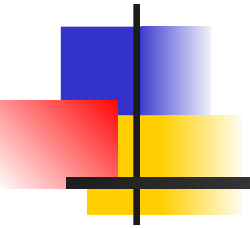
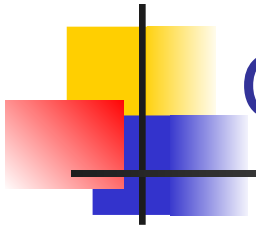


Introducción a la Teoría de Grafos

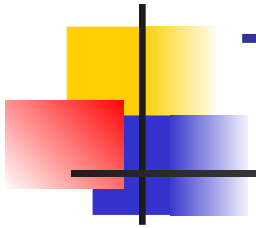
(Semana 15)

SI138 - Estructura de Datos y Algoritmos
MS Cynthia Muñoz Jugo
Universidad Peruana de Ciencias Aplicadas
2006 – 01



Objetivos de Hoy

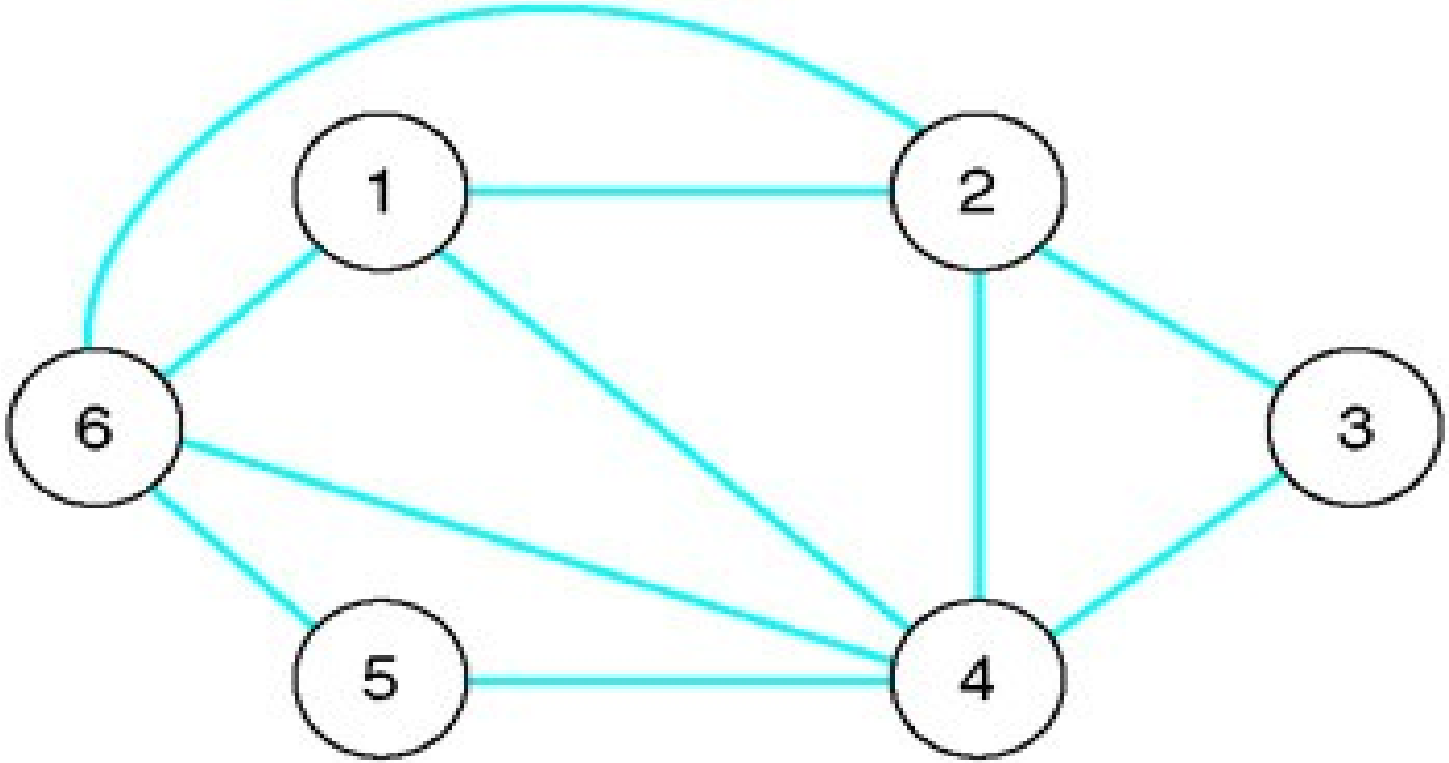
- Teoría de Grafos
  - Definiciones, terminología
  - Formas de representaciones:
    - Listas
    - matrices
- Problemas resueltos con Grafos
  - Enunciados
- Algoritmo de la Ruta Más Corta: Dijkstra



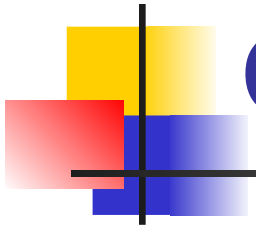
Teoría de Grafos

- Muchos problemas computacionales se pueden resolver definiéndolos en términos de grafos
- La teoría de grafos estudia las propiedades de los grafos y muestra soluciones a problemas específicos.
- Un grafo es una representación donde se puede almacenar data de elementos y sus relaciones, así una arista representa una conexión de los dos elementos que une.

Un Grafo que puede representar varias ideas



Los nodos pueden representar ciudades y cada arista puede representar el precio o el kilometraje de una ciudad a otra.



Grafos

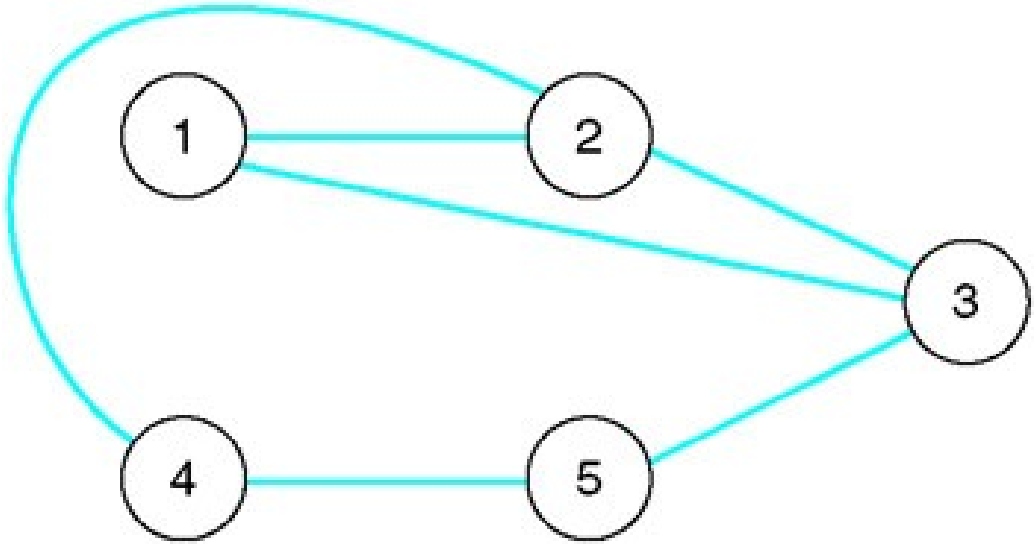
- Un Grafo G se denota por G = (V, E)
- donde
- V es el conjunto de Vértices (Vertex) o Nodo o Puntos
- E es el conjunto de aristas (Edges)

Mas terminología de Grafos

- Como máximo E puede ser V^2
- El grado de un grafo es el número de vértices que tiene.
- Un grafo que no tiene ningún vértice o solo tiene uno, es un grafo Trivial
- Hay grafos dirigidos y no dirigidos

Grafos No Dirigidos

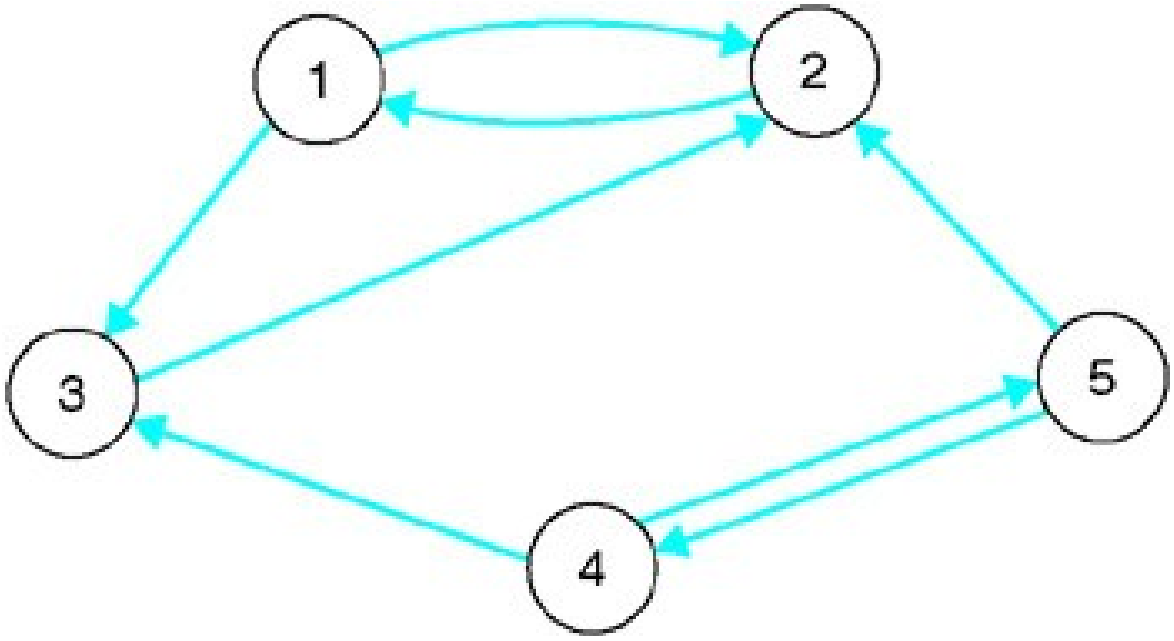
- Este grafo se denotaría por:
- G = ({1, 2, 3, 4, 5,}, {{1, 2}, {1, 3}, {2, 3}, {2, 4}, {3, 5}, {4, 5}})



[Analysis of Algorithms: An Active Learning Approach by Jeffrey J. McConnell]

Grafo Dirigido

- G = ({1, 2, 3, 4, 5}, {(1, 2), (1, 3), (2, 1), (3, 2), (4, 3), (4, 5), (5, 2), (5, 4)})



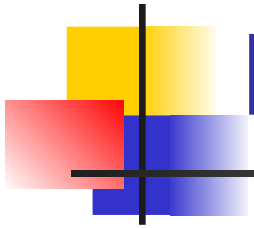
[Analysis of Algorithms: An Active Learning Approach by Jeffrey J. McConnell]

Mas Terminología

- Grado de un Nodo: es el número de aristas que tienen al nodo como fin
- Un nodo que no tiene ninguna arista que llega a él, se llama un nodo aislado y tiene como grado: 0.

Camino (Path)

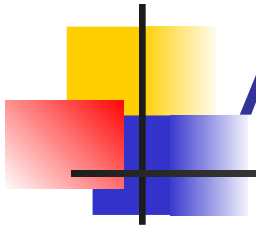
- Camino o Trayectoria es el conjunto de vértices que se usan para llegar de un vértice a otro pasando a través de aristas que no pueden ser repetidas y deben estar ubicadas adyacentemente, igualmente, no se debe pasar nuevamente por un mismo vértice.
- Aristas adyacentes son las que aristas que tienen un vértice en común.



Representaciones de Grafos

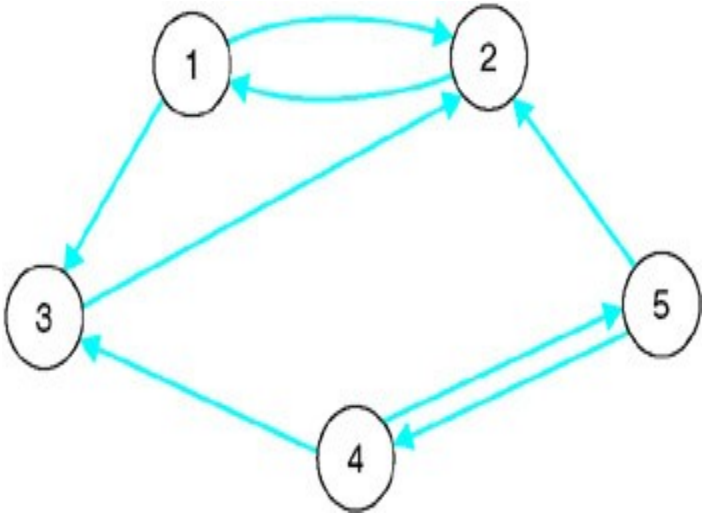
- Usando Matrices de Adyacentes
- Usando Listas de Adyacentes

Usando Matrices de Adyacentes



- Usar una matriz de (N x N), donde N es el numero de vértices, así en cada elemento de la matriz dirá si hay conexión entre vértice *i* y vértice *j*.
- Así M[i,j] = 0 si no hay conexión entre vértice i y vértice j.
- Así M[i,j] = 1 si hay conexión entre vértice i y vértice j.

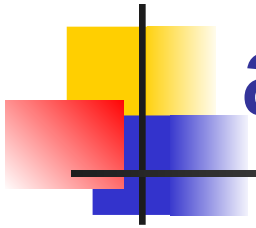
Grafos usando Matrices de Adyacentes



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

Así si hay una arista de vértice “a” hacia un vértice “b”, la celda (a, b) será 1, en caso contrario, sería 0.

Grafos usando listas de adyacentes

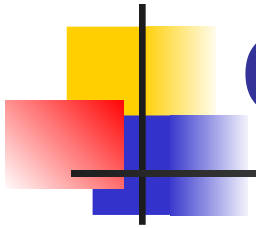


- Tener un arreglo en donde cada ítem i es una lista de vértices adyacentes que son los vértices que están conectados por una arista desde el vértice *i*.

- La estructura será así:

```
pnodo = ^nodo;
nodo = record
   vertice: integer;
   siguiente: pnodo;
End;
listaArista = array[1..n] of pnodo;
```

Problemas más comunes resueltos con Teoría de Grafos



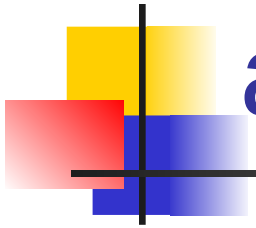
- **Graph coloring**
- Many problems have to do with various ways of coloring graphs, for example:
- The four-color theorem

- **Route problems**
- Hamiltonian path and cycle problems (NP-Complete)
- Minimum spanning tree
- Shortest path problem
- Traveling salesman problem (NP-Complete)

- **Network flow**
- Max flow min cut theorem

- [Fuente: http://en.wikipedia.org/wiki/Graph_theory]

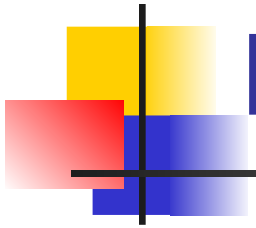
El Camino mas corto: El algoritmo de Dijkstra



- Busca el camino mas corto en un grafico que puede ser cíclico y donde las aristas tienen peso positivo.
- Es un algoritmo ávido, voraz (greedy)
- Escoge a un nodo para iniciar el camino
- Explora en etapas, para lo cual escoge la mejor opción.
- Depende de la implementación de las estructuras utilizadas para analizar sus tiempos.
- Se usará una estructura para:
  - Guardar los pesos de las aristas en una matriz p donde el peso de ir de a hacia b sea p[a,b]
  - Guardar los vértices ya explorados: S,
  - Guardar los vértices no explorados totalmente: Q.
  - Guardar las distancias mínimas desde el inicio hasta el vértice: d. Al comienzo d[v] = infinito
  - Guardar el vértice anterior al momento de encontrar las distancias mínima de un vértice v: r[v] = vértice anterior en la solución hasta el momento.
  - Al comienzo todos las respuestas r[v] = 0

Algoritmo de Dijkstra (en palabras)

- Seleccionar el nodo que se quiere tomar como de inicio y colocar su d[inicio] = 0.
- Agregar a la lista de vértices no explorados Q todos los vértices en V
- Mientras Q no esta vacío
  - Escoger el nodo “n” con el camino más corto desde el inicio (esto se hace viendo quien tiene la menos distancia d[i]. Al comienzo, será el inicio el que tiene 0)
  - Colocar el nodo “n” en la lista de vértices ya explorados S y quitarlo de Q.
  - Para cada vértice v que se encuentra en la lista de adyacentes del nodo “n” ver:
    - Si d[v] > d[n] + p[n, v] entonces
      d[v] ← d[n] + p[n, v]
      r[v] ← n
- Al final, la solución se encontrará en r para ver el vértice anterior y en d para ver las distancia hasta ese vértice.



Bibliografía

- Cormen, Leiserson, Rivest, Stein. Introduction to Algorithms. 2001.
- Mark Allen Weiss. Estructura de Datos y Algoritmos. 1995
- Jeffrey J. McConnell Analysis of Algorithms: An Active Learning Approach. 1994
- http://en.wikipedia.org/wiki/Graph_theory