

16주차 세션

김나현, 이은빈

목차

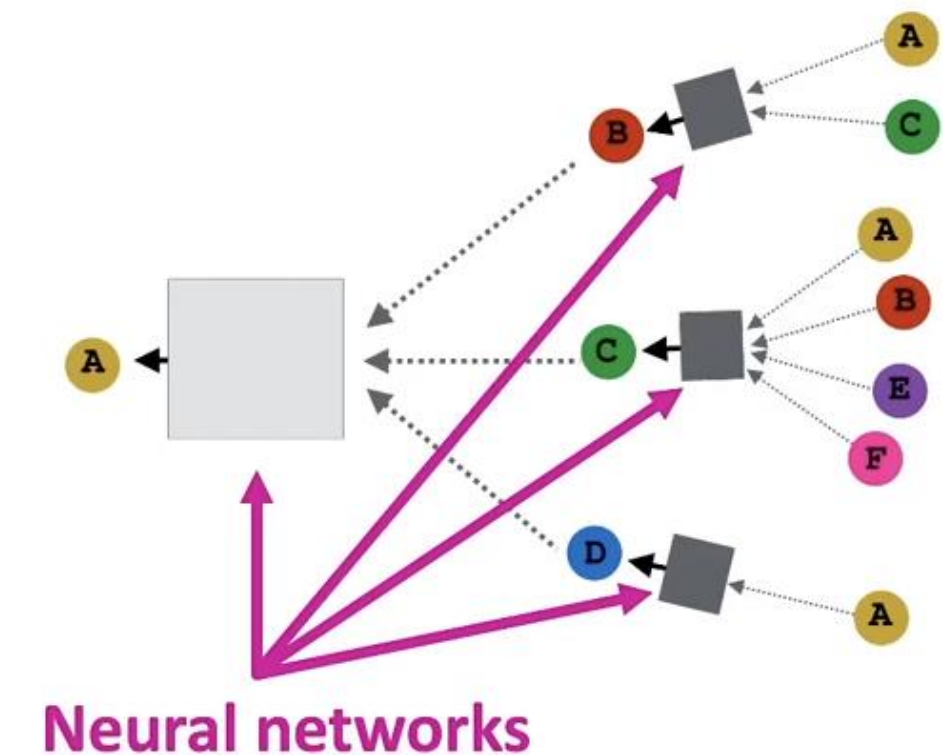
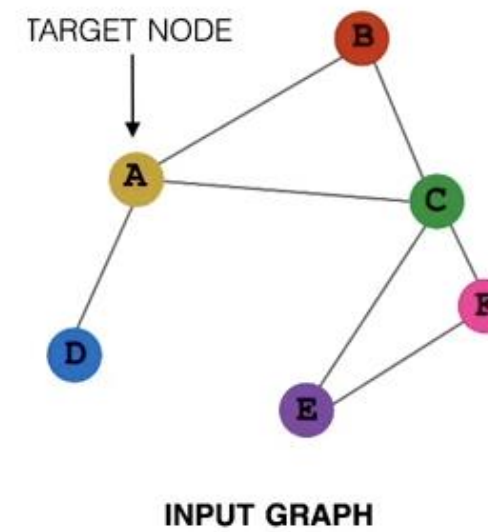
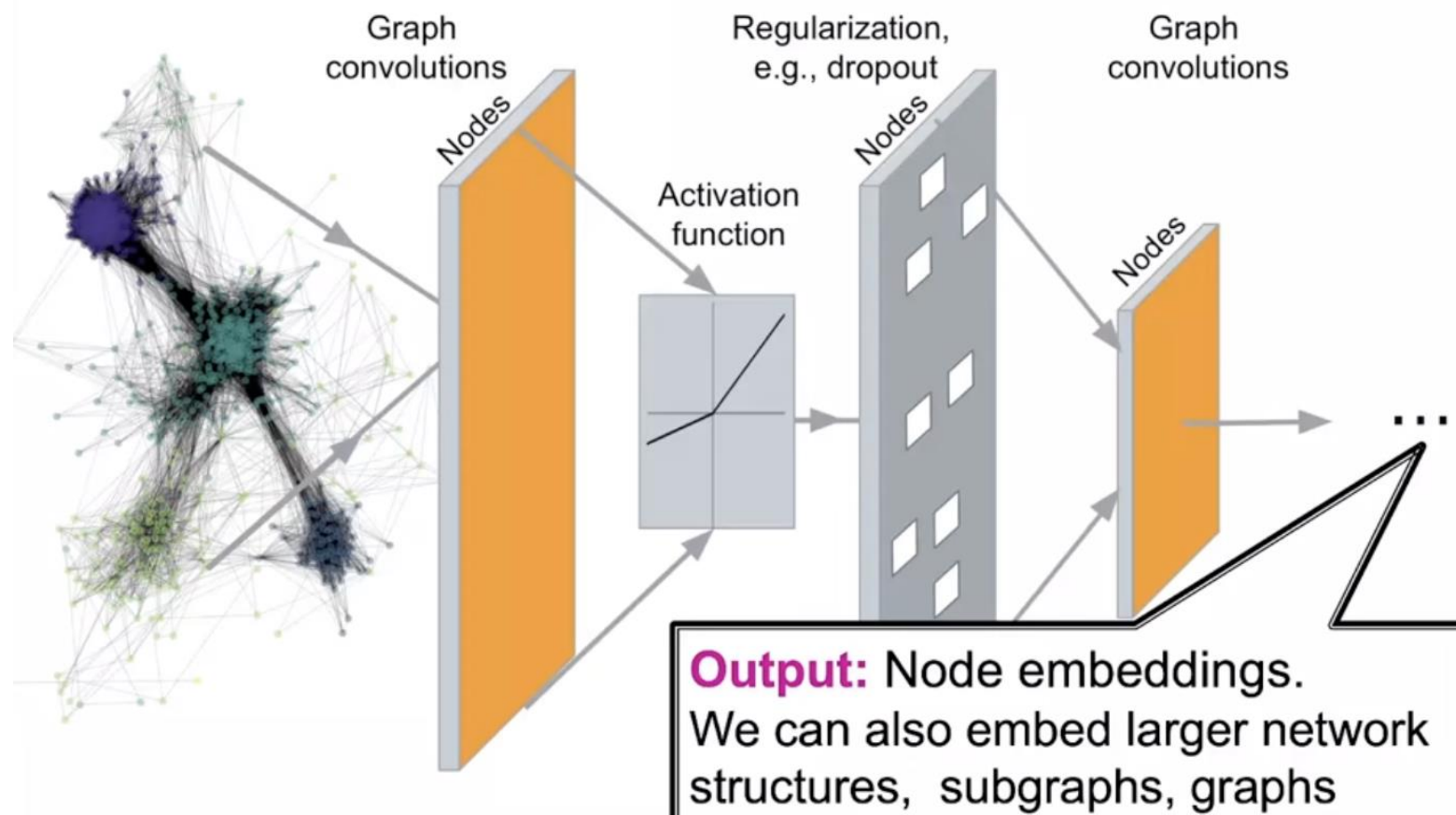
#01 How Expressive are Graph Neural Networks

#02 Designing the Most Powerful Graph Neural Networks



Recap : Graph Neural Networks

- Key idea of GNN : **Aggregate Neighbors**
 - We are trying to come up with the embedding of nodes, embedding of networks
 - node의 Local network neighborhood에 기반하여 Aggregation을 통해 node Embedding 생성
 - node는 Neural Network를 사용하는 neighbors의 information을 aggregate

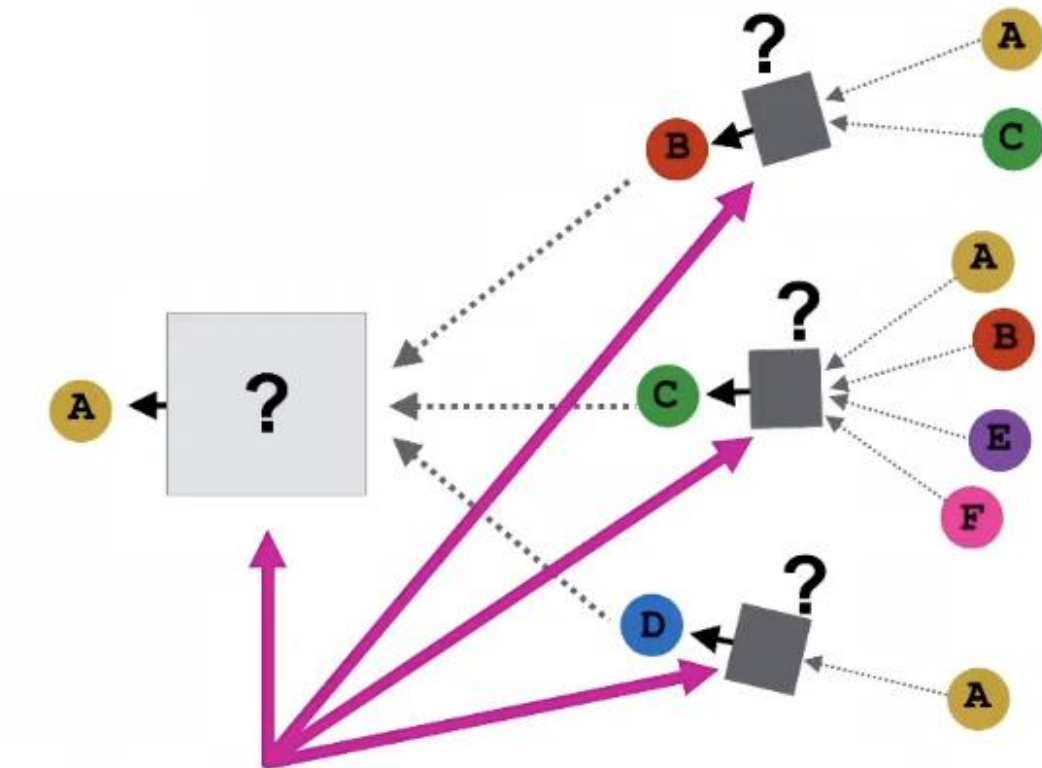
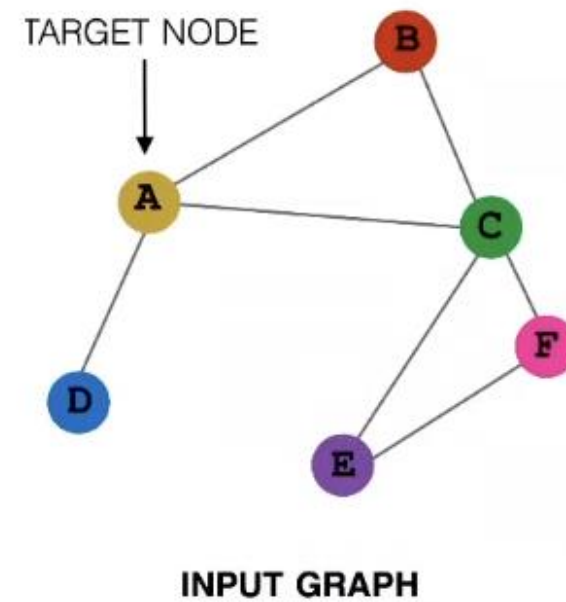


Theory of GNNs

- How powerful are GNNs?
 - GNN 기반의 많은 모델들이 제안되고 있다 (e.g., GCN, GAT, GraphSAGE, design space)
 - GNN의 expressive power는 어디에서 오는가? (node와 graph structure를 어떻게 구분하는가?)
 - 어떻게 GNN의 표현력을 극대화할 수 있는가?

Background : Many GNN Models

- Many GNN models have been proposed :
 - GCN, GraphSAGE, GAT, Design Space etc.

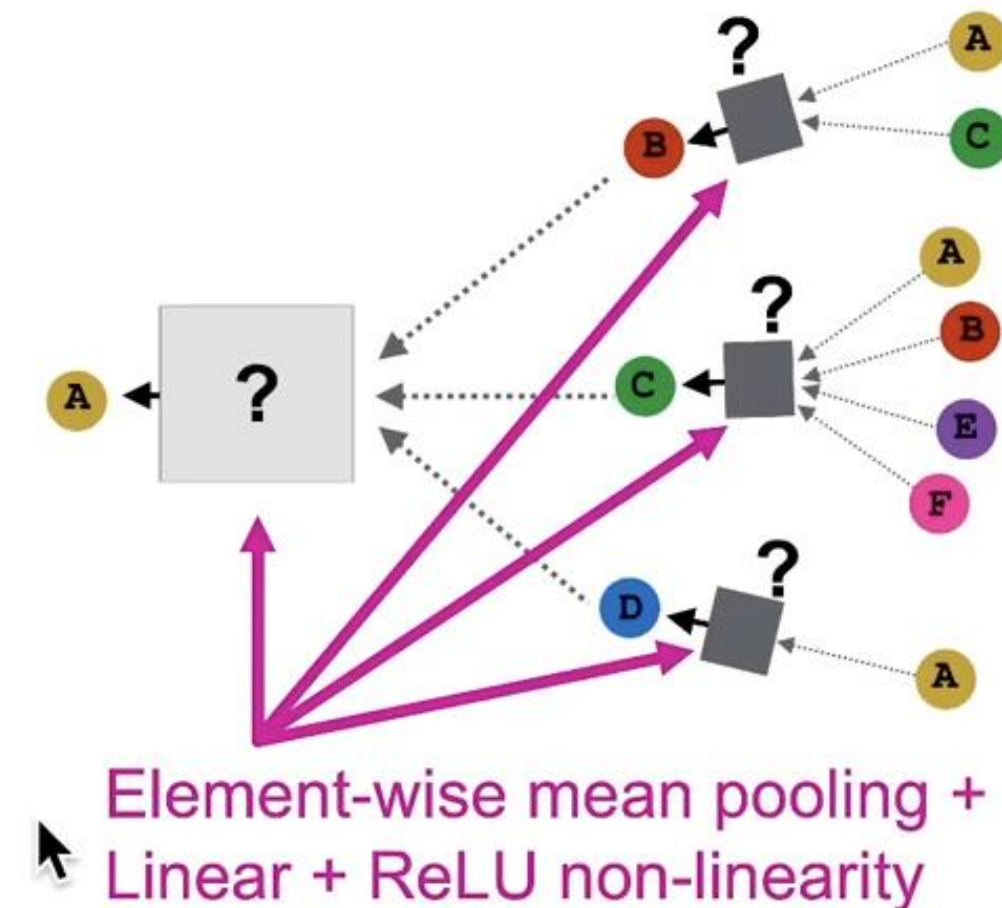
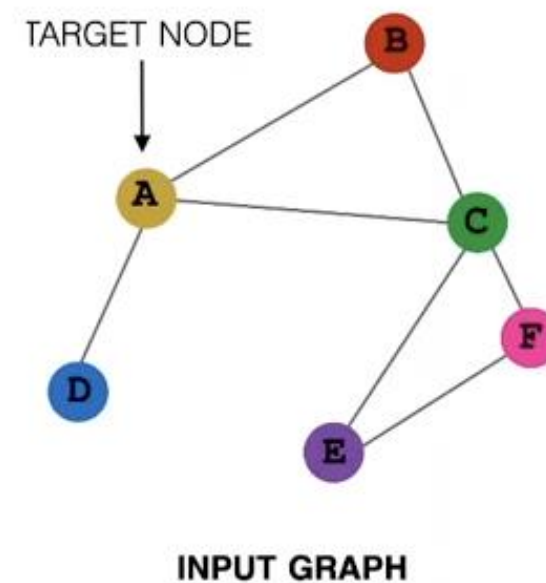


Different GNN models use different neural networks in the box

GNN Model Example (1) : GCN

- GCN (mean-pool) [Kipf and Welling ICLR 2017]

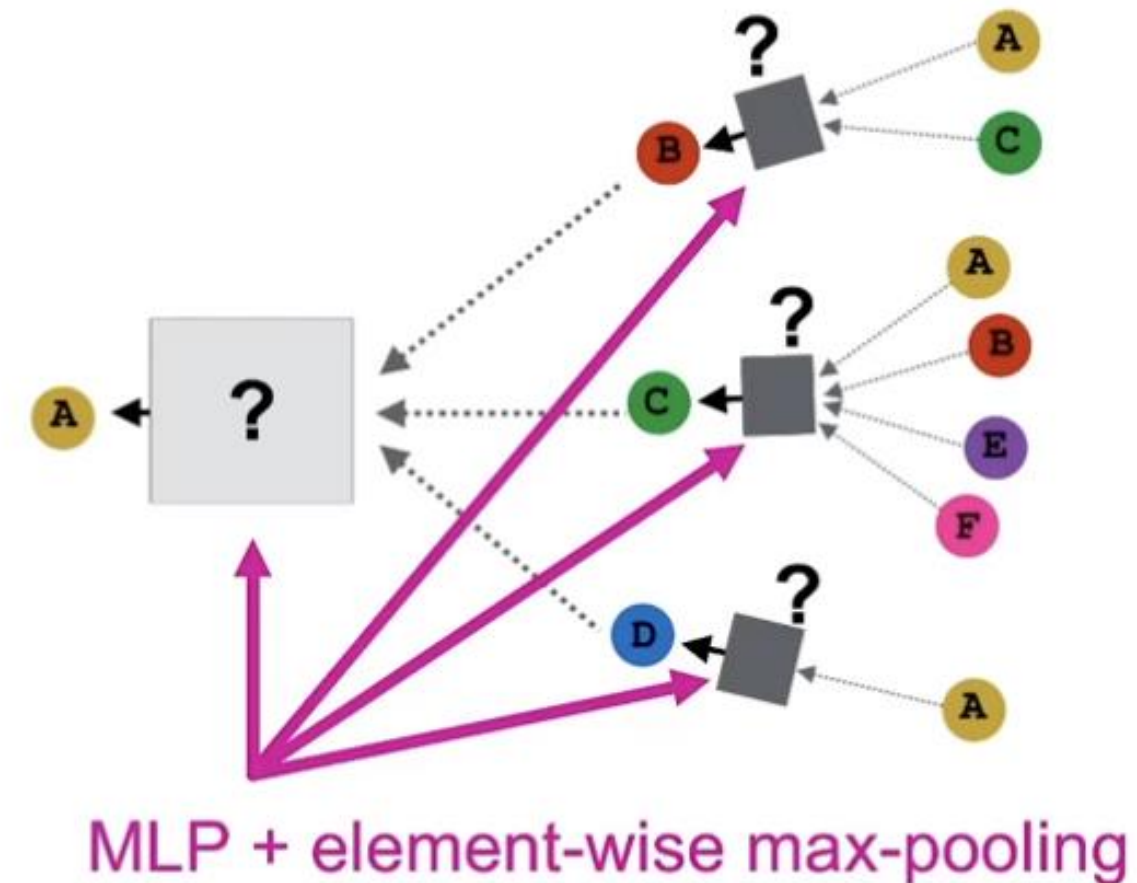
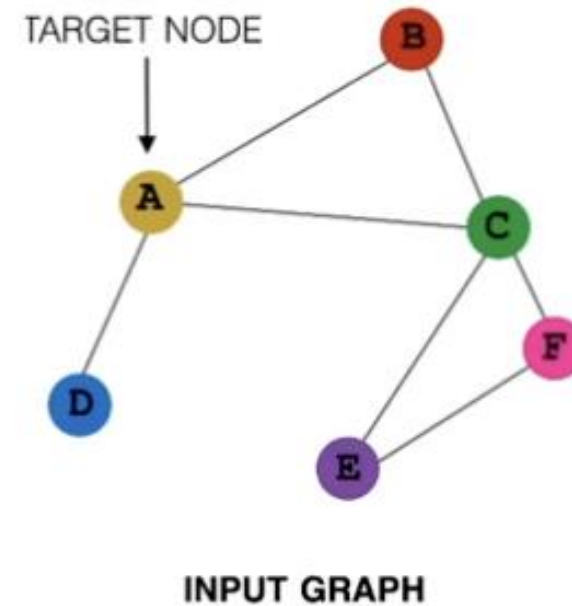
- Element-Wise Mean Pooling으로 Aggregation
- Linear + Relu non-linearity



GNN Model Example (2) : GraphSAGE

- GraphSAGE (max-pool) [Hamilton et al. NeurIPS 2017]

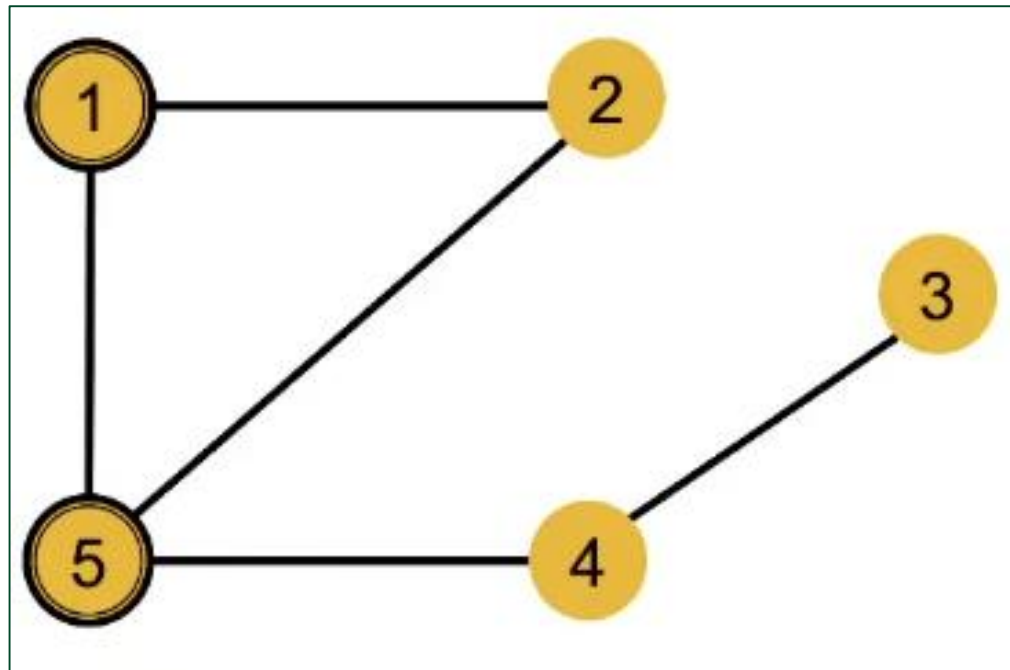
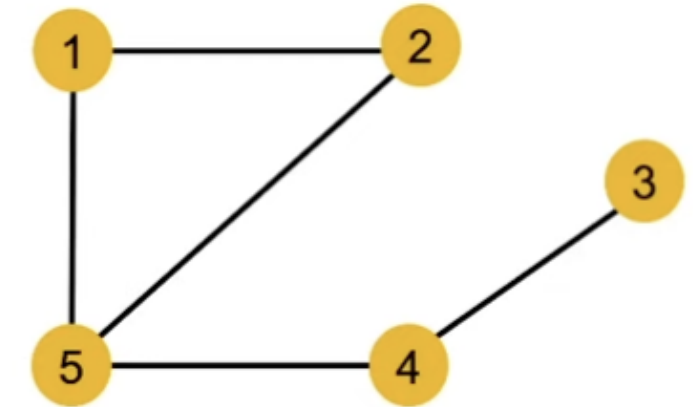
- Element-Wise Max Pooling으로 Aggregation
- 이후 MLP 적용



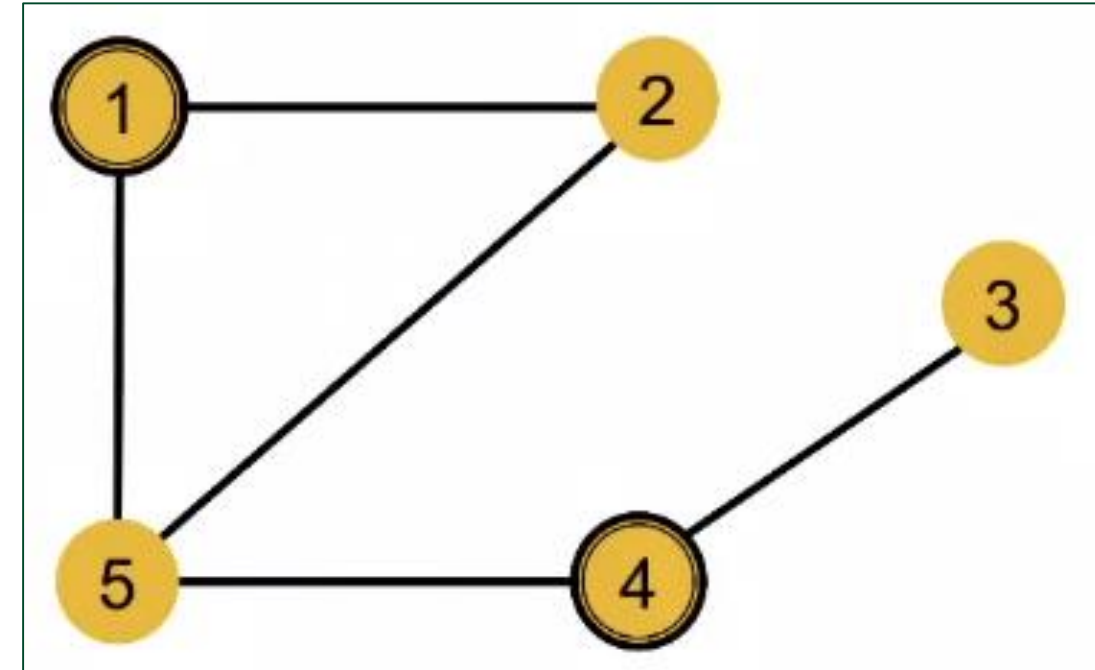
Local Neighborhood Structures

- **Local Neighborhood Structure**

- 모든 node가 같은 feature를 가지고 있는 graph에서 node를 구별하는 방법



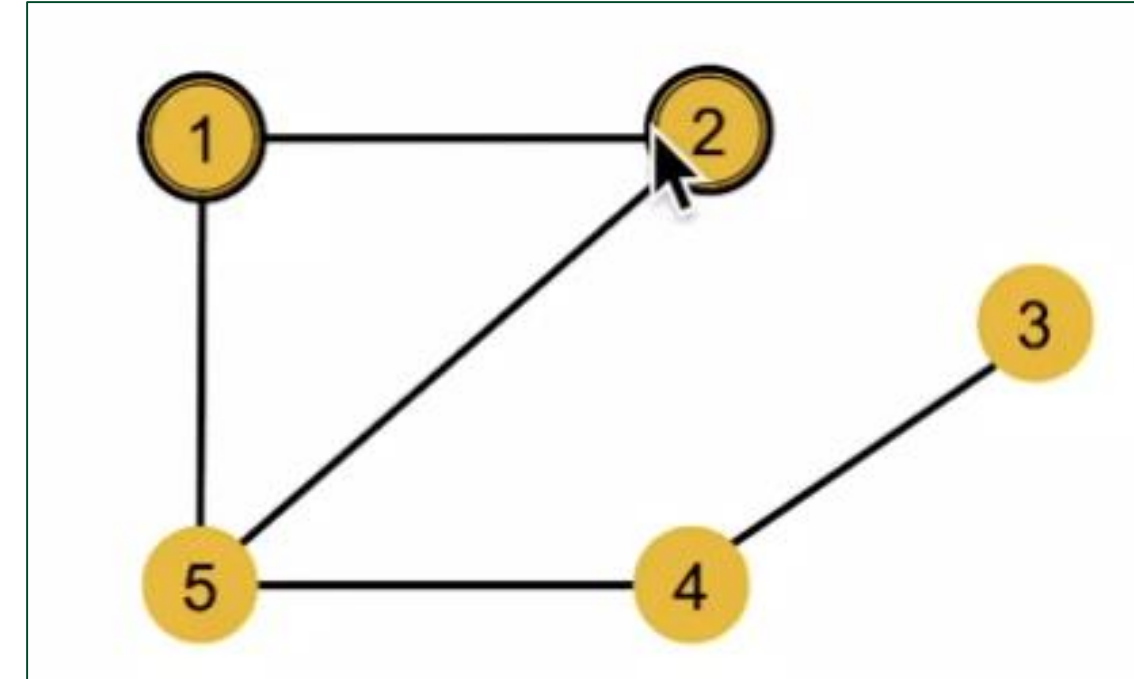
Node 1과 5는 서로 다른 node degree이기 때문에
different neighborhood structure를 가짐



Node 1과 4는 동일한 node degree이지만
Neighbor들이 서로 다른 node degree이기 때문에
different neighborhood structure를 가짐

Local Neighborhood Structures

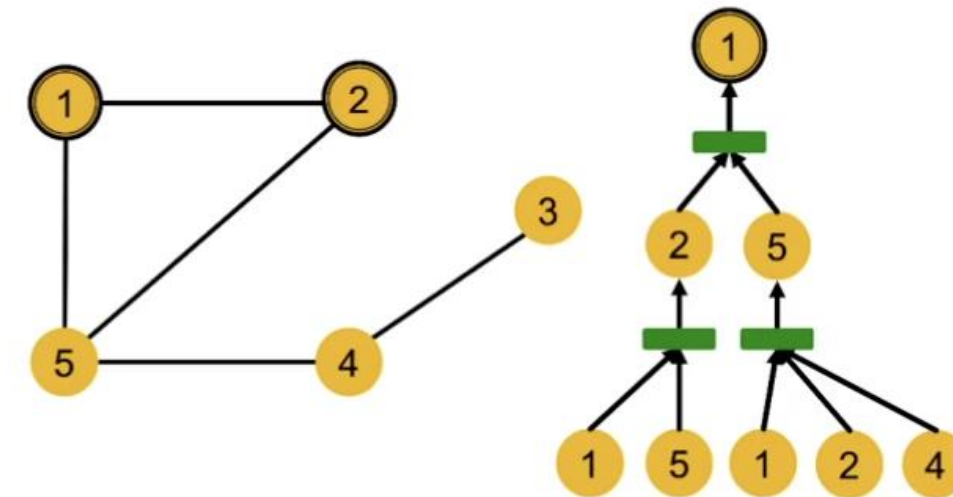
- However...
- Node 1과 2는 그래프 상에서 symmetric
- They have same neighborhood structures



- Questions..
- GNN의 node embedding은 서로 다른 node의 local neighborhood structure를 구분할 수 있는가?
- 어떤 조건에서 가능/불가능한가?

Computational Graph

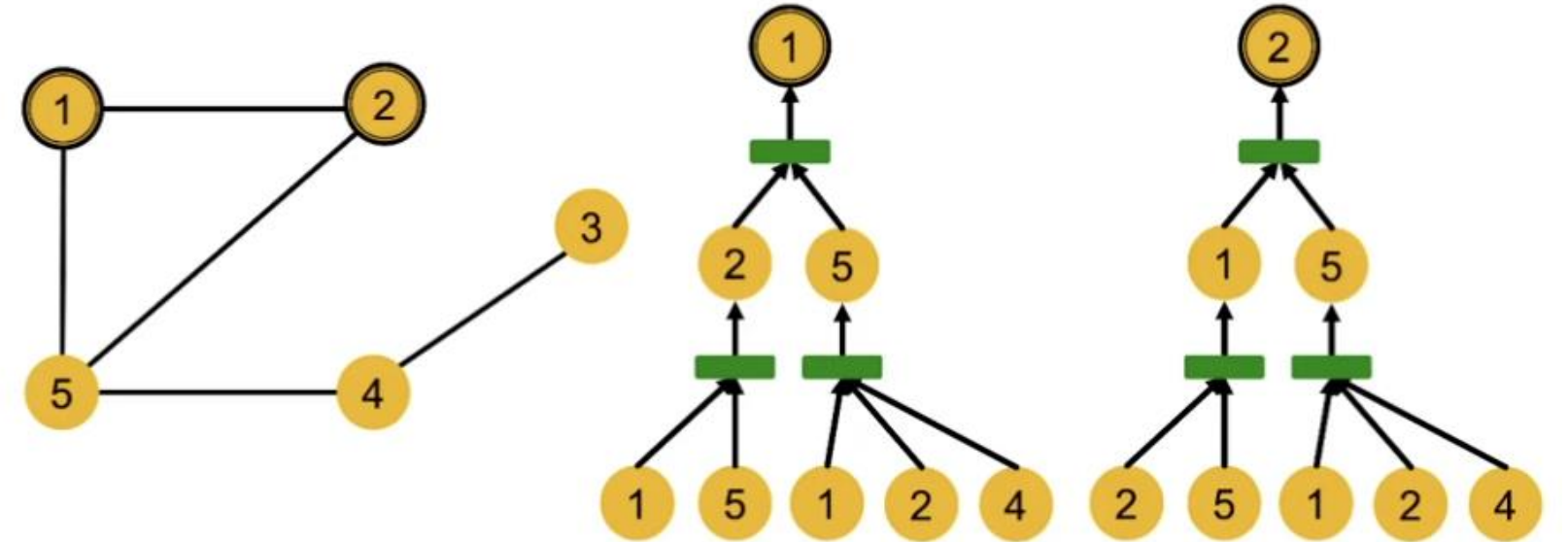
- Key Concept : We need to understand how a GNN captures Local Neighborhood Structures
- Node 1의 computational graph (2 layer GNN)
 - Node 1은 Node 2, 5로부터 information aggregate
 - Node 2는 Node 1, 5로부터 information aggregate



⇒ GNN은 neighbor에 의한 Computational Graph로부터 node embedding 생성

Computational Graph

- Node 1, 2의 computational graph (2 layer GNN)
- GNN은 node feature만 확인 (not IDs)
- 동일한 feature를 가진 Graph가 같은 모양으로 연결

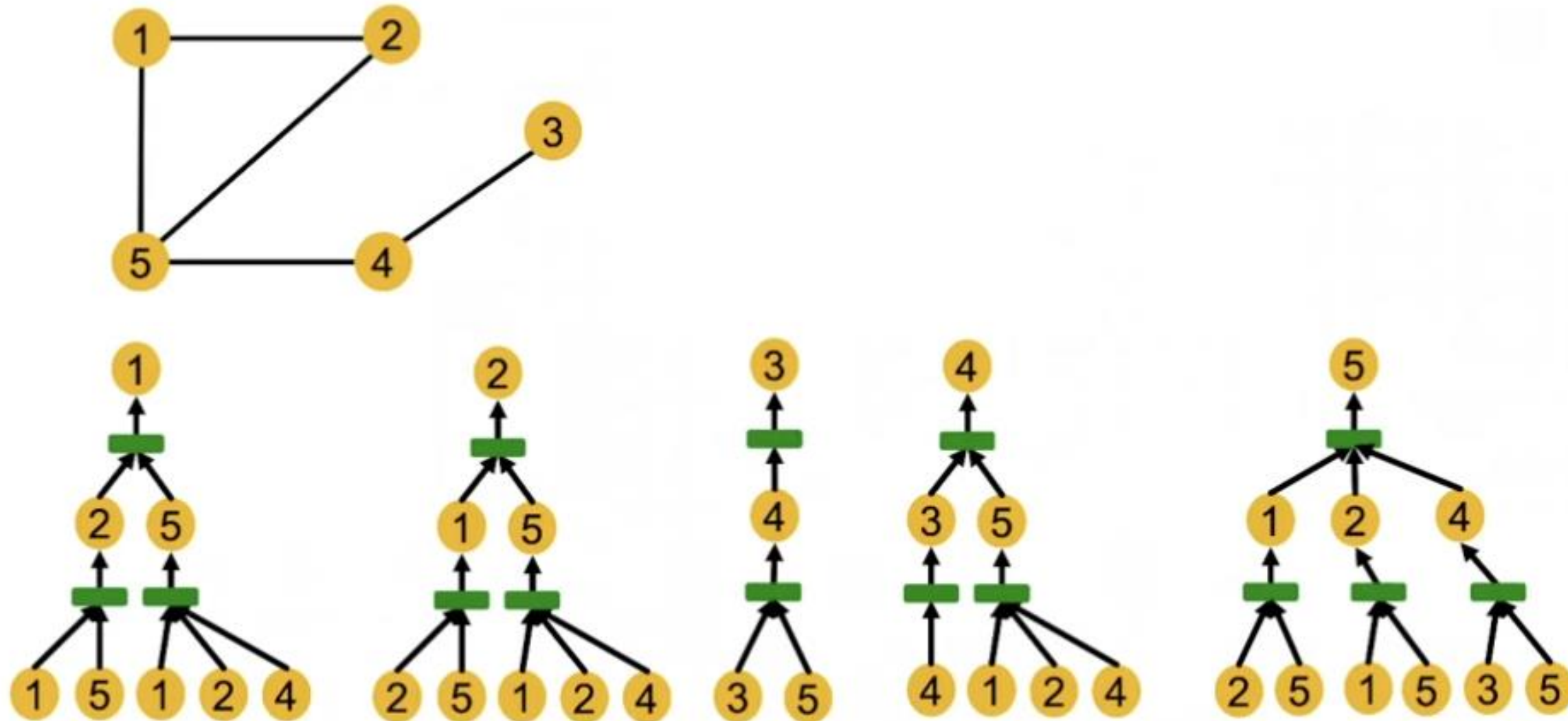


- Computational Graph가 동일하고 Node feature가 identical하기 때문에
- Node 1, 2에 대해서 각각 동일한 embedding을 생성하게 될 것이다
- GNN은 node1과 2를 구별하지 못할 것이다

Computational Graph

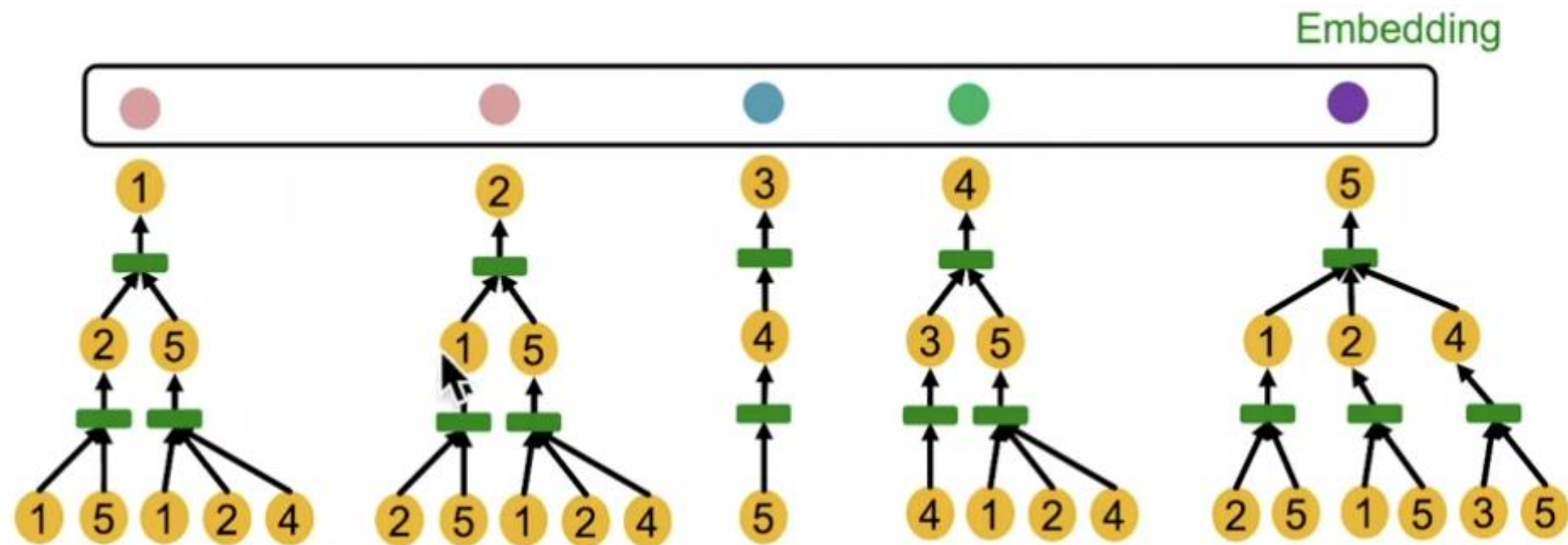
- 일반적으로 서로 다른 local neighborhoods는 서로 다른 computational graph를 정의한다

(앞의 Node 1, 2의 경우는 GNN의 한계, 이외의 경우에는 잘 구별 가능)



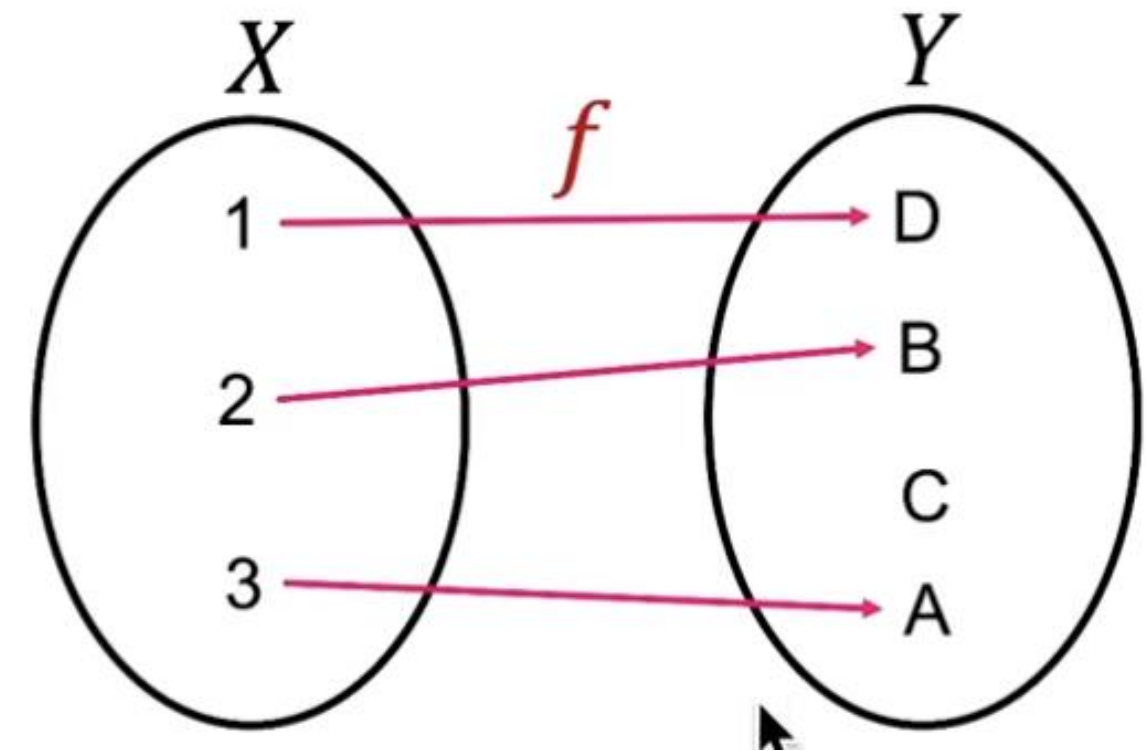
Computational Graph

- Computational Graph는 각 Node의 Rooted Subtree Structure에 따라 결정된다
- Expressive GNN은 다른 rooted subtree를 다른 Node Embedding으로 매핑



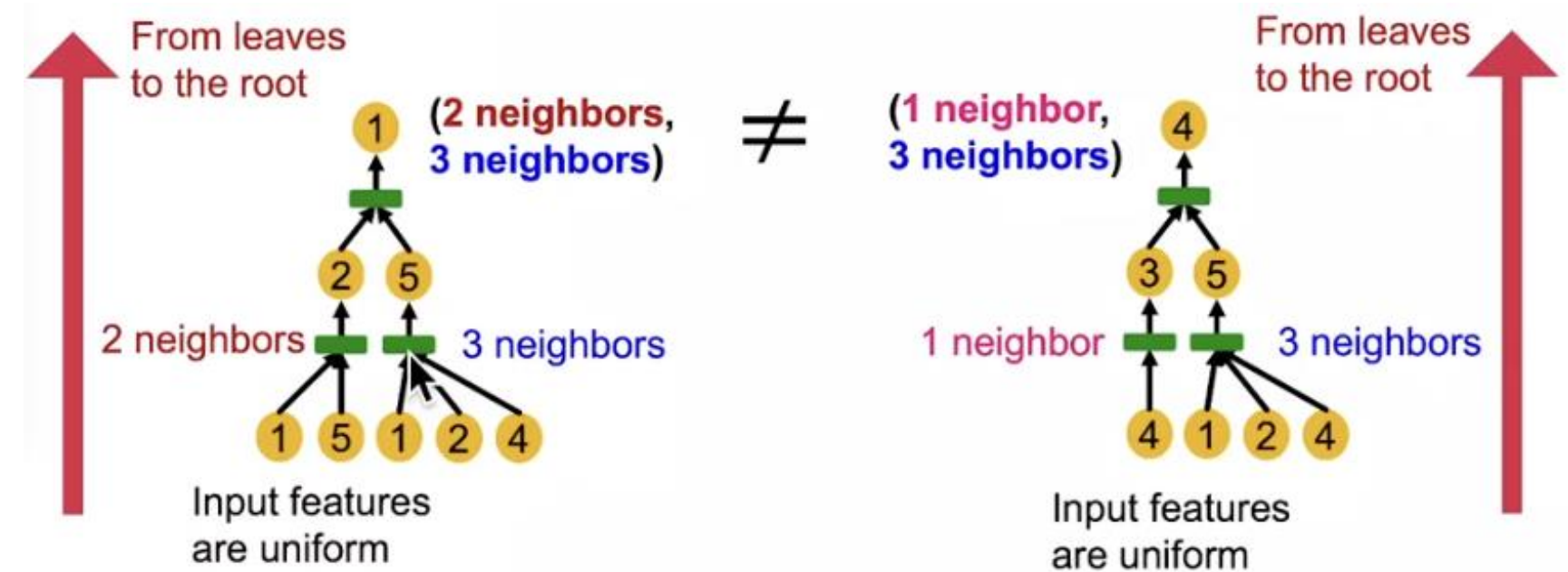
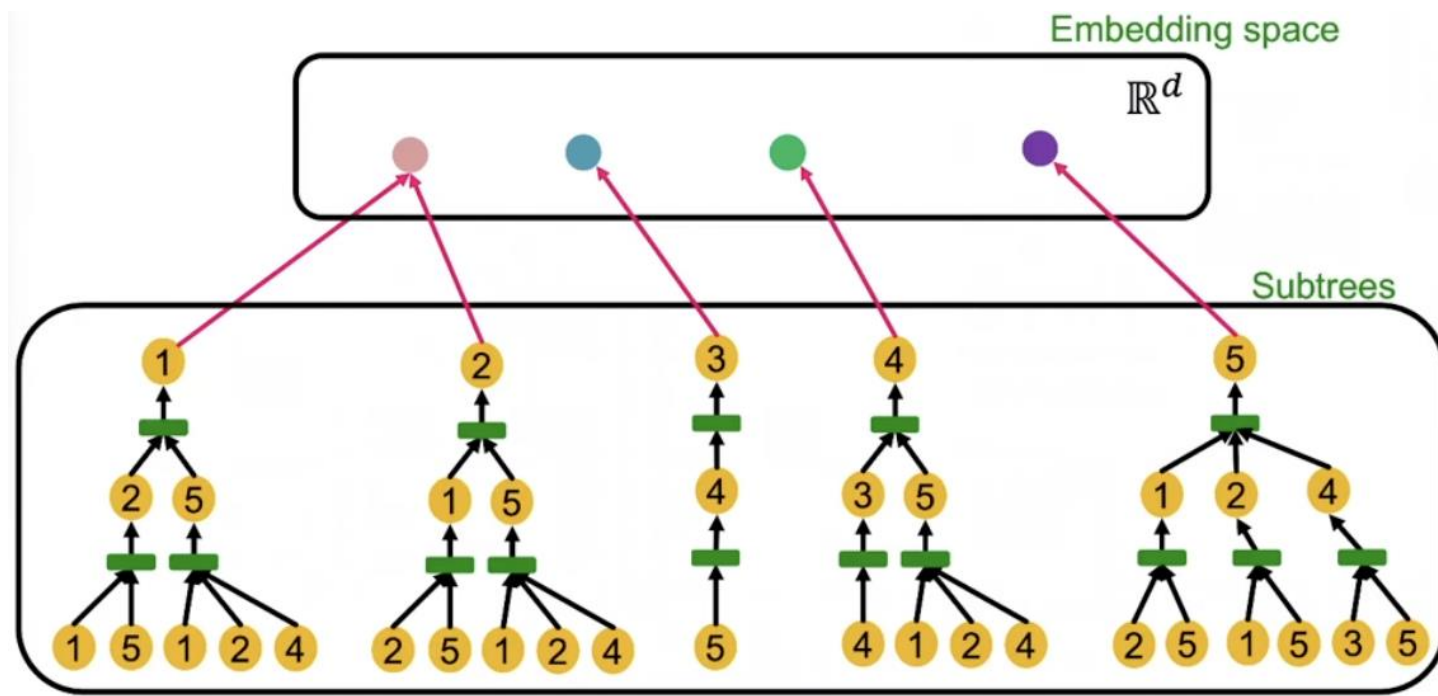
(Recall) Injective Function

- Injective Function (일대일 함수)
 - $f : X \rightarrow Y$
 - 서로 다른 x 에 대하여 서로 다른 y 가 매핑,
 - 즉 일대일 대응일 경우 injective 만족



Injective in GNN

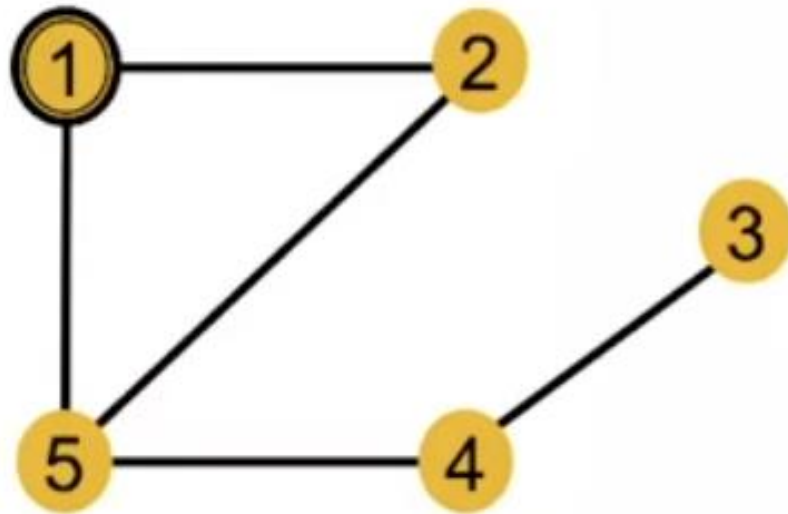
- Expressive GNN은 subtree를 node embedding으로 mapping할 때 **injective 만족**
- 같은 depth의 subtree일 때 root node 뿐 아니라 각 depth에서도 neighborhood structure에 따라 다른 embedding을 갖도록 함
- Aggregation 함수가 injective를 만족하면 ==> 강력한 GNN



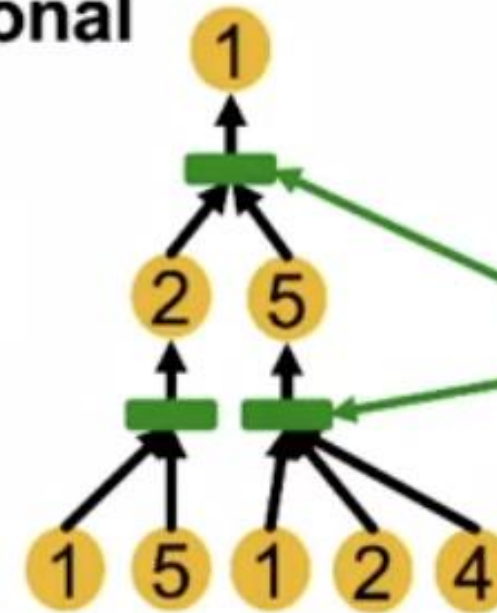
Summary so far

- GNN은 Node embedding을 생성하기 위해 각 노드의 subtree rooted에 Computational graph 사용
- GNN은 각 step에서 neighbor aggregation이 injective할 때 충분히 다른 subtree 구조를 구별할 수 있다

Input graph



Computational graph
= **Rooted subtree**



Using injective
neighbor
aggregation
→ distinguish
different
subtrees

Designing the Most Powerful Graph Neural Networks



Expressive Power of GNNs

- GNN의 표현력

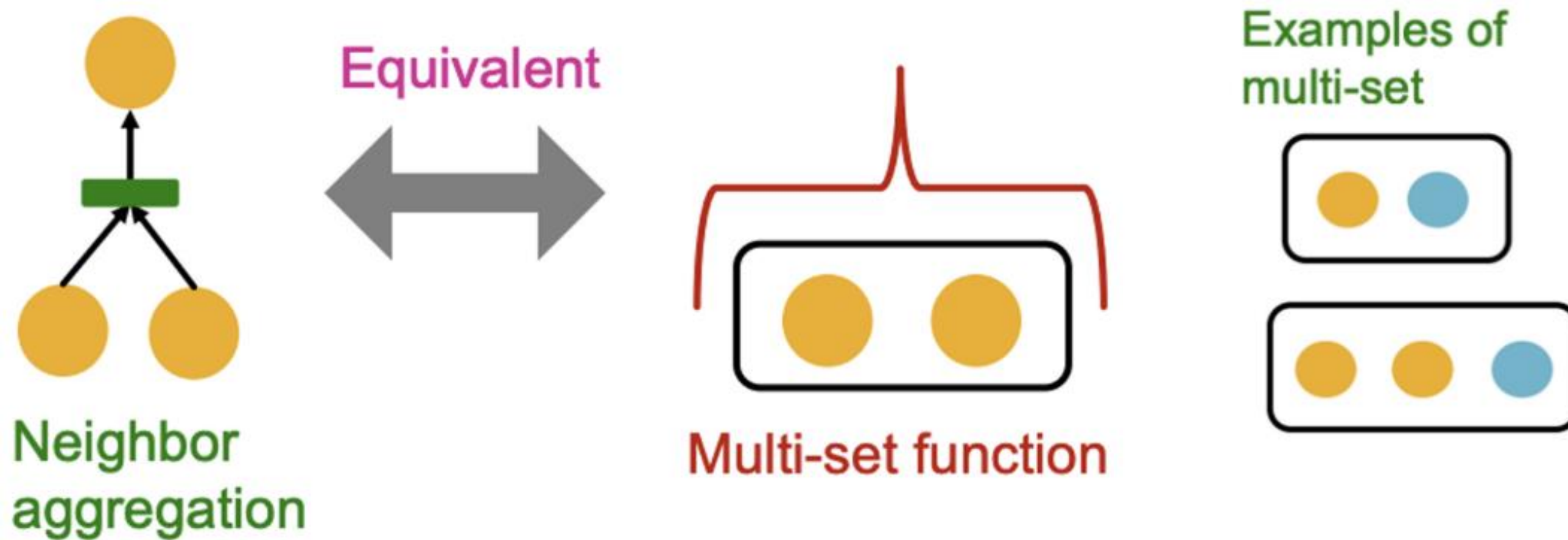
- ☞ 어떤 Neighbor aggregation 함수를 사용하는지

- expressive aggregation function \rightarrow expressive GNN

- Injective aggregation function

- ☞ 가장 강한 표현력을 가진 GNN

Neighbor Aggregation



Definition 1 (Multiset). A multiset is a generalized concept of a set that allows multiple instances for its elements. More formally, a multiset is a 2-tuple $X = (S, m)$ where S is the *underlying set* of X that is formed from its *distinct elements*, and $m : S \rightarrow \mathbb{N}_{\geq 1}$ gives the *multiplicity* of the elements.

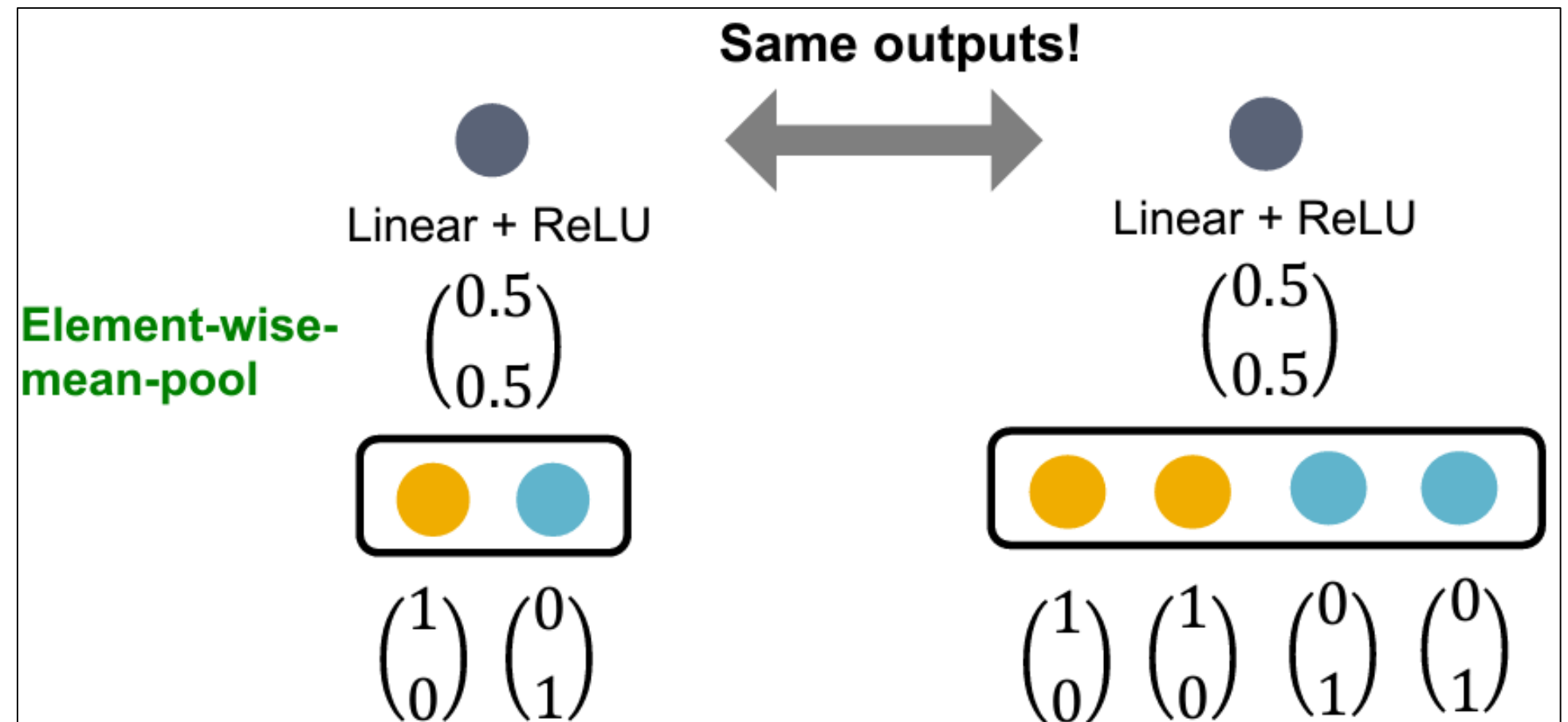
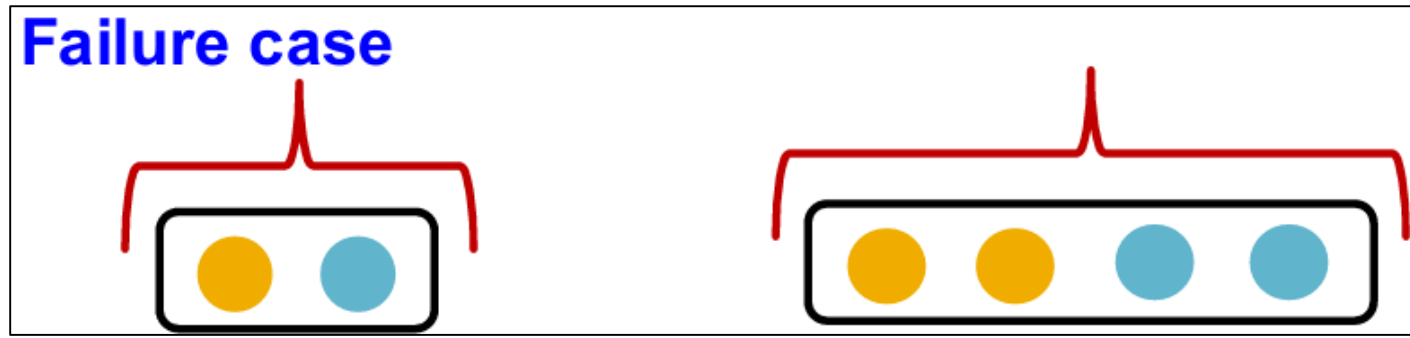
Neighbor aggregation \Rightarrow multiset에 대한 aggregation function

GNN 모델이 강력한 representational power를 갖기 위해서는,
다른 multiset들을 서로 다른 representation으로 aggregate 해야 한다.

GCN (mean-pool)

$$\text{Mean}(\{x_u\}_{u \in N(v)})$$

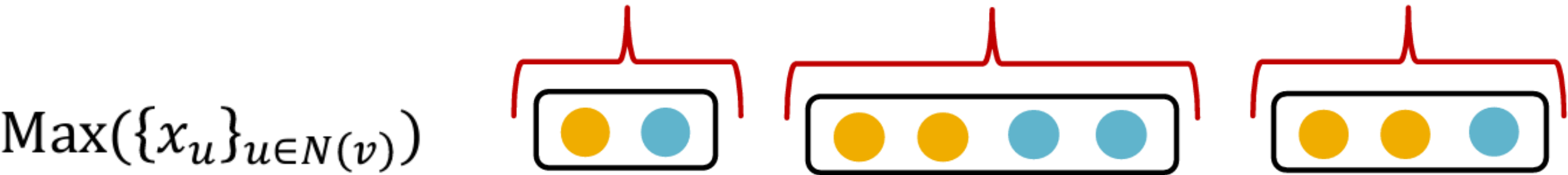
Failure case



- multi-sets with the same color proportion 가진 경우 → 구별 못함.
- element-wise mean 취하면 결과적으로 average가 같아짐

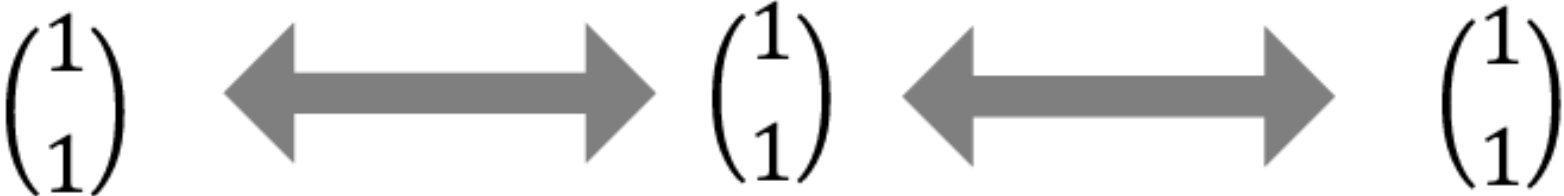
GraphSAGE (max-pool)

Failure case



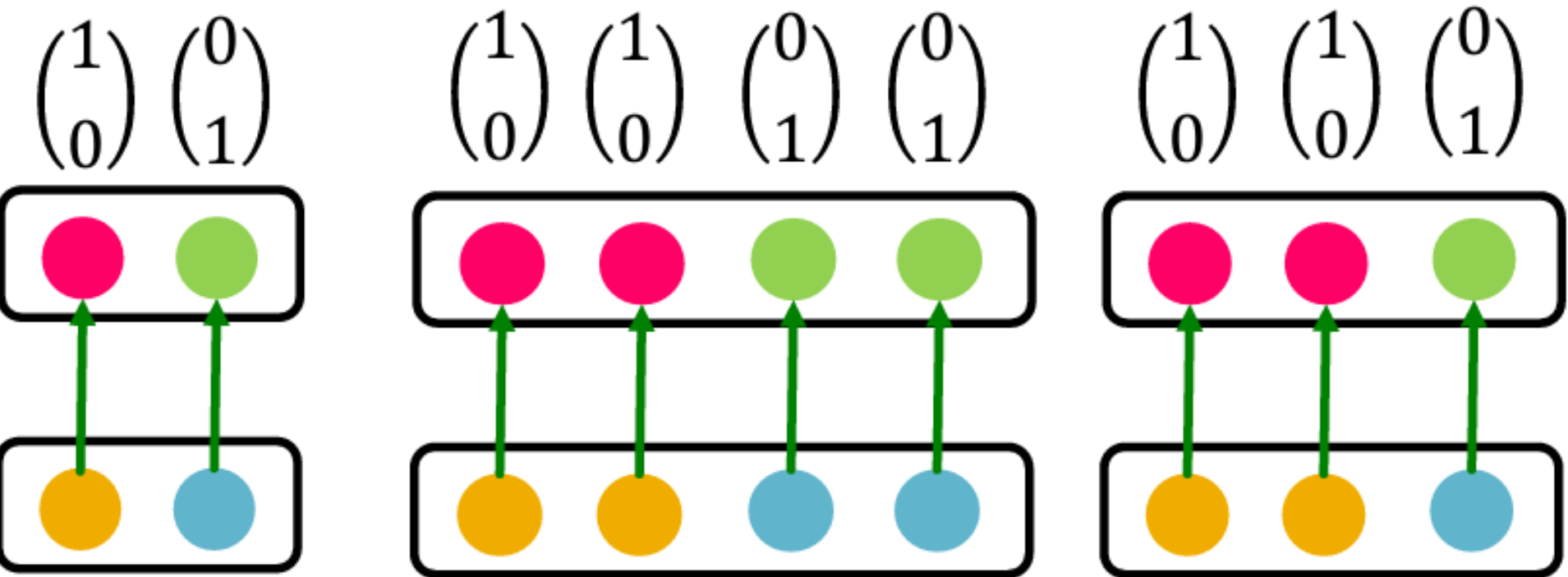
The same outputs!

Element-wise-max-pool



For simplicity, assume the one-hot encoding after MLP.

MLP



- # of children
 - color ratio
- 에 상관없이, 동일한 representation 낳는다.

GCN & GraphSAGE

- GCN 과 GraphSAGE 는 **NOT injective**
 - ↳ 서로 다른 inputs 이더라도 동일한 output 으로 맵핑됨
 - ↳ 정보 유실 발생
 - ↳ not maximally powerful GNNs.
- 어떤 aggregation function을 활용해야
Injective 하면서 표현력 높은 GNN을 만들 수 있을까?

Injective Multi-Set Function

Theorem [Xu et al. ICLR 2019]


Any injective multi-set function can be expressed as:

Some non-linear function $\rightarrow \Phi \left(\sum_{x \in S} f(x) \right)$

Some non-linear function $\rightarrow f(x)$

Sum over multi-set $\rightarrow \sum_{x \in S}$

S : multi-set

 $\rightarrow \Phi \left[f(\text{yellow}) + f(\text{blue}) + f(\text{blue}) \right]$

Example: $\Phi \left[f(\text{yellow}) + f(\text{blue}) + f(\text{blue}) \right]$

One-hot $\begin{pmatrix} 1 \\ 0 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} + \begin{pmatrix} 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}$

Injective

- multi-set 내의 모든 요소에 함수 f 적용
- multi-set 합산
- 합산한 결과에 non-linear 함수 적용

Most Expressive GNN

- **Graph Isomorphism Network (GIN)** [Xu et al. ICLR 2019]

- Apply an MLP, element-wise **sum**, followed by another MLP.

$$\text{MLP}_{\Phi} \left(\sum_{x \in S} \text{MLP}_f(x) \right)$$

Universal Approximation Theorem

: 충분히 큰 Hidden Dimension과 Non-Linear 함수가 있다면

1-Hidden-Layer NeuralNet으로 어떤 연속함수든 근사할 수 있다.

GIN : failure case 없음 / 가장 강한 표현력 가진 GNN

aggregation 함수가 injective 하다!

Relation to WL Graph Kernel

$$c^{(k+1)}(v) = \text{HASH} \left(c^{(k)}(v), \{c^{(k)}(u)\}_{u \in N(v)} \right),$$

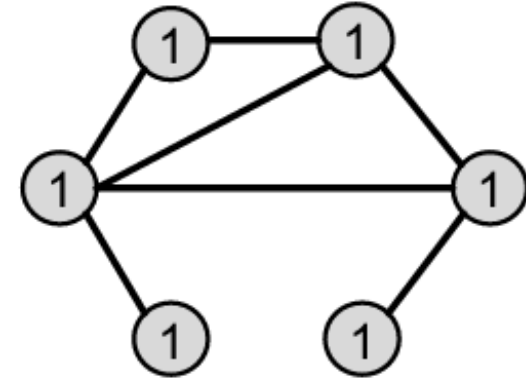
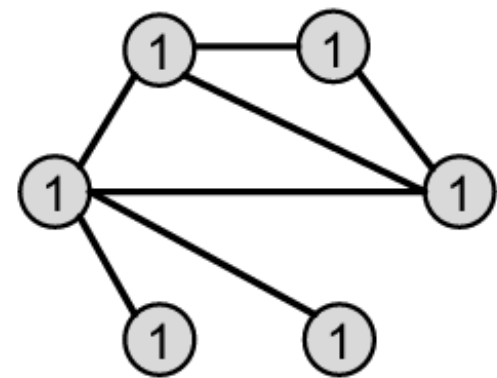
- 이웃 node로부터 message 가져온 다음
- node v 자신의 message 와 결합
- HASH 함수를 통해 새로운 color 생성

Weisfeiler-lehman kernel

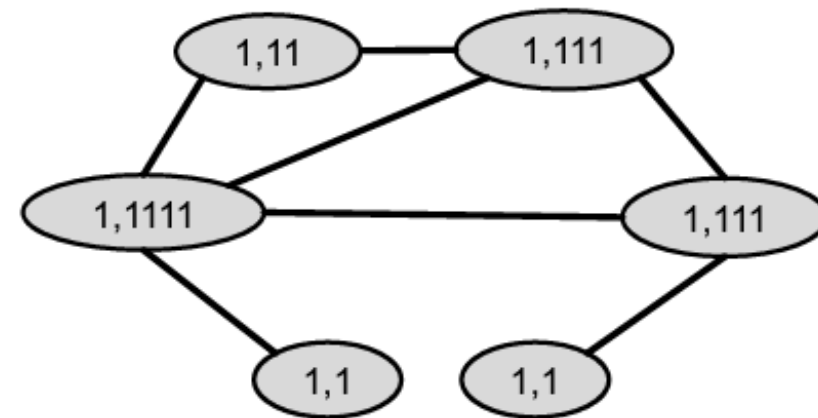
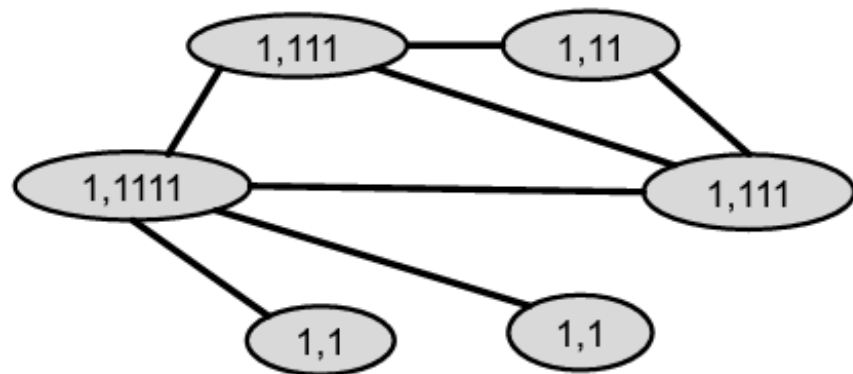
: HASH함수를 기반으로 서로 다른 neighbor node에 대해 서로 다른 색을 부여해서 graph가 isomorphic한지 아닌지 판별하는 알고리즘

Color Refinement

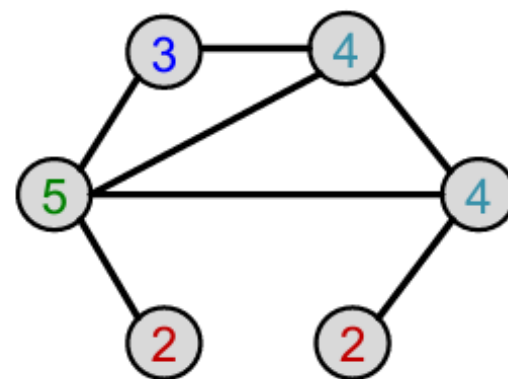
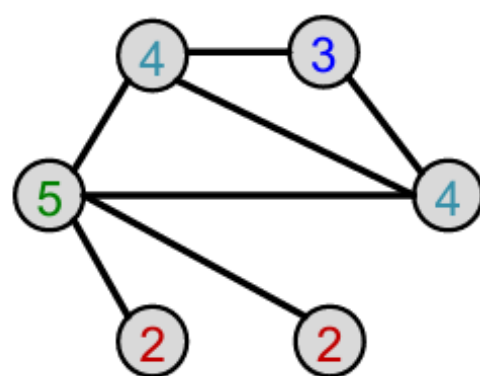
- Assign initial colors



- Aggregate neighboring colors



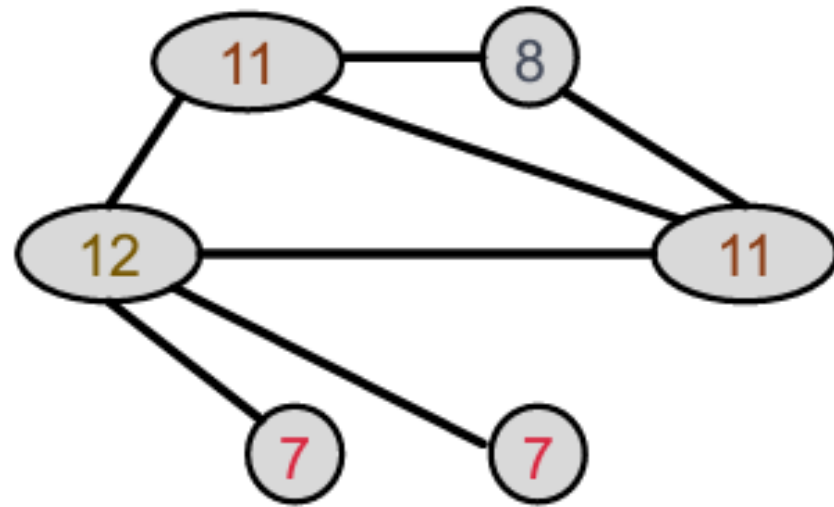
- Injectively** HASH the aggregated colors



HASH table: **Injective!**

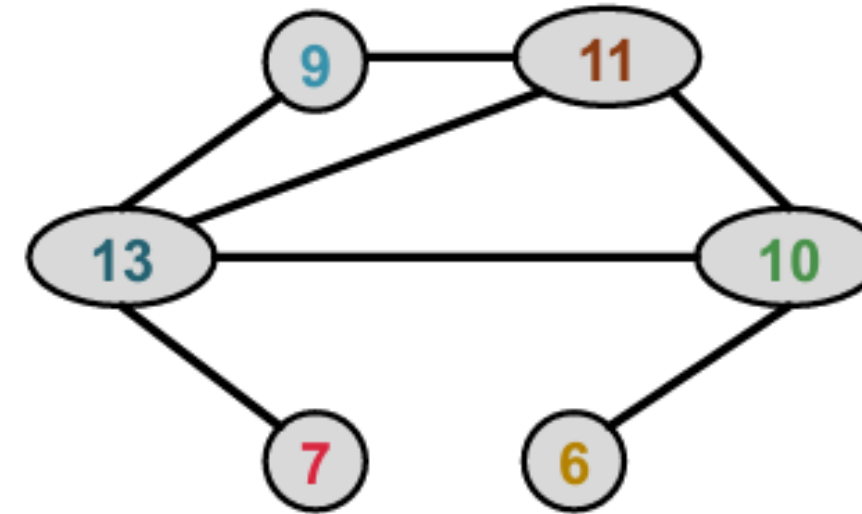
1,1	-->	2
1,11	-->	3
1,111	-->	4
1,1111	-->	5

Color Refinement



non - isomorphic

\neq



- 위 과정은 stable coloring 단계에 이르기까지 계속 반복한다.

The Complete GIN Model

$$\text{MLP}_{\Phi} \left(\overset{\text{small learnable scalar}}{(1 + \epsilon) \cdot \underset{\substack{\text{transform our} \\ \text{own message}}}{\text{MLP}_f(c^{(k)}(v)))}} + \sum_{u \in N(v)} \text{MLP}_f(c^{(k)}(u)) \right)$$

- aggregate children
- transform with MLP
- sum them up

GIN and WL Graph Kernel

GIN = WL graph kernel 의 미분가능한 neural version

- GIN과 WL graph kernel은 둘 다 low-dimensional한 node embedding을 갖는다.
 - ➡ 서로 다른 node의 미세한 유사성을 확인할 수 있습니다.
- GIN & WL graph kernel
 - : update하는 target과 update function이 유사하기 때문에 정확히 같은 expressive power를 갖는다.
 - : 두 방법 모두 대부분의 graph 를 구별할 수 있다.

THANK YOU

