Departamento de Matemática y Ciencia de la Computación

Licenciatura en Ciencia de la Computación

Laboratorio N°3

Implementación de Grafos

Matemática Computacional
Profesor: Nicolas Thériault

Matías Daza A.
María Maldonado P.

Índice general

1. Introducción

El siguiente informe tiene como objetivo la implementación de algoritmos que resuelven el problema de la representación gráfica de un grafo (dibujo)en un ambiente computacional. Existen varias maneras de representar grafos en un computador, las principales son:

- Listado de aristas
- Matriz de adyacencia
- Listado de adyacencia
- Listado dinámico (linked list)

2. Generación aleatoria de grafos

Para experimentar técnicas de trabajo en grafos (grandes), es práctico comprobarlas con grafos generados aleatoriamente que para grafos dados de manera explícita. En este laboratorio, generaremos grafos conectados no-dirigidos (1) simples (2) y de grado par (3), es decir: (1) se puede ir desde cualquier vértice hasta cualquier otro vértices pasando por aristas del grafo; (2) entre cada par de vértices hay a lo más una arista (no dirigida) y no hay aristas que vuelven a su vértice de origen; (3) cada vértice tiene un número par de aristas.

2.1. Matriz de adyacencia

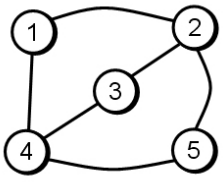
La matriz de adyacencia es una matriz cuadrada que se utiliza como una forma de representar relaciones binarias.

2.2. Construcción de la matriz a partir de un grafo

1. Se crea una matriz cero, cuyas columnas y filas representan los nodos del grafo.
2. Por cada arista que une a dos nodos, se suma 1 al valor que hay actualmente en la ubicación correspondiente de la matriz. Si tal arista es un bucle y el grafo es no dirigido, entonces se suma 2 en vez de 1.

Finalmente, se obtiene una matriz que representa el número de aristas (relaciones) entre cada par de nodos (elementos).

Existe una matriz de adyacencia única para cada grafo (sin considerar las permutaciones de filas o columnas), y viceversa.



M	1	2	3	4	5
1	0	1	0	1	0
2	1	0	1	0	1
3	0	1	0	1	0
4	1	0	1	0	1
5	0	1	0	1	0

Observación: La matriz de adyacencia es muy utilizada en la programación, porque su representación binaria y matricial se ajusta a la representación en los computadores..

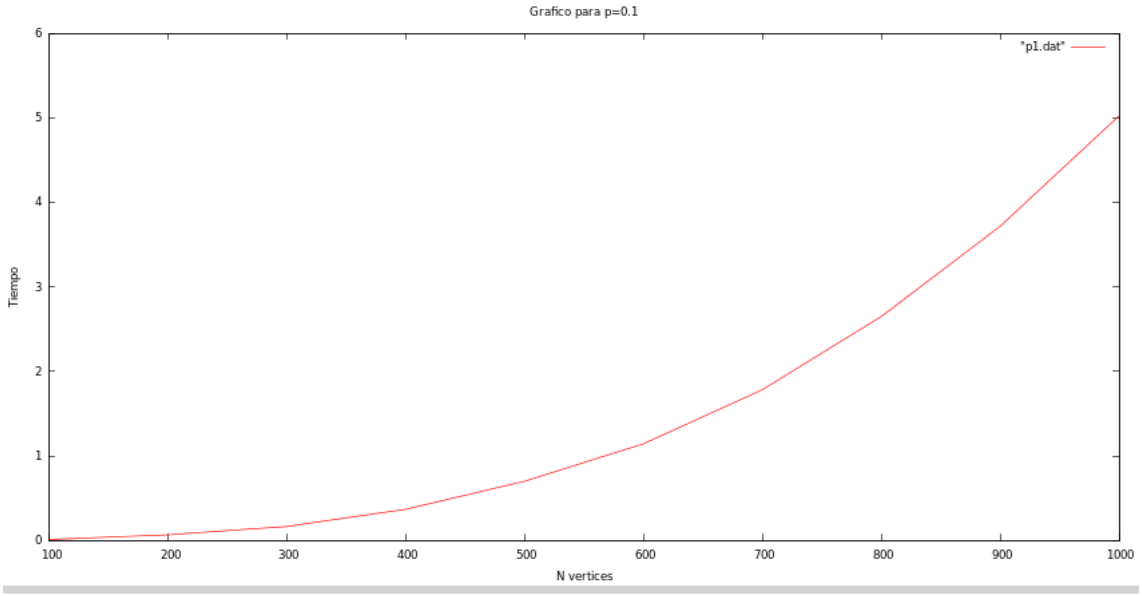
2.3. Formulación del experimento

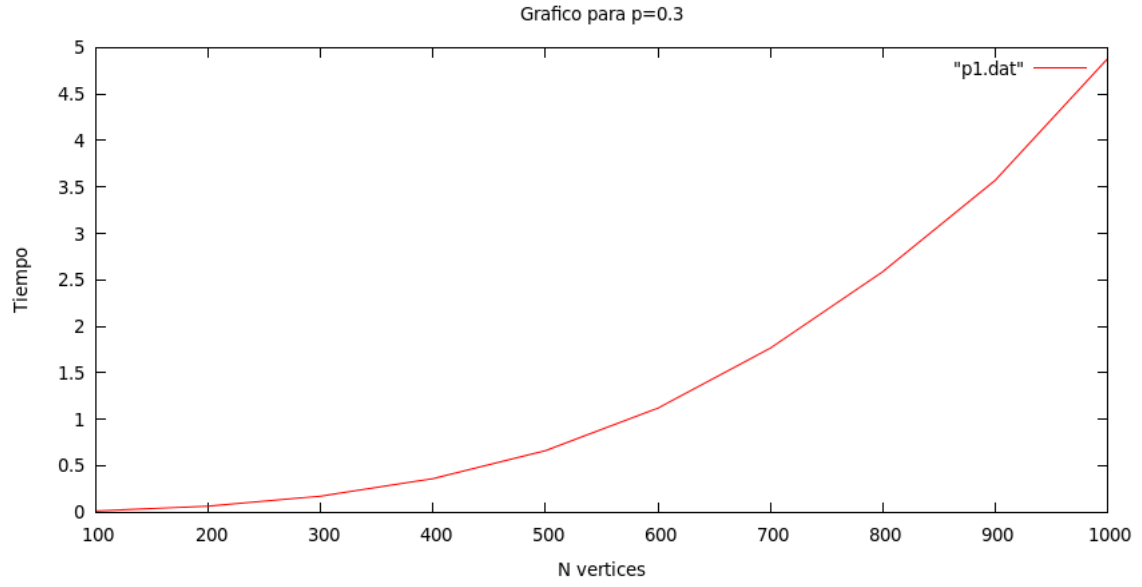
Experimento para n entre 100 y 1000 y $p \in \{ 0.1, 0.3, 0.5, 0.7, 0.9 \}$.

N	Probabilidad (p)	Tiempo de ejecución (s)
100	0.100000	0.009397
100	0.300000	0.013995
100	0.500000	0.013180
100	0.700000	0.010374
100	0.900000	0.007058
200	0.100000	0.065881
200	0.300000	0.065089
200	0.500000	0.047240
200	0.700000	0.050331
200	0.900000	0.019430
300	0.100000	0.163957
300	0.300000	0.173368
300	0.500000	0.144893
300	0.700000	0.130680
300	0.900000	0.048617
400	0.100000	0.367209
400	0.300000	0.361760
400	0.500000	0.327455
400	0.700000	0.256380
400	0.900000	0.101657
500	0.100000	0.698884
500	0.300000	0.662391
500	0.500000	0.596022
500	0.700000	0.495728
500	0.900000	0.201988
600	0.100000	1.143101
600	0.300000	1.119918
600	0.500000	1.010842
600	0.700000	0.813768
600	0.900000	0.330873
700	0.100000	1.782896
700	0.300000	1.765171
700	0.500000	1.570323
700	0.700000	1.302843
700	0.900000	0.453700
800	0.100000	2.649109
800	0.300000	2.587949
800	0.500000	2.319096
800	0.700000	1.875852
800	0.900000	0.685026
900	0.100000	3.716891
900	0.300000	3.570874
900	0.500000	3.278253
900	0.700000	2.620573
900	0.900000	0.931476
1000	0.100000	5.023305
1000	0.300000	4.879386
1000	0.500000	4.436449
1000	0.700000	3.588393
1000	0.900000	1.276908

2.4. Gráficos de desempeño

Para obtener la curva de desempeño de los algoritmos se graficará en función del tiempo que se demora con un n entre 100 y 1000 y una probabilidad dada, a continuación las curvas obtenidas:





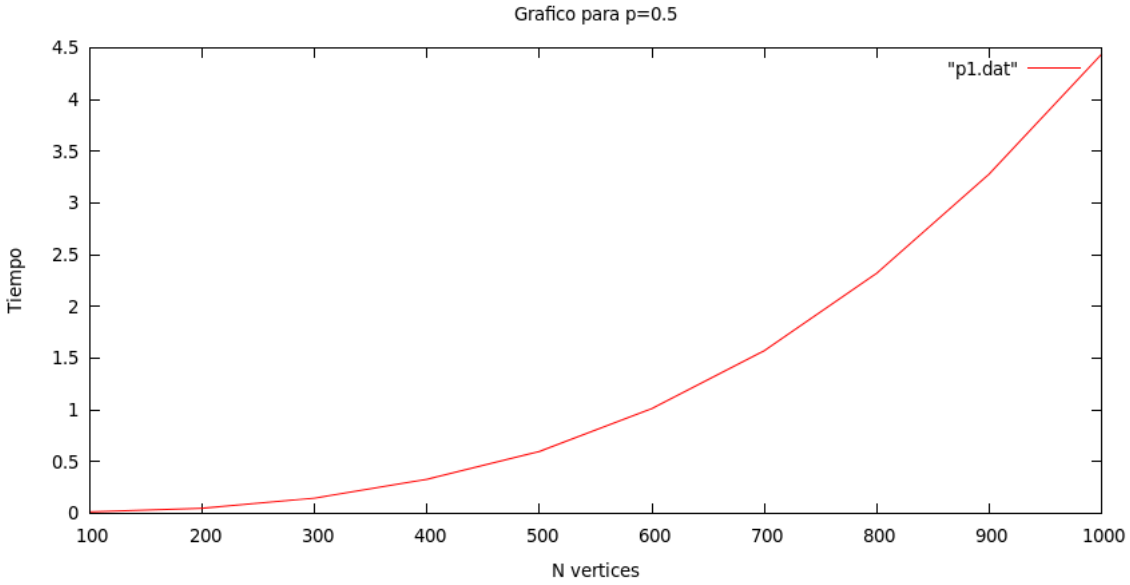


Grafico para p=0.5
"p1.dat"
Tiempo
N vertices
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
100
200
300
400
500
600
700
800
900
1000

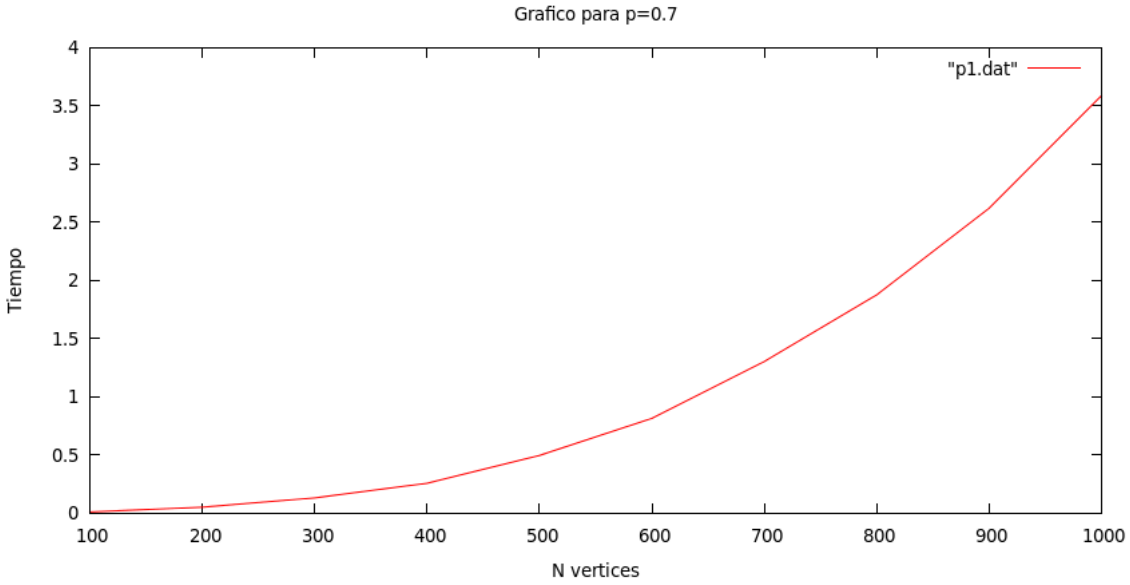


Grafico para p=0.7
"p1.dat"
Tiempo
N vertices
0
0.5
1
1.5
2
2.5
3
3.5
4
100
200
300
400
500
600
700
800
900
1000

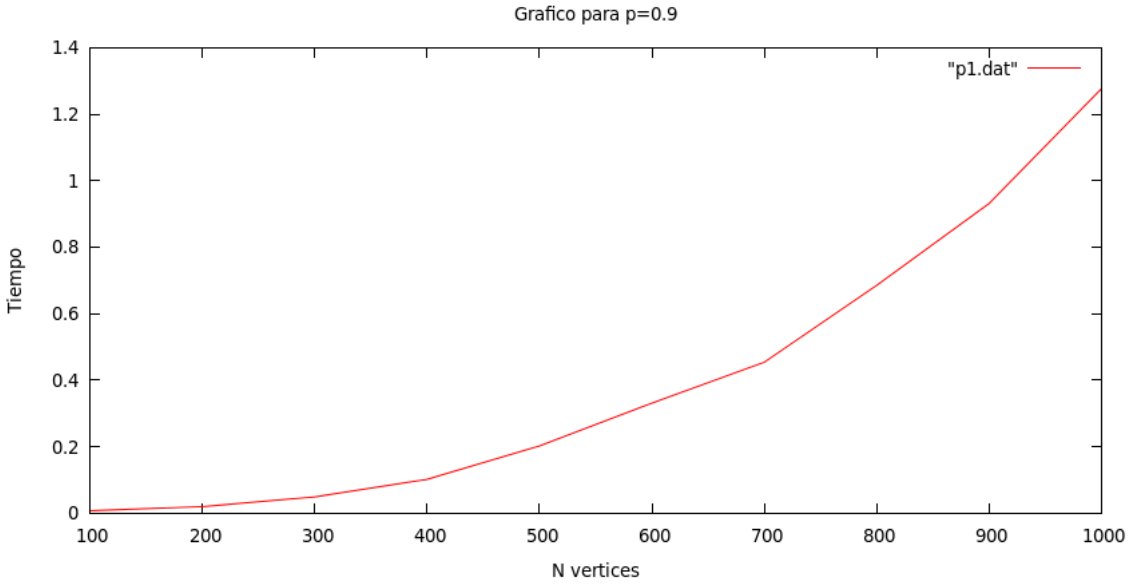


Grafico para p=0.9
"p1.dat"
Tiempo
N vertices
0
0.2
0.4
0.6
0.8
1
1.2
1.4
100
200
300
400
500
600
700
800
900
1000

3. Conclusiones

Lo que se observa al realizar los experimentos es que los gráficos forman una curva exponencial, esto se repite para cualquier probabilidad p, también se observa que mientras mayor es la probabilidad p este se demora menos en algunos casos (p=0.7 y p=0.9) ya que hay menos unos en la matriz, es decir, se demora menos en encontrar un ciclo de euler.

Podemos inferir que con probabilidad p=1, el tiempo de ejecución sería menor independiente de la cantidad de vértices de la matriz adyacente.

Se utilizó la matriz adyacente ya que facilita la implementación computacional para la teoría de grafos, sin embargo, es más fácil identificar las aristas y los vértices en un grafo convencional y luego completar la matriz adyacente, pero esto es válido solo para valores pequeños de n.

4. Estructuras de Datos Utilizadas

Se utilizaron matrices y TDA's como punteros y pilas.

5. Lectura de programa desde la consola

La lectura del programa se indica a continuación:

Para compilar:

- gcc -o Ejecutar lab3.c

Para ejecutar:

- ./Ejecutar

6. Implementación

El algoritmo está implementado en lenguaje C.

7. Plataforma Computacional

El programa fue ejecutado en un computador con las siguientes características:

- **Procesador:** Intel Core i3-350M CPU 2.27GHz
- **Memoria RAM:** 5,6 GiB
- **Tipo de SO:** 64 bits
- **Sistema Operativo:** Linux - Ubuntu 14.04 LTS

7.1. Librerías

- stdio.h
- stdlib.h
- time.h

8. Bibliografía

- Libro : Matemática Discreta y Combinatoria , Ralph P. Grimaldi