GB13604 - Maths for Computer Science

Lecture 4 – Graphs Part I

Claus Aranha
caranha@cs.tsukuba.ac.jp

College of Information Science

2018-10-24

Last updated October 30, 2019

This course is based on Mathematics for Computer Science, Spring 2015, by Albert Meyer and Adam Chlipala, Massachusetts Institute of Technology OpenCourseWare.

Week 3: Exercise Discussion

Week 4 and 5 summary

Graphs

Lecture I: Chapter 9

- Walks and Paths
- Scheduling and Partial Orders
- Equivalence Relations
- Idea of Isomorphism

Lecture II: Chapter 11

- Using Isomorphism
- Coloring and Connectivity
- Spanning Trees
- Matching

Idea: Course Registration

Imagine a university where you can take any subject that you want, as long as you satisfy the requirements.

Code	Lecture	Prerequisites
0000	Social Questions	*none*
0001	Intro to Programming	*none*
0002	Calculus I	*none*
0003	Programming Theory	*0001*
0004	Linear Algebra	*0000, 0002*
0005	Programming Challenges	*0000, 0001, 0003*
0006	Computer Graphics	*0003, 0004*

How long would it take for you to graduate, if you could take 2 lectures per semester?

Idea: Course Registration

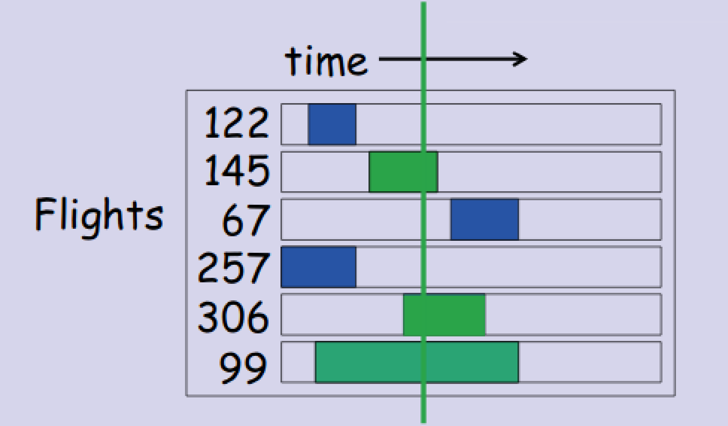
A new lecture is proposed, *Maths for Computer Science*, as below.

Code	Lecture	Prerequisites
0000	Social Questions	*none*
0001	Intro to Programming	*none*
0002	Calculus I	*none*
0003	Programming Theory	*0001,* **0007**
0004	Linear Algebra	*0000, 0002*
0005	Programming Challenges	*0000, 0001, 0003*
0006	Computer Graphics	*0003, 0004*
0007	**Maths for Computer Science**	***0005***

Now, how many semesters would it take to graduate, if you could take two lectures per semester?

Idea2: Airplane

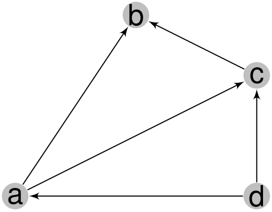
- In an airport, each airplane needs a gate when it is in the ground.
- How many gates do we need, if we know the times of the planes?



Problems as Graphs

- Many problems can be described by the *relationship* between the entities in the problem (planes, courses, etc).
- This relationship can be described mathematically using the graph structure.

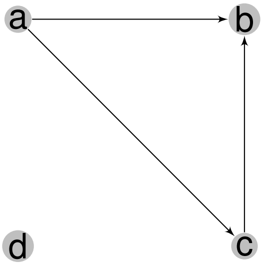
Directed Graphs (DiGraphs)



A graph is defined by a set of Vertices (V) and a set of Edges (E). The set of edges is also a relation from V to V.

- $V = \{a, b, c, d\}$
- $E = \{(a, b), (a, c), (c, b), (d, c), (d, a)\}$
  - $E : V \to V$;
  - $(a, c) \in E$;
  - $E(a) = c$; careful!
  - $a \to c$

Relations (Week 2) and Graphs (Week 4)

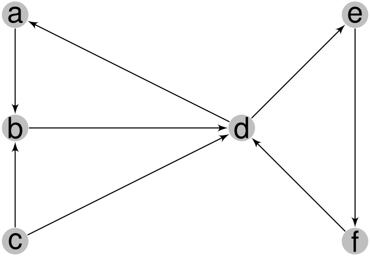


$$V = \{a, b, c, d\}$$
$$E = \{(a, c), (a, b), (c, b)\}$$

A digraph with vertices V is the same as a binary relation on V.

Every Binary Relation can also be written as a directed graph!

Digraphs: Matrix Representation



	a	b	c	d	e	f
a	0	1	0	0	0	0
b	0	0	0	1	0	0
c	0	1	0	1	0	0
d	1	0	0	0	1	0
e	0	0	0	0	0	1
f	0	0	0	1	0	0

Adjacency Matrix

A value in the adjacency matrix is one if there is an edge between the two vertices, 0 otherwise.

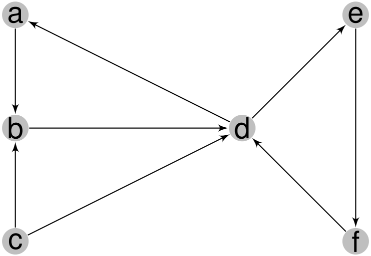
$$A(v_i, v_j) = 1 \textbf{ iff } E(v_i) = v_j$$

Walks and Paths

There are many properties that we can study in a graph.

- A **Walk** is a sequence of successive edges;
- A **Path** is a walk that does not repeat vertices;

Walk example

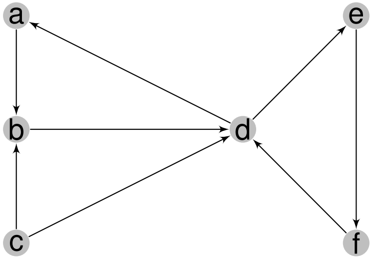


Walk - sequence of successive edges

b → d → e → f → d → e

- **lengh:** 5 edges (NOT 6 vertices)
- **as a relation:** $E(E(E(E(E(a)))))$

Path example



Path - walk without repeating vertices

e ⟶ f ⟶ d ⟶ a ⟶ b **Stuck!**

- **lengh:** 4 edges
- (NOT 5 vertices)

Walks and Paths: Shortest Path

Lemma: The shortest walk between two vertices is a Path.

1. **Proof (by contradiction):** Suppose the lemma is not true
2. The shortest walk between v_s and v_e is not a path.
   $w_s = v_s \rightarrow v_i \rightarrow \ldots \rightarrow v_j \rightarrow v_e$
3. This means that this path has at least one repeated vertice, and a walk of size ≥ 1 between the repetitions.
   $w_s = v_s \rightarrow \ldots \rightarrow v_k \rightarrow \ldots \rightarrow v_k \rightarrow \ldots \rightarrow v_e$
4. We can create a new, shorter walk by removing the edges from w_s between v_k and v_k. Contradiction!
   $w'_s = v_s \rightarrow \ldots \rightarrow v_k \rightarrow \ldots \rightarrow v_e$

The length n walk relation

$$vG^n w \tag{1}$$

- "There is a walk of length n from v to w"
- G^n is called the length n walk relation for G

- G^1 is the relation of nodes directly connected by edges.
- **lemma:** $G^n \circ G^m = G^{n+m}$ (remember that $R \circ S = R(S)$)
- $x\ G^m \circ G^n\ y \to \exists z, x\ G^m\ z\ G^n\ y$ ($z \in G^n(y)$ and $x \in G^m(z)$)

Adjacency Matrices and Composition

- A_G ::= Adjacency Matrix for relation G
- **lemma:** $A_{G \circ H} = A_G \odot A_H$
  where $\odot$ is the Boolean Matrix Multiplication

- This allows us to compute A_{G^n} by Fast Matrix Exponentiation
- $A_{G^n} = A_{G^{n/2}} \odot A_{G^{n/2}} = \dots$

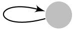
Walk Relation of a DiGraph

- G^* is the walk relation of G
- uG^*v **iff** $\exists$ walk from u to v of any length

How do we calculate calculte G^*? Walks can be infinite, so is there a **finite** algorithm to calculate it?

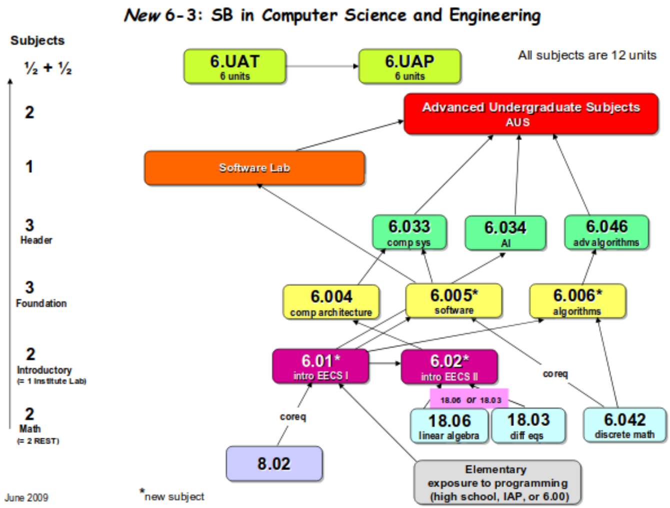
Walk Relation of a DiGraph

Algorithm to calculate G^*:

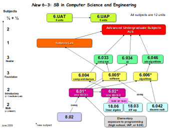
- Add self loops to the graph: 
- This is equivalent to defining $G^{\leq} = G \cup G^0$
- $G^{\leq}$ has a walk length n **iff** G has a walk of length $\leq n$
- $G^* = (G^{\leq})^{n-1}$

QUIZ: Why is it necessary to add the self-loop?

Applications of Walks: Prerequisite Trees



Prerequisite Trees



- Direct Prerequisite: Req(6.046) = 6.006
- Indirect Prerequisite: Req(6.046) = $\{6.006, 6.042, 6.01, 6.00, 8.02\}$

u is an indirect prerequisite of v means that:

There is a positive length walk from u to v in graph D.

$$uD^+v$$

Requisites, Cycles and DAGs

- A closed walk is a walk that starts and ends at the same vertex.

  Q: How long does it take to graduate if there is a closed walk in the prerequisite graph?

- A cycle is a closed walk where the only repeat vertex is at the beginning and end.
  $v_0 \to v_1 \to \ldots \to v_n - 1 \to v_0 | v_{i>0}$ does not repeat.
  - **OR** A cycle is a path from $v \to w + (w, v) \in E$

- A Directed Acyclic Graph (DAG) is a digraph that has no positive length cycles.

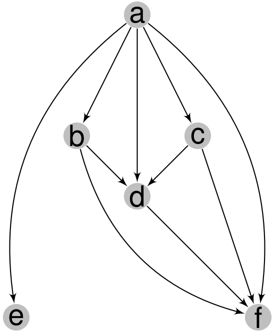
DAG Examples

- Class Prerequisite Graphs;
- Ordered Task List:
  "first add rice, then add water, then press cook button"

Some weird things can be described as DAGs:

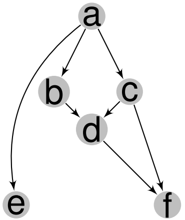
- The Successor Relation: $n \longrightarrow n+1$, $a < b$
- The Subset Relation $\subset$: $\{1, 2\} \subset \{1, 2, 3\}$
- Dynamic Programming;
- Induction Proofs;

DAG Walk Relation



Given a DAG A, what is the smallest DAG B with the same Walk Relation?

- $a \to e$
- $a \to c \to f$
- $a \to b \to d \to f$



B is the Covering Edges of DAG A;

Using DAGs for Scheduling

$18.01 \to 6.042$
$18.01 \to 18.02$
$18.01 \to 18.03$

$6.001 \to 6.034$
$6.042 \to 6.046$
$8.02 \to 6.002$
$18.03, 6.002 \to 6.004$

$6.001, 6.004 \to 6.033$
$6.033 \to 6.857$
$6.046 \to 6.840$

u is a indirect prerequisite of *v* if there is a posiive length walk in graph *R*: uR^+v

$$18.01 \to 6.042 \to 6.046 \to 6.840$$

Scheduling: Minimal Subject

- A minimal subject is does not have any prerequisites.

  nothing $\rightarrow$ 18.01, nothing $\rightarrow$ 6.001, nothing $\rightarrow$ 8.02

- A minimum subject comes before all other subjects.
  Indirect Prerequisite of all subjects!

  none in this example...

- Maximal and Maximum subjects have similar definition.
  QUIZ: What is a Maximum subject?

Scheduling

$18.01 \rightarrow 6.042$
$18.01 \rightarrow 18.02$
$18.01 \rightarrow 18.03$

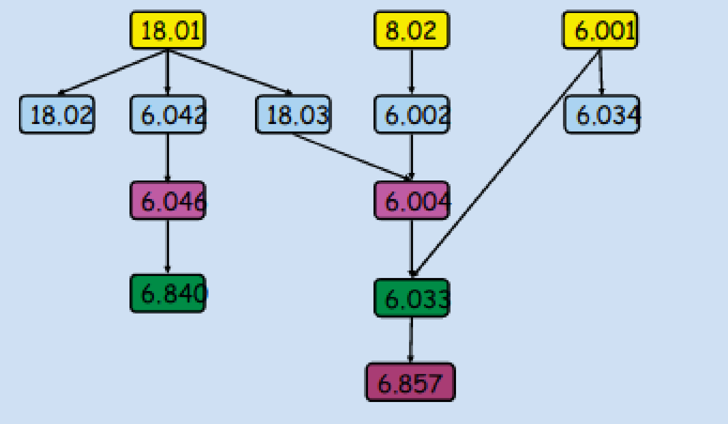
$6.001 \rightarrow 6.034$
$6.042 \rightarrow 6.046$
$8.02 \rightarrow 6.002$
$18.03, 6.002 \rightarrow 6.004$

$6.001, 6.004 \rightarrow 6.033$
$6.033 \rightarrow 6.857$
$6.046 \rightarrow 6.840$

Greedy Scheduling:

1. Identify Minimal Subjects;
2. Add Minimal Subjects to Schedule;
3. Remove Minimal Subjects;
4. Return to Step 1

Greedy Scheduling



Anti-Chains

- An anti-chain is a set of subjects which does not include indirect requisites
- The subjects in an anti-chain can be taken in any order
- They are also called incomparable (no path)

Example: $\{6.046, 6.004\}, \{6.001, 6.002, 6.046, 18.02\}$

A Lazy Scheduling

- Can we take only 1 subject per term?

  18.01, 6.001, 8.02, 6.002, 18.03, 6.034, 6.042, 18.02, 6.004, 6.046, 6.033, 6.840, 6.857

- This is called a “topological sort”.

- A chain is a set of subjects which all have a prerequisite relation to each other.

- It is possible to show that the maximum chain is the requisite and necessary numbers of terms to finish a program.

Parallel Processing

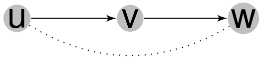
- The schedule of courses in terms is an example of the more general idea of parallel scheduling
  - minimum terms to graduate: Minimal parallel time (assuming no limit on parallel tasks)
  - Minimal Parallel Time = Max Chain Size
  - maximum term load: Number of processors needed;
  - Processors for minimum time $\leq$ maximum anti-chain size.

- Minimum Load with n tasks and m max chain size:

$$\text{min. load} \geq \lceil n/m \rceil \tag{2}$$

Walks and transitivity

- If there is a walk from u to v, and a walk from v to w
- This implies there is a walk from u to w



Expressing this idea as a walk relation in G:

$$uG^{+}v \wedge vG^{+}w \implies uG^{+}w \tag{3}$$

Transitivity in Relations

Any relation **R** is transitive if: $xRy \wedge yRz \implies xRz$

DAG and assimetry

In an Acyclic Graph D we can see that

- A positive length path from u to v implies no path from v to u;
- $uD^+v \implies \text{NOT}(vD^+u)$
- Property of Assimetry or Assimetry Relation R

Strict Partial Order

A relation R is a Strict Partial Order **iff** it is **Transitive** and **Assimetric**.

Examples:

- The $\subset$ relation on sets
- The “indirect prerequisite” relationship on subjects.
- The $<$ relationship on $\mathbb{R}$

R is an SPO **iff** $R = D^+$ for some DAG D.

Path Total Orders

A partial order is also Path Total if for any two distinct elements, one will be "greater than" another.

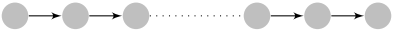
Example: $<$ or $\leq$ on $\mathbb{R}$: if $x, y \in \mathbb{R}, x \neq y \implies x > y$ or $y > x$

Counter-Example: $\subset$ in POW($\mathbb{N}$): $\{1,3\} \not\subset \{2,5\} \not\subset \{1,3\}$

- Relation R is path total: if $x \neq y \implies xRy \vee yRx$
- This means there are no imcomparable elements

Path totality

In a path total relation, the whole graph is a chain



A weak partial order is the same as a strict partial order R, except that *aRa* always holds:

- Examples: $\subseteq$ on sets, $\leq$ on $\mathbb{R}$
- Weak Partial Orders define the property of Reflexivity
- Relation R on A is reflexive **iff** $aRa, \forall a \in A$

Assimetry and Antissimetry

Assimetry

- Reflexibility is never allowed
- R is assimetric **iff**:

$$xRy \implies \text{NOT}(yRx) \tag{4}$$

Antissimetry

- Reflexibility is sometimes allowed
- R is antissimetric **iff**

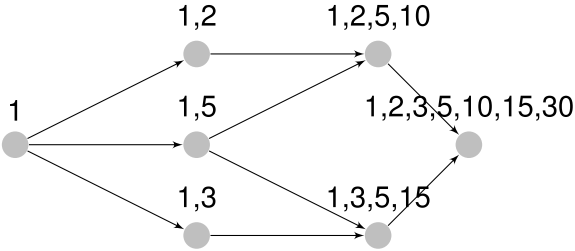
$$xRy \implies \text{NOT}(yRx), \text{ for } x \neq y \tag{5}$$

Definition of Weak Partial Order

R is a WPO **iff**
$R = D^*$ for some DAG D

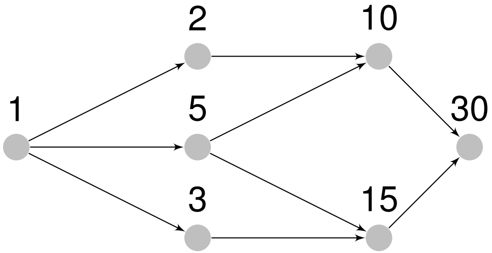
Proper Subset Relation

- $A \subset B$ means that B has everything A has, and something extra ($B \not\subset A$)
- Example of proper subset relation:



Partial Order: Proper Divides

- a proper divides b if $a|b$ and $a \neq b$



- The proper divide relation has **the same shape** as the $\subset$ relation.
- Both relations are isomorphic.

Isomorphism

- Two graphs are isomorphic if they have the same connections
- More formally, two graphs are isomorphic if there is a edge preserve matching (bijection) between their vertices.
- G_1 isomorphic $G_2 \iff \exists$ bijection $f : V_1 \to V_2$ with $(u, v) \in E_1 \iff (f(u), f(v)) \in E_2$

Isomorphism, $\subset$ and partial orders

Theorem: Every strict p.o. R is isomorphic to some collection of sets partially ordered by $\subset$.

Proof (by construction):

- Map element a to the set of elements below it.
- in other words, a maps to $\{b \in A | bRa \vee b = a\}$ (remember that NOT(aRa))
- in other words, $f(a) ::= R^{-1}(a) \cup \{a\}$

Example: from divides

- $f(10) = 1|10, 2|10, 5|10, \cup\{10\} = \{1, 2, 5, 10\}$
- $f(3) = 1|3, \cup\{3\} = \{1, 3\}$

Symmetric Relations and Equivalence Relations

- If there is a walk from u to v and a walk from v to u, then we say that u and v are strongly connected. uG^*v and vG^*u
- Relation R is symmetric if $aRb \implies bRa$. The strongly connected relation is symmetric.
- An equivalence relation R is: transitive, symmetric and reflexive.
- R is an equivalence relation **iff** R is the strongly connected relation of some DiGraph.

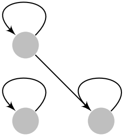
Equivalence Relations Examples

Examples:

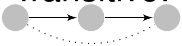
- Equality: $=$
- $\equiv$(mod n)
- Same Size, Same Color, etc.

Relation Properties: Graphical Review

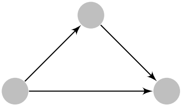
Reflexive:



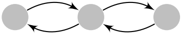
Transitive:



Assymetric:



Symetric:



Representing Equivalence

- For a total function $f : A \to B$
- We can define an equivalence relation: $\equiv_f$ on A:

$$a \equiv_f a' \iff f(a) = f(a') \tag{6}$$

- **Theorem:** Relation R on set A is an equiv. relation **iff**: R is $\equiv_f$ for some $f : A \to B$

- **Example:** $\equiv$ (mod n) is $\equiv_f$ where $f(k) ::= \text{rem}(k, n)$

Equivalence and Partition

- We define a partition Π of a set A, where \$*Pi* is a collection of subsets of A that cover all elements but do not overlap.

  Example: For A = $\{a, b, c, d, e\}$ one partition could be: $\{a, b\}, \{c, e\}, \{d\}$

- We define a relatin $\equiv_\Pi$ on A: $a \equiv_\Pi a'$ if both a and a' are in the same subset of Π

- A relation R on set A is an equivalence relation **iff** R is $\equiv_\Pi$ for some partition Π of A.

Lecture Summary

- Graphs can be seen as relations on vertices;
- Vertex connectivity by n-1 power of adjacency matrix;
- DAG have no cycles, and define **order** relationships;
- Properties of Relations: Symmetry, Transitivity, Reflexivity;
- Equivalence relations;