Topological Sorting

Ioan Despi
despi@turing.une.edu.au

University of New England

August 14, 2013
Outline

1 Motivation

2 Definition

3 Example

4 Topological Sorting Algorithm

5 Scheduling
Motivation

- A partial order relation can be used to do a topological sorting, which may find applications in compiler construction, planning, scheduling, etc.
Motivation

- A partial order relation can be used to do a **topological sorting**, which may find applications in compiler construction, planning, scheduling, etc.
- For any finite set $A$ and a partial order relation $\preceq$ on the set, the purpose of **topological sorting** is to sort all the elements of the set $A$ into an ordered list such that its sequential order preserves the partial order dictated by the relation $\preceq$. 

  - For example:
    - prerequisites must be completed prior to certain course
    - building a house
      - walls
      - windows
      - roof
      - plumbing
      - decorating
      - foundations
Motivation

- A partial order relation can be used to do a **topological sorting**, which may find applications in compiler construction, planning, scheduling, etc.
- For any finite set $A$ and a partial order relation $\preceq$ on the set, the purpose of **topological sorting** is to sort all the elements of the set $A$ into an ordered list such that its sequential order preserves the partial order dictated by the relation $\preceq$.
- Topological sorting is useful for defining the order things must be done. For example:
Motivation

- A partial order relation can be used to do a topological sorting, which may find applications in compiler construction, planning, scheduling, etc.
- For any finite set $A$ and a partial order relation $\leq$ on the set, the purpose of topological sorting is to sort all the elements of the set $A$ into an ordered list such that its sequential order preserves the partial order dictated by the relation $\leq$.
- Topological sorting is useful for defining the order things must be done. For example:
  - prerequisites must be completed prior to certain course
Motivation

A partial order relation can be used to do a **topological sorting**, which may find applications in compiler construction, planning, scheduling, etc.

For any finite set $A$ and a partial order relation $\preceq$ on the set, the purpose of **topological sorting** is to sort all the elements of the set $A$ into an ordered list such that its sequential order preserves the partial order dictated by the relation $\preceq$.

Topological sorting is useful for defining the order things must be done. For example:
- prerequisites must be completed prior to certain course
- building a house
Motivation

- A partial order relation can be used to do a **topological sorting**, which may find applications in compiler construction, planning, scheduling, etc.
- For any finite set \( A \) and a partial order relation \( \preceq \) on the set, the purpose of **topological sorting** is to sort all the elements of the set \( A \) into an ordered list such that its sequential order preserves the partial order dictated by the relation \( \preceq \).
- Topological sorting is useful for defining the order things must be done. For example:
  - prerequisites must be completed prior to certain course
  - building a house
Motivation

- A partial order relation can be used to do a **topological sorting**, which may find applications in compiler construction, planning, scheduling, etc.
- For any finite set $A$ and a partial order relation $\preceq$ on the set, the purpose of **topological sorting** is to sort all the elements of the set $A$ into an ordered list such that its sequential order preserves the partial order dictated by the relation $\preceq$.
- Topological sorting is useful for defining the order things must be done. For example:
  - prerequisites must be completed prior to certain course
  - building a house
In other words, if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.

More precisely, topologically sorting a set $A$ of $n$ elements with respect to (w.r.t.) a partial order relation $\preceq$ is to find an ordered enumeration, $a_1, a_2, \ldots, a_n$, such that for all $i$ and $j$, $a_i \preceq a_j$ implies $i \leq j$, where $a_k$ for $k = 1, 2, \ldots, n$ denotes the element placed to the $k$-th position of the resulting list.

Note that the topologically sorted list $a_1, \ldots, a_n$ can also be characterised by another partial order relation $\preceq'$ defined by $\preceq'$ def $= \{(a, b) \in A \times A | \exists i, j \text{ such that } a = a_i, b = a_j, i \leq j\}$. (recall a binary relation on $A$ is a subset of the Cartesian product $A \times A$)

We observe that the relation $\preceq'$ is in fact a total order relation and is defined so that $a_1 \preceq' a_2, a_2 \preceq' a_3, \ldots, a_{n-1} \preceq' a_n$.

Hence if we sort the set $A$ w.r.t. the ordering $\preceq'$ using an usual sorting algorithm such as the insertion sort, we will then obtain exactly the same resulting list $a_1, \ldots, a_n$. 
Definition

- In other words,
  - if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.
In other words,

- if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.

More precisely,
Definition

- In other words,
  - if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.

- More precisely,
  - topologically sorting a set $A$ of $n$ elements with respect to (w.r.t.) a partial order relation $\preceq$ is to find an ordered enumeration, $a_1, a_2, \ldots, a_n$, such that for all $i$ and $j$, $a_i \preceq a_j$ implies $i \leq j$, where $a_k$ for $k = 1, 2, \ldots, n$ denotes the element placed to the $k$-th position of the resulting list.
Definition

- In other words,
  - if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.
- More precisely,
  - topologically sorting a set $A$ of $n$ elements with respect to (w.r.t.) a partial order relation $\preceq$ is to find an ordered enumeration, $a_1, a_2, ..., a_n$, such that for all $i$ and $j$, $a_i \preceq a_j$ implies $i \leq j$, where $a_k$ for $k = 1, 2, ..., n$ denotes the element placed to the $k$-th position of the resulting list.
- Note that the topologically sorted list $a_1, ..., a_n$ can also be characterised by another partial order relation $\preceq'$ defined by

$$
\preceq' \overset{\text{def}}{=} \{(a, b) \in A \times A \mid \exists i, j \text{ such that } a = a_i, b = a_j, i \leq j\}.
$$

(recall a binary relation on $A$ is a subset of the Cartesian product $A \times A$)
Definition

- In other words,
  - if $a$ and $b$ are 2 arbitrary elements of $A$ and $a$ precedes $b$, i.e., $a \preceq b$, then $a$ must appear before $b$ in the resulting topologically sorted list.

- More precisely,
  - topologically sorting a set $A$ of $n$ elements with respect to (w.r.t.) a partial order relation $\preceq$ is to find an ordered enumeration, $a_1, a_2, \ldots, a_n$, such that for all $i$ and $j$, $a_i \preceq a_j$ implies $i \leq j$, where $a_k$ for $k = 1, 2, \ldots, n$ denotes the element placed to the $k$-th position of the resulting list.

- Note that the topologically sorted list $a_1, \ldots, a_n$ can also be characterised by another partial order relation $\preceq'$ defined by
  \[
  \preceq' \overset{\text{def}}{=} \{ (a, b) \in A \times A \mid \exists i, j \text{ such that } a = a_i, b = a_j, i \leq j \}.
  \]
  (recall a binary relation on $A$ is a subset of the Cartesian product $A \times A$)

- We observe that the relation $\preceq'$ is in fact a total order relation and is defined so that $a_1 \preceq' a_2, a_2 \preceq' a_3, \ldots, a_{n-1} \preceq' a_n$. 


Definition

• In other words,
  ➤ if \( a \) and \( b \) are 2 arbitrary elements of \( A \) and \( a \) precedes \( b \), i.e., \( a \preceq b \), then \( a \) must appear before \( b \) in the resulting topologically sorted list.

• More precisely,
  ➤ topologically sorting a set \( A \) of \( n \) elements with respect to (w.r.t.) a partial order relation \( \preceq \) is to find an ordered enumeration, \( a_1, a_2, \ldots, a_n \), such that for all \( i \) and \( j \), \( a_i \preceq a_j \) implies \( i \leq j \), where \( a_k \) for \( k = 1, 2, \ldots, n \) denotes the element placed to the \( k \)-th position of the resulting list.

• Note that the topologically sorted list \( a_1, \ldots, a_n \) can also be characterised by another partial order relation \( \preceq' \) defined by

\[
\mathrel{\preceq'} \overset{\text{def}}{=} \{(a, b) \in A \times A \mid \exists i, j \text{ such that } a = a_i, b = a_j, i \leq j\}.
\]

(recall a binary relation on \( A \) is a subset of the Cartesian product \( A \times A \))

• We observe that the relation \( \preceq' \) is in fact a total order relation and is defined so that \( a_1 \preceq' a_2, a_2 \preceq' a_3, \ldots, a_{n-1} \preceq' a_n \).

• Hence if we sort the set \( A \) w.r.t. the ordering \( \preceq' \) using an usual sorting algorithm such as the insertion sort, we will then obtain exactly the same resulting list \( a_1, \ldots, a_n \).
Compatible Partial Order Relations

Two partial order relations $R_1$ and $R_2$ on the same set $A$ are said to be compatible if, whenever $a$ and $b$ are comparable under both $R_1$ and $R_2$, we have $(a, b) \in R_1$ iff $(a, b) \in R_2$. 

Hence the topological sorting of a set $A$ w.r.t. a partial order relation $\preceq$ can be regarded as the construction of a total order relation $\preceq'$ such that $\preceq'$ is compatible with the existing partial order $\preceq$.

In general, a topological sorting doesn't produce a unique result, unless the existing partial order relation $\preceq$ is in fact also a total order relation.
Compatible Partial Order Relations

- Two partial order relations $R_1$ and $R_2$ on the same set $A$ are said to be compatible if, whenever $a$ and $b$ are comparable under both $R_1$ and $R_2$, we have $(a, b) \in R_1$ iff $(a, b) \in R_2$.

- Hence the topological sorting of a set $A$ w.r.t. a partial order relation $\preceq$ can be regarded as the construction of a total order relation $\preceq'$ such that $\preceq'$ is compatible with the existing partial order $\preceq$.
Compatible Partial Order Relations

- Two partial order relations $R_1$ and $R_2$ on the same set $A$ are said to be **compatible** if, whenever $a$ and $b$ are comparable under both $R_1$ and $R_2$, we have $(a, b) \in R_1$ iff $(a, b) \in R_2$.

- Hence the topological sorting of a set $A$ w.r.t. a partial order relation $\preceq$ can be regarded as the construction of a total order relation $\preceq'$ such that $\preceq'$ is compatible with the existing partial order $\preceq$.

- In general, a topological sorting doesn’t produce a unique result, unless the existing partial order relation $\preceq$ is in fact also a total order relation.
Example

Let $A = \{F, M, D, S\}$ denote a set of family members, $F$ (father), $M$ (mother), $D$ (daughter) and $S$ (son). Suppose the family have just acquired a computer game and all wish to play it as soon as possible. In what order can the family take turns to play the game, if the family tradition that children be given priority when it comes to playing games is to be observed?

Solution.

- It is obvious that there are 4 acceptable solutions. They are
Example

Let $A = \{F, M, D, S\}$ denote a set of family members, $F$ (father), $M$ (mother), $D$ (daughter) and $S$ (son).

Suppose the family have just acquired a computer game and all wish to play it as soon as possible. In what order can the family take turns to play the game, if the family tradition that children be given priority when it comes to playing games is to be observed?

Solution.

It is obvious that there are 4 acceptable solutions. They are

- (i) $D, S, M, F$;
- (ii) $D, S, F, M$;
- (iii) $S, D, M, F$ and
- (iv) $S, D, F, M$.
Example

Let $A = \{F, M, D, S\}$ denote a set of family members, $F$ (father), $M$ (mother), $D$ (daughter) and $S$ (son).

Suppose the family have just acquired a computer game and all wish to play it as soon as possible. In what order can the family take turns to play the game, if the family tradition that children be given priority when it comes to playing games is to be observed?

Solution.

- It is obvious that there are 4 acceptable solutions. They are
  - (i) $D, S, M, F$; (ii) $D, S, F, M$; (iii) $S, D, M, F$ and (iv) $S, D, F, M$.
- In obtaining any of the above 4 solutions, we have implicitly done a topological sorting!
Example

Let \( A = \{F, M, D, S\} \) denote a set of family members, \( F \) (father), \( M \) (mother), \( D \) (daughter) and \( S \) (son). Suppose the family have just acquired a computer game and all wish to play it as soon as possible. In what order can the family take turns to play the game, if the family tradition that children be given priority when it comes to playing games is to be observed?

Solution.

- It is obvious that there are 4 acceptable solutions. They are
  - (i) \( D, S, M, F \); (ii) \( D, S, F, M \); (iii) \( S, D, M, F \) and (iv) \( S, D, F, M \).
- In obtaining any of the above 4 solutions, we have implicitly done a topological sorting!
- In fact, the family tradition that children be given priority can be precisely represented by a partial order relation \( \preceq \) where all the comparable pairs are list below

\[
D \preceq M, \quad D \preceq F, \quad S \preceq M, \quad S \preceq F.
\]
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\leq$.
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\leq$.
- First we compare $D$ with $S$, $M$ and $F$. 

Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\leq$.
- First we compare $D$ with $S, M$ and $F$.
  - Since $D$ and $S$ are not comparable because neither has the priority, the order these 2 appear in the resulting list is not relevant.
Solution. (cont.)

Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,

- the list $D, S, F, M$ preserves the partial order relation $\leq$.

First we compare $D$ with $S$, $M$ and $F$.

- Since $D$ and $S$ are not comparable because neither has the priority, the order these 2 appear in the resulting list is not relevant.
- Since $D$ comes ahead of both $M$ and $F$ is consistent with the existing partial order, $D \leq M$ and $D \leq F$ respectively, we conclude that the 1st element, $D$, observes the existing partial order $\leq$. 
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\preceq$.
- First we compare $D$ with $S$, $M$ and $F$.
  - Since $D$ and $S$ are not comparable because neither has the priority, the order these 2 appear in the resulting list is not relevant.
  - Since $D$ comes ahead of both $M$ and $F$ is consistent with the existing partial order, $D \preceq M$ and $D \preceq F$ respectively, we conclude that the 1st element, $D$, observes the existing partial order $\preceq$.
- We then compare the 2nd element, $S$, with all of its later elements, $M$ and $F$, in the list (i).
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\preceq$.
- First we compare $D$ with $S, M$ and $F$.
  - Since $D$ and $S$ are not comparable because neither has the priority, the order these 2 appear in the resulting list is not relevant.
  - Since $D$ comes ahead of both $M$ and $F$ is consistent with the existing partial order, $D \preceq M$ and $D \preceq F$ respectively, we conclude that the 1st element, $D$, observes the existing partial order $\preceq$.
- We then compare the 2nd element, $S$, with all of its later elements, $M$ and $F$, in the list (i).
  - We can show likewise that $S$ also preserves the existing partial order.
Solution. (cont.)

- Let us now check that the list (i) is indeed a topological sorting of the set of people $A$. That is,
  - the list $D, S, F, M$ preserves the partial order relation $\preceq$.
- First we compare $D$ with $S$, $M$ and $F$.
  - Since $D$ and $S$ are not comparable because neither has the priority, the order these 2 appear in the resulting list is not relevant.
  - Since $D$ comes ahead of both $M$ and $F$ is consistent with the existing partial order, $D \preceq M$ and $D \preceq F$ respectively, we conclude that the 1st element, $D$, observes the existing partial order $\preceq$.
- We then compare the 2nd element, $S$, with all of its later elements, $M$ and $F$, in the list (i).
  - We can show likewise that $S$ also preserves the existing partial order.
- Similarly it can be verified that all elements in the list (i) are ordered consistently with the family tradition characterised by the relation $\preceq$. 
Topological Sorting Algorithm

- The topological sorting algorithm for a (nonempty) finite set $A$ with respect to a partial order relation $\preceq$ is in fact straightforward.
The topological sorting algorithm for a (nonempty) finite set $A$ with respect to a partial order relation $\preceq$ is in fact straightforward.

The algorithm is as follows:
The topological sorting algorithm for a (nonempty) finite set $A$ with respect to a partial order relation $\preceq$ is in fact straightforward.

The algorithm is as follows:
The **topological sorting algorithm** for a (nonempty) finite set $A$ with respect to a partial order relation $\preceq$ is in fact straightforward.

The algorithm is as follows:

**Topological sorting algorithm**

(i) Set the resulting list to empty initially.

(ii) Pick any minimal element in $A$.

   Append the element to the end of the resulting list and remove the element from the set $A$.

(iii) Go back to step (ii) if $A$ is still nonempty.

   Otherwise the algorithm terminates.
Topological Sorting Algorithm with Hasse Diagrams

With the assistance of Hasse diagrams, the above topological sorting algorithm can be simplified to the following steps:

1. Set the resulting list to empty initially.
2. Pick any minimal element of the Hasse diagram. Append the element to the end of the resulting list and remove the element, along with all the edges that are directly connected to it, from the Hasse diagram.
3. Go back to step 2 if the Hasse diagram is still nonempty. Otherwise, the algorithm terminates.

We note that minimal elements in a Hasse diagram are those bottom vertices in the diagram. By a bottom vertex we mean a vertex that is not downwardly connected to any other vertices in the Hasse diagram.
Topological Sorting Algorithm with Hasse Diagrams

With the assistance of Hasse diagrams, the above topological sorting algorithm can be simplified to the following steps:

1. Set the resulting list to empty initially.
2. Pick any minimal element of the Hasse diagram.
   Append the element to the end of the resulting list and remove the element, along with all the edges that are directly connected to it, from the Hasse diagram.
3. Go back to step 2 if the Hasse diagram is still nonempty. Otherwise, the algorithm terminates.

We note that minimal elements in a Hasse diagram are those bottom vertices in the diagram. By a bottom vertex, we mean a vertex that is not downwardly connected to any other vertices in the Hasse diagram.
Topological Sorting Algorithm with Hasse Diagrams

- With the assistance of Hasse diagrams, the above topological sorting algorithm can be simplified to the following steps:

**Topological sorting algorithm using Hasse diagrams**

(i’) Set the resulting list to empty initially.

(ii’) Pick any minimal element of the Hasse diagram. Append the element to the end of the resulting list and remove the element, along with all the edges that are directly connected to it, from the Hasse diagram.

(iii’) Go back to step (ii’) if the Hasse diagram is still nonempty. Otherwise the algorithm terminates.
With the assistance of Hasse diagrams, the above topological sorting algorithm can be simplified to the following steps:

1. Set the resulting list to empty initially.
2. Pick any minimal element of the Hasse diagram.
   Append the element to the end of the resulting list and remove the element, along with all the edges that are directly connected to it, from the Hasse diagram.
3. Go back to step (ii’) if the Hasse diagram is still nonempty. Otherwise the algorithm terminates.

We note that minimal elements in a Hasse diagram are those bottom vertices in the diagram. By a bottom vertex we mean a vertex that is not downwardly connected to any other vertices in the Hasse diagram.
**Example**

Let $A = \{1, 2, 3, 9, 18\}$ and, for any $a, b \in A$, $a \preceq b$ iff $a \mid b$. Construct a topological sorting for the relation $\preceq$ on the set $A$.

**Solution.**

- The following Hasse diagrams were created and utilised in the topological sorting algorithm.
Example

Let $A = \{1, 2, 3, 9, 18\}$ and, for any $a, b \in A$, $a \preceq b$ iff $a \mid b$. Construct a topological sorting for the relation $\preceq$ on the set $A$.

Solution.

- The following Hasse diagrams were created and utilised in the topological sorting algorithm.
- To start with, we first set the resulting list $T$ to an empty list, and observe that the Hasse diagram in $\text{A}$ shows ”1” is the only minimal element in $A$. 

Example

Let $A = \{1, 2, 3, 9, 18\}$ and, for any $a, b \in A$, $a \preceq b$ iff $a \mid b$. Construct a topological sorting for the relation $\preceq$ on the set $A$.

Solution.

- The following Hasse diagrams were created and utilised in the topological sorting algorithm.
- To start with, we first set the resulting list $T$ to an empty list, and observe that the Hasse diagram in $A$ shows ”1” is the only minimal element in $A$. 
Example

Let \( A = \{1, 2, 3, 9, 18\} \) and, for any \( a, b \in A \), \( a \preceq b \) iff \( a | b \). Construct a topological sorting for the relation \( \preceq \) on the set \( A \).

Solution.

- The following Hasse diagrams were created and utilised in the topological sorting algorithm.
- To start with, we first set the resulting list \( T \) to an empty list, and observe that the Hasse diagram in \( A \) shows ”1” is the only minimal element in \( A \).

\[
T = \emptyset
\]
Solution. (cont.)

- Hence we choose ”1” as the 1st element of the resulting list \( T \), and remove ”1” and the two edges that are directly connected to ”1”.
Solution. (cont.)

- Hence we choose ”1” as the 1st element of the resulting list $T$, and remove ”1” and the two edges that are directly connected to ”1”.
- The resulting Hasse diagram for the new set $\{2, 3, 9, 18\}$, the original set $A$ after the removal of ”1”, is then given in $B$. 

![Diagram of Hasse diagram for the set $\{2, 3, 9, 18\}$ after removing ”1”](image.png)
Solution. (cont.)

- Hence we choose ”1” as the 1st element of the resulting list \( T \), and remove ”1” and the two edges that are directly connected to ”1”.
- The resulting Hasse diagram for the new set \{2, 3, 9, 18\}, the original set \( A \) after the removal of ”1”, is then given in B.
Solution. (cont.)

- Hence we choose ”1” as the 1st element of the resulting list $T$, and remove ”1” and the two edges that are directly connected to ”1”.

- The resulting Hasse diagram for the new set $\{2, 3, 9, 18\}$, the original set $A$ after the removal of ”1”, is then given in $B$.

- There are 2 minimal elements in this case, ”2” and ”3”. We choose ”3” although we could also choose ”2”.

\[ T = \{1\} \]
Solution. (cont.)

- By removing the selected minimal element "3" and its directly connected edge from the Hasse diagram \( B \), we obtain the new \( T \) and the new Hasse diagram for \( \{2, 9, 18\} \) in \( C \).

\[
\begin{array}{c}
\text{18} \\
\text{9} \\
\text{3} \\
\text{2} \\
\text{1} \\
\text{2} \\
\text{9} \\
\text{18} \\
\end{array}
\]

\[
\begin{array}{c}
\text{18} \\
\text{9} \\
\text{3} \\
\text{2} \\
\text{1} \\
\text{2} \\
\text{9} \\
\text{18} \\
\end{array}
\]
Solution. (cont.)

By removing the selected minimal element "3" and its directly connected edge from the Hasse diagram $B$, we obtain the new $T$ and the new Hasse diagram for $\{2, 9, 18\}$ in $C$. 

\[ T = \{1, 3\} \]
Solution. (cont.)

- By removing the selected minimal element ”3” and its directly connected edge from the Hasse diagram \( B \), we obtain the new \( T \) and the new Hasse diagram for \( \{2, 9, 18\} \) in \( C \).

\[
T = \{1, 3\}
\]

- There are again 2 minimal elements, ”2” and ”9”. This time we pick ”2”.
Solution. (cont.)

- The resulting list $T$ and the shrunk Hasse diagram for $\{9, 18\}$ then become those in $D$. 

![Diagram showing Hasse diagrams for different sets and the list $T$.]
Solution. (cont.)

- The resulting list $T$ and the shrunk Hasse diagram for $\{9, 18\}$ then become those in $D$. 

\[ T = \{1, 3, 2\} \]
Solution. (cont.)

- The resulting list $T$ and the shrunk Hasse diagram for $\{9, 18\}$ then become those in $D$.

\[ T = \{1, 3, 2\} \]
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see \( F \).

![Hasse diagram with elements 1, 2, 3, 9, 18, 2, 3, 9, 18, 9, 18, 18, 18, F]
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $F$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$. 

![Hasse diagram with elements 1, 2, 3, 9, 18 ordered by topological sorting]
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $\text{F}$. 
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$. 
- We note that the topological sorting is not unique in this case.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $F$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.

![Hasse diagram](image)
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $\mathbf{F}$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $F$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $\mathbf{F}$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.

For instance, the list $T = \{1, 3, 9, 2, 18\}$ is another valid topological sorting.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see F.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see F.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.
Solution. (cont.)

- We proceed similarly until all the elements of the Hasse diagram have been moved to the resulting list $T$, see $F$.
- The list produced by the topological sorting is thus $T = \{1, 3, 2, 9, 18\}$.
- We note that the topological sorting is not unique in this case.

For instance, the list $T = \{1, 3, 9, 2, 18\}$ is another valid topological sorting.
Example

Let $A$ be the set of all subsets of set $\{a, b, c\}$, i.e.

$$A = \{ \emptyset, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\} \},$$

and a partial order relation $\leq$ on $A$ be defined by $u \leq v$ iff $u \subseteq v$. Construct a topological sorting for the relation $\leq$ on the set $A$.

Solution.

The intermediate results, obtained by carrying out steps (i)-(iii) in the topological sorting algorithm, are summarised in the table below.

<table>
<thead>
<tr>
<th>elements of set $A$</th>
<th>minimal elements</th>
<th>pick</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\emptyset, {a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>${a}, {b}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}$</td>
<td>${a}, {b}, {c}$</td>
<td>${b}$</td>
</tr>
<tr>
<td>${a}, {c}, {a, b}, {a, c}, {b, c}, {a, b, c}$</td>
<td>${a}, {c}$</td>
<td>${a}$</td>
</tr>
<tr>
<td>${c}, {a, b}, {a, c}, {b, c}, {a, b, c}$</td>
<td>${a, b}, {c}$</td>
<td>${a, b}$</td>
</tr>
<tr>
<td>${c}, {a, c}, {b, c}, {a, b, c}$</td>
<td>${c}$</td>
<td>${c}$</td>
</tr>
<tr>
<td>${a, c}, {b, c}, {a, b, c}$</td>
<td>${a, c}, {b, c}$</td>
<td>${a, c}$</td>
</tr>
<tr>
<td>${b, c}, {a, b, c}$</td>
<td>${b, c}$</td>
<td>${b, c}$</td>
</tr>
<tr>
<td>${a, b, c}$</td>
<td>${a, b, c}$</td>
<td>${a, b, c}$</td>
</tr>
<tr>
<td>none</td>
<td>none</td>
<td>stop</td>
</tr>
</tbody>
</table>
Solution. (cont).

\[
\{a, b, c\} \\
\{a, b\} \quad \{a, c\} \\
\{b, c\} \\
\{a\} \quad \{b\} \quad \{c\}
\]
Solution. (cont).

A

B

C

D

E

F

G

H

Ioan Despi – AMTH140 16 of 19
Solution. (cont).

[Diagram A]

[Diagram B]

[Diagram C]
Solution. (cont).
Solution. (cont.)
Solution. (cont).

Graphs A, B, C, D, E, and F illustrate various configurations of the sets \{a, b, c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a\}, \{b\}, \{c\}. Each graph represents a different arrangement of these sets, demonstrating the relationships between them.
Solution. (cont.)

Graph A shows the sets {a, b, c}, {a, b}, {a, c}, and {b, c} connected.

Graph B includes the sets {a, b, c}, {a, b}, {a, c}, and {b, c} with an additional connection.

Graph C features the sets {a, b, c}, {a, b}, {a, c}, and {b, c} with an extra link.

Graph D illustrates the sets {a, b, c}, {a, b}, and {b, c} with a specific connection.

Graph E displays the sets {a, b, c}, {a, b}, and {b, c} with a particular arrangement.

Graph F contains the sets {a, b, c}, {a, b}, and {b, c} with a specific configuration.

Graph G shows the sets {a, b, c} and {b, c} with a particular connection.

Ioan Despi – AMTH140
Solution. (cont).

[Diagram of sets A through H with labeled subsets]
Solution. (cont).
Solution.  (cont).

Hence the topological sorting for the set $A$ gives

$$\emptyset, \{b\}, \{a\}, \{a, b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}.$$
Solution. (cont).

- Hence the topological sorting for the set $A$ gives

$$\emptyset, \{b\}, \{a\}, \{a, b\}, \{c\}, \{a, c\}, \{b, c\}, \{a, b, c\}.$$ 

- We note that the construction of the above table may, or may not, make use of the following list of Hasse diagrams for locating the minimal elements.
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.
- Create a vertex $v$ for each task.
Application: Scheduling

### Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

### Graph model.
- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$. 
Scheduling.

Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.

- Create a vertex \( v \) for each task.
- Create an edge \( v \rightarrow w \) if task \( v \) must precede task \( w \).
- The model must be a DAG (Directed Acyclic Graph).
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.
- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
Application: Scheduling

**Scheduling.**
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

**Graph model.**
- Create a vertex $v$ for each task.
- Create an edge $v \to w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.
- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.
- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.

Topological order: B A D C E
Application: Scheduling

Scheduling.

Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.

- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.
Application: Scheduling

Scheduling.

Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.

- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.

Topological order: $B \ A \ D \ C \ E$
Scheduling.

Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.

- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.

- Create a vertex $v$ for each task.
- Create an edge $v \rightarrow w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.
Application: Scheduling

Scheduling.
Given a set of tasks to be completed with precedence constraints, in what order should we schedule the tasks?

Graph model.
- Create a vertex $v$ for each task.
- Create an edge $v \to w$ if task $v$ must precede task $w$.
- The model must be a DAG (Directed Acyclic Graph).
- Redraw DAG so all edges point left to right.
- Schedule tasks in topological order.

Topological order:
B A D C E
Application: Scheduling

Topological sorting: D E A B C
Application: Scheduling

Topological sorting:

D E A B C
Application: Scheduling

Topological sorting: D E A B C
Application: Scheduling

Topological sorting:
D E A B C
Application: Scheduling

Topological sorting:
D E A B C
Application: Scheduling

Topological sorting: D E A B C