Universidad Veracruzna

Inteligencia Artificial

Análisis de Algoritmos

Actividad 3

Leonardo Flores Torres

20 de septiembre de 2022

Programar un algoritmo de su elección (no tan sencillo) y analizarlo de la siguiente forma:

- 1. Graficar el tiempo de ejecución en función de N,
- 2. sobre los mismos ejes graficar 2 cotas superiores y dos cotas inferiores,
- 3. repetir el punto 1 y 2 ejecutando el programa en otra computadora de distinto desempeño,
- 4. analizar los resultados y discutirlos. Escribir de la manera más completa las características de las 2 computadoras.

El algoritmo que se eligió fue computar un fractal, más específicamente, el ...

El extracto de código que se muestra a continuación es una representación del modo de trabajo que se lleva en julia usando el REPL¹, asemeja un ambiente de trabajo y ejecución de comandos en la terminal.

```
julia> ns = 10:10:1600;
                                             # valores que puede tomar N
julia> reps = 20;
                                             # numero de repeticiones
julia> timings = zeros(length(ns), 2);
                                            # arreglo bidimensional
julia> for rep in 1:reps
            for (index, n) in enumerate(ns)
                time = @elapsed mf.fractalCMap(n, n, maxiter=100)
                if rep == 1
                    timings[index, 1] = n
                timings[index, 2] += time
            end
        end
julia> timings[:,2] = timings[:,2] / reps; # promedio de tiempo
julia> timings = vcat([0 0], timings);
                                            # agregar entrada extra para el tiempo cero
julia> nvalues = timings[:,1];
                                            # lista de iteraciones
julia> time = timings[:,2];
                                            # lista de tiempos
```

Para graficar el tiempo de ejecución se hicieron 20 repeticiones indicadas por reps , y se definió una variable ns para guardar el conjunto de valores que puede tomar $N=10,\ 20,\ 30,\ \ldots,\ 1600$. La variable timings guarda en la

¹REPL es un acrónimo para Read-Eval-Print loop.

primera columna el valor de N, mientras que en la segunda columna guarda el tiempo t(N) que le toma al algoritmo computar el fractal. Por cada iteración del loop se suman los tiempos t(N) a sus respectivas entradas, y al final toda la columna de tiempos se divide entre la cantidad de repeticiones reps lo que resulta en tiempos promedio $\tilde{t}(N)$. Se usaron los tiempos promedio debido a que si hay procesos en ejecución en los ordenadores pueden generar cambios en los tiempos de cómputo.

Los tiempos de ejecución en el ordenador diannao se pueden observar en la figura 1 donde las dos líneas continuas representan las cotas superiores, mientras que las líneas no continuas con las pertenecientes a las cotas inferiores. Se tuvo que utilizar un factor de escalamiento para los cuatro casos ya que los tiempos eran muy pequeños incluso para el caso en el que N=1600 donde el tamaño del fractal corresponderia a una imagen de 1600×1600 pixeles con un número de iteraciones máximo de 100.

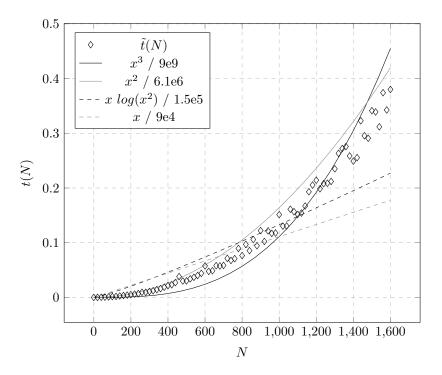


Figura 1: Tiempos de ejecución usando hongdiannao.

Las características de las computadoras usadas, diannao y hongdiannao, se muestran en la figuras 2 y 3, respectivamente.

Figura 2: características del ordenador diannao.

```
• • •
       ,xXc
.l0MMMMMO
                           leo@hongdiannao
   .knmmmmmmmmn,
                           OS: Linux Lite 6.0 x86_64
                           Host: HP Pavilion 11 x360 PC 0977100000405F00010420180
                           Kernel: 5.15.0-47-generic
  Uptime: 3 mins
                           Packages: 2419 (dpkg), 6 (flatpak), 8 (snap)
Shell: bash 5.1.16
Resolution: 1366x768
  . WMMMMX0MMMMW.
 oMMMMMMXWMMMMM:
WMMMMMNkMMMMMO
: MMMMMOXMMMMW
.0MMMMM×MMMMM;
                           WM: Xfwm4
;cKMMWxMMMMO
                           WM Theme: Materia
                           Theme: Materia-compact [GTK2/3]
Icons: Papirus-Adapta [GTK2], Adwaita [GTK3]
Terminal: xfce4-terminal
 MMWMMXOMMMMl
 kMMMMKOMMMMMX:
  WMMMMKOWMMM0c
                           Terminal Font: mononoki Nerd Font Mono 15
                           CPU: Intel Pentium N3520 (4) @ 2.415GHz
GPU: Intel Atom Processor Z36xxx/Z37xxx Series Graphics & Displ
Memory: 636MiB / 3815MiB
```

Figura 3: características del ordenador hongdiannao.

Apéndice

```
module MandelbrotFractal
17
18
   19
20
   Required libraries
   ==================================#
21
22
   using Images
23
   using Plots
24
25
   26
  Image operations
27
28
```

```
29
    function imageSave(path, img)
30
        save(path, img)
31
32
    end
33
    function visualize(rgbmap)
34
        return plot(rgbmap, ticks=false)
35
36
    end
37
    #-----
38
    Fractal genereration
39
    40
41
    function mandelbrot(c, maxiter)
42
43
        z = 0
        n = 0
44
        while abs(z) <= 2 && n < maxiter</pre>
45
            Z = Z*Z + C
46
47
            n += 1
        end
48
49
        return n
50
    end
51
    function canvasRGB(height, width)
52
        return zeros(RGB, height, width)
53
54
    end
55
    function canvasHSV(height, width)
56
        return zeros(HSV, height, width)
57
58
    end
59
    xc(i, lx, nx) = lx * (2*i - 1) / nx
60
61
    function fractalGrays(nx, ny; lx=2, ly=2, maxiter=80)
62
        cv = canvasRGB(ny, nx)
63
64
        for row in 1:ny
65
            y = xc(row, ly, ny) - ly
66
67
            for col in 1:nx
68
69
                x = xc(col, lx, nx) - lx
                c = x + y * 1im
70
                m = mandelbrot(c, maxiter) / maxiter
71
72
                pixel = 1 - m ▷ RGB
73
                cv[row, col] = pixel
74
75
            end
        end
76
77
        return cv
78
    end
79
    function fractalColors(nx, ny; lx=2, ly=2, maxiter=80)
80
        cv = canvasHSV(ny, nx)
81
82
```

```
for row in 1:ny
83
              y = xc(row, ly, ny) - ly
84
85
              for col in 1:nx
86
                  x = xc(col, lx, nx) - lx
87
                  c = x + y * 1im
88
                  m = mandelbrot(c, maxiter) / maxiter
89
90
91
                  hue = 360 \times m
                                           # H between 0 and 360 (color wheel)
                  sat = 0.85
                                           # S between 0 and 1 (saturation)
92
                  val = m < 1 ? 1 : 0  # V between 0 and 1 (brightness)</pre>
93
                  cv[row, col] = HSV(hue, sat, val)
94
95
              end
         end
96
97
          return cv
     end
98
99
     function fractalCMap(nx, ny; lx=2, ly=2, maxiter=80, cname="Oranges")
100
         cv = canvasRGB(ny, nx)
101
102
         cmap_divs = 200
103
         cmap = colormap(cname, cmap_divs, logscale=true) ▷ reverse
104
105
         for row in 1:ny
106
              y = xc(row, ly, ny) - ly
107
108
              for col in 1:nx
109
                  x = xc(col, lx, nx) - lx
110
                  c = x + y * 1im
111
                  m = mandelbrot(c, maxiter) / maxiter
112
113
                  # To select a color of the colormap
114
                  cv[row, col] = cmap[ ceil(m * cmap_divs) > Int ]
115
              end
116
117
         end
118
          return cv
119
     end
120
     end # module MandelbrotFractal
121
```