Massachusetts Institute of Technology | Handout 21
6.042J/18.062J: Mathematics for Computer Science | September 17, 1995
Professors David Karger and Tom Leighton

Notes 8

1 Partial Orders

Last few lectures we talked about relations and functions. Among other things, we saw that a function is a special type of relation. Today, we are going to talk about another special type of relation that is known as a partial order.

Definition 1.1 *A relation* $R \subseteq A \times A$ *is a* **partial order** *if it is reflexive, transitive, and antisymmetric.*

Quick review:

- reflexive: aRa
- transitive: $aRb \wedge bRc \Rightarrow aRc$
- symmetric: $aRb \Rightarrow bRa$
- antisymmetric: $aRb \wedge bRa \Rightarrow a = b$. i.e., $\forall a \neq b, aRb \Rightarrow \neg bRa$, NEVER symmetric!

Remember that a relation which is reflexive, transitive, and symmetric is an equivalence relation,

Examples of partial orders:

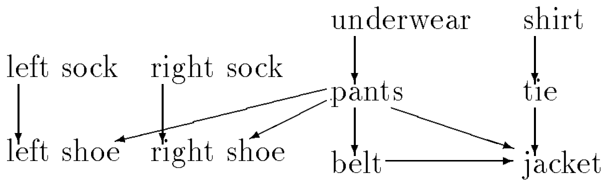
- $A = \mathbb{N}, R = \leq$, easy to check reflexive, transitive, antisymmetric
- $A = \mathbb{N}, R = \geq$, same
- $A = \mathbb{N}, R = <$, NOT because not reflexive
- $A = \mathbb{N}, R = \mid$ (divides), easy to check reflexive, transitive, antisymmetric
- $A = P(\mathbb{N}), R = \subseteq$, check reflexive: $S \subseteq S$, transitive: $S \subseteq S' \wedge S' \subseteq S'' \Rightarrow S \subseteq S''$, antisymmetric: $S \subseteq S' \wedge S' \subseteq S \Rightarrow S = S'$.

- $A =$ "set of all computers", $R =$ "is connected to", NOT because it is not true that $aRb \wedge bRa \Rightarrow a = b$. In fact is symmetric, so it is equivalence relation.
- $A =$ "set of all propositions", $R = \Rightarrow$, NOT because not antisymmetric. Not symmetric either, so not equivalence relation.
- $A =$ set of tasks, $R =$ "must be done before". Does $aRb \wedge bRa \Rightarrow a = b$? Hopefully, because otherwise a, b can never be done! So we will only deal with tasks with this property. Note not reflexive. This is flaw in book. Change R to "= or must be done before", and we can talk about a partial order on a set of tasks.

2 Posets

Definition 2.1 *A set A together with a partial order $\preceq$ is called a* **poset** $(A, \preceq)$

Posets can be nicely drawn as a graph where there is an arc or arrow from a to b if aRb. (Similar to what we've already done with relations.) Let's draw a poset that describes the order in which one would put on a suit. The set is of clothes, and the partial order specifies what should be put on before what.



Some arcs aren't drawn because they are obvious from transitivity. An example is shirt $\rightarrow$ jacket. Loops (reflexive arcs) also aren't shown.

Definition 2.2 *The* **Hasse diagram** *is the graph of a poset without loops or arcs implied by transitivity.*

So remove the pants $\rightarrow$ jacket arc above to get the Hasse diagram.

Definition 2.3 *The* **transitive closure** *is the graph of a poset with all loops and arcs implied by transitivity.*

Usually the transitive closure is too messy, so we just draw the Hasse diagram.

Notice that the graph above has directed paths, that is, sequences of nodes $a_1 \cdots a_n$ where $\forall i < n a_i R a_{i+1}$. However, it does not have any directed cycles, a directed path where $a_1 = a_n$. Good thing, because otherwise we could get stuck trying to get dressed. This is in fact true of all posets.

Theorem 2.4 *A poset has no directed cycles (other than 1-edge loops)*

Proof. By contradiction.
Assume there is a directed cycle of length n, so $x_1 \preceq x_2 \preceq \cdots \preceq x_n \preceq x_1, \forall i \neq j x_i \neq x_j$
$x_n \preceq x_1$
By transitivity, $x_1 \preceq x_n$
These imply by antisymmetry that $x_1 = x_n$, which is a contradiction. ∎

We have a special name for graphs with this property:

Definition 2.5 *A* **DAG** *(directed acyclic graph) is a directed graph without directed cycles. (We forget about the loops since we took them away. Technically speaking, DAGs have no loops)*

More on graphs will come later. Let's go back to posets.

A Partial order is called a partial order because it is not necessary that an ordering exist between every pair of items in the set. For example, Lshoe and Rshoe have no prescribed ordering between them. When there is no prescribed order between two items we say that they are incomparable.

Definition 2.6 *a and b are* **incomparable** *if neither $a \preceq b$ nor $b \preceq a$.*

Definition 2.7 *a and b are* **comparable** *if $a \preceq b$ or $b \preceq a$.*

For some partial orders there is a specified order between every pair of items. Such partial orders are called total orders.

Definition 2.8 *A poset $(S, \preceq)$ is* **totally ordered** *if $\forall a, b \in S a \preceq b \vee b \preceq a$.*

Examples:

- $S = \mathbb{N}, \preceq = \leq$
- $S = \mathbb{N}, \preceq = |$, NOT because neither $3|5$ nor $5|3$
- $S = P(\mathbb{N}), \preceq = \subseteq$, NOT because neither $\{3\} \subseteq \{5\}$ nor $\{5\} \subseteq \{3\}$

The Hasse diagram of a total order is a line. You prove this for HW.

3 Topological Sorting

One nice property of a total order is that the items are totally ordered or ranked. For example, if the order denotes strength, you know who is strongest, who is next strongest, etc. Items in a poset can also be ranked in a manner that is consistent with the partial order, although the ranking is not necessarily unique. For example, there are several ways of putting on a suit that are consistent with the picture above.

The task of finding an ordering that is consistent with a partial order is known as topological sorting.

Definition 3.1 *A* **topological sort** *of a finite poset* $(A, \preceq)$ *is an ordering of all the items of* A*,* $a_1, a_2, \cdots, a_n$ *so that* $\forall i < j$ *either* $a_i \preceq a_j$ *or* a_i *and* a_j *are incomparable.*

For example, underwear, shirt, Lsock, Rsock, pants, tie, Lshoe, Rshoe, belt, jacket is a topological sort of how to put on a suit.

One of the nice facts about posets is that such an ordering always exists and is even easy to find.

Theorem 3.2 *Every finite poset has a topological sort*

We'll prove this by induction. The basic idea is to pick a "smallest" element to start and then proceed recursively.

Definition 3.3 *A* **minimal item** a *in a poset* $(A, \preceq)$ *is one for which* $\forall b \in A a \preceq b$ *or* a *and* b *are incomparable.*

Notice that there can be more than one minimal item. There are 4 in the clothes example: Lsock, Rsock, underwear, shirt.

Lemma 3.4 *Every finite poset has a minimal item.*

Proof. By contradiction.
Assume no minimal items, which implies $\forall a \in A, \exists b \neq a b \preceq a$.
Let p be the longest simple (no repeats) path in the poset. Finite since A finite (runs out of items to include).
If u is the first item in this chain, then by assumption there exists a $u', u' \preceq u$. u' can't be in the path, because that would make a cycle, so we are adding a new item, making a longer path. This contradicts the assumption. ∎

Now we can prove the existence of the topological sort.

Proof. By induction.
$P(n)$ = any poset with n items has a topological sort.

Base case ($P(1)$): for a poset with one item the one item makes a topological sort.

Inductive step: Assume ($P(n)$). Consider a poset of $n+1$ items. By lemma it has a least item u. Consider the relation on $A - \{u\}$. It is easy to check that it is still reflexive, transitive, and anti-symmetric. So by the inductive hypothesis $A - \{u\}$ has a topological sort. We put u first and now have a topological sort on the whole thing.

Thus every poset has a topological sort, by induction. ■

4 Parallel Task Scheduling

When items of a poset are tasks that need to be done and the partial order is precedence constraints, topological sorting provides us with a legal way to execute tasks sequentially, i.e., without violating any precedence constraints. But what if we have the ability to execute more than one task at the same time? For example, say tasks are programs, partial order indicates data dependence, and we have a parallel machine with lots of processors instead of a sequential machine with only one? How should we schedule the tasks? (For simplicity, say all the tasks take the same amount of time and all the processors are identical.)

Assume that all tasks take 1 unit of time and we have an unlimited number of processors. Given a poset of tasks, how long does it take to do them all?

On clothes example, we could do all the minimal items first (Lsock, Rsock, underwear, shirt), remove them and repeat. (So do pants and tie next, and then Lshoe, Rshoe, and belt, and finally jacket.) Can't do any better, because the sequence shirt, pants, belt, jacket must be don in that order. A sequence like this is called a chain.

Definition 4.1 *A* **chain** *is a sequence $a_1 \preceq a_2 \preceq \cdots \preceq a_t$. Each item is comparable to the next in chain and is smaller in the partial order.*

Parallel time $\geq$ length of the longest chain. For this reason longest chain is also known as critical path.

This is true, because if we used less time, then by pigeonhole two tasks in the chain would have to be done at the same time, but by definition of chain this violates precedence constraints.

So we need at least t steps, where t is the length of the longest chain. Happily, it is always possible to use only t.

Theorem 4.2 *Given any finite poset $(A, \preceq)$ for which the longest chain has length t, it is* *possible to partition A into t subsets, $A_1, A_2, \cdots, A_t$ such that $\forall i \in [1,t] \forall a \in A_i \forall b \preceq a, b \in A_1 \cup A_2 \cup \cdots \cup A_i$.*

Corollary 4.3 *$\forall i$ items in A_i can be scheduled for step i.*

(Because all preceding tasks are scheduled in preceding steps and thus already completed)

Corollary 4.4 *parallel time = length of longest chain*

(Because the last task is scheduled at step t, where t is length of longest chain.)

Let's prove the theorem:

Proof. $\forall a$ put a in A_i where i is the length of the longest path ending at a.

To show this works, we need to show that $\forall i \forall a \in A_i$ and $\forall b \preceq a, b \in A_1 \cup A_2 \cup \cdots \cup A_{i-1}$.

Proof by contradiction.

If $b \in A_1 \cup A_2 \cup \cdots \cup A_{i-1}$ then there is a path of length $> i-1$ ending in b, which means there is a path of length $> i$ ending in a, which means $a \notin A_i$. This is a contradiction. ∎

So with an unlimited number of processors, the time to complete all the tasks is the length of the longest chain. For HW you'll see what happens when there are limited processors.

It turns out that this theorem is good for more than parallel scheduling. It is usually stated as follows.

Corollary 4.5 *If t is the length of the longest chain i a poset $(A, \preceq)$ then A can be partitioned into t antichains.*

Definition 4.6 *An* **antichain** *is a set of incomparable items.*

This is a corollary because we showed before how to partition the tasks into sets that could be done at the same time, which means that these were incomparable. (If any two comparable then one would have to precede other and so couldn't be done at same time.)

Review:

- **chain**: all items comparable (total order)
- **antichain**: all items incomparable

5 Dillworth's theorem

We can use this to prove a famous result about posets:

Theorem 5.1 *(Dillworth) $\forall t$, every poset with n items must have a chain of size $> t$ or an antichain of size $\geq \frac{n}{t}$.*

Proof. By contradiction:
Assume the theorem is false. No chain of length $> t$ means longest chain has length $\leq t$. So the n items can be partitioned into $\leq t$ antichains, so by pigeonhole there is an antichain with $\geq \frac{n}{t}$ items. This contradicts the assumption. ∎

Corollary 5.2 *Every poset with n items has a chain or an antichain of size $\geq \sqrt{n}$.*

Set $t = \sqrt{n}$ and this is immediate.

Posets arise in all sorts of contexts, and when they do Dillworth's theorem can have interesting implications.

Example: In any sequence of n different numbers, there is either an increasing or decreasing subsequence of length $\geq \sqrt{n}$.

For example, 6 4 7 8 1 2 5 3 8 has the decreasing sequence 6 4 1 and the increasing sequence 1 2 3 8.

We can get this directly from Dillworth's theorem if we find an appropriate poset.

So define $a \preceq b$ if $a \leq b$ and a is "left of or equal to" b.

We can check that this is reflexive, transitive, and antisymmetric.

A chain corresponds to a sequence that is less than and left of, which is an increasing sequence.

But what does an antichain correspond to?

Lemma 5.3 *If a, b are incomparable and $a > b$ then a is left of b.*

Proof. By contradiction. Assume b is left of a. Then $b \leq a$ and b is left of a so $b \preceq a$, which contradicts that they are incomparable. ∎

Lemma 5.4 *If $a_1, a_2, \cdots, a_t$ are incomparable and $a_1 > a_2 > \cdots > a_t$ then $a_1, a_2, \cdots a_t$ form a decreasing subsequence.*

Proof. $\forall i a_i > a_{i+1} \Rightarrow a_i$ left of a_{i+1} by previous lemma. ∎

So given an antichain, arrange the items so that they are in decreasing order and the "left of" relation follows, giving a decreasing subsequence.

Thus by Dillworth's theorem there is either an increasing or decreasing subsequence of length $\geq \sqrt{n}$.

There are more applications on the HW.