Warshall's Algorithm

A more efficient way of computing the transitive closure of a relation with digraph on vertices $\{v_1, v_2, \ldots, v_n\}$:

Theorem (p. 606). Let $W_k = (w_{ij}^{[k]})$ be the 0,1 matrix $w_{ij}^{[k]} = 1$ iff there is a path from v_i to v_j with any interior vertices in the set $\{v_1, v_2, \ldots v_k\}$. Then
$w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$
$W_0 = W_R$, $W_n = W_{R^*}$.

Proof: We'll use induction.

Base case: k=0. $W_0 = W_R$ because there can be no interior vertices, so just a single edge.

Induction step: If true for k-1, show
$w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$
because there is a path from v_i to v_j using interior vertices from $\{v_1, v_2, \ldots, v_k\}$ iff

- There is a path without v_k as an interior vertex (so $w_{ij}^{[k-1]} = 1$) or
- There is path with v_k as an interior vertex, in which case both $w_{ik}^{[k-1]}$ and $w_{kj}^{[k-1]}$ are 1. (there must be a k-1path from v_i to v_k and from v_k to v_j)

Using Warshall's Algorithm

As shown in the book, the formula giving Warshall's Algorithm easily translates to computer code.

If you do it by hand, just note that in $w_{ij}^{[k]} = w_{ij}^{[k-1]} \vee (w_{ik}^{[k-1]} \wedge w_{kj}^{[k-1]})$ you go from W_{k-1} to W_k by looking at the matrix for W_{k-1}. If you can go from v_i to v_k in W_{k-1} then in W_k you can add an entry ij if v_k goes to v_j in W_{k-1}. (this is easier than it sounds)

Transitive Closure via Warshall's Algorithm

$$W_0 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad W_1 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$W_2 = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \qquad W_3 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$M_{R^*} = W_3 = W_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Equivalence Relations (Section 9.5)

Equivalence relations are used to relate objects that are similar in some way. (section 9.5)

Definition: A relation on a set A is called an equivalence relation if it is reflexive, symmetric, and transitive.

Two elements that are related by an equivalence relation R are called **equivalent** under that relation.

Equivalence Relations

Since an equivalence relation R is **symmetric**, a is equivalent to b whenever b is equivalent to a.

Since R is **reflexive**, every element is equivalent to itself.

Since R is **transitive**, if a and b are equivalent and b and c are equivalent, then a and c are equivalent.

Obviously, these three properties are necessary for a reasonable definition of equivalence.

Equivalence Relations

Example: Suppose that R is the relation on the set of strings that consist of English letters such that aRb iff l(a) = l(b), where l(x) is the length of the string x. Is R an equivalence relation?

Solution:

- R is reflexive, because l(a) = l(a) and therefore aRa for any string a.
- R is symmetric, because if l(a) = l(b) then l(b) = l(a), so if aRb then bRa.
- R is transitive, because if l(a) = l(b) and l(b) = l(c), then l(a) = l(c), so aRb and bRc implies aRc.

R is an equivalence relation.

Equivalence Classes

Definition: Let R be an equivalence relation on a set A. The set of all elements that are related to an element a of A is called the **equivalence class of a**.

The equivalence class of a with respect to R is denoted by $[a]_R$.

When only one relation is under consideration, we will delete the subscript R and write **[a]** for this equivalence class.

If $b \in [a]_R$, b is called a **representative** of this equivalence class.

Equivalence Classes

Example: In the previous example (strings of identical length), what is the equivalence class of the word mouse, denoted by [mouse] ?

Solution: [mouse] is the set of all English words containing five letters.

For example, 'horse' would be a representative of this equivalence class.

Equivalence Classes

Theorem: Let R be an equivalence relation on a set A. The following statements are equivalent:

(i) aRb (meaning $(a,b) \in R$)

(ii) [a] = [b]

(iii) $[a] \cap [b] \neq \emptyset$

Proof: we'll prove that (i) → (ii), (ii) → (iii), and (iii) → (i), when R is an equiv. relation

(i) → (ii)

Suppose aRb. If $x \in [a]$ then xRa, so xRb by transitivity, and $x \in [b]$. By symmetry, $x \in [b] \rightarrow x \in [a]$

(ii) → (iii) if [a]=[b] then $a \in [a] \cap [b]$.

(iii) → (i)

Suppose $x \in [a] \cap [b]$. Then xRa and xRb, so by symmetry aRx and xRb, so aRb by transitivity.

Equivalence Classes

Definition: A **partition** of a set S is a collection of disjoint nonempty subsets of S that have S as their union. In other words, the collection of subsets A_i, $i \in I$, forms a partition of S if and only if

(i) $A_i \neq \emptyset$ for $i \in I$

(ii) $A_i \cap A_j = \emptyset$, if $i \neq j$

(iii) $\cup_{i \in I} A_i = S$

Equivalence Classes

Examples: Let S be the set {u, m, b, r, o, c, k, s}.
Do the following collections of sets partition S ?

{{m, o, c, k}, {r, u, b, s}}	yes.
{{c, o, m, b}, {u, s}, {r}}	no (k is missing).
{{b, r, o, c, k}, {m, u, s, t}}	no (t is not in S).
{{u, m, b, r, o, c, k, s}}	yes.
{{b, o, o, k}, {r, u, m}, {c, s}}	yes ({b,o,o,k} = {b,o,k}).
{{u, m, b}, {r, o, c, k, s}, ∅}	no (∅ not allowed).

Equivalence Classes

Theorem: Let R be an equivalence relation on a set S. Then the **equivalence classes** of R form a **partition** of S. Conversely, given a partition $\{A_i \mid i \in I\}$ of the set S, there is an equivalence relation R that has the sets A_i, $i \in I$, as its equivalence classes.

This is typically used to identify elements regarded as equivalent. The equivalence classes become the "points" of a new, smaller space.

A typical example of this is forming Z_n from Z under the equivalence relation aRb is $a \equiv b \pmod{n}$.

Equivalence Classes

Example: Let us assume that Frank, Suzanne and George live in Boston, Stephanie and Max live in Lübeck, and Jennifer lives in Sydney.

Let R be the **equivalence relation** {(a, b) | a and b live in the same city} on the set P = {Frank, Suzanne, George, Stephanie, Max, Jennifer}.

Then R = {(Frank, Frank), (Frank, Suzanne), (Frank, George), (Suzanne, Frank), (Suzanne, Suzanne), (Suzanne, George), (George, Frank), (George, Suzanne), (George, George),

(Stephanie, Stephanie), (Stephanie, Max), (Max, Stephanie), (Max, Max),

(Jennifer, Jennifer)}.

Equivalence Classes

And the **equivalence classes** of R are:

{{Frank, Suzanne, George}, {Stephanie, Max}, {Jennifer}}.

This is a **partition** of P.

The equivalence classes of any equivalence relation R defined on a set S constitute a partition of S, because every element in S is assigned to **exactly one** of the equivalence classes.

Equivalence Classes

Another example: Let R be the relation {(a, b) | $a \equiv b \pmod{3}$} on the set of integers.

Is R an equivalence relation?

Yes, R is reflexive, symmetric, and transitive.

What are the equivalence classes of R ?

{{…, -6, -3, 0, 3, 6, …},
{…, -5, -2, 1, 4, 7, …},
{…, -4, -1, 2, 5, 8, …}}

The integers mod n

The relation $a \equiv b \pmod{n}$ produces n distinct equivalence classes, [0], [1], …, [n-1].

We shorten their names to 0,1,…,n-1 and call the set of equivalence classes Z_n the integers mod n.

We saw earlier that we can define + and * nicely on Z_n, producing a nice finite arithmetic system.

Arithmetic in Z_n

- If a ≡ b (mod n) and c ≡ d (mod n) then a + c ≡ b + d (mod n).
- This shows we can define [u] + [v] = [u+v] and we'll get the same answer no matter what representatives u and v we choose for the equivalence classes.

Arithmetic in Z_n

- If a ≡ b (mod n) and c ≡ d (mod n) then a*c ≡ b*d (mod n).
- This shows we can define [u] * [v] = [u*v] and we'll get the same answer no matter what representatives u and v we choose for the equivalence classes.

Arithmetic in Z_n

- If a ≡ b (mod n) and c ≡ d (mod n) then a*c ≡ b*d (mod n).
- This shows we can define [u] * [v] = [u*v] and we'll get the same answer no matter what representatives u and v we choose for the equivalence classes.

Arithmetic in Z_n

- 13 ≡ 76 (mod 7) and 2 ≡ 79 (mod 7) and [13+2] = [15] = [1] = [76+79] = [155] in Z_7, so we know [13] = [76] and [2] = [79] and [13 + 2] = [76 + 79].
- Thus we can define [13] + [2] = [15]
- In defining the sum we picked 13 and 2 as representatives of the equivalence classes, but any representatives we picked would give the same answer.

Multiplication in Z_7

*	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

Partial Orderings

Sometimes, relations do not specify the equality of elements in a set, but define an **order** on them.

Definition: A relation R on a set S is called a **partial ordering** or **partial order** if it is reflexive, antisymmetric, and transitive.

A set S together with a partial ordering R is called a **partially ordered set**, or **poset**, and is denoted by (S, R).

Partial Orderings

Example: Consider the "greater than or equal" relation ≥ (defined by {(a, b) | a ≥ b}).

Is ≥ a **partial ordering** on the set of integers?

- ≥ is **reflexive**, because a ≥ a for every integer a.
- ≥ is **antisymmetric**, because if a ≠ b, then a ≥ b ∧ b ≥ a is false.
- ≥ is **transitive**, because if a ≥ b and b ≥ c, then a ≥ c.

Consequently, (Z, ≥) is a partially ordered set.

Partial Orderings

Another example: Is the "inclusion relation" ⊆ a **partial ordering** on the power set of a set S?

- ⊆ is **reflexive**, because A ⊆ A for every set A.
- ⊆ is **antisymmetric**, because if A ≠ B, then A ⊆ B ∧ B ⊆ A is false.
- ⊆ is **transitive**, because if A ⊆ B and B ⊆ C, then A ⊆ C.

Consequently, (P(S), ⊆) is a partially ordered set.

Partial Orderings

In a poset the notation a ≤ b denotes that (a, b)∈ R.

Note that the symbol ≤ is used to denote the relation in **any** poset, not just the usual "less than or equal" relation in numbers.

The notation a < b denotes that a ≤ b, but a ≠ b.

If a < b we say "a is less than b" or "b is greater than a".

Partial Orderings

For two elements a and b of a poset (S, ≤) it is possible that neither a ≤ b nor b ≤ a.

Example: In (P(Z), ⊆), {1, 2} is not related to {1, 3}, and vice versa, since neither is contained within the other.

Definition: The elements a and b of a poset (S, ≤) are called **comparable** if either a ≤ b or b ≤ a.
When a and b are elements of S such that neither a ≤ b nor b ≤ a, then a and b are called **incomparable**.

Partial Orderings

For some applications, we require **all** elements of a set to be comparable.

For example, if we want to write a dictionary, we need to define an order on **all** English words (alphabetic order).

Definition: If (S, ≤) is a poset and **every two elements** of S are comparable, S is called a **totally ordered** or **linearly ordered set**, and ≤ is called a **total order** or **linear order**. A totally ordered set is also called a **chain**.

Partial Orderings

Example I: Is (Z, ≤) a totally ordered poset?

Yes, because a ≤ b or b ≤ a for all integers a and b.

Example II: Is (Z^+, |) a totally ordered poset?

No, because it contains incomparable elements such as 5 and 7.

Lexicographic Order

How can we define a **lexicographic ordering** on the set of English words?

This is a **special case** of an ordering of strings on a set constructed from a partial ordering on the set.

We already have an **ordering of letters** (such as $a \leq b$, $b \leq c$, …), and from that we want to derive an **ordering of strings**.

Let us take a look at the **general case**, that is, how the construction works in any poset.

Lexicographic Order

First step: Construct a partial ordering on the Cartesian product of two posets, $(A_1, \leq_1)$ and $(A_2, \leq_2)$:
$(a_1, a_2) \leq (b_1, b_2)$ if $(a_1 <_1 b_1) \vee [(a_1 = b_1) \wedge (a_2 \leq_2 b_2)]$
Note: this gives us also:

$(a_1, a_2) < (b_1, b_2)$ if $(a_1 <_1 b_1) \vee [(a_1 = b_1) \wedge (a_2 <_2 b_2)]$

Examples: In the poset $(Z \times Z, \leq)$, …

… is (5, 5) < (6, 4) ? yes.

… is (6, 5) < (6, 4) ? no.

… is (3, 3) < (3, 3) ? no.

Lexicographic Order

Second step: Extend the previous definition to the Cartesian product of **n** posets $(A_1, \leq_1)$, $(A_2, \leq_2)$, …, $(A_n, \leq_n)$:

$(a_1, a_2, \ldots, a_n) < (b_1, b_2, \ldots, b_n)$
if $(a_1 <_1 b_1) \vee$
$\exists i>0\ (a_1 = b_1, a_2 = b_2, \ldots, a_i = b_i, a_{i+1} <_{i+1} b_{i+1})$

Examples: Is (1, 1, 1, 2, 1) < (1, 1, 1, 1, 2)?

No.

Is (1, 1, 1, 1, 1) < (1, 1, 1, 1, 2)?

Yes.

Lexicographic Order

Final step: Define lexicographic ordering of strings:

Consider the strings $a_1a_2\ldots a_m$ and $b_1b_2\ldots b_n$ on a partially ordered set S. Let t be the minimum of m and n. The definition of **lexicographic ordering** is that the string $a_1a_2\ldots a_m$ is less than $b_1b_2\ldots b_n$ if and only if

$(a_1, a_2, \ldots, a_t) < (b_1, b_2, \ldots, b_t)$, for $t = \min(m, n)$, or
$[(a_1, a_2, \ldots, a_m) = (b_1, b_2, \ldots, b_m) \wedge m < n]$

Lexicographic Order

Examples: If we apply this concept to lowercase English letters, …

… is discreet < discrete ?

Yes, because in the 7th position, e < t.

… is discreetness < discreet ?

No, because discreet is a prefix of discreetness.

… is discrete < discretion ?

Yes, because in the 8th position, e < i.

Hasse Diagrams

The digraph of a partial order can be simplified to form a Hasse Diagram.

- We omit any edge (a,a)
- We omit any edge that can be deduced by transitivity.
- We draw the edge (a,b), $a \leq b$, with a below b in the graph.

See the examples on pages 622 ff
(6th ed. 572 ff.)

Maximal & Minimal elements

An element a is minimal in a poset (S,≤) if there is no b with b<a.

An element a is maximal in a poset (S,≤) if there is no b with b>a.

Maximal (and minimal) elements are easy to spot in a Hasse diagram.

They are elements with nothing above (or below) them.

29 Oct 2015 CS 320 37

Maximal & Minimal elements

a is the greatest element of a poset (S,≤) if b≤a for all b ∈ S.

c is the least element of a poset (S,≤) if c≤b for all b ∈ S.

If a greatest or least element exists it must be unique.

(Make sure you can prove this fact).

29 Oct 2015 CS 320 38

Upper Bound of a set A

If a ≤ u for all a∈A then u is an upper bound for A.

If u is an upper bound for A and u≤x for every upper bound x for A then u is a least upper bound for A.

Not every set has an upper bound, for a general poset.

29 Oct 2015 CS 320 39

Lower Bound for a set A

If v ≤ a for all a∈A then v is a lower bound for A.

If v is a lower bound for A and x≤v for every lower bound x for A then v is a greatest lower bound for A.

Not every set has a lower bound, for a general poset.

See the examples on pages 522, 523

29 Oct 2015 CS 320 40

Lattices

A lattice is a poset in which every pair of elements has a least upper bound (lub) and a greatest lower bound (glb).

Lattices occur in lots of places and have a lot of known structure.

An example of a lattice is the poset of all subsets of a set U under ⊆.

29 Oct 2015 CS 320 41

Topological Sort

Sometimes it's convenient to derive a linear order or total order from a given partial order on a set.

This process is called topological sorting.

You can think of it as projecting a Hasse diagram horizontally onto a straight line so that no two vertices hit the same point on the line.

29 Oct 2015 CS 320 42

Topological Sort

We can construct an algorithm to do this by noting that every non empty subset in a poset has a minimal element.

We can construct a linear order on a poset (S,$\subseteq$) by successively choosing a minimal element from the elements left.

These elements form an increasing sequence in the linear order $\leq$.

The linear order is compatible in that $a \subseteq b$ guarantees that $a \leq b$ in the linear order.

The reverse is guaranteed only if $\subseteq$ is linear.