“Topological sort”

From a given partial order, produce a compatible total order

The problem

Partial order: ordering constraints between elements of a set, e.g.
- “Remove the dishes before discussing politics”
- “Walk to Uetliberg before lunch”
- “Take your medicine before lunch”
- “Finish lunch before removing dishes”

Total order: sequence including all elements of set

Compatible: the sequence respects all ordering constraints
- Uetliberg, Medicine, Lunch, Dishes, Politics: OK
- Medicine, Uetliberg, Lunch, Dishes, Politics: OK
- Politics, Medicine, Lunch, Dishes, Uetliberg: not OK

Why we are doing this!

- Very common problem in many different areas
- Interesting, efficient, non-trivial algorithm
- Illustration of many algorithmic techniques
- Illustration of data structures, complexity (big-Oh notation), and other topics of last lecture
- Illustration of software engineering techniques: from algorithm to component with useful API
- Opportunity to learn or discover important mathematical concepts: binary relations (order relations in particular) and their properties
- It’s just beautiful!

Today: problem and math basis
Next time: detailed algorithm and component
Topological sort: example uses

From a dictionary, produce a list of definitions such that no word occurs prior to its definition.

Produce a complete schedule for carrying out a set of tasks, subject to ordering constraints.
(This is done for scheduling maintenance tasks for industrial tasks, often with thousands of constraints.)

Produce a version of a class with the features reordered so that no feature call appears before the feature's declaration.

Rectangles with overlap constraints

Constraints: [B, A], [D, A], [A, C], [B, D], [D, C]

Possible solution:

B → D → E → A → C

Rectangles with overlap constraints

Constraints: [B, A], [D, A], [A, C], [B, D], [D, C]

Possible solution:

B → D → E → A → C

The problem

From a given partial order, produce a compatible total order.

Partial order: ordering constraints between elements of a set, e.g.
- "Remove the dishes before discussing politics"
- "Walk to Üetliberg before lunch"
- "Take your medicine before lunch"
- "Finish lunch before removing dishes"

Total order: sequence including all elements of set

Compatible: the sequence respects all ordering constraints
- Üetliberg, Medicine, Lunch, Dishes, Politics: OK
- Medicine, Üetliberg, Lunch, Dishes, Politics: OK
- Politics, Medicine, Lunch, Dishes, Üetliberg: not OK

Pictured as a graph

Üetliberg → Medicine → Lunch → Dishes → Politics

"Remove the dishes before discussing politics"
"Walk to Üetliberg before lunch"
"Take your medicine before lunch"
"Finish lunch before removing dishes"
Sometimes there is no solution

- "Introducing recursion requires that students know about stacks"
- "You must discuss abstract data types before introducing stacks"
- "Abstract data types rely on recursion"

The constraints introduce a cycle

Overall structure (1)

Given:
- A type \( \mathcal{G} \)
- A set of elements of type \( \mathcal{G} \)
- A set of constraints between these elements

Required:
- An enumeration of the elements, in an order compatible with the constraints

class ORDERABLE(\( \mathcal{G} \)) feature
  elements: LIST(\( \mathcal{G} \))
  constraints: LIST(TUPLE(\( \mathcal{G}, \mathcal{G} \)))
  tosort: LIST(\( \mathcal{G} \)) is
  ensure compatible(Result, constraints)
end

Some mathematical background...

Binary relation on a set

Any property that either holds or doesn’t hold between two elements of a set

On a set PERSON of persons, example relations are:
- mother: \( a \) mother \( b \) holds if and only if \( a \) is the mother of \( b \)
- father
- child
- sister
- sibling (brother or sister)

Notation: \( a \, r \, b \) to express that \( r \) holds of \( a \) and \( b \).

Example: the before relation

The set of interest:
- Tasks = (Politics, Lunch, Medicine, Dishes, Uetliberg)

The constraining relation:
- Dishes before Politics
- Uetliberg before Lunch
- Medicine before Lunch
- Lunch before Dishes

"Remove the dishes before discussing politics"
"Walk to Uetliberg before lunch"
"Take your medicine before lunch"
"Finish lunch before removing dishes"

Some special relations on a set \( X \)

universal \( [X] \): holds between any two elements of \( X \)

id \( [X] \): holds between every element of \( X \) and itself

empty \( [X] \): holds between no elements of \( X \)
Relations: a more precise mathematical view

We consider a relation \( r \) on a set \( P \) as:

A set of pairs in \( P \times P \),
containing all the pairs \([x, y]\) such that \( x r y \).

Then \( x r y \) simply means that \([x, y]\) \( \in r \).

See examples on next slide.

Example: the \textit{before} relation

"Remove dishes before discussing politics"
"Walk to Uetliberg before lunch"
"Take your medicine before lunch"
"Finish lunch before removing dishes"

The set of interest:
\[ \text{elements} = \{ \text{Politics, Lunch, Medicine, Dishes, Uetliberg} \} \]

The constraining relation:
\[ \text{before} = \{ ([\text{Dishes, Politics}], [\text{Uetliberg, Lunch}], [\text{Medicine, Lunch}], [\text{Lunch, Dishes}]) \} \]

Using ordinary set operators

\[
\begin{align*}
\text{spouse} &= \text{wife} \cup \text{husband} \\
\text{sibling} &= \text{sister} \cup \text{brother} \cup \text{id} \ [\text{Person}] \\
\text{family\_head} &\subseteq \text{sibling} \\
\text{universal} \ [X] &= X \times X \quad \text{(cartesian product)} \\
\text{empty} \ [X] &= \emptyset
\end{align*}
\]

Possible properties of a relation

\[
\begin{align*}
\text{Total*}: & \quad (a \neq b) \implies ((a r b) \lor (b r a)) \\
\text{Reflexive}: & \quad a r a \\
\text{Irreflexive}: & \quad \neg (a r a) \\
\text{Symmetric}: & \quad a r b \implies b r a \\
\text{Antisymmetric}: & \quad (a r b) \land (b r a) \implies a = b \\
\text{Asymmetric}: & \quad \neg ((a r b) \land (b r a)) \\
\text{Transitive}: & \quad (a r b) \land (b r c) \implies a r c
\end{align*}
\]

*Definition of "total" is specific to this discussion (there is no standard definition). The other terms are standard.

Examples (on a set of persons)

\[
\begin{align*}
\text{sibling} &\quad \text{Reflexive, symmetric, transitive} \\
\text{family\_head} &\quad \text{Reflexive, antisymmetric} \\
\text{mother} &\quad \text{Asymmetric, irreflexive}
\end{align*}
\]

Total order relation (strict)

Relation is strict total order if:
\[
\begin{align*}
\text{Total} \\
\text{Irreflexive} \\
\text{Transitive}
\end{align*}
\]

Example: "less than" \(<\) on natural numbers

\[
\begin{array}{cccccc}
0 & 1 & 2 & 3 & 4 & 5 \\
\hline
0 < 1 & 0 < 2 & 0 < 3 & 0 < 4 & \ldots \\
1 < 2 & 1 < 3 & 1 < 4 & \ldots \\
2 < 3 & 2 < 4 & \ldots \\
\end{array}
\]
**Theorem**

A strict (total) order is asymmetric

**Total order relation (strict)**

Relation is strict total order if:
- Total
- Irreflexive
- Transitive

**Total order relation (non-strict)**

Relation is non-strict total order if:
- Total
- Reflexive
- Transitive
- Antisymmetric

**Partial order relation (strict)**

Relation is strict partial order if:
- Irreflexive
- Transitive

**Theorems**

A strict (total) order is asymmetric

A total order is a partial order

("partial" order really means possibly partial)
Example partial order

Here the following hold:

- No link between a and c, b and c:
  - e.g., neither a \(\perp\) c nor c \(\perp\) a
- \(a \perp b\) c \(\perp d\)
- \(a \perp d\)

Possible topological sorts

Possible topological sorts:

- a \(\perp\) b c \(\perp\) d
- a \(\perp\) d

Topological sort understood

Here the relation \(\uplus\) is:

\[[a, b], [a, d], [c, d]\]

One of the solutions is:

- a \(\perp\) b \(\perp\) c \(\perp\) d

Final statement of topological sort problem

From a given partial order, produce a compatible total order

where:

A partial order \(p\) is compatible with a total order \(t\) if and only if

\(p \subseteq t\)

From constraints to partial orders

Is a relation defined by a set of constraints, such as

\text{constraints} = 
\{ [\text{Dishes, Politics}], [\text{"Uetliberg, Lunch}], [\text{Medicine, Lunch}], [\text{Lunch, Dishes}] \}

always a partial order?

Powers and transitive closure of a relation

Transitive closure

\(r^n = r^1 \cup r^2 \cup \ldots\) always transitive
**Reflexive transitive closure**

\[ r^* = \text{id}([X]) \]  
where \(X\) is the underlying set

\[ r^* = r^0 \cup r^1 \cup r^2 \cup ... \]  
always transitive

\[ r^* = r^0 \cup r^1 \cup r^2 \cup ... \]  
always transitive and reflexive

---

**Acyclic relation**

A relation \(r\) on a set \(X\) is acyclic if and only if:

\[ r^* \cap \text{id}([X]) = \emptyset \]

---

**Acyclic relations and partial orders**

**Theorems:**

- Any (strict) order relation is acyclic.
- A relation is acyclic if and only if its transitive closure is a (strict) order.

(Also: if and only if its reflexive transitive closure is a nonstrict partial order)

---

**From constraints to partial orders**

The partial order of interest is before:

\[[\text{Dishes, Politics}], [\text{Uetliberg, Lunch}], \]
\[[\text{Medicine, Lunch}], [\text{Lunch, Dishes}]\]

---

**Back to software...**

---

**The basic algorithm idea**

The partial order of interest is before:

\[[\text{Dishes, Politics}], [\text{Uetliberg, Lunch}], \]
\[[\text{Medicine, Lunch}], [\text{Lunch, Dishes}]\]

---

Theorems:

- Any (strict) order relation is acyclic.
- A relation is acyclic if and only if its transitive closure is a (strict) order.

(Also: if and only if its reflexive transitive closure is a nonstrict partial order)
What we have seen

The topological sort problem and its applications
Mathematical background:
  > Relations as sets of pairs
  > Properties of relations
  > Order relations: partial/total, strict/nonstrict
  > Transitive, reflexive-transitive closures
  > The relation between acyclic and order relations
  > The basic idea of topological sort

Next: how to do it for
  > Efficient operation ($O(m+n)$ for $m$ constraints & $n$ items)
  > Good software engineering: effective API

Reading assignment for next Monday

Touch of Class, chapter on topological sort: 17

End of lecture 19