시간 라벨 데이터에서의 효율적인 KNN 탐색

발표자 : 김수지

목차

1. Intro

Q. 2000-2023년 사이에서 '주토피아'와 가장 유사한 영화 3개를 알고 싶다면?

어떻게 시간과 관련된 벡터 데이터에서 특정 시간 간격 내 빠르고 정확한 Proximity search를 할 수 있을까?

2. Related work

Proximity search는 query(질의)에 가까운(유사한) 데이터를 찾는 문제

- **k-nearest neighbor (kNN) search**

  k-nearest neighbor (kNN) search은 Proximity search 중 하나

  Query point와 가까운 k개의 데이터 포인트를 찾는 것을 목적으로 함

  하지만 Exact kNN search는 특히 고차원 데이터에서 비용이 많이 요구됨

  -> Approximate k-nearest neighbor search (ANN) 등장

2. Related work

- **Approximate k-nearest neighbor search (ANN)**

  비용을 줄이기 위해 정확도를 약간 포기하면서 대략적인 k개의 nearest neighbors를 찾는 것이 목표

  인덱싱하는 방법(데이터 구조)에 따라 크게 네개 정도로 나눌 수 있음

  -> Tree-based, Graph-based, Hashing-based, Quantization-based

  최근에는 Graph-based & Quantization-based가 SOTA

BUT 기존 ANN Method들로 시간 관련 데이터 문제들을 풀 수 있을까?

-> 바로 적용시키기 어려움

2. Related work

- **Time-Restricted kNN (TkNN) Search**

  대부분의 기존 연구들은 2-3차원을 가정한 method 제공

  -> R*-Tree, Quadtree, PPR-Tree 기반

  데이터의 시공간적 차원이 늘어나면서 TkNN 문제 더욱 복잡해짐

  BUT 기존 TkNN Method들로 앞서 말한 문제들을 풀 수 있을까?

  -> 차원의 저주 등의 문제로 tree 기반 method들로는 고차원 데이터에서의 TkNN 문제 해결 X

-> 새로운 방법 제안

3. Method

- **challenge**

(1) How can we make the query process remain efficient regardless of the length of the query time window?

(2) How can we efficiently add new vectors that occur over time to MBI?

(3) How should we select a set of blocks to process the query most efficiently?

3. Method

- **Binary Search and Brute-Force (BSBF)**

  Binary Search와 Brute-Force 방법을 합친 것

  데이터를 날짜 순서대로 정렬하여 인덱스로 사용

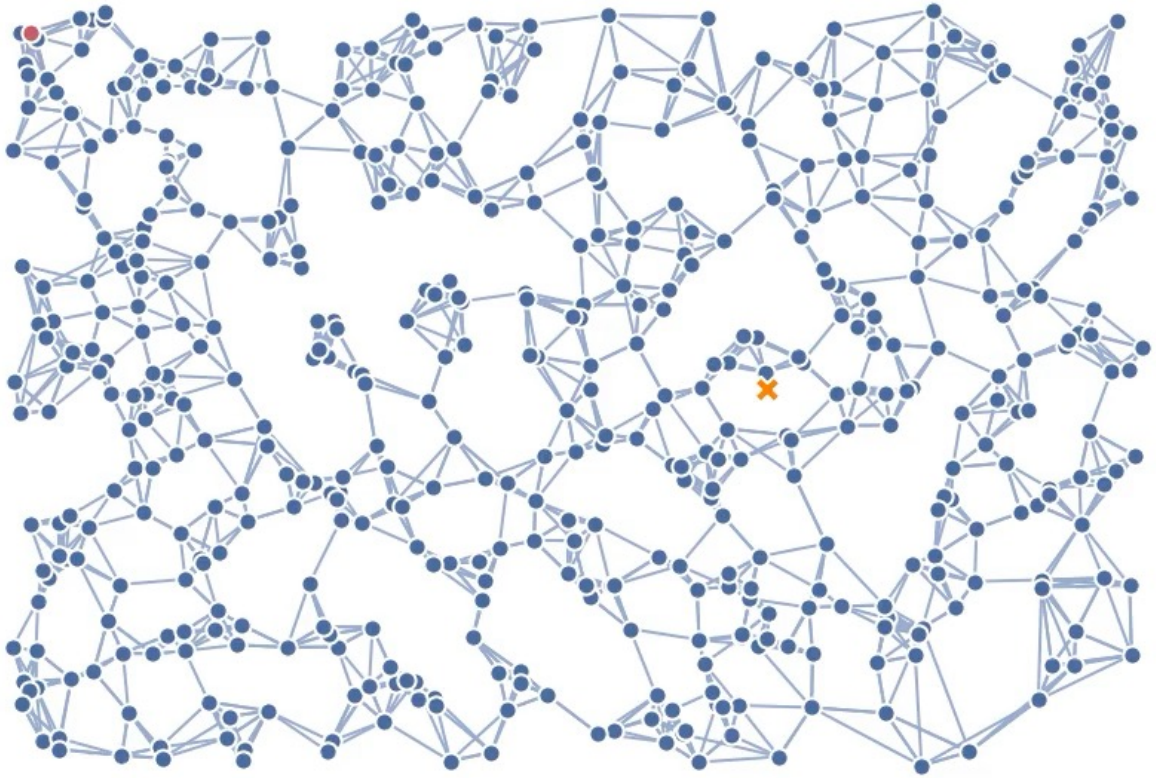
  query $q = (w, k, ts, te)$ 가 주어지면,
  기간 $\mathcal{D}[ts : te]$ 사이에서 Binary Search 수행하고
  Brute-Force 방법을 사용하여 k개의 유사한 vector들을 찾는 방법

w : query vector
k : 찾으려는 vector의 수
ts : period 시작
te : period 끝
$\mathcal{D}$: Spatio-temporal database

-> 쿼리 기간이 짧을 때 속도가 빠르지만 기간이 길어질수록 매우 비효율적

3. Method

- 기존 kNN Search 동작 방식



3. Method

- **Search and Filtering (SF)**

  기존 kNN method를 조금 수정한 방법

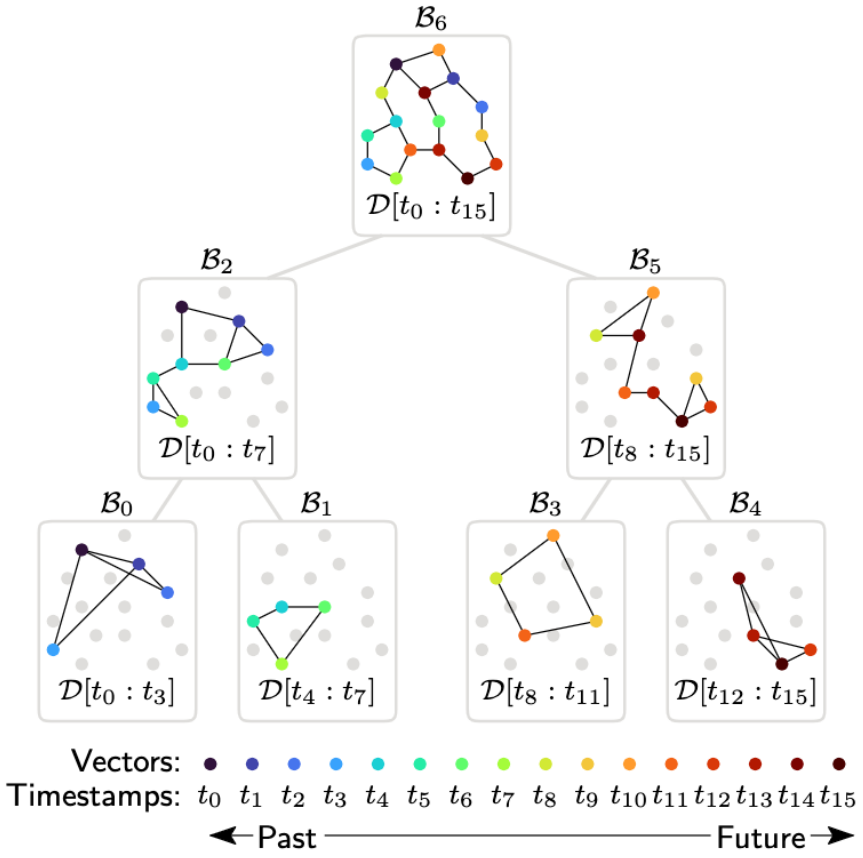
  전체 데이터에 대해 그래프를 만들어 kNN 탐색하고 결과 리스트에 쿼리 기간이 포함된 vector만 추가

  쿼리 기간이 충분히 길 때는 매우 효율적

  하지만 쿼리 기간이 좁을 때는 매우 비효율적

3. Method

- **Multi-level Block Indexing (MBI)**



BSBF와 SF를 결합한 방법

시간 구간에 따라 데이터를 여러 블럭으로 나누고
각 블럭을 kNN 질의를 처리할 수 있도록 구조화

각 블럭은 Binary tree로 되어있으며 특정 시간대의 모든 벡터들과 그 벡터들로 생성한 KNNG로 구성

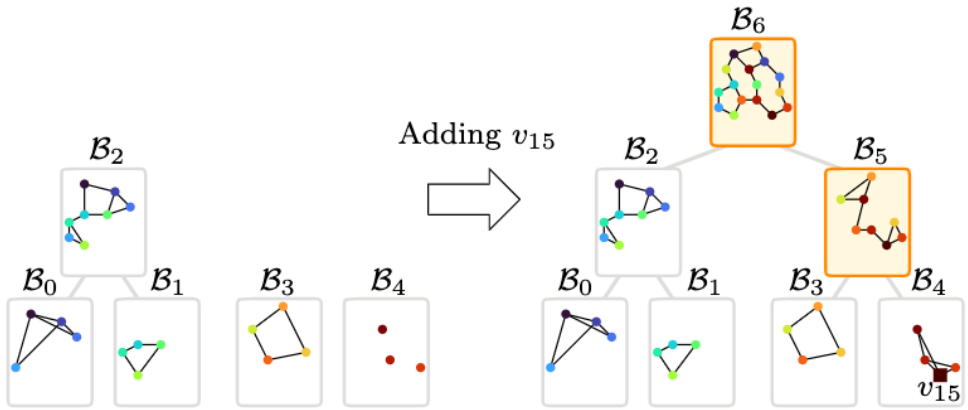
각 블럭은 왼쪽 자식, 오른쪽 자식이 있으며 이는
부모 블록의 절반 벡터들을 가짐

왼쪽 자식이 earlier timestamp, 오른쪽 자식이 later timestamp

위 계층으로 올라갈수록 블럭의 크기 2배씩 커짐

3. Method

- **Multi-level Block Indexing (MBI)**



Bottom-up block merging

블럭 크기까지 데이터가 다 차면 블럭을 만드는 방법

아래 레벨의 블럭 두개가 완성되면 윗 레벨의 부모 블럭이 만들어짐

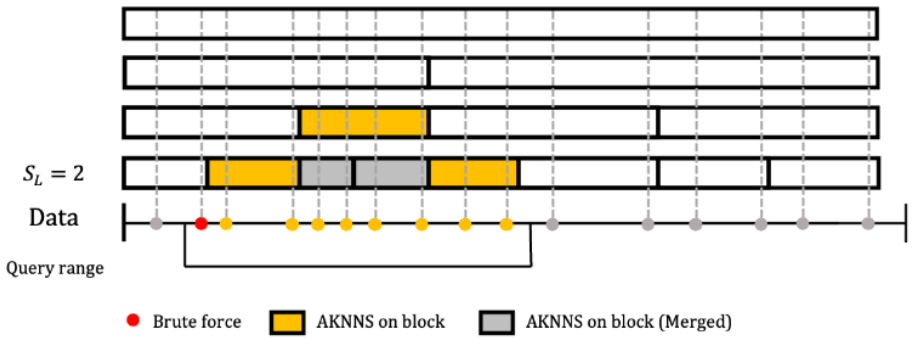
데이터가 추가될 때마다 전체 kNN 그래프를 빌드하지 않아도 됨

아직 블럭이 만들어지지 않은 부분은 Brute-Force 방법으로 처리

-> Challenge 2 해결

3. Method

- **Multi-level Block Indexing (MBI)**



Query Processing on MBI

Top-down block selection 방식을 통한 효율적인 Search 범위 선택

(1) BSBF와 동일한 방법으로 시점과 종점을 찾아 각 블럭이 질의에 포함되었는지 판단

(2) 질의 구간에 포함된 leaf block들을 선택 후 병합 최대한 수행 (leaf block을 완전히 채우지 못하는 끝 영역은 Brute force로 처리, 검색된 블럭이 전체 블럭의 일정 부분 넘으면 SF로 처리)

(3) 각각의 block들에서 KNN 질의를 처리한 결과 값들과 Brute-force로 얻은 결과값들의 병합을 통해 가장 가까운 K개의 데이터를 찾아 반환

-> Challenge 3 해결

4. Experiment

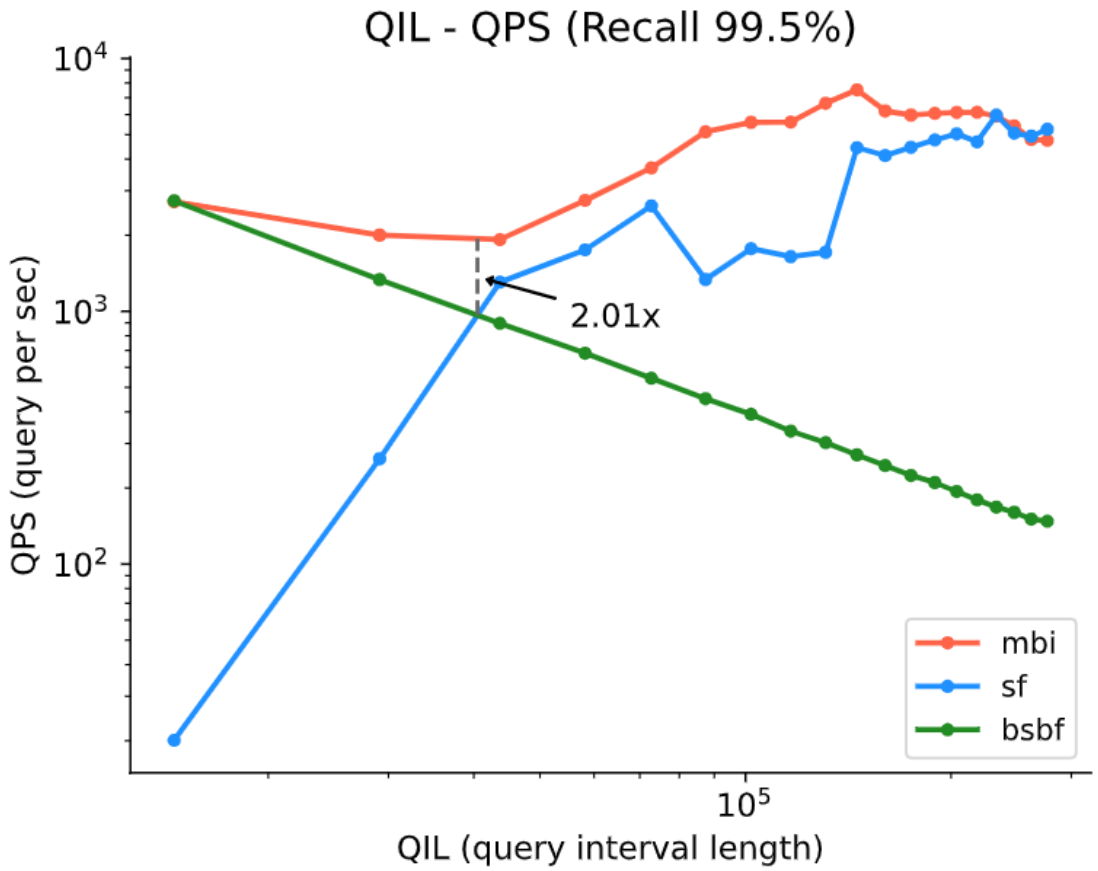
- **Experimental Setting**

Dataset	# items	Dim	Source
Movielens	57,771	32	GroupLens[6]
GK2A	291,380	128	KMA[7]
GloVe-25	1,193,514	25	Jeffrey et al.[8][?]

무비렌즈, GK2A, GloVe-25 세가지 real-world 데이터셋 사용

4. Experiment

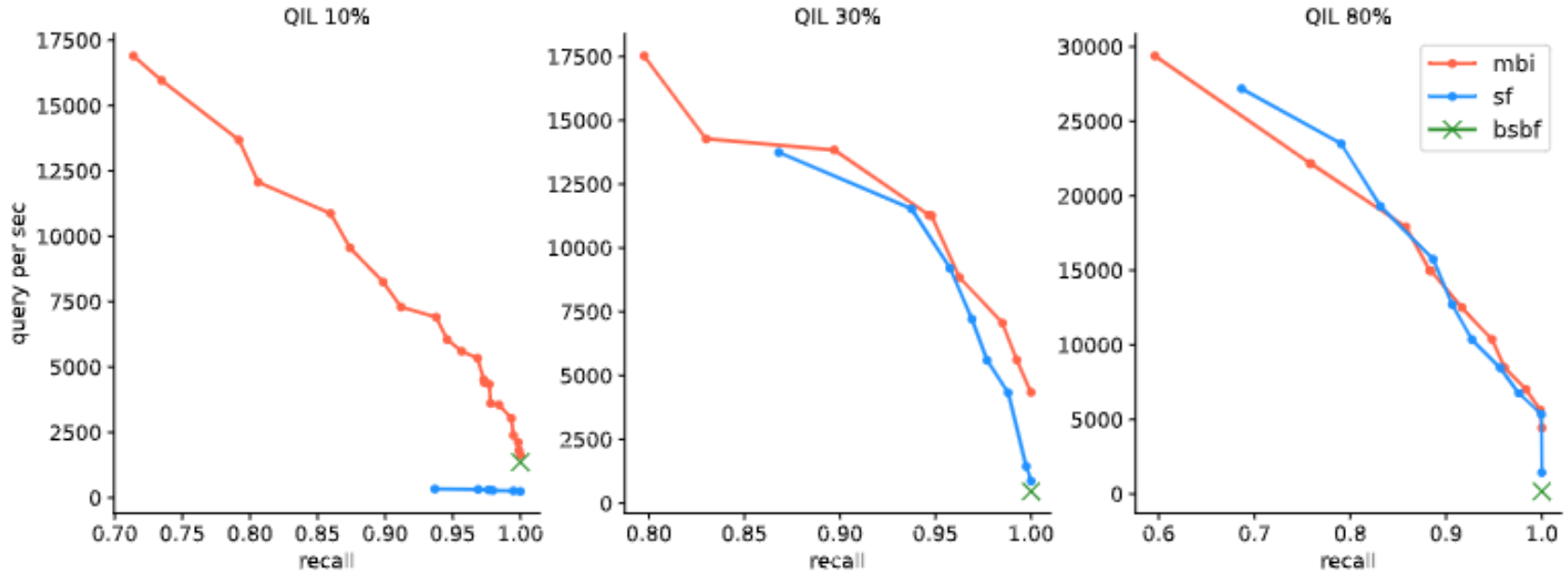
- **Search Performance**



QPS : Query Per Second
QIL : Query Interval Length

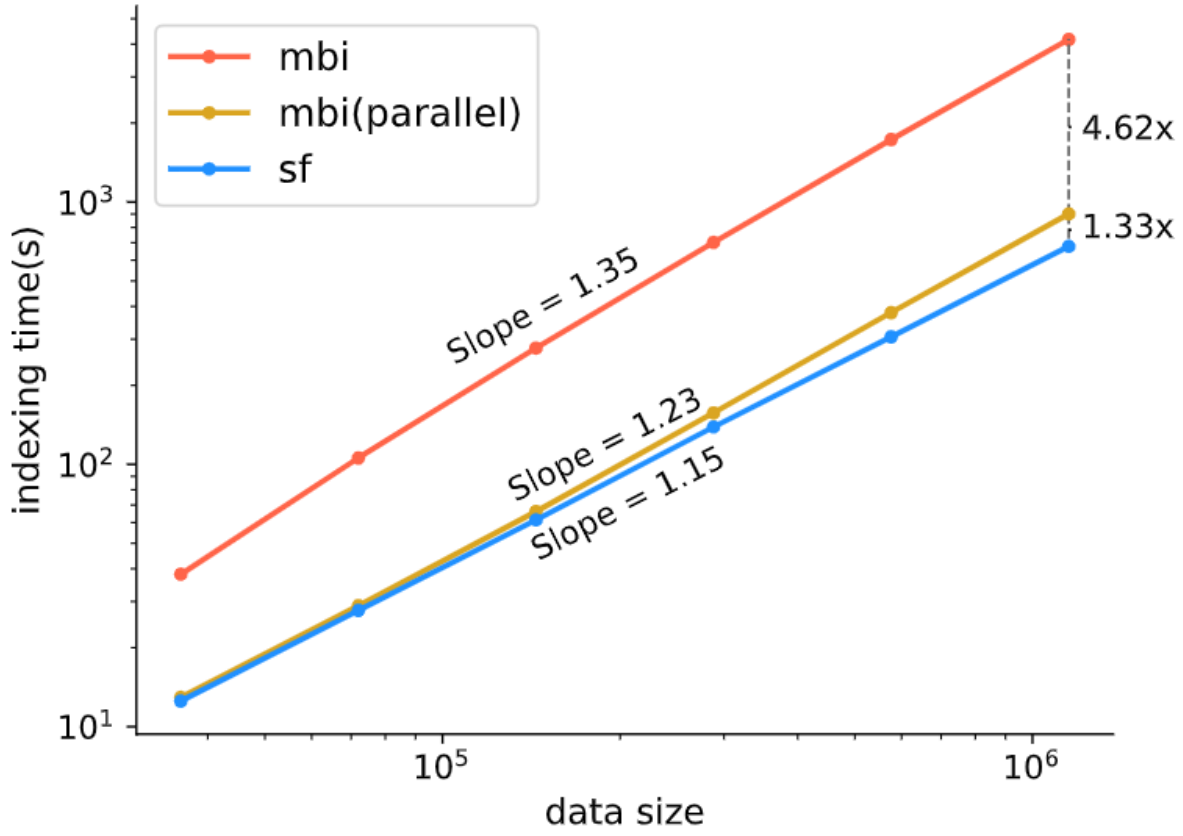
4. Experiment

- **Search Performance**



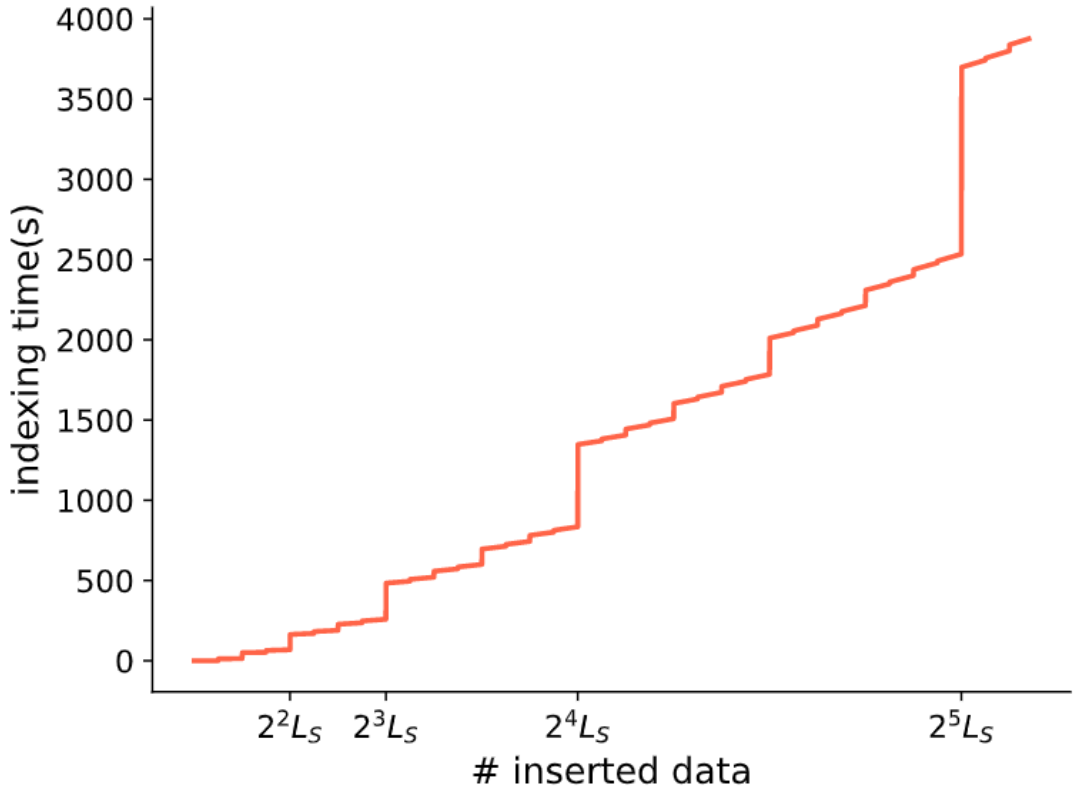
4. Experiment

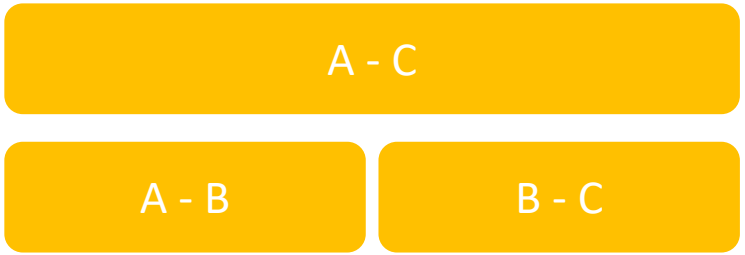
- **Indexing time of MBI**



4. Experiment

- **Insertion time of MBI**





S_L : 리프 블럭 데이터의 크기