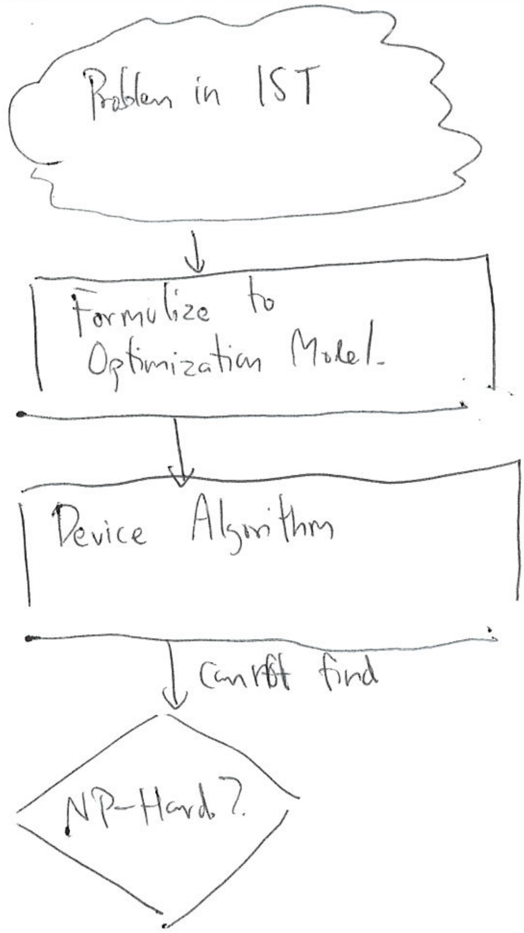
Approximation and Online
Algorithms with Applications,
#2

Last time

(1)



- Input
- Output
- Constraint
- Objective Function.

Idea: Noone will solve it ?!!! (Too hard to say).

(3)

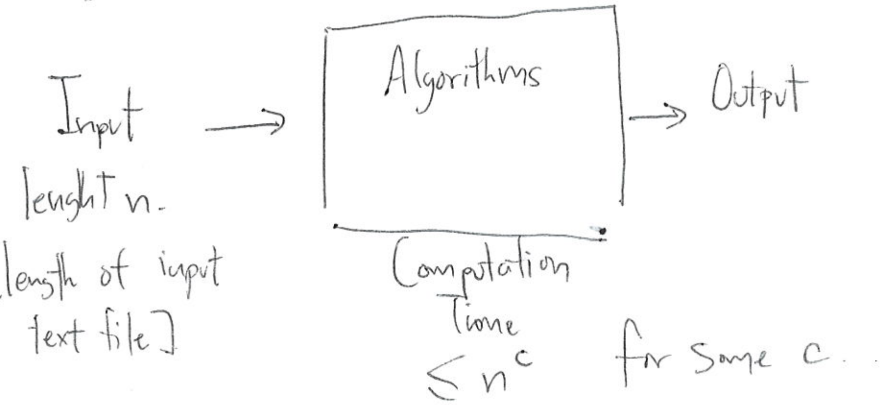
Many people have tried, but noone solved it.

(Easier to say, but you have just introduced the problem.).

---

(2)

Solved?



Definition

An optimization model is solvable, if there is an algorithm with running time $\leq n^c$ for some c and all input length n, Popular Problem

Many have tried, but failed

Our model

We have just proposed it.

If I cannot solve popular problem, then I cannot solve our model.

I am not likely to solve it, like many.

I and many are not likely to solve the model.

$$P \longrightarrow Q$$

P : I cannot solve popular problems

Q : I cannot solve our model.

If I can solve our model, I can solve popular problem

$$\neg Q \longrightarrow \neg P.$$

$\neg Q$: I can solve our model

$\neg P$: I can solve popular problem.

Difficulty Level

Our Model.

$\Uparrow$

Popular Problem

```
Output algorithm for OurModel(Input i1, Input i2, ..., Input ip) {
    \\ We don't know what is here but let assume
        that we can solve our model ~~the~~ by this code.
}

Output algorithm for PopularProblem(Input i1, Input i2, ..., Input iq) {
    \\ code for this popular problem
    \\ we can call the function "algorithmforOurModel()" here.


}
```

Popular Problem? : Satisfiability.

Input: $\cancel{S_1, \dots, S_p \in \{1, \dots, n\}}$ Boolean circuit 2 levels (first level $\vee$, second level $\wedge$)

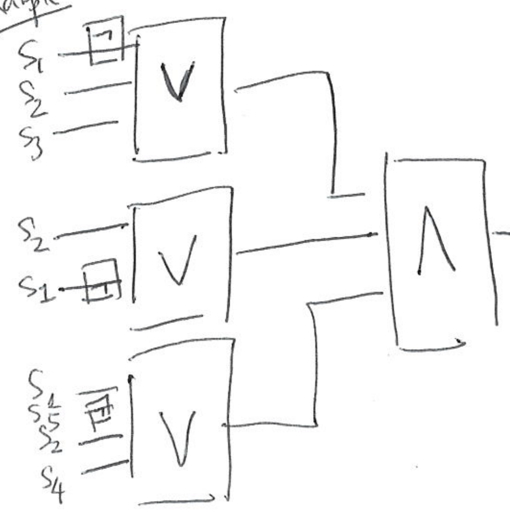
Output: Yes or No

Constraint: $\cancel{S_1, \dots, S_p \in \{}$

Yes, when there is an assignment to $S_1, \dots, S_p$ that makes the circuit output true.

No, otherwise.

Example



When S_2 = true circuit output = true.

Your model :

Input: ~~B~~ Same as satisfiability

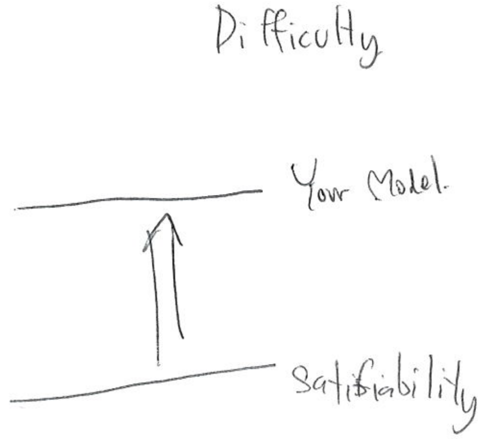
Output: $s_1, \dots, s_p \in \{\text{True}, \text{False}\}$.

Constraint: $s_1, \dots, s_p$ is an assignment that makes the circuit output true, if there exists any.

Assignment.

~~Output~~ algorithm for Model(Circuit C); → library.

```
Output satisfiability (Circuit C){
    s1,...,sp = algorithm for Model(C);
    If C(s1,...,sp) = true , return true;
    else return false;
}
```



The model is NP-Hard.

Definition NP-Hard problem is a problem ~~a~~ harder than Satisfiability.

Definition NP-Complete problem is when ~~a problem as hard as satisfiability.~~

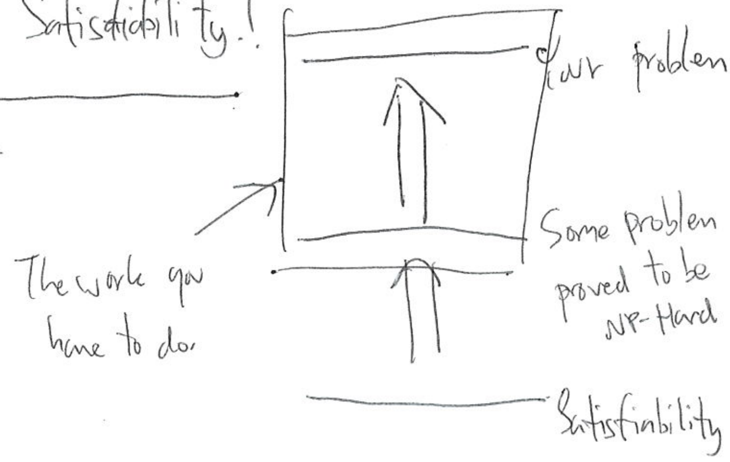
1) a problem is harder than satisfiability
2) satisfiability is harder than the problem

```
Output algorithm for Popular Problem (Input ... ) { → library
Output algorithm for Our Model (Input ... ) {
    Your code for solving our model

}
```

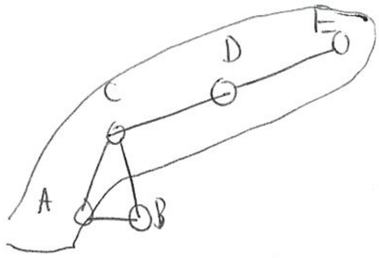
But, our problem is so different from Satisfiability!

~~A million~~ Use other NP-Hard problem.



Comprehensive (??) List of NP-Hard problem [Garey, Johnson
A guide to the theory of NP-completeness].

k-densest subgraph



k=

Input: Social Network, ~~k~~ integer k

Output: A set of ~~k~~ persons S.

Constraint: ~~None~~ $|S| = k$.

Objective Function: Maximize # relationship in S, $f(S)$

Example $k = 4$

$S = \{A, C, D, E\}$

$f(S) = 3$

$S = \{A, B, C, D\}$

$f(S) = 4$

Optimal output for this case is 4.

∴ Use to find a set of popular persons in social networks.

k-clique

Input: A social network, integer k.

Output: Yes or No

Constraint: Yes : If there is a set of persons S such that

1: $|S| = k$

2: All pairs of person in S can communicate together

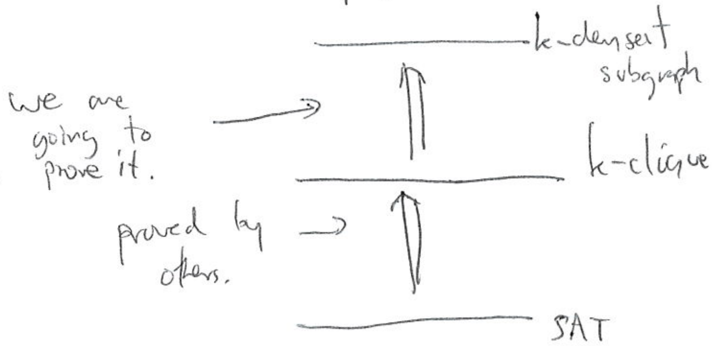
No : Otherwise

Example: $k=3 \rightarrow$ Yes. ($S = \{A, B, C\}$)

$k=4 \rightarrow$ No

Proved to be an NP-Hard problem!

If we can solve k-densest subgraph (our model), we can solve k-clique (popular model)




```
Sets kDensestSubgraph (Network N, int k); → library.
boolean kCliqee (Network N, int k) {
    S = kDensestSubgraph (N, k);
    if (All pairs of person in S can communicate) → Then, S is a set that satisfies 2 conditions
        return true;
    else return false;   ← If there is a set that everyone can communicate, that should maximize communication.
}
```

Product Selection Problem [Xu and Lui KDD 2014]

- We are a notebook company.
- We want to have k more products choosing from n products proposed by developer.
- We found that, when a customer chooses a notebook, they will have 3 requirement, weight, CPU, and harddisk space.
- They will buy a cheapest notebook that satisfies the requirement.
- Our competitors currently have m products.

Ex $k = 2$

Candidate Products,

A (2.0 kg, 1.5 MHz, 1TB) – 70,000 Yen

B (1.3 kg, 1.5 MHz, 500 GB) – 100,000 Yen

C (1.3 kg, 1.8 MHz, 1TB) – 170,000 Yen

Customer.

α ($\leq$ 1.4 kg, $\geq$ 1.6 MHz, $\geq$ 500 GB)

β ($\leq$ 3 kg, $\geq$ 1 MHz, $\geq$ 1TB)

γ ($\leq$ 2 kg, $\geq$ 1.5 MHz, $\geq$ 1TB).

η ($\leq$ 2 kg, $\geq$ 1 MHz, $\geq$ 500 GB)

Competitor Products

D (1 kg, 1.8 MHz, 2 TB) – 300,000 Yen

E (2.5 kg, 1.5 MHz, 500 GB) – 50,000 Yen

F (0.9 kg, 0.7 MHz, 200 GB) – 100,000 Yen

If we choose A and B

$\alpha \to \{D\} \to D.$ $\quad\eta \to \{A, B, D, E\} \to E$

$\beta \to \{A, D\} \to \boxed{A}$

$\gamma \to \{A, D\} \to \boxed{A}$ $\longrightarrow$ 2 customers.

If we choose A, C.

$\alpha \rightarrow \{C, D\} \rightarrow C$

$\beta \rightarrow \{A, C, D\} \rightarrow \cancel{C}A$ $\quad \rightarrow$ 3 customers.

$\gamma \rightarrow \{A, C, D) \rightarrow A$

$\eta \rightarrow \{A, B, D, E\} \rightarrow E$

$\therefore$ A, C is the best choices possible

<u>Optimization Model</u> : k- Most Marketable Products (k-MMP)

<u>Input</u>: # candidates n, # new products k

$P_1, \dots, P_n$ when P_i is qualification of Product i (3-d vectors)

competitors m.

$Q_1, \dots, Q_m$ when Q_i is qualification of Competitor i (3-d vectors)

customers d.

$C_1, \dots, C_d$ when C_i is qualification required by (3-d vectors) Customer i.

Product Cost :

$c_1, \dots, c_n \in \mathbb{R}_{>0}$: c_i is a cost for P_i.

$c'_1, \dots, c'_m \in \mathbb{R}_{>0}$: c'_i is a cost for Q_i

<u>Output:</u> $S \subseteq \{1, \dots, n\}$ (new products)

<u>Constraint:</u> $|S| = k$.

Objective Function :

$$f_P(C_i) = \min_{\substack{\cancel{C_i \le P_j} \\ P_j \ge C_i \text{ and } j \in S}} c_j$$ (cost that customer i has to pay for our product)

$$f_Q(C_i) = \min_{Q_j \ge C_i} c'_j$$ (cost that customer i has to pay for competitors)

~~Minimize # of~~

Maximize # C_i such that $f_P(C_i) \le f_Q(C_i)$.

Popular Problem : Top k-representative Skyline Product. [Lin, Yuan, Zhang, Zhang, ICDE 2007]

k-MMP

$\Downarrow$

k-MMP without competition

Input : # candidates n, # new products k

$P_1, \ldots, P_n$ when P_i is a ~~d~~ 3-dimensional vector.

customers d

$C_1, \ldots, C_d$ when C_i is a 3-dimensional vector.

Output : $S \subseteq \{1, \ldots, n\}$

Constraint : $|S| = n$.

Objective Function : Maximize # C_i such that $C_i \le P_j$ for some $j \in S$.

Our problem when there is <u>no competitor.</u>

$$f_P(C_i) = \min_{P_j \geq C_i \text{ and } j \in S} c_j$$

$< \infty$ if there is $j \in S$ such that $P_j \geq C_i$

$$f_Q(C_i) = \min_{Q_j \geq C_i} c_j' = \infty$$

$f_P(C_i) \lneq f_Q(C_i)$ when there is $j \in S$ such that $P_j \geq C_i$

<u>Maximize $\# C_i$ $f_P(C_i) < f_Q(C_i)$</u> $\Longleftrightarrow$

Our model

<u>Maximize $\# C_i$ such that there is $j \in S$ where $P_j \geq C_i$</u>

classical model

Sets k-MMP (int n, int k, Vectors P, int m, Vectors Q, int d, ~~Cost~~ Vectors C, doubles cost Products, doubles cost Competitors);

Sets Top-k (Int n, int k, Vectors P, int d, Vectors C) {
return k-MMP (n, k, P, 0, [], d, C, (random Arrays), (random Arrays));

}

$\therefore$ k-MMP is an NP-Hard problem.