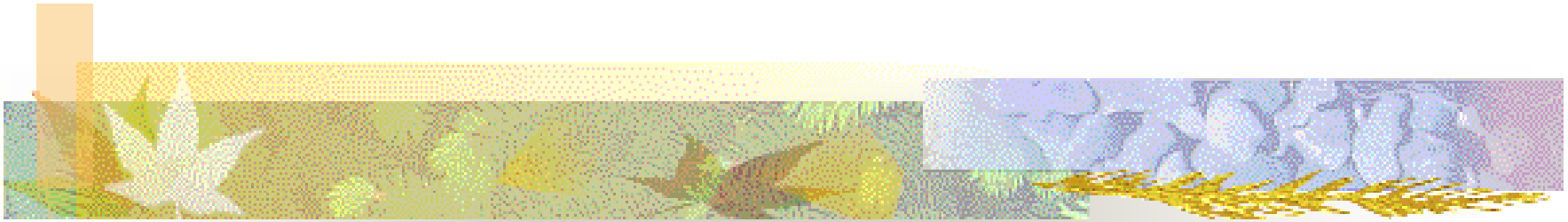
Large-Scale Knowledge Processing Guidance



Faculty of Information Science and Technology, Hokkaido Univ.

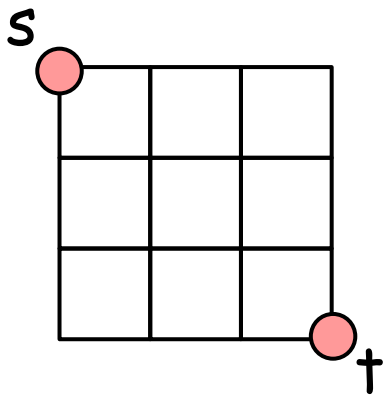
Takashi Horiyama
Kazuhisa Seto

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.

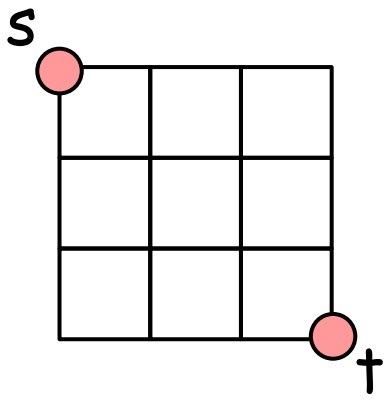
Quiz : s-t path (shortest path)

Q : Enumerate (or count) all **shortest paths** from s to t



Quiz : s-t path (shortest path)

Q : Enumerate (or count) all **shortest paths** from s to t



Any combination of ⬇⬇⬇➡➡➡

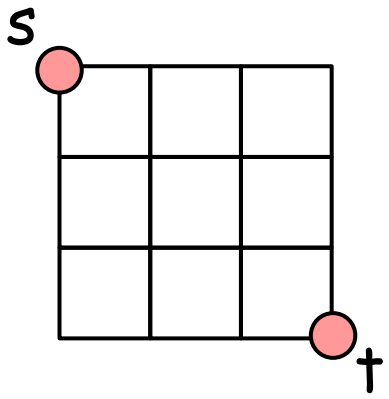
gives a shortest path

#(shortest path) is $_6C_3 = \frac{6 \cdot 5 \cdot 4}{3 \cdot 2 \cdot 1} = 20$

Quiz : s-t path ~~(shortest path)~~

Q : Enumerate (or count) all ~~shortest~~ **paths** from s to t

- **Detours** are **allowed**
- Self avoiding : Any path **cannot** go through the **same vertex twice** (or more)



- There is no formula for counting the number of paths

(The answer of the above problem is 184)

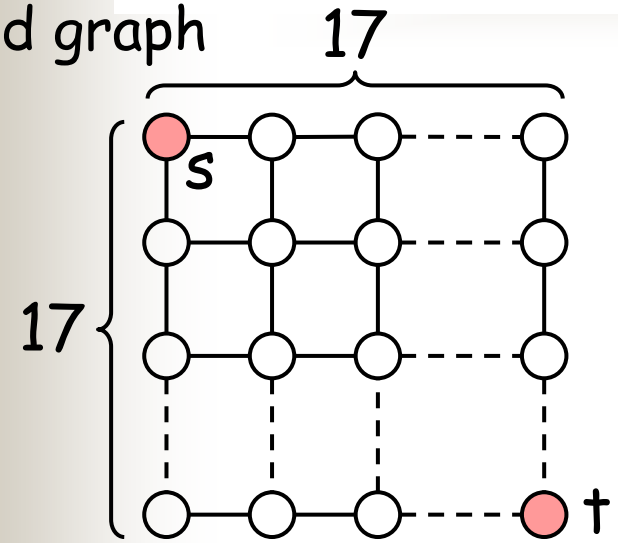
Enumeration:

- Find all solutions satisfying the given conditions

- What is required
  - Enumerate **efficiently** (Time complexity)
  - Store the solutions **compactly** (Space complexity)
  - Use the solutions **easily**
    - How many ? / Sampling
    - Retrieve the solutions by various queries

Ex.) s-t path (Self-avoiding path)

Grid graph



#(s-t paths) 6,344,814,
611,237,963,971,310,297,540,
795,524,400,449,443,986,866,
480,693,646,369,387,855,336

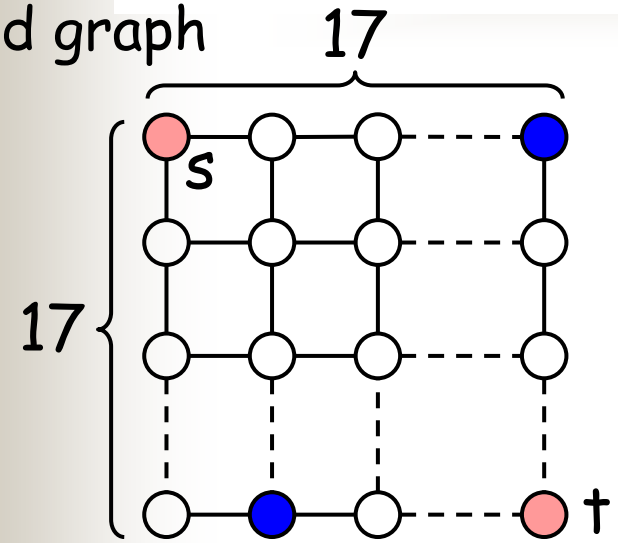
Approx. 6.3×10^{61} (63 那由他)

"Sufficiently large number" in Sanskrit

- What is required
  - Enumerate **efficiently** (Time complexity)
  - Store the solutions **compactly** (Space complexity)
  - Use the solutions **easily**
    - How many ? / Sampling
    - Retrieve the solutions by various queries

Ex.) s-t path (Self-avoiding path)

Grid graph



#(s-t paths) 6,344,814,
611,237,963,971,310,297,540,
795,524,400,449,443,986,866,
480,693,646,369,387,855,336

Approx. 6.3×10^{61} (63 那由他)

"Sufficiently large number" in Sanskrit

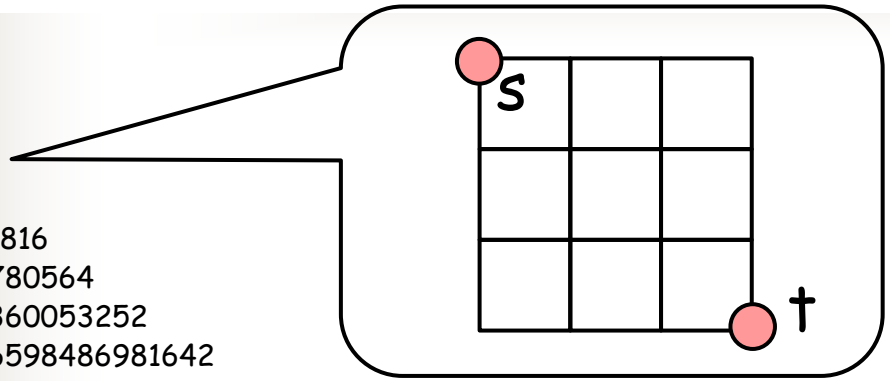
- What is required
  - Enumerate **efficiently** (Time complexity)
  - Store the solutions **compactly**
  - Use the solutions **easily**
    - How many ? / Sampling
    - Retrieve the solutions by various queries

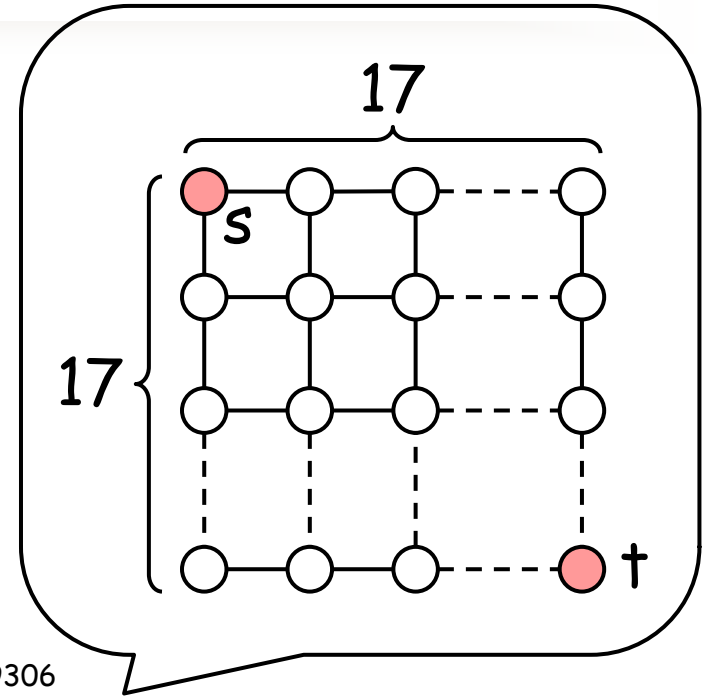
The number of paths of length at most 50?

Longest path ?

Paths going through both of two ●s?

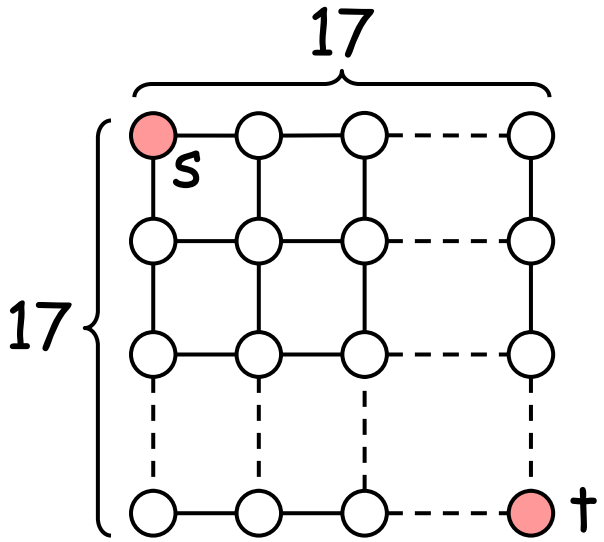
#(s-t paths)





1 2
2 12
3 184
4 8512
5 1262816
6 575780564
7 789360053252
8 3266598486981642
9 41044208702632496804
10 1568758030464750013214100
11 182413291514248049241470885236
12 64528039343270018963357185158482118
13 69450664761521361664274701548907358996488
14 22744971467681273963182645932798963387613323440
15 226674556886267274637456739671309893486632488540831902 8
16 68745445609149931587631563132489232824587945968099457285419306
17 63448146112379639713102975407955244004494439868664806936463693878553 36
18 17821128408420651298933849466523252751678380657047676559314524746058266927825 32
19 15233449717048799930807428103192296908994542553232945557760298667373550605928775692558 44
20 39628921998230375602072995171333625021063397057394637715152371133770106823640357067044720649403 98

#(s-t paths)



- 21
  3137475105013710272042053813738221451310331219369872365306135199134643337938938579396557699224602131646
  3868
- 22
  7559702866673453396615191233152226193531037320724094811673914104795179257927436312349870388833176349872
  71171404439792
- 23
  5543542935523747700991431848906143793069037997096433133255695864648400840733488554456638692402087571
  12420600854085134829339457 20
- 24
  1237171223120706475833874486267357083237304198901294353967872708048495169551593048564139455079215303719
  1858028212512280926600304581386791094
- 25
  8402974857881133471007083745436809127296054293775383549824742623937028497898215256929178577083970960121
  6256025060273165497184021064940499783756042474 08
- 26
  1736993158627927293117544042123649890037222958828814060466370372091034241327613476278921819349800610708
  2296223143380491348290026721931129627708738890853908108906396

Algorithms for large-scale knowledge processing

- **How to solve the problems** (= strategies)
- Why algorithms are **important** ?
  - Improve **implementations**: **2x or 3x** speed-up
    Improve **algorithms**: **1000x** speed-up
    → We can handle larger/practical problems
  - **Basic theory** can connect to various **applications**

    Ex.)
    - "Electoral district" and "disaster evacuation site"
    - "Developments" and "origami by cells"
    - …

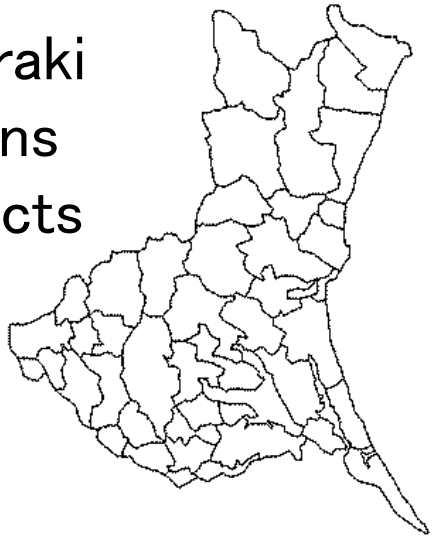
Electoral district

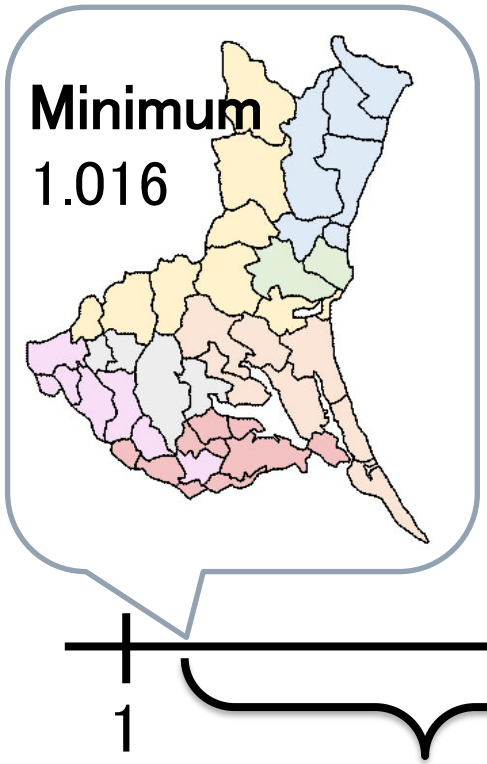
4,893,281,393,039,250,022,519,012,101,206 ways for partitioning Osaka prefecture (We cannot store them in the usual way) We can find all of them in 0.34 seconds.

- **Political equality**

Partition into districts with almost equal populations according to the census results

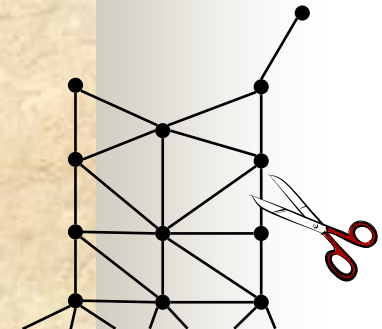
Ex.: Ibaraki
41 regions
7 districts





We are interested in the gap.
Many ways for partitioning or not ?

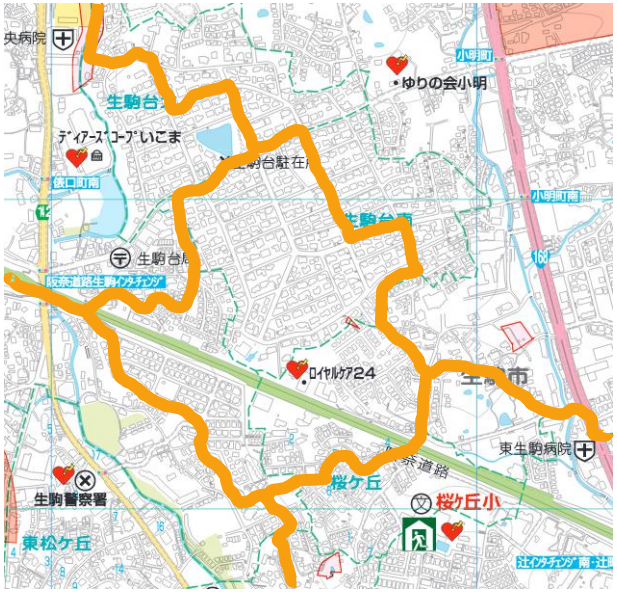
How to tackle problems
- Not all at once
- Step-by-step approach



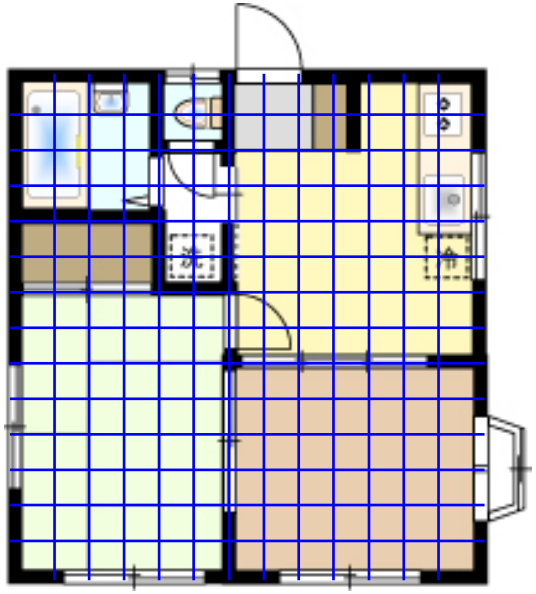
Ex.: model the problem as a partition of a graph

Other applications

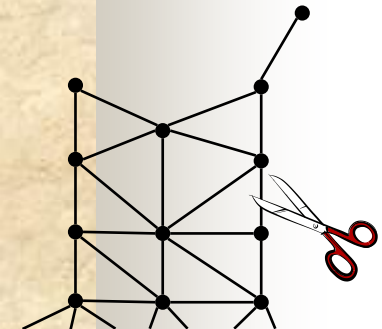
- Similar ideas can be **applied to other areas**



Evacuation plan
(Allocation of
evacuation sites)



Floor plan



Ex.: model the problem
as a partition of a graph

Folding Mechanism

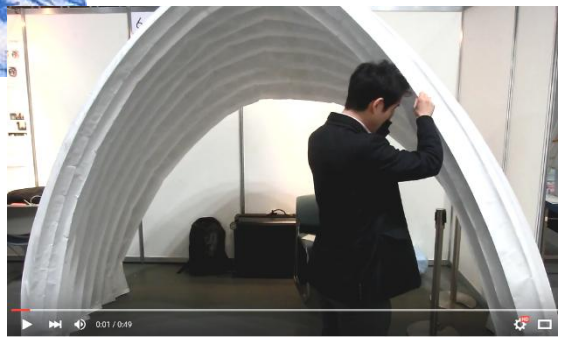
Packages

Folding Mechanism

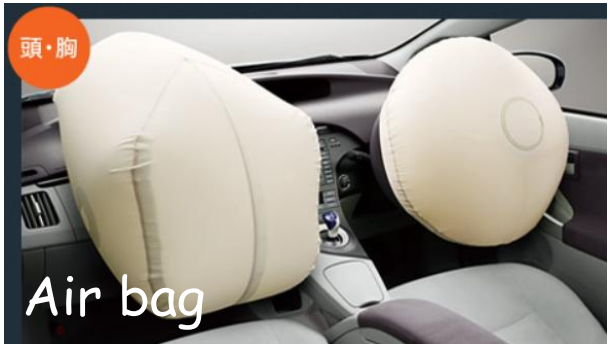
Packages

Satellite panel (Miura fold)

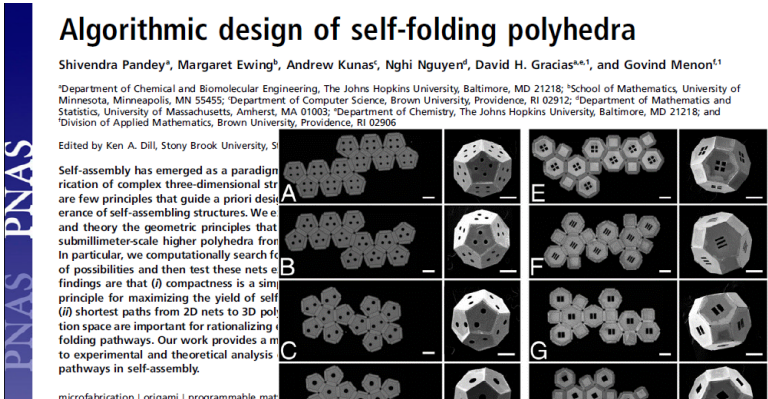
Architecture



Canned beverage (Yoshimura pattern)



Research on Origami



Algorithmic design of self-folding polyhedra

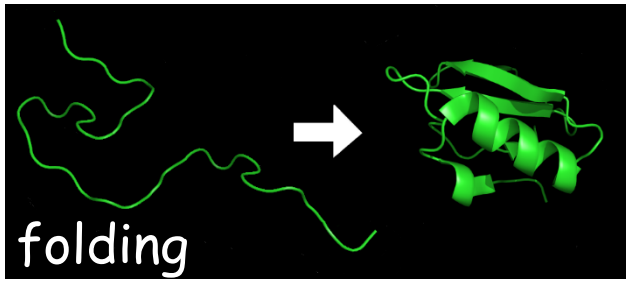
Shivendra Pandey[a], Margaret Ewing[b], Andrew Kunas[c], Nghi Nguyen[d], David H. Gracias[a,e,1], and Govind Menon[f,1]

[a]Department of Chemical and Biomolecular Engineering, The Johns Hopkins University, Baltimore, MD 21218; [b]School of Mathematics, University of Minnesota, Minneapolis, MN 55455; [c]Department of Computer Science, Brown University, Providence, RI 02912; [d]Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003; [e]Department of Chemistry, The Johns Hopkins University, Baltimore, MD 21218; and [f]Division of Applied Mathematics, Brown University, Providence, RI 02906

Edited by Ken A. Dill, Stony Brook University, S

Self-assembly has emerged as a paradigm
rication of complex three-dimensional str
are few principles that guide a priori desi
erance of self-assembling structures. We e
and theory the geometric principles that
submillimeter-scale higher polyhedra fro
In particular, we computationally search f
of possibilities and then test these nets e
findings are that (*i*) compactness is a sim
principle for maximizing the yield of self
(*ii*) shortest paths from 2D nets to 3D pol
tion space are important for rationalizing
folding pathways. Our work provides a m
to experimental and theoretical analysis
pathways in self-assembly.

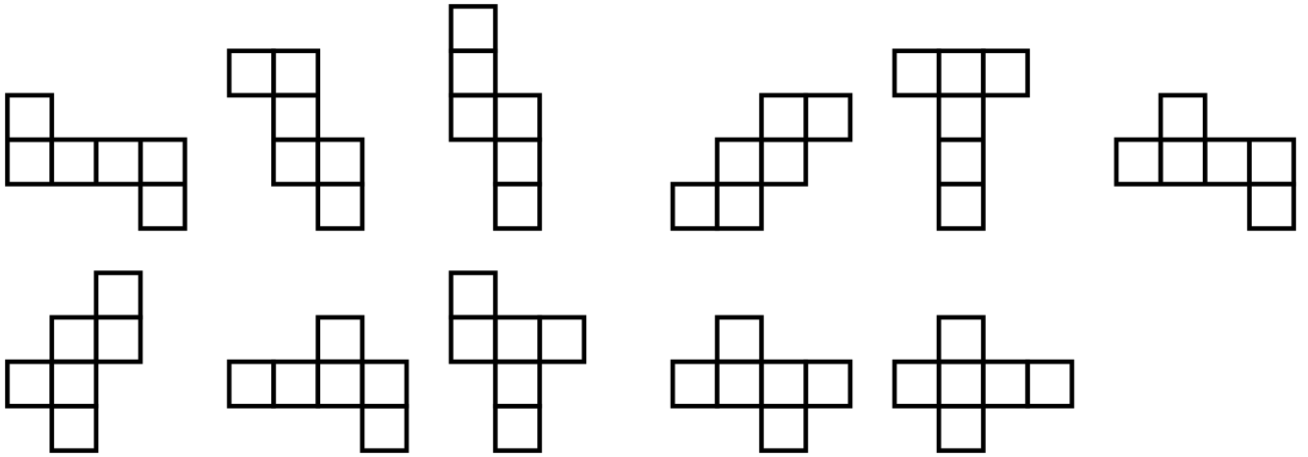
microfabrication | origami | programmable mat



Protein folding

Developments of polyhedra

11 ways for a cube

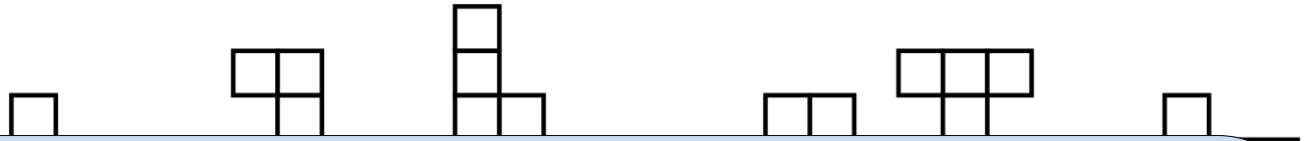


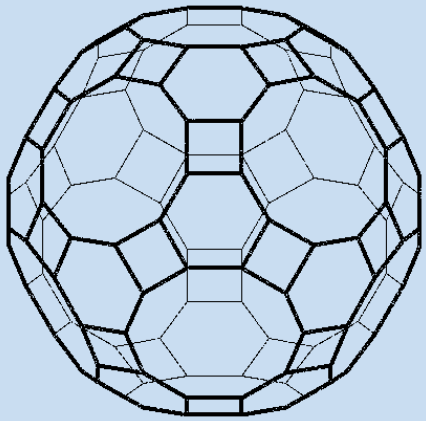
Q.2 How many ways for a soccer ball？

A. 3,127,432,220,939,473,920

Developments of polyhedra

11 ways for a cube





181,577,189,197,376,045,
928,994,520,239,942,164,480

A. 3,127,432,220,939,473,920

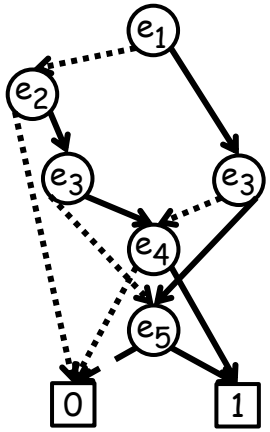
This class: From the viewpoint of discrete structures

- Support our society **by algorithms**
  (algorithms on enumeration/optimization and their complexity)
- Various connections with our society

Discrete structure

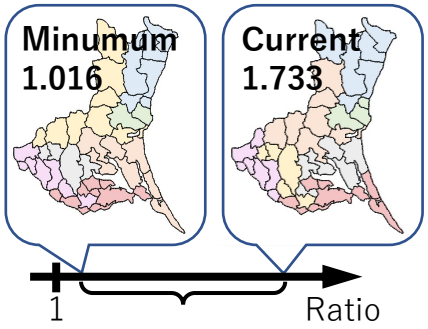
How to **represent/manipulate discrete structure** (combinations, graphs, and so on)

$\{ \{e_1 e_3 e_5\}, \{e_2 e_3 e_4\}, \{e_1 e_4\}, \{e_2 e_5\} \}$



Various applications

Electoral district



Evacuation plan

Computational origami

Origami with cells