Lecture 19: Topological sort
Part 1: Problem and math basis
“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 rediscover 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
Reading assignment for next Monday

Touch of Class, chapter on topological sort: 17

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]
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
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

Intro. to Programming, lecture 19: Topological Sort I

Overall structure (1)

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

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

Some mathematical background...

```plaintext
class ORDERABLE(G) feature
    elements: LIST[G]
    constraints: LIST[TUPLE[G, G]]
    tosort: LIST[G] is
        ensure
            compatible(Result, constraints)
end
```
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.

Note: relation names will appear in green.

Example: the before relation

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

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

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 \( x \mathrel{r} y \).

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

See examples on next slide.

Example: the \textit{before} relation

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

The set of interest:
\textit{elements} = \{Politics, Lunch, Medicine, Dishes, Üetliberg\}

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

Using ordinary set operators

- \textit{spouse} = \textit{wife} \cup \textit{husband}
- \textit{siblings} = \textit{sister} \cup \textit{brother} \cup \textit{id} [\textit{Person}]
- \textit{sister} \subseteq \textit{siblings}
- \textit{father} \subseteq \textit{ancestor}
- \textit{universal} \([X] = X \times X\) (cartesian product)
- \textit{empty} \([X] = \emptyset\)
Possible properties of a relation

(On a set \( X \). All definitions must hold for every \( a, b, c \in X \))

- **Total**: \((a \neq b) \Rightarrow ((a \leq b) \lor (b \leq a))\)
- **Reflexive**: \(a \leq a\)
- **Irreflexive**: \(\neg (a \leq a)\)
- **Symmetric**: \(a \leq b \Rightarrow b \leq a\)
- **Antisymmetric**: \((a \leq b) \land (b \leq a) \Rightarrow a = b\)
- **Asymmetric**: \(\neg ((a \leq b) \land (b \leq a))\)
- **Transitive**: \((a \leq b) \land (b \leq c) \Rightarrow a \leq c\)

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

Examples (on a set of persons)

- **Sibling**: Reflexive, symmetric, transitive
- **Sister**: Symmetric, irreflexive
- **Family head**: Reflexive, antisymmetric
- **Mother**: Asymmetric, irreflexive

Total order relation (strict)

- **Total**: \((a \neq b) \Rightarrow ((a < b) \lor (b < a))\)
- **Irreflexive**: \(a \neq a\)
- **Symmetric**: \(a < b \Rightarrow b < a\)
- **Antisymmetric**: \((a < b) \land (b < a) \Rightarrow a = b\)
- **Asymmetric**: \(\neg ((a < b) \land (b < a))\)
- **Transitive**: \((a < b) \land (b < c) \Rightarrow a < c\)

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

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

A strict (total) order is asymmetric

Total order relation (strict)

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

Theorem:

A strict total order is asymmetric

Total order relation (non-strict)

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

Example: "less than or equal" \( \leq \) on natural numbers

Example: 0, 1, 2, 3, 4, 5

Example: 0 \( \leq \) 0, 0 \( \leq \) 1, 0 \( \leq \) 2, ...
Total order relation (strict)

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

Total: \((a \neq b) \Rightarrow ((a \prec b) \lor (b \prec a))\)

Irreflexive: not \((a \prec a)\)

Symmetric: \(a \prec b \Rightarrow b \prec a\)

Antisymmetric: \((a \prec b) \land (b \prec a) \Rightarrow a = b\)

Transitive: \((a \prec b) \land (b \prec c) \Rightarrow a \prec c\)

Partial order relation (strict)

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

Total: \((a \neq b) \Rightarrow ((a \leq b) \lor (b \leq a))\)

Irreflexive: not \((a \leq a)\)

Symmetric: \(a \leq b \Rightarrow b \leq a\)

Antisymmetric: \((a \leq b) \land (b \leq a) \Rightarrow a = b\)

Transitive: \((a \leq b) \land (b \leq c) \Rightarrow a \leq c\)

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 $c$, and $b$ and $c$:
  - $a \not\less b$
  - $c \not\less d$
  - $a \not\less d$

Possible topological sorts

Here the relation $\less$ is:

- $(a, b, [a, d], [c, d])$

One of the solutions is:

- $a, b, c, d$

Topological sort understood

Here the relation $\less$ is:

- $[(a, b), [a, d], [c, d]]$

One of the solutions is:

- $a, b, c, d$
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

\[
 r^{i+1} = r^i \cdot r \text{ where } \cdot \text{ is composition}
\]

Transitive closure

\[
r^* = r^1 \cup r^2 \cup \ldots \text{ always transitive}
\]

\[
r^1
\]

\[
r^2
\]

\[
r^3
\]
**Reflexive transitive closure**

\[ r^0 = \text{id} \left[ X \right] \text{ where } X \text{ is the underlying set} \]
\[ r^i = r^{i-1} \circ r \text{ where } \circ \text{ is composition} \]

**Transitive closure**

\[ r^* = r^0 \cup r^1 \cup r^2 \cup \ldots \text{ always transitive} \]

**Reflexive transitive closure:**

\[ r^* = r^0 \cup r^1 \cup r^2 \cup \ldots \text{ always transitive and reflexive} \]

---

**Acyclic relation**

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

\[ r^* \cap \text{id} \left[ X \right] = \emptyset \]

---

**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 + before = 

\{\\{\text{Dishes, Politics}\}, \{\text{Üetliberg, Lunch}\}, \\
\{\text{Medicine, Lunch}\}, \{\text{Lunch, Dishes}\}\}

Back to software...

The basic algorithm idea
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