DS.GR.1

Graph Algorithms

Chapter 9 Overview

- **Definitions**
- **Representation**
- **Topological Sort**
- **Graph Matching (for Program 3)**
- **Shortest Path Algorithms**
- **Network Flow Problems**
- **Minimum Spanning Trees**
- **Depth-First and Breadth-First Search**
- **NP-Completeness**

DS.GR.2

Graphs and Digraphs

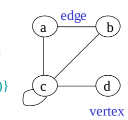
A graph is a pair G = (V,E) where

V is a set of vertices (or nodes)
E is a set of edges (or arcs)

Example:

V = {a, b, c, d}

E = { (a,b), (b,a),(a,c)
(c,a),(b,c),(c,b)
(c,d), (d,c),(c,c)}



or E = { {a,b}, {a,c}
{b,c}, {c,d} {c,c} }

An (undirected) graph represents a symmetric relation.

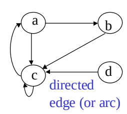
DS.GR.3

A digraph G = (V,E) is a graph in which the edges are directed from the first node to the second.

Example:

V = {a, b, c, d}

E = { (a,b), (a,c), (b,c),(c,a),(c,c),(d,c) }

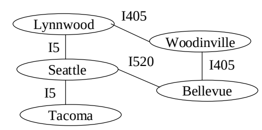


This graph represents a binary relation that is not symmetric.

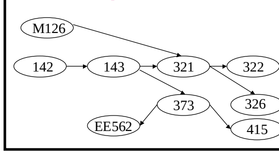
DS.GR.4

More Examples

Undirected Graph

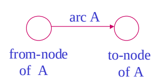


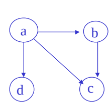
Directed Graph



DS.GR.5

Some Additional Digraph Terminology





outdegree(a) = 3
indegree(a) = 0

outdegree(c) = 0
indegree(c) = 2

Node **a** is a **source;** Node **c** is a **sink.**

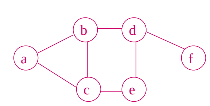
For graphs, we just have the degree of a node.

DS.GR.6

Paths through Graphs

A path in a graph is a sequence of vertices w1, w2, … , wN such that {wi,w[i+1]} is in E for $1 \le i < N$.

The length of the path is N–1, the number of edges on the path.



A path from a node to itself with no repeated edges is a cycle.

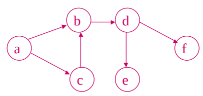
What are the cycles of this graph?
What are all the paths from a to f?

DS.GR.7

Paths through Digraphs

A path in a digraph is a sequence of vertices w1, w2, … , wN such that (wi,w[i+1]) is in E for 1 ≤ I < N.

The only difference is moving along directed edges in the proper direction.



What are the cycles of this digraph?
What are all the paths from a to f?

This is an **acyclic** digraph.

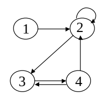
DS.GR.8

Graph Representations

- N x N Adjacency Matrix for N node graph

Digraphs:

$$A[i,j] = \begin{cases} 1 & \text{if there is an arc from node i to node j} \\ 0 & \text{otherwise} \end{cases}$$

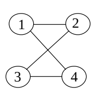


	1	2	3	4
1	0	1	0	0
2	0	1	1	0
3	0	0	0	1
4	0	1	1	0

DS.GR.9

Graphs:

$$A[i,j] = \begin{cases} 1 & \text{if there is an edge connecting node i and node j} \\ 0 & \text{otherwise} \end{cases}$$



	1	2	3	4
1				
2		??		
3				
4				

Note: since A[i,j] = 1 iff A[j,i] = 1, we only need to store half the matrix.

DS.GR.10

Adjacency Matrices

- Are usually bit matrices, unless the graph has weights on the edges, in which case the 1-bits are replaced by the weights.
- Are used in certain algorithms that make use of simple matrix operations, such as AND and OR.
- Are inefficient for some algorithms, because

  a. They waste space when the graph is sparse

  b. You have to search a whole row to find those nodes adjacent to a given one.

DS.GR.11

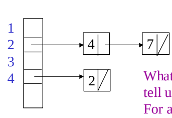
2. Linked Representation: Adjacency Lists

N element array of lists

V[i] points to a list of nodes that are adjacent to node i.

For digraphs, this is usually the list of nodes reachable by following one arc out of node i.

But, we can have another set of lists for nodes whose arcs go into node i.



What does this structure tell us for a graph?
For a digraph?

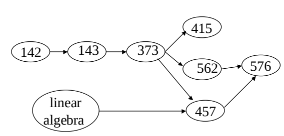
DS.GR.12

Topological Sort

Given an acyclic digraph G, where (<<)
(Ni,Nj) ∈ E means that Ni precedes Nj

Find an ordering of the nodes
N1, N2, N3, . . . , Nn

so that N1 << N2 << N3 << . . . << Nn.



Find an ordering in which all these courses can be taken, satisfying their prerequisites.

DS.GR.13

Complexity of Topological Sort

Assuming the adjacency list representation,

- The indegree of each vertex is computed in an initialization step. $|V|$
- Each node will go into the queue and come out exactly once. $|V|$
- Each edge will be examined once (in the for loop when its from-node is processed). $|E|$

So the complexity is $O(|V| + |E|)$.

DS.GR.14

Graph Matching

Input: 2 digraphs $G1 = (V1,E1)$, $G2 = (V2,E2)$

Questions to ask:

- Are G1 and G2 isomorphic?
- Is G1 isomorphic to a subgraph of G2?
- How similar is G1 to G2?
- How similar is G1 to the most similar subgraph of G2?

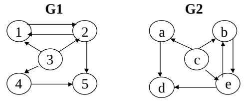
DS.GR.15

Isomorphism for Digraphs

G1 is isomorphic to G2 if there is a 1-1, onto mapping $h: V1 \rightarrow V2$ such that

$(vi,vj) \in E1$ iff $(h(vi), h(vj)) \in E2$

G1 **G2**



Find an isomorphism $h: \{1,2,3,4,5\} \rightarrow \{a,b,c,d,e\}$.
Check that the condition holds for every edge.

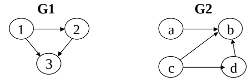
DS.GR.16

Subgraph Isomorphism for Digraphs

G1 is isomorphic to a subgraph of G2 if there is a 1-1 mapping $h: V1 \rightarrow V2$ such that

$(vi,vj) \in E1 \Rightarrow (h(vi), h(vj)) \in E2$

G1 **G2**



Isomorphism and subgraph isomorphism are defined similarly for undirected graphs.

In this case, when $(vi,vj) \in E1$, either (vi,vj) or (vj,vi) can be listed in E2, since they are equivalent and both mean $\{vi,vj\}$.

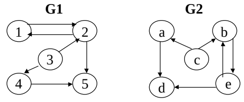
DS.GR.17

Similar Digraphs

Sometimes two graphs are close to isomorphic, but have a few "errors."

Let h(1)=b, h(2)=e, h(3)=c, h(4)=a, h(5) = d.

G1 **G2**



(1,2)	(b,e)
(2,1)	(e,b)
X	(c,b)
(4,5)	(a,d)
(2,5)	(e,d)
(3,2)	X
(3,4)	(c,a)

The mapping **h** has **2 errors**.

$(c,b) \in G2$, but $(3,1) \notin G1$

$(3,2) \in G1$, but $(c,e) \notin G2$

DS.GR.18

Error of a Mapping

Intuitively, the error of mapping h tells us
- how many edges of G1 have no corresponding edge in G2 and
- how many edges of G2 have no corresponding edge in G1.

Let G1=(V1,E1) and G2=(V2,E2), and let $h:V1 \rightarrow V2$ be a 1-1, onto mapping.

forward error: $EF(h) = |\{(vi,vj) \in E1 \mid (h(vi),h(vj)) \notin E2\}|$
edge in E1 — corresponding edge not in E2

backward error: $EB(h) = |\{(vi,vj) \in E2 \mid (h^{-1}(vi),h^{-1}(vj)) \notin E1\}|$
edge in E2 — corresponding edge not in E1

total error: $Error(h) = EF(h) + EB(h)$

relational distance: $GD(G1,G2) = \min Error(h)$
for all 1-1, onto $h:V1 \rightarrow V2$

DS.GR.19

Variations of Relational Distance

- normalized relational distance:
  Divide by the sum of the number of edges in E1 and those in E2.
- undirected graphs:
  Just modify the definitions of EF and EB to accommodate.
- one way mappings:
  h is 1-1, but need not be onto
  Only the forward error EF is used.
- labeled graphs:
  When nodes and edges can have labels, each node should be mapped to a node with the same label, and each edge should be mapped to an edge with the same label.

DS.GR.20

Graph Matching Algorithms

- graph isomorphism
- * subgraph isomorphism
- * relational distance
- attributed relational distance (uses labels)

Subgraph Isomorphism

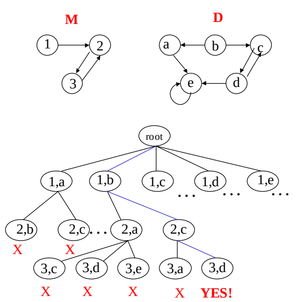
Given model graph M = (VM,EM)
data graph D = (VD,ED)

Find 1-1 mapping $h:VM \rightarrow VD$

satisfying $(vi,vj) \in EM \Rightarrow ((h(vi),h(vj)) \in ED$.

DS.GR.21

Method: Backtracking Tree Search



DS.GR.22

Treesearch for Subgraph Isomorphism in Digraphs

```
procedure Treesearch(VM, VD, EM, ED, h)
  {
  v = first(VM);
  for each w ∈ VD
    {
    h' = h ∪ {(v,w)};
    OK = true;
    for each edge (vi,vj) in EM satisfying that
        either 1. vi = v and vj ∈ domain(h')
            or 2. vj = v and vi ∈ domain(h')
          if ( (h'(vi),h'(vj)) ∉ ED )
             {OK = false; break;};

    if OK  {
            VM' = VM – v;
            VD' = VD – w'
             if isempty(VM') output(h');
             else Treesearch(VM',VD',EM,ED,h')
             } } }
```

DS.GR.23

Branch-and-Bound Tree Search

Keep track of the least-error mapping.