 llega a tardar hasta 3.1 mins en terminar la ejecución.

Por otro lado en la Gráfica 3, línea naranja también podemos ver la tendencia lineal para el método algebraico, con el uso del coeficiente de Oro. Esta misma tendencia se muestra en la Gráfica 4 para su mejor visualización, donde la línea azul es la tendencia lineal respecto a la real, línea naranja. Donde para el núm. de Fibonacci de 41 tarda solo 0.0016 seg. Vs los 3.1 mins del método recursivo.