Lecture 19: Topological sort
Part 1: Problem and math basis
Un dîner en famille.

— Surtout ! ne parlons pas de l’affaire Dreyfus!

... Ils en ont parlé ...
“Topological sort”

From a given partial order, produce a compatible total order
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
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

Introduction to Programming, lecture 19: Topological sort
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
"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
**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

---

```plaintext
class ORDERABLE[G] feature

  elements: LIST[G]

  constraints: LIST[TUPLE[G, G]]

  topsort: LIST[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

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

The set of interest:

\[ \text{Tasks} = \{ \text{Politics, Lunch, Medicine, Dishes, Üetliberg} \} \]

The constraining relation:

\[ \text{Dishes} \text{ before Politics} \]
\[ \text{Üetliberg} \text{ before Lunch} \]
\[ \text{Medicine} \text{ before Lunch} \]
\[ \text{Lunch} \text{ before Dishes} \]
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 $x \sim r y$.

Then $x \sim r y$ simply means that $[x, y] \in r$

See examples on next slide
Example: the *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:

```
elements = {Politics, Lunch, Medicine, Dishes, Üetliberg}
```

The constraining relation:

```
before = 
{{Dishes, Politics], [Üetliberg, Lunch],
  [Medicine, Lunch], [Lunch, Dishes]}
```
Using ordinary set operators

\[\text{spouse} = \text{wife } \cup \text{ husband}\]

\[\text{Sibling} = \text{sister } \cup \text{ brother } \cup \text{id} [\text{Person}]\]

\[\text{sister } \subseteq \text{sibling}\]

\[\text{father } \subseteq \text{ancestor}\]

universal \( [X] = X \times X \) \quad \text{(cartesian product)}

empty \( [X] = \emptyset \)
Possible properties of a relation

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

- **Total**: $(a \neq b) \implies ((a \, r \, b) \lor (b \, r \, a))$
- **Reflexive**: $a \, r \, a$
- **Irreflexive**: not $(a \, r \, a)$
- **Symmetric**: $a \, r \, b \implies b \, r \, a$
- **Antisymmetric**: $(a \, r \, b) \land (b \, r \, a) \implies a = b$
- **Asymmetric**: not $((a \, r \, b) \land (b \, r \, a))$
- **Transitive**: $(a \, r \, b) \land (b \, r \, c) \implies a \, r \, c$

*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
(a family_head b means a is the head of b’s family, with one head per family)

**Mother**
Asymmetric, irreflexive

<table>
<thead>
<tr>
<th>Condition</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total: ((a \neq b) \Rightarrow (a \text{ r} b) \lor (b \text{ r} a))</td>
<td></td>
</tr>
<tr>
<td>Reflexive: (a \text{ r} a)</td>
<td></td>
</tr>
<tr>
<td>Irreflexive: not ((a \text{ r} a))</td>
<td></td>
</tr>
<tr>
<td>Symmetric: (a \text{ r} b \Rightarrow b \text{ r} a)</td>
<td></td>
</tr>
<tr>
<td>Antisymmetric: ((a \text{ r} b) \land (b \text{ r} a) \Rightarrow a = b)</td>
<td></td>
</tr>
<tr>
<td>Asymmetric: not (((a \text{ r} b) \land (b \text{ r} a)))</td>
<td></td>
</tr>
<tr>
<td>Transitive: ((a \text{ r} b) \land (b \text{ r} c) \Rightarrow a \text{ r} c)</td>
<td></td>
</tr>
</tbody>
</table>
Total order relation (strict)

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

Total: \( (a \neq b) \Rightarrow ((a \, \mathcal{R} \, b) \lor (b \, \mathcal{R} \, a)) \)

Irreflexive: \( \text{not} \ (a \, \mathcal{R} \, a) \)

Symmetric: \( a \, \mathcal{R} \, b \Rightarrow b \, \mathcal{R} \, a \)

Asymmetric: \( \text{not} \ ((a \, \mathcal{R} \, b) \land (b \, \mathcal{R} \, a)) \)

Transitive: \( (a \, \mathcal{R} \, b) \land (b \, \mathcal{R} \, c) \Rightarrow a \, \mathcal{R} \, c \)

Example: “less than” < on natural numbers

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

A strict (total) order is asymmetric
Total order relation (strict)

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

Total: \((a \neq b) \Rightarrow ((a \, r \, b) \lor (b \, r \, a))\)
Irreflexive: not \((a \, r \, a)\)
Symmetric: \(a \, r \, b \Rightarrow b \, r \, a\)
Asymmetric: not \(((a \, r \, b) \land (b \, r \, a))\)
Transitive: \((a \, r \, b) \land (b \, r \, c) \Rightarrow a \, r \, c\)

**Theorem:**

A **strict total order** is **asymmetric**
Total order relation (non-strict)

Relation is **non-strict** total order if:
- **Total**: \( (a \neq b) \Rightarrow ((a r b) \lor (b r a)) \)
- **Irreflexive**: not \( (a r a) \)
- **Symmetric**: \( a r b \Rightarrow b r a \)
- **Antisymmetric**: \( (a r b) \land (b r a) \Rightarrow a = b \)
- **Transitive**: \( (a r b) \land (b r c) \Rightarrow a r c \)

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

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( 0 \leq 0 )</td>
<td>( 0 \leq 1 )</td>
<td>( 0 \leq 2 )</td>
<td>( 0 \leq 3 )</td>
<td>( 0 \leq 4 )</td>
<td>( 0 \leq 5 )</td>
</tr>
<tr>
<td></td>
<td>( 1 \leq 1 )</td>
<td>( 1 \leq 2 )</td>
<td>( 1 \leq 3 )</td>
<td>( 1 \leq 4 )</td>
<td>( 1 \leq 5 )</td>
<td>( 2 \leq 5 )</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>( 2 \leq 2 )</td>
</tr>
</tbody>
</table>
Total order relation (strict)

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

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

Irreflexive: \(\text{not } (a \ r \ a)\)

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

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

Transitive: \((a \ r \ b) \land (b \ r \ c) \Rightarrow a \ r \ c\)
Partial order relation (strict)

Relation is **strict partial order** if:

- **Total**
  - Irreflexive
  - Transitive

Total: 
\[(a \neq b) \Rightarrow ((a \not r b) \lor (b \not r a))\]

Irreflexive: not \((a \not r a)\)

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

Antisymmetric: 
\[(a r b) \land (b r a) \Rightarrow a = b\]

Transitive: 
\[(a r b) \land (b r c) \Rightarrow a r c\]

Example: relation between points in a plane:

\[p \ll q\] if both
- \(x_p < x_q\)
- \(y_p < y_q\)
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:

- $a \preceq b$
- $c \preceq d$
- $a \preceq d$

No link between $a$ and $c$, $b$ and $c$:

- e.g. neither $a \preceq c$ nor $c \preceq a$

$p \preceq q$ if both

- $x_p < x_q$
- $y_p < y_q$
Possible topological sorts

\[ a < b \quad \text{and} \quad c < d \]

\[ a \preceq d \]
Here the relation $\triangleleft$ is:

$\{[a, b], [a, d], [c, d]\}$

One of the solutions is:

$a, b, c, d$

We are looking for a total order relation $t$ such that $\triangleleft \subseteq t$
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

\[
\text{constraints} = \{(\text{Dishes, Politics}), (\text{Üetliberg, Lunch}), (\text{Medicine, Lunch}), (\text{Lunch, Dishes})\}
\]

always a partial order?
Powers and transitive closure of a relation

\[ r^{i+1} = r^i ; r \]
where \( ; \) is composition

Transitive closure
\[ r^+ = r^1 \cup r^2 \cup ... \text{ always transitive} \]
**Reflexive transitive closure**

\[
r^0 = id \ [X] \quad \text{where } X \text{ is the underlying set}
\]

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

Transitive closure

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

**Reflexive transitive closure:**

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

A relation $r$ on a set $X$ is acyclic if and only if:

$$r^+ \cap \text{id}[X] = \emptyset$$

before$^+$

id $[X]$
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)
The partial order of interest is $\text{before}^*$

$\text{before} = \{[\text{Dishes, Politics}], [\text{Üetliberg, Lunch}], [\text{Medicine, Lunch}], [\text{Lunch, Dishes}]\}$

Introduction to Programming, lecture 19: Topological sort
Back to software..
The basic algorithm idea

topsort
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