Large-Scale Knowledge Processing #14 Using Binary Decision Diagrams (Part 3)

Faculty of Information Science and Technology, Hokkaido Univ.

Takashi Horiyama

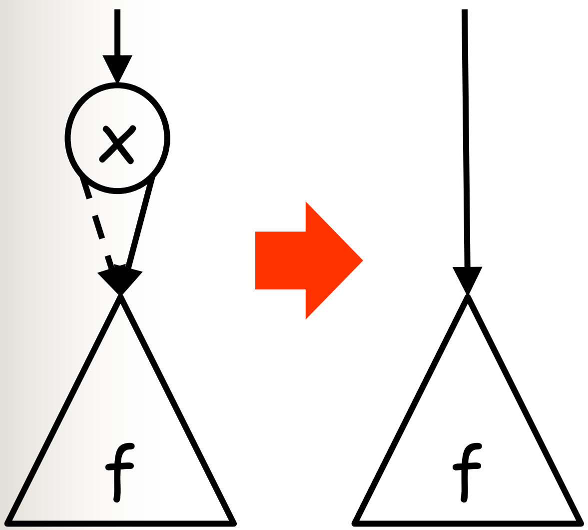
ZDD (Zero-suppressed BDD)

- ZDD: Variant of BDD
- The frontier-based search (Top-down construction of BDD/ZDD)

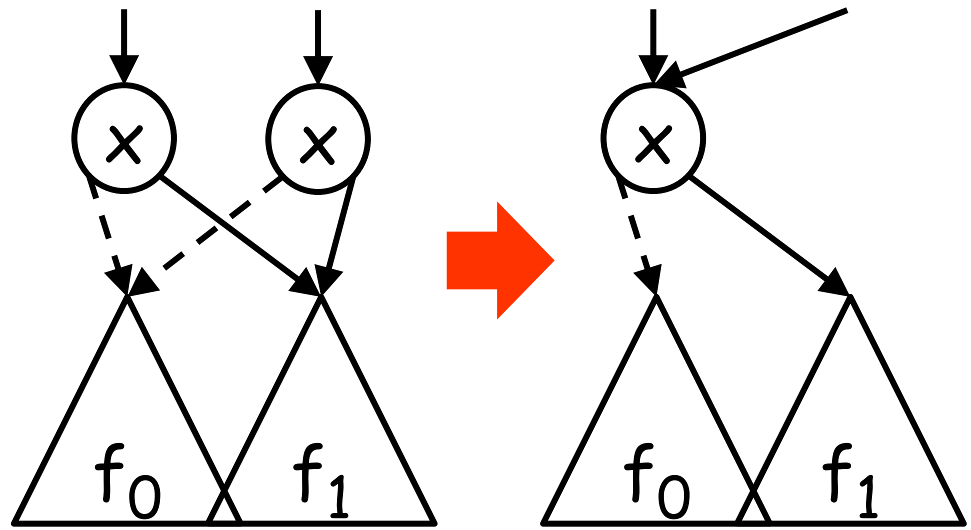
Prev. class : Binary Decision Diagram (BDD)

- **Variable order** : Variables appear according to a total order
- Two rules for **reducing** (i.e., simplifying) BDDs
  - **Reduction** : Repeat until we have no redundant and equivalent nodes

Remove a **redundant node**



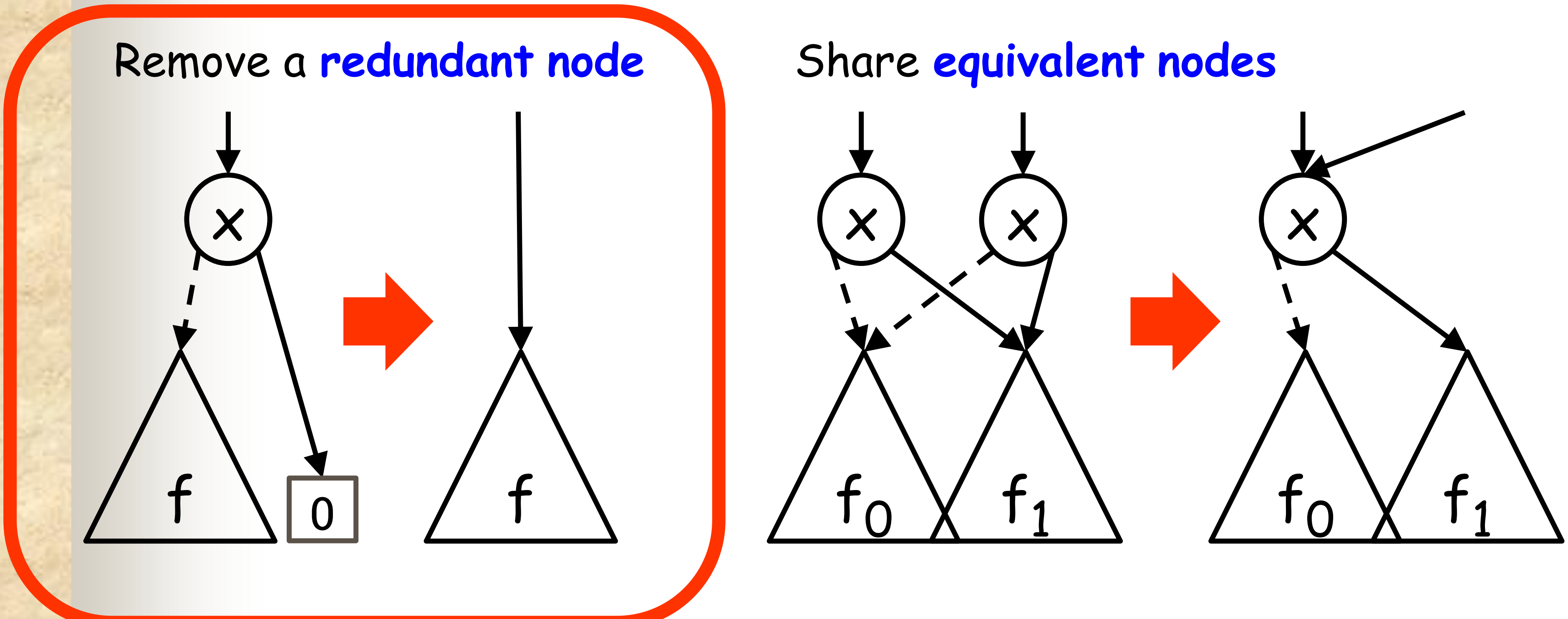
Share **equivalent nodes**



[Minato 1993]

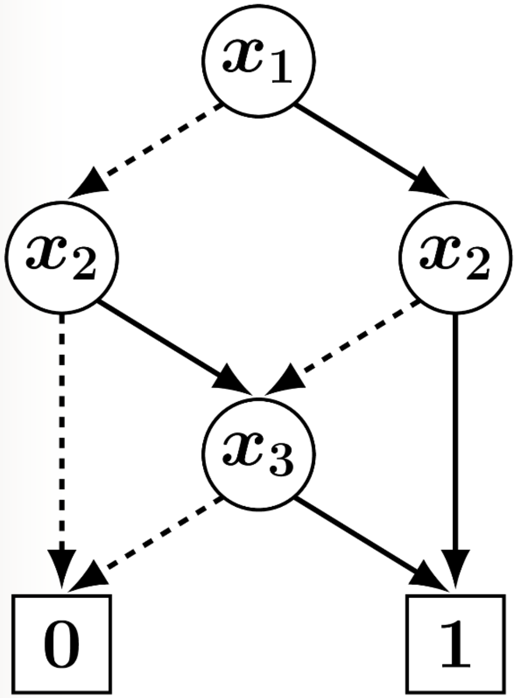
ZDD (Zero-Suppressed BDD)

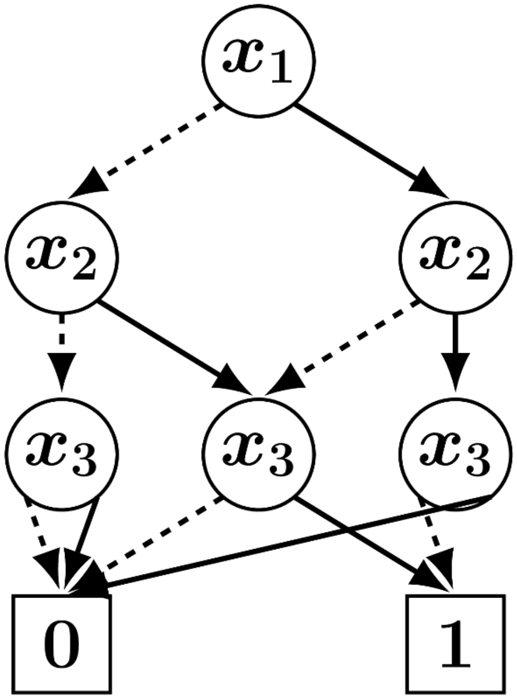
- **Variable order** : Variables appear according to a total order
- Two rules for **reducing** (i.e., simplifying) ZDDs
  - **Reduction** : Repeat until we have no redundant and equivalent nodes



ZDD

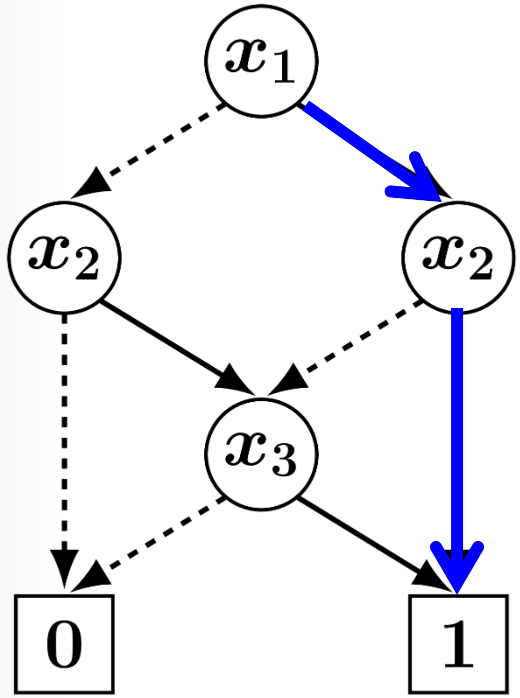
Ex.) $\{ \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\} \}$

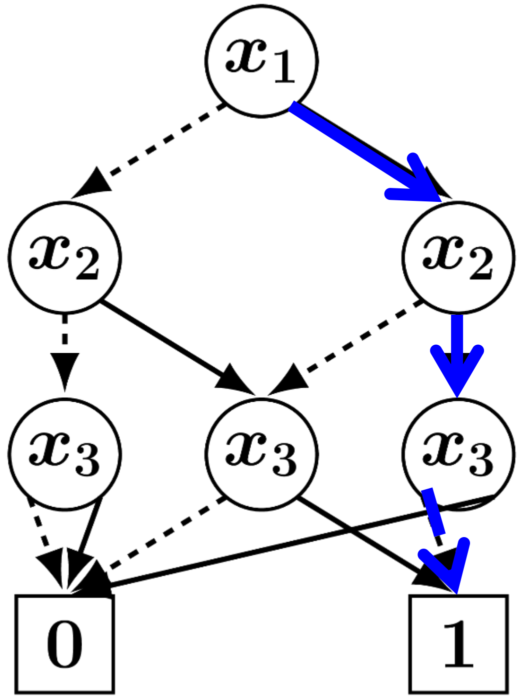




Find the family of sets represented by a ZDD (Method 1)

Ex.) { $\{x_1, x_2\}$, $\{x_1, x_3\}$, $\{x_2, x_3\}$ }

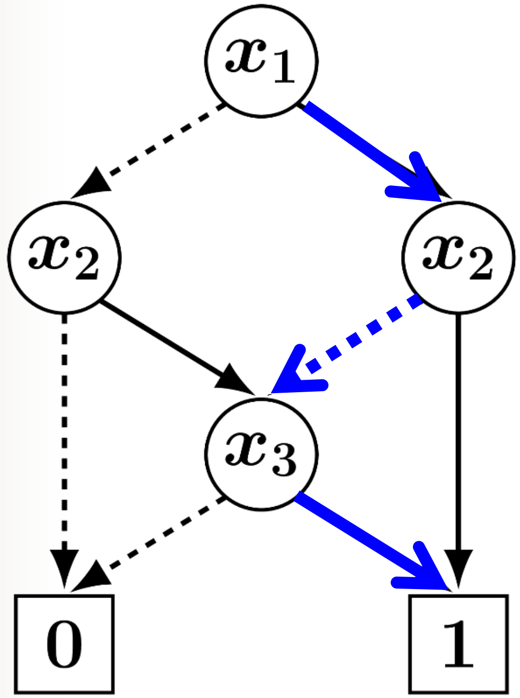




- Each 1-path corresponds to a set

Find the family of sets represented by a ZDD (Method 1)

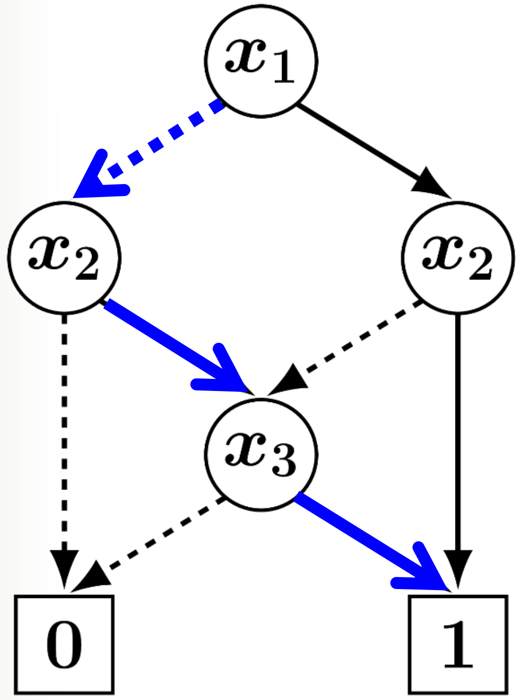
Ex.) $\{ \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\} \}$



- Each 1-path corresponds to a set

Find the family of sets represented by a ZDD (Method 1)

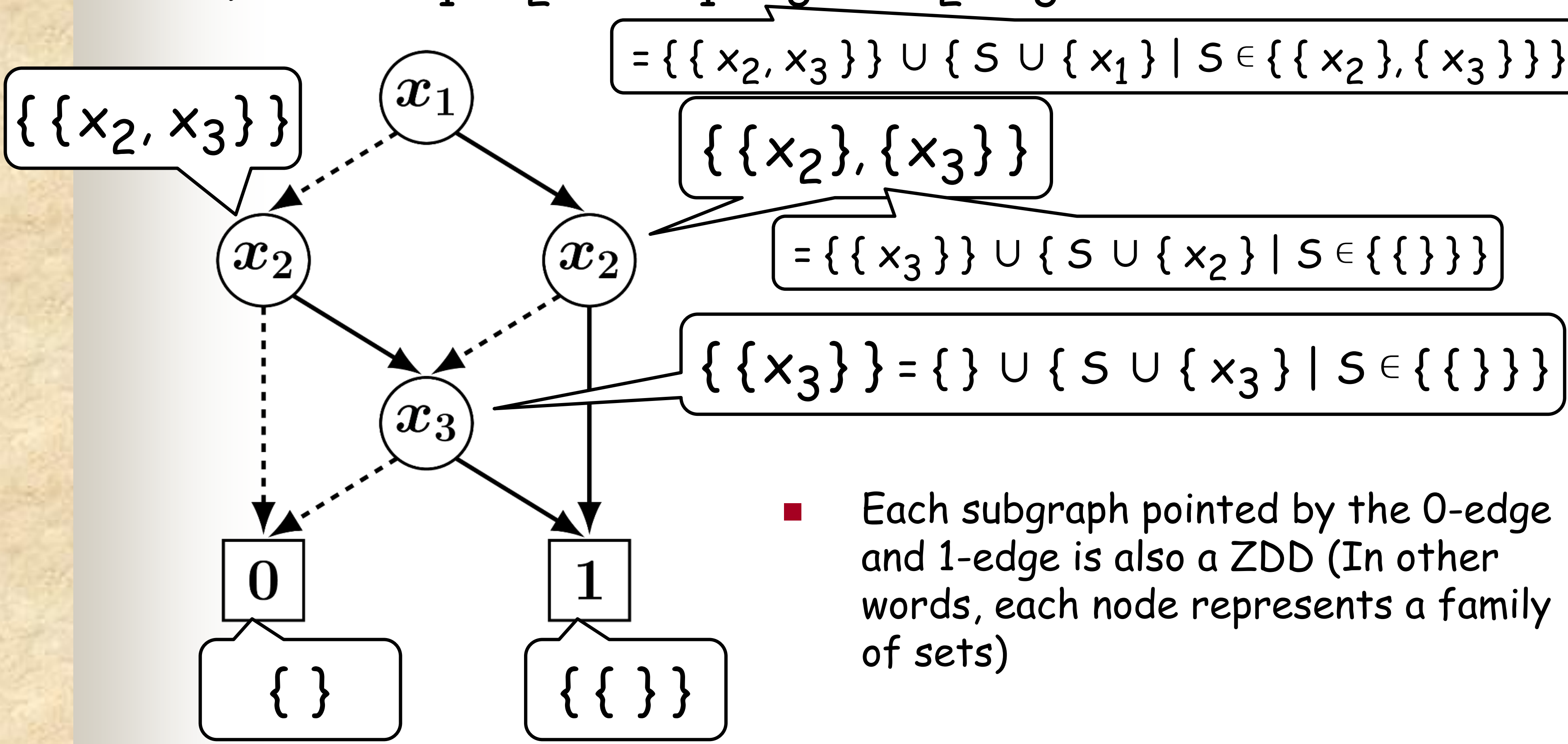
Ex.) $\{ \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\} \}$



- Each 1-path corresponds to a set

Find the family of sets represented by a ZDD (Method 2)

Ex.) $\{ \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\} \}$



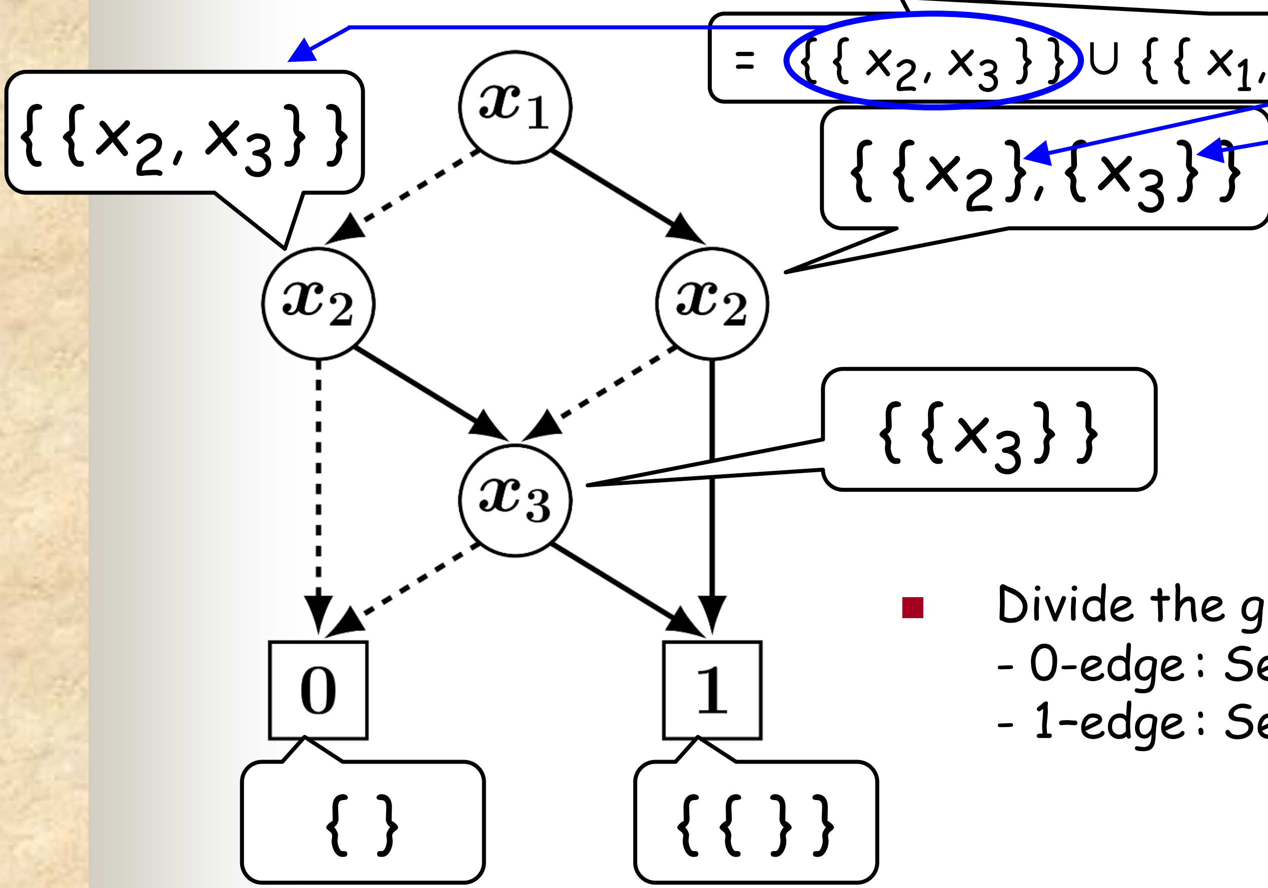
- Each subgraph pointed by the 0-edge and 1-edge is also a ZDD (In other words, each node represents a family of sets)

- $F = F_0 \cup \{ S \cup \{x_i\} \mid S \in F_1 \}$

Construct a ZDD from a family of sets

Ex.) $\{ \{x_1, x_2\}, \{x_1, x_3\}, \{x_2, x_3\} \}$

$= \{ \{x_2, x_3\} \} \cup \{ \{x_1, x_2\}, \{x_1, x_3\} \} \}$

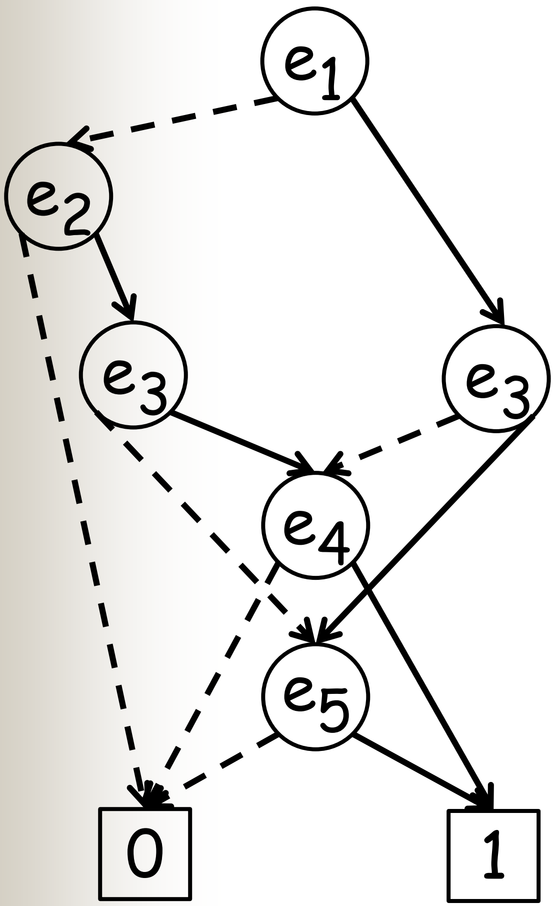


- Divide the given family of sets into two
  - 0-edge : Sets that do not contain x_i
  - 1-edge : Sets that contain x_i

- $F = F_0 \cup \{ S \cup \{x_i\} \mid S \in F_1 \}$

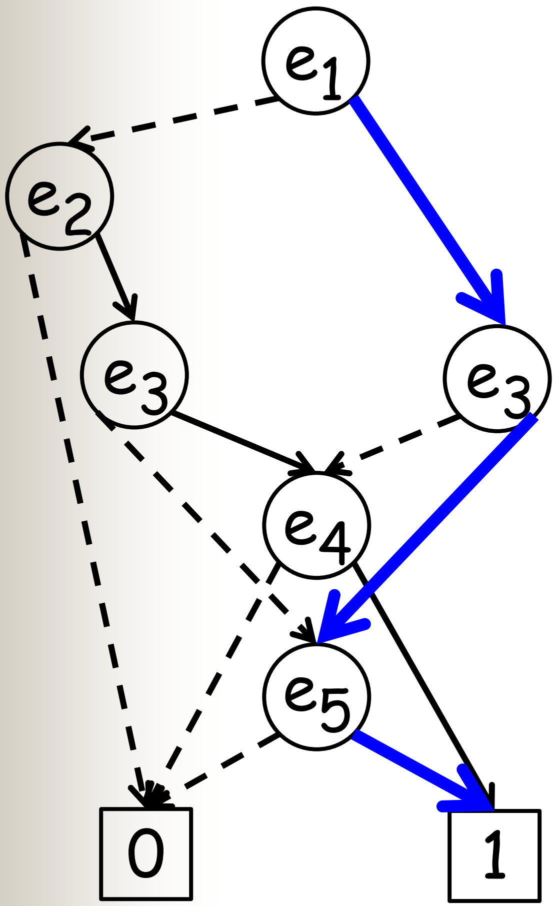
practice： ZDD

1. Find the family of sets represented by the ZDD below
2. Construct the ZDD representing the family of sets obtained in 1 with variable order e_2, e_1, e_3, e_4, e_5



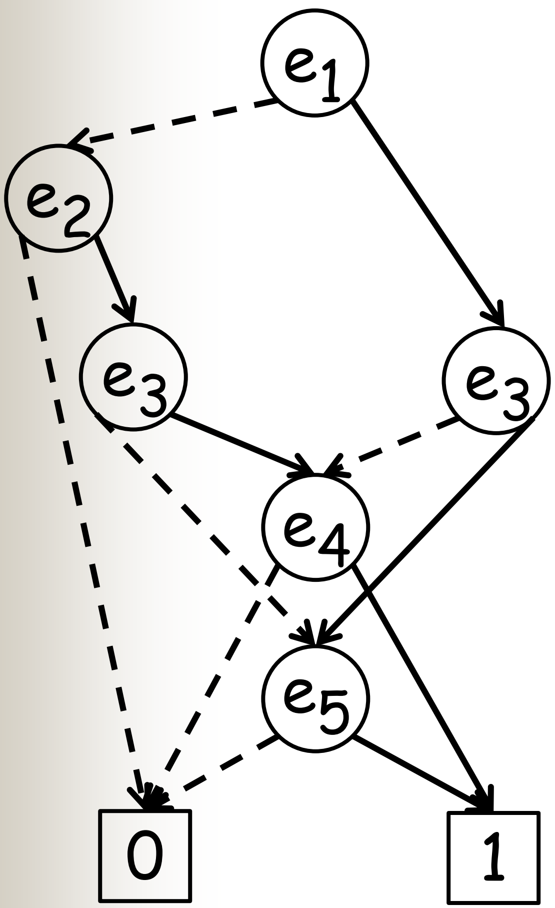
practice: ZDD (cont.)

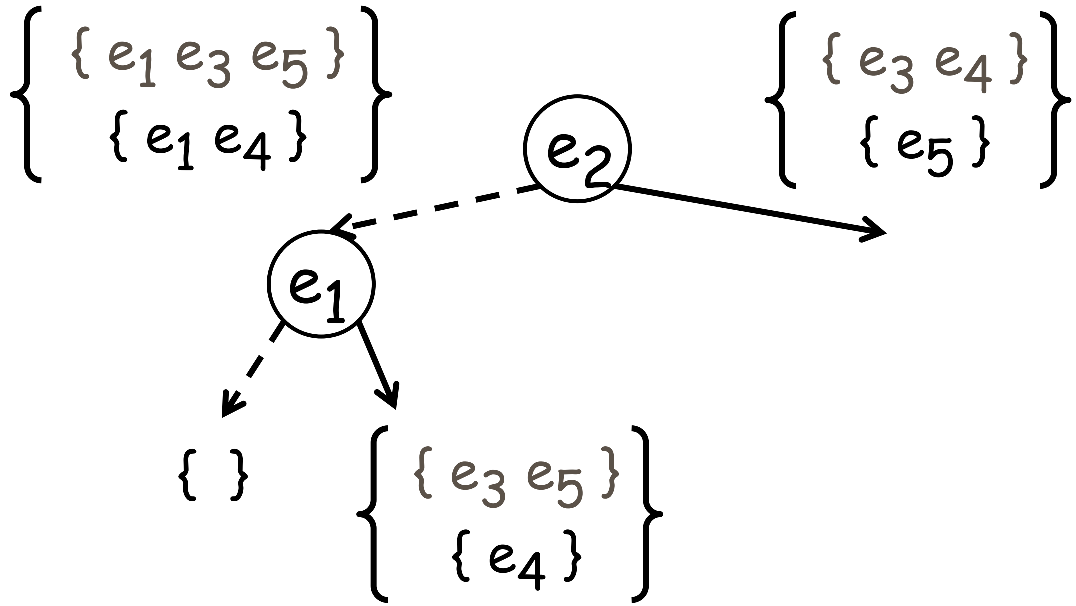
$$\left\{ \begin{matrix} \{e_1\, e_3\, e_5\} & \{e_1\, e_4\} \\ \{e_2\, e_3\, e_4\} & \{e_2\, e_5\} \end{matrix} \right\}$$



practice: ZDD (cont.)

$$\left\{ \begin{array}{ll} \{e_1\, e_3\, e_5\} & \{e_1\, e_4\} \\ \{e_2\, e_3\, e_4\} & \{e_2\, e_5\} \end{array} \right\}$$

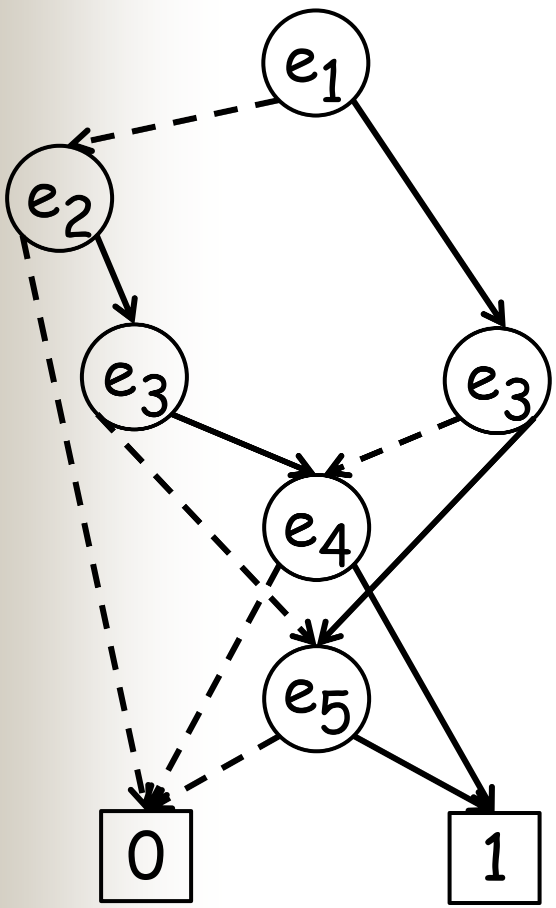




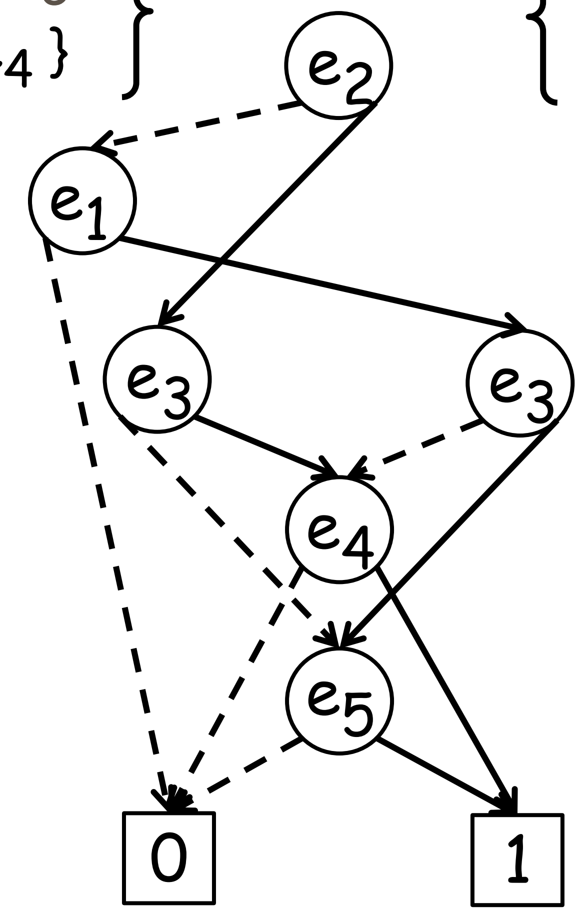
practice : ZDD (cont.)

$$\left\{ \begin{array}{ll} \{e_1\, e_3\, e_5\} & \{e_1\, e_4\} \\ \{e_2\, e_3\, e_4\} & \{e_2\, e_5\} \end{array} \right\}$$

$$\left\{ \begin{array}{c} \{e_1\, e_3\, e_5\} \\ \{e_1\, e_4\} \end{array} \right\}$$



$$\left\{ \begin{array}{c} \{e_3\, e_4\} \\ \{e_5\} \end{array} \right\}$$



Short break

- Take a deep breath, and relax yourself

The frontier-based search
(top-down construction of BDD/ZDD)

Top-down construction of BDD/ZDD

- **Simpath** [Knuth 2008]
  - Enumeration of s-t paths
  - Construct a ZDD in a top-down manner like **dynamic programming**
- **Frontier-based search** [Kawahara et al. 2014, 2017]
  - Generalization of Simpath
  - Enumeration of spanning trees, forests, cycles, Hamiltonian cycles, Hamiltonian paths, etc.

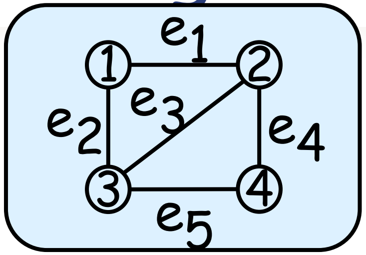
※ Similar approaches were attempted to problems in other fields

- Jones polynomials (knot theory)
- Spanning trees
- Network reliability [Hardy]

} [Sekine, Imai 1995]

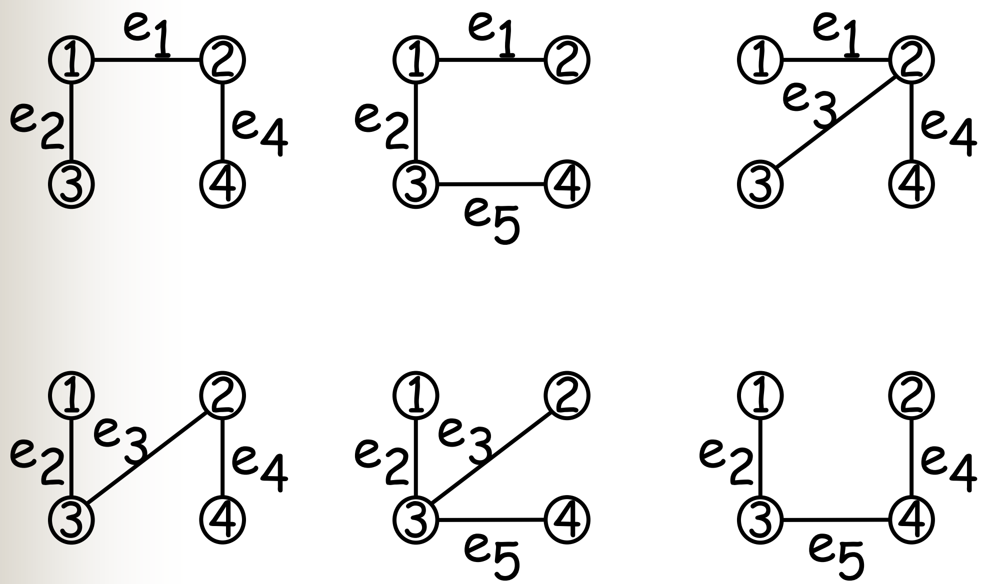
Enumeration of spanning trees

- Do not contain cycles
- All vertices are connected

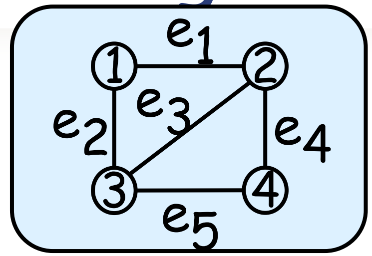


Input : a graph

Output : spanning trees of a given graph



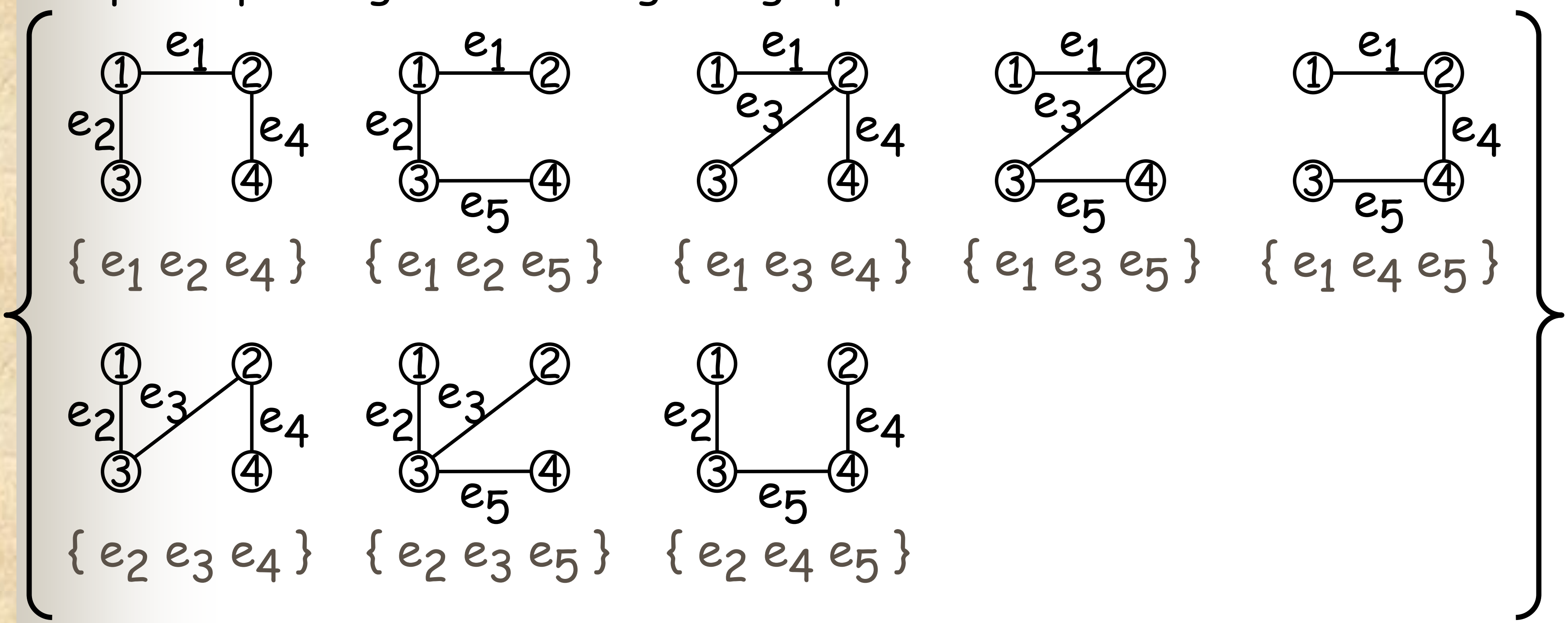
Enumeration of spanning trees



Input : a graph

- Do not contain cycles
- All vertices are connected

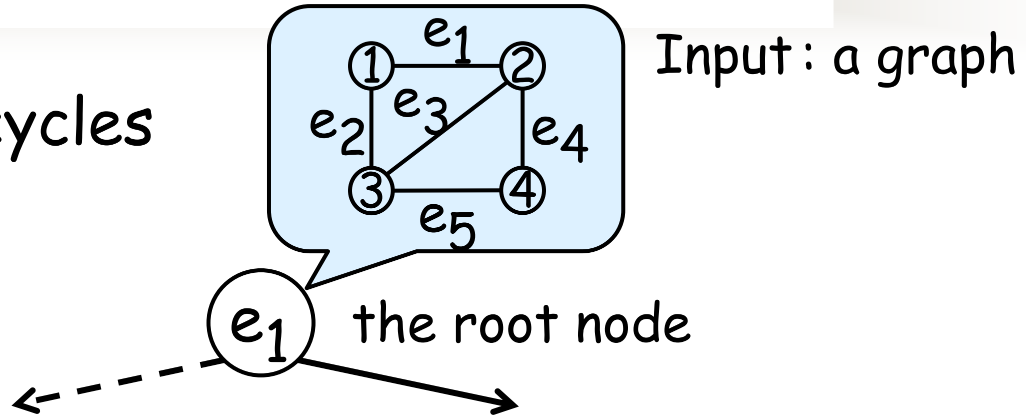
Output : spanning trees of a given graph



A spanning tree is represented as a set of its edges

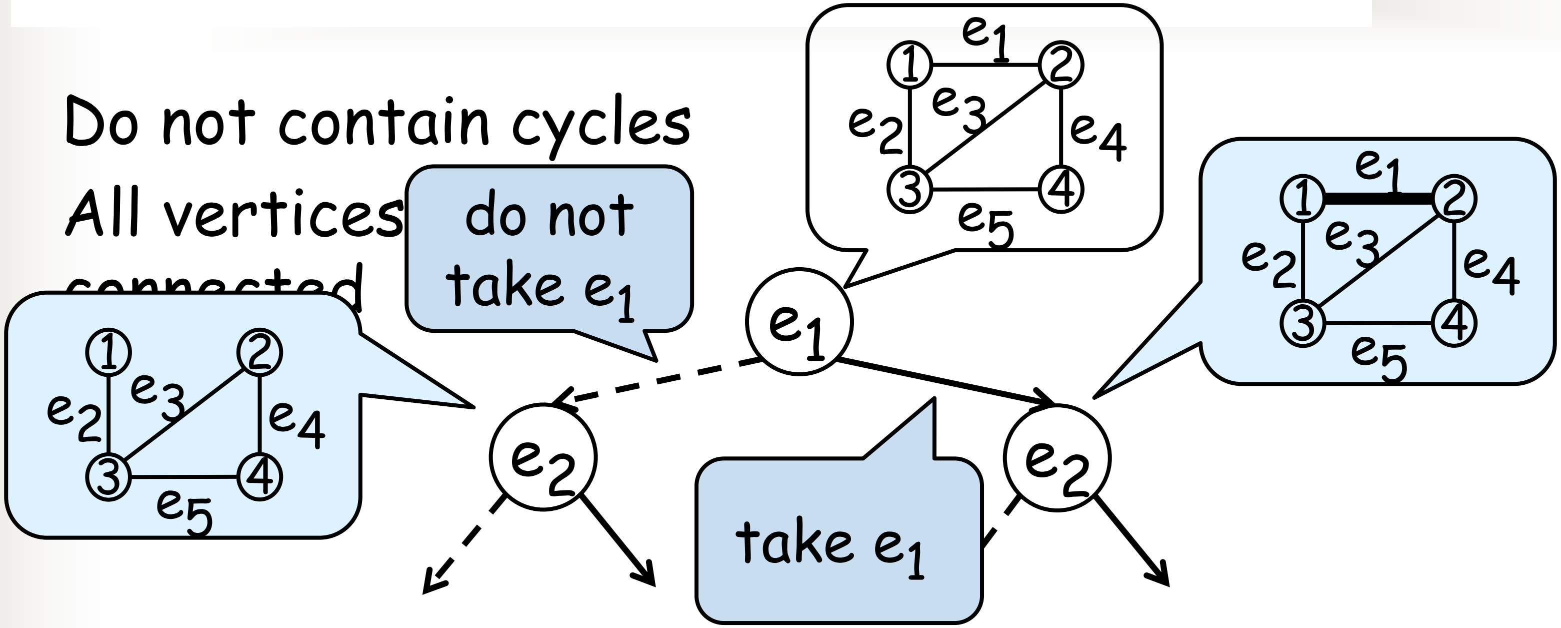
Enumeration of spanning trees : Construction of a ZDD

- Do not contain cycles
- All vertices are connected



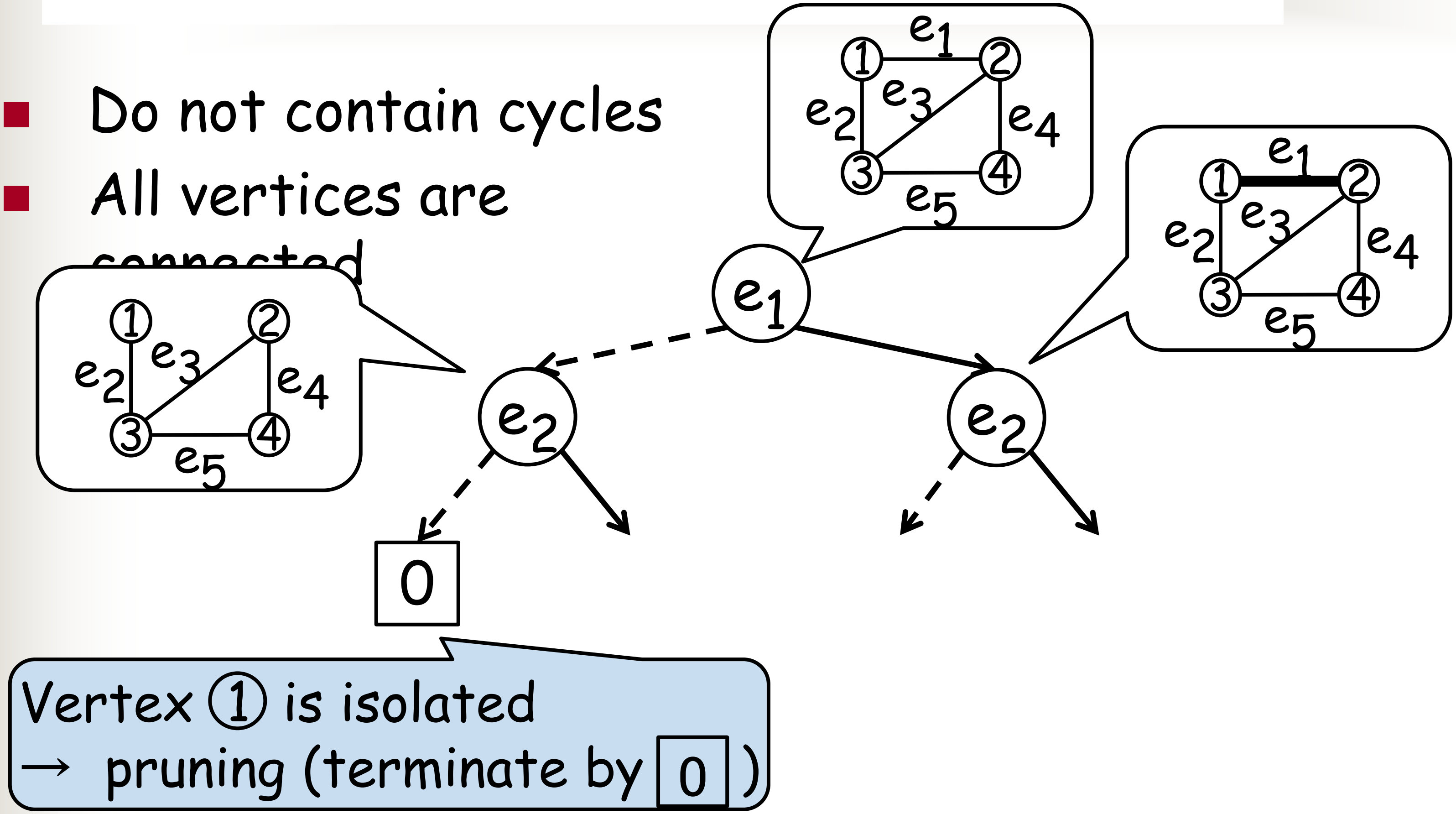
Enumeration of spanning trees : Construction of a ZDD

- Do not contain cycles
- All vertices connected



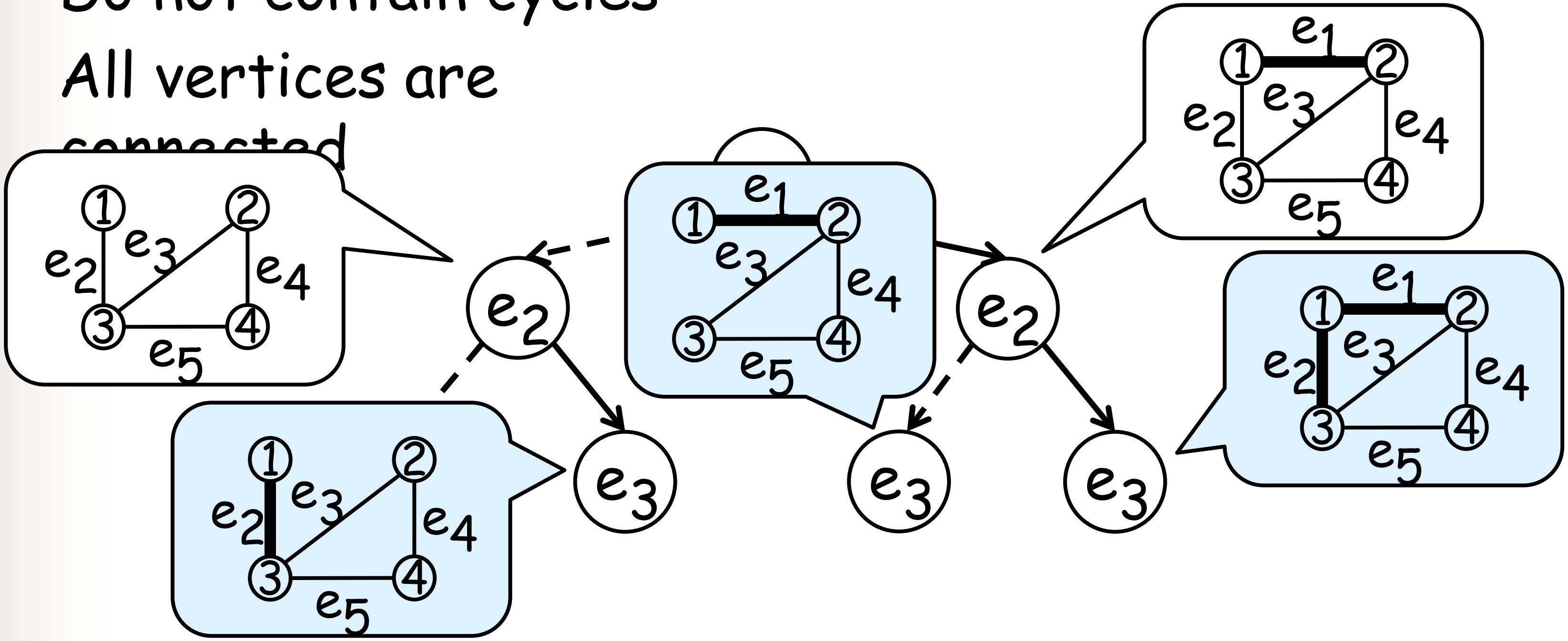
Enumeration of spanning trees : Construction of a ZDD

- Do not contain cycles
- All vertices are connected



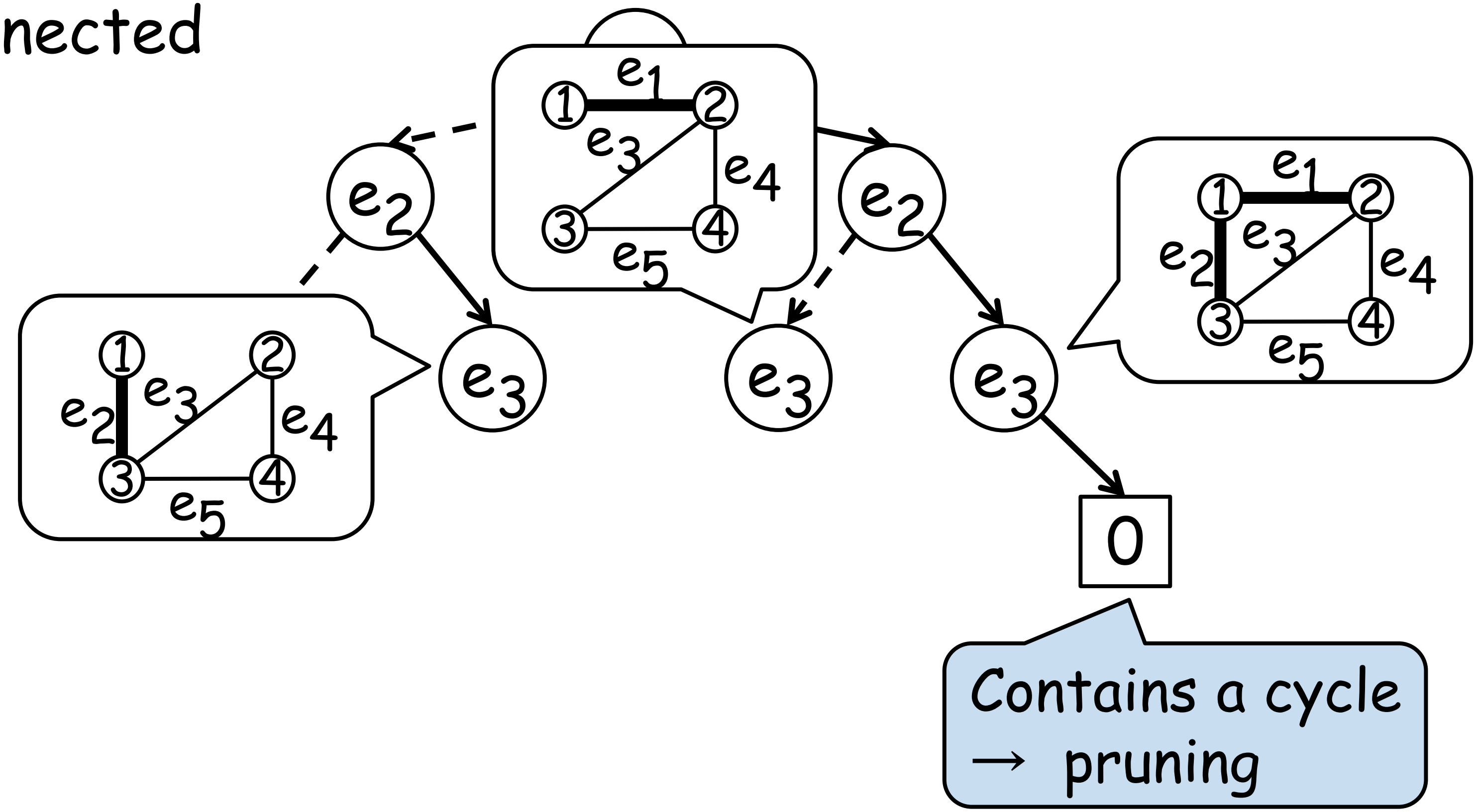
Enumeration of spanning trees : Construction of a ZDD

- Do not contain cycles
- All vertices are connected



Enumeration of spanning trees : Construction of a ZDD

- Do not contain cycles
- All vertices are connected

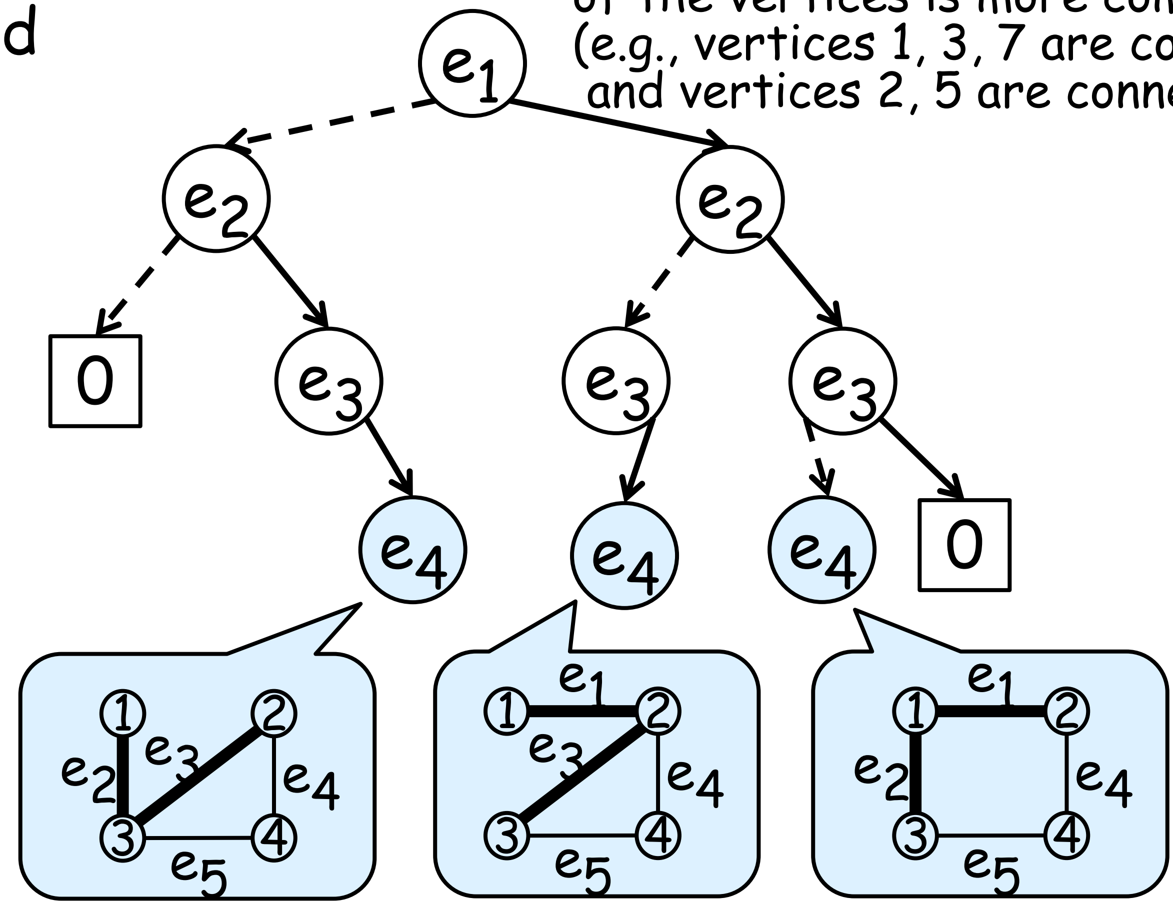


Enumeration of sp
Construction of a

The three nodes in the ZDD are essentially equivalent
- Vertices 1, 2, and 3 are connected
- Remaining subproblems on e_4, e_5 are the same

- Do not contain cycles
- All vertices are connected

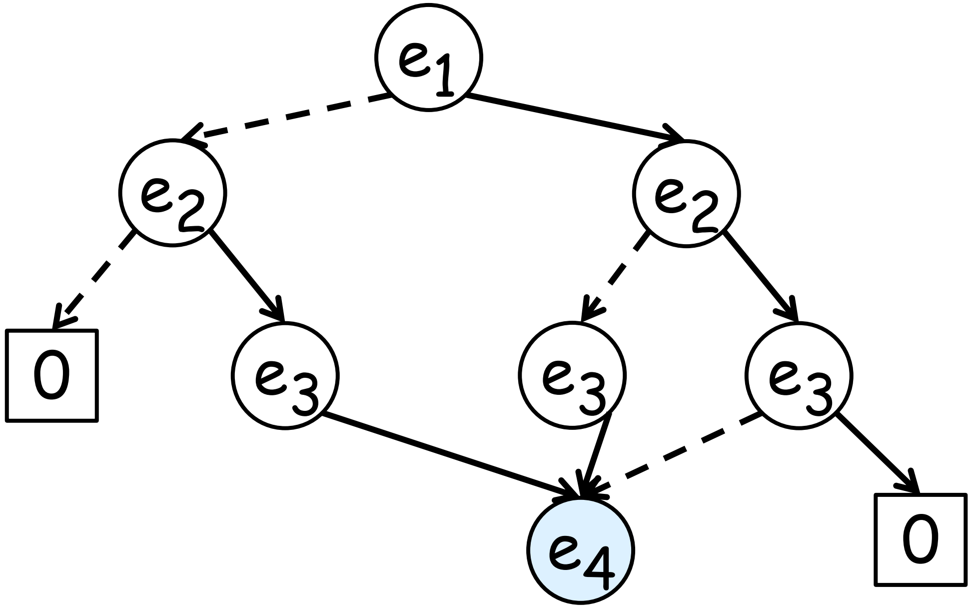
※ For a large graph, the connectivity of the vertices is more complex (e.g., vertices 1, 3, 7 are connected and vertices 2, 5 are connected ...)



Enumeration of sp
Construction of a

The three nodes in the ZDD
are essentially equivalent
 - Vertices 1, 2, and 3 are connected
 - Remaining subproblems on e_4, e_5
 are the same

- Do not contain cycles
- All vertices are connected



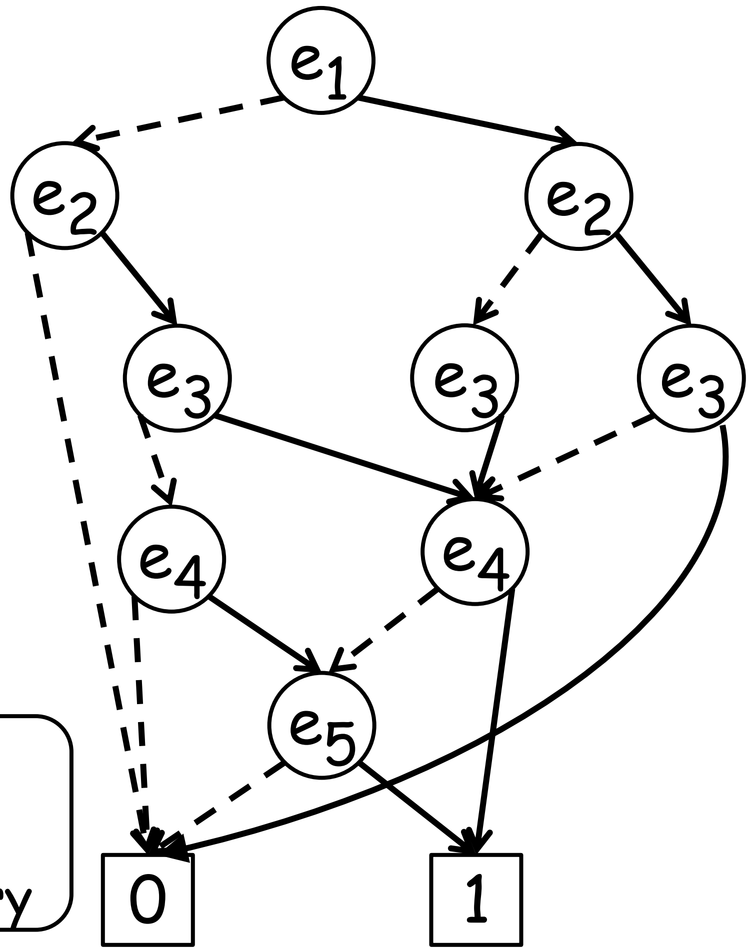
Equivalent nodes are
shared in the ZDD
→ high speed, less memory

Enumeration of sp
Construction of a

The three nodes in the ZDD are essentially equivalent
- Vertices 1, 2, and 3 are connected
- Remaining subproblems on e_4, e_5 are the same

- Do not contain cycles
- All vertices are connected

How to check these two properties?
(It is inefficient to keep a whole graph in each ZDD node and to check the whole graph in each ZDD node)

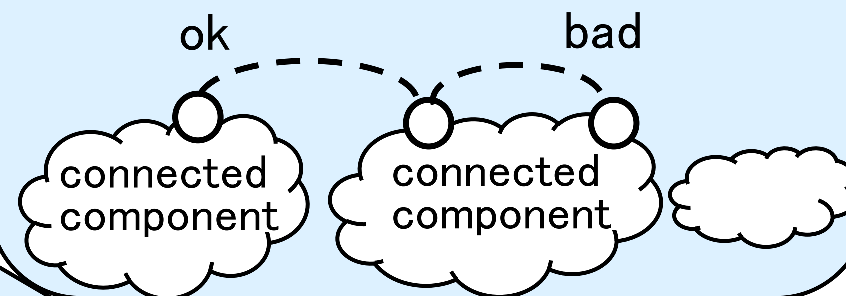


Equivalent nodes are shared in the ZDD
→ high speed, less memory

Conditions for pruning (termination by 0)

- Do not contain cycles
- All vertices are connected

Pruning if we take an edge whose both ends belong to the same connected component,



How to check these two properties?

(It is inefficient to keep a whole graph in each ZDD node and to check the whole graph in each ZDD node)

Pruning if it is confirmed that the number of connected component is two or more

Idea:

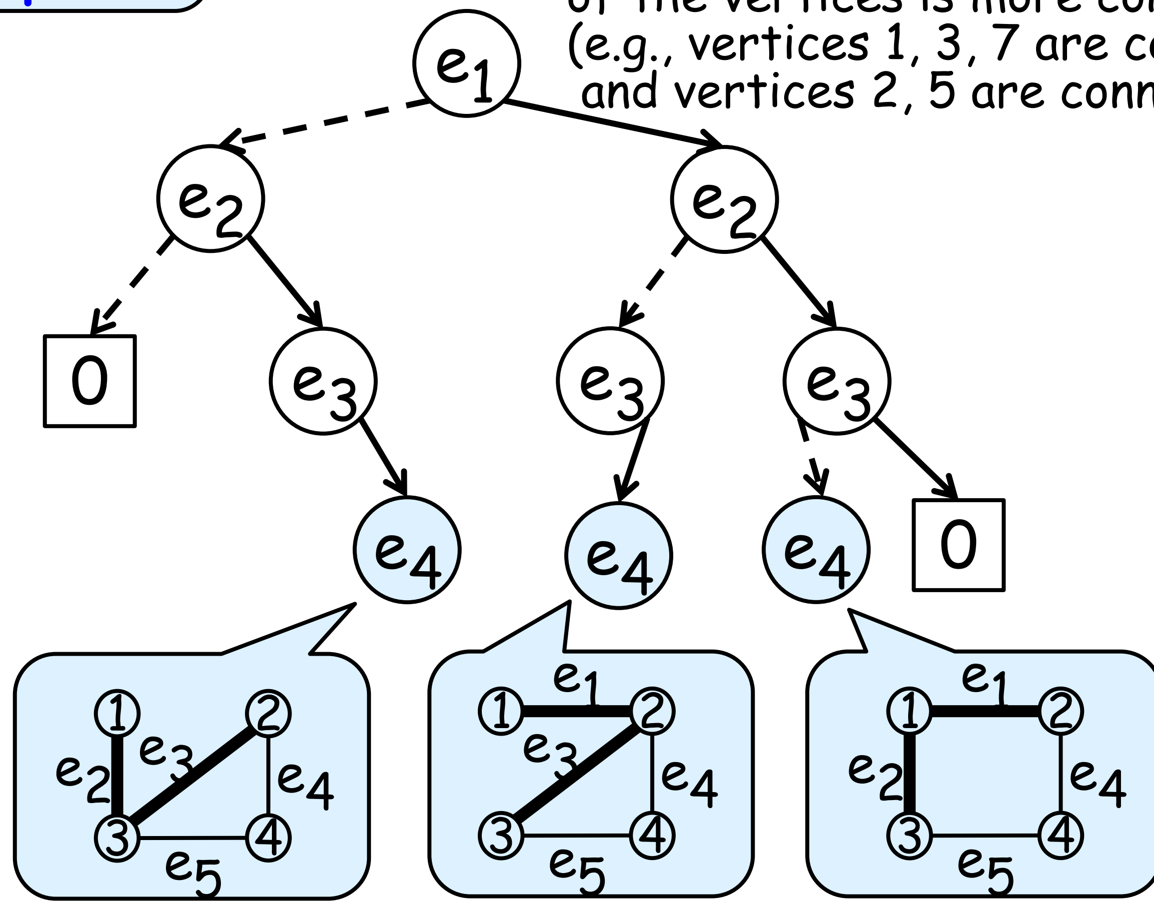
- Do not keep a whole graph, but keep **which vertex belongs to which connected component**
- In the initial graph, all vertices are isolated
- If we take an edge, two connected components are connected by the edge

Enumeration of sp
Construction of a

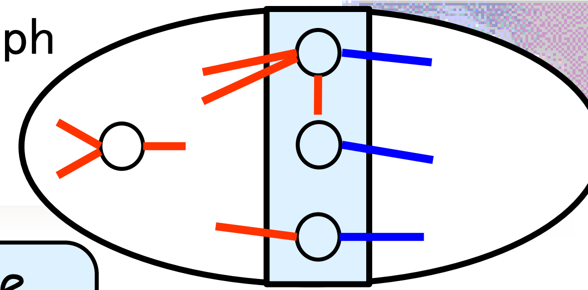
The three nodes in the ZDD are essentially equivalent
- Vertices 1, 2, and 3 are connected
- Remaining subproblems on e_4, e_5 are the same

We can use the information on **which vertex belongs to which connected component**

※ For a large graph, the connectivity of the vertices is more complex (e.g., vertices 1, 3, 7 are connected and vertices 2, 5 are connected ...)



The frontier



(The frontier is uniquely defined for each level of the ZDD)

We can use the information on **which vertex belongs to which connected component**

We'd like to reduce the information we need to memorize

- Vertices can be classified into three types
  - Vertex that is adjacent **only** to **unexplored edges**
    - Always isolated → there is no need to memorize the information (connected component) of this vertex
  - Vertex that is adjacent to both **unexplored edges** and **explored edges** (We say that this vertex is in the frontier)
    - Memorize the information of this vertex

      to memorize
  - Vertex that is adjacent **only** to **explored edges**
    - We have another vertex that belongs to the same connected component and is in the frontier (Otherwise, the search was already pruned since the vertex and any vertices in the frontier belong to different connected component and have no chance of being the same)
    - It' is sufficient to check only the vertices in the frontier → no need to memorize the information

We can understand the details on the construction of a ZDD if we recheck the slides on the construction with the knowledge of this page

Application of ZDDs: Analysis of power grid design

Switching control of power grid

- Example of standard power distribution network

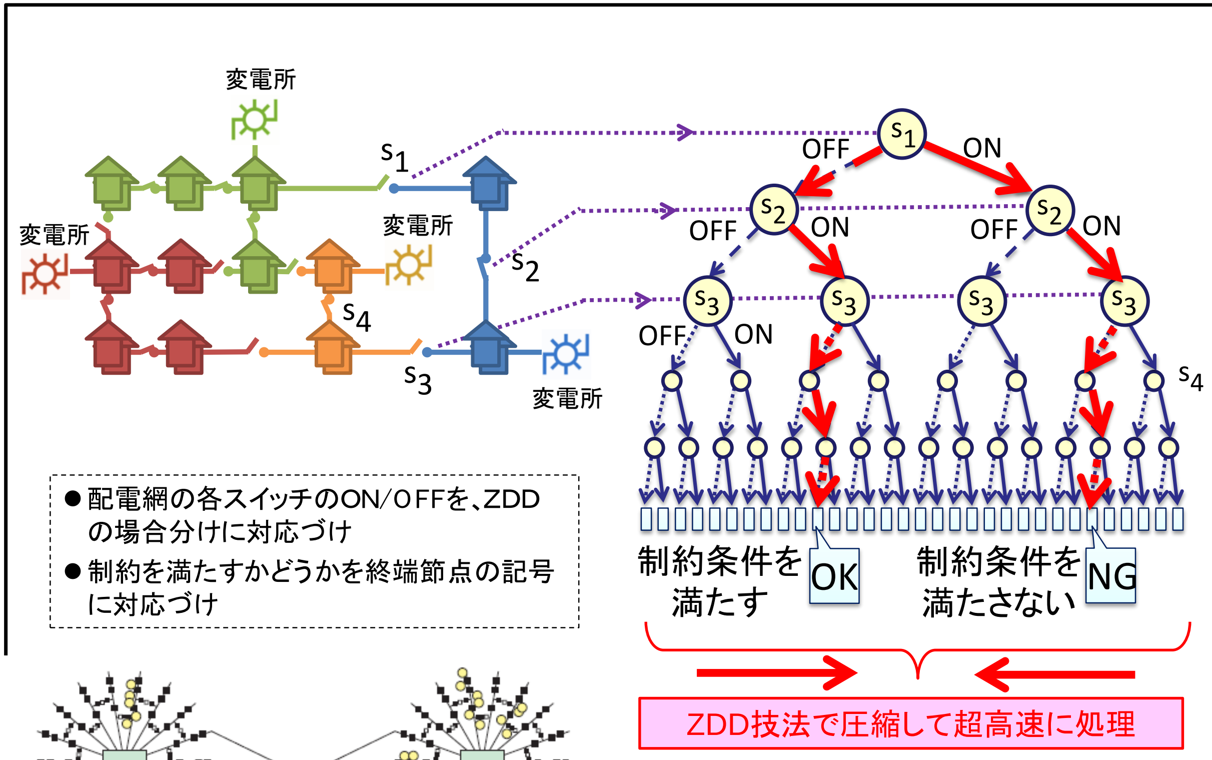
468 switches

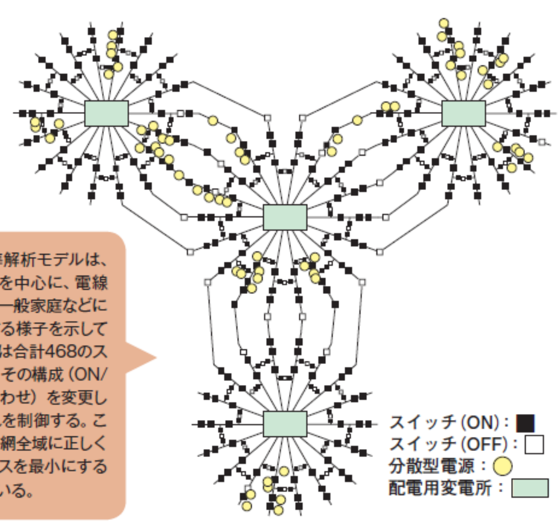
10^{140} combinations

Graph theory constraint

+

Electrical constraints (e.g., voltage drop)





配電網の標準解析モデルは、4つの変電所を中心に、電線に接続された一般家庭などに電力を供給する様子を示している。電線には合計468のスイッチがあり、その構成（ON/OFFの組み合わせ）を変更して電力の流れを制御する。この図では配電網全域に正しく給電、送電ロスを最小にする構成を表している。

High-speed, large scale, and high reliable analysis method on power grids, where its importance increased after the Great East Japan Earthquake

Supplementary materials

- ERATO Minato discrete structure manipulation system project, S. Minato (ed.), **Ultra high-speed graph enumeration algorithm**, Morikita publishing, 2015. (in Japanese)
- R. E. Bryant, Graph-based algorithms for Boolean function manipulation. IEEE Trans. Comput. C-35(8), pp. 677–691, 1986.
- S. Minato, Zero-suppressed BDDs for set manipulation in combinatorial problems, In Proc. of the 30th International Design Automation Conference, 1993, pp. 272–277.
- J. Kawahara, T. Inoue, H. Iwashita, S. Minato, Frontier-Based Search for Enumerating All Constrained Subgraphs with Compressed Representation, IEICE Transactions, 100-A(9), pp. 1773-1784, 2017.

Large-Scale Knowledge Processing

- Course objectives
  - This lecture aims to learn the techniques of knowledge processing, which are essential to intellectual information processing, such as editing, classifying, analyzing, and indexing of knowledge.
- Topics on large-scale knowledge processing: (Lectures are given in parallel or sequentially.)
  - Optimization techniques
  - Fundamentals of Boolean functions and computational complexity
  - Exact algorithms and approximation algorithms
  - Manipulation of discrete structure by BDDs/ZDDs
- Report assignments will be assigned.