Product Quantization in Similarity Search

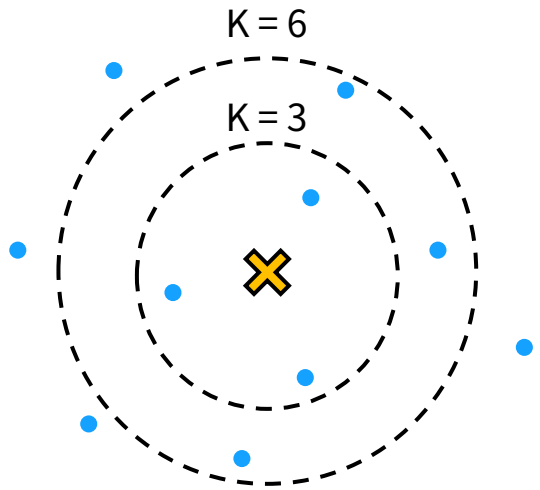
발표자 : 김수지

Index

- Background
- Product Quantization
- IVF
- IVF+PQ
- TKNN

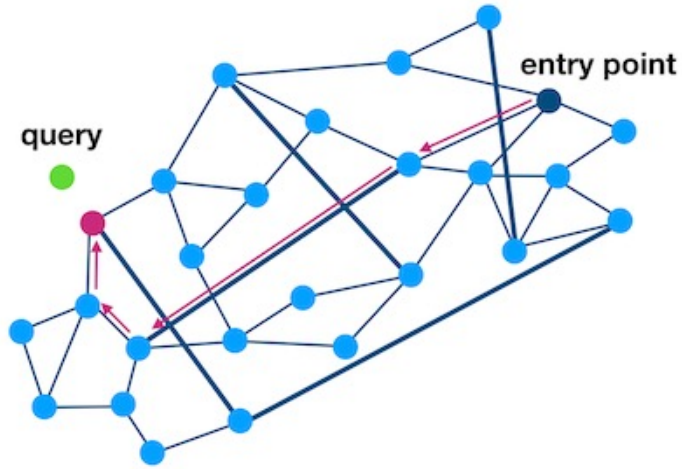
Similarity Search

- k-Nearest Neighbor search
  - 쿼리 지점과 가장 가까운 (유사한) 벡터 k 개를 찾는 방법



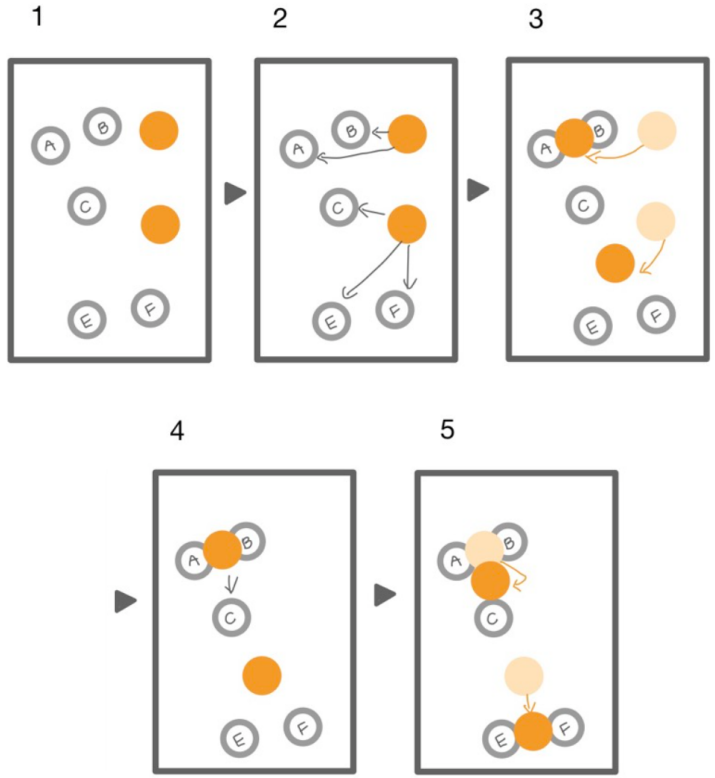
- Approximate k-Nearest Neighbor search
  - 정확도를 약간 포기하고, 속도를 높인 방법
  - 최근 ANN 연구 방법 중,

    Product quantization을 사용한 방법이 SOTA



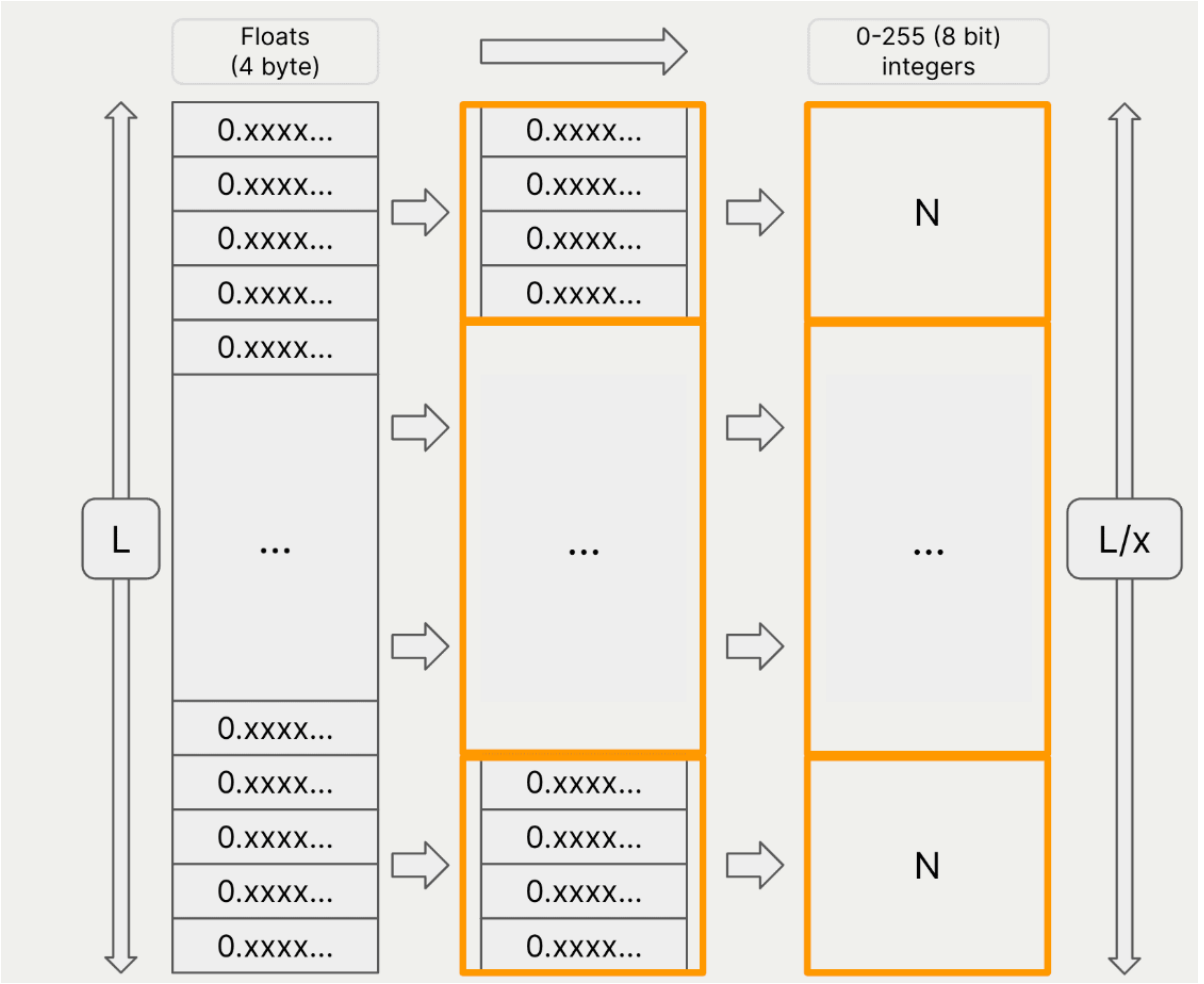
k-means

- K-means clustering
  - K개로 클러스터링하는 방법 중 하나
  - 순서
    - 원하는 군집 수 만큼 임의의 군집 중심점(centroid) 설정
    - 각 데이터는 가장 가까운 중심점에 소속
    - 중심점에 할당된 데이터들의 평균 중심으로 중심점 이동
    - 각 데이터는 이동된 중심점 기준으로 가장 가까운 중심점에 소속
    - 다시 중심점에 할당된 데이터들의 평균 중심점으로 중심점 이동



Product Quantization

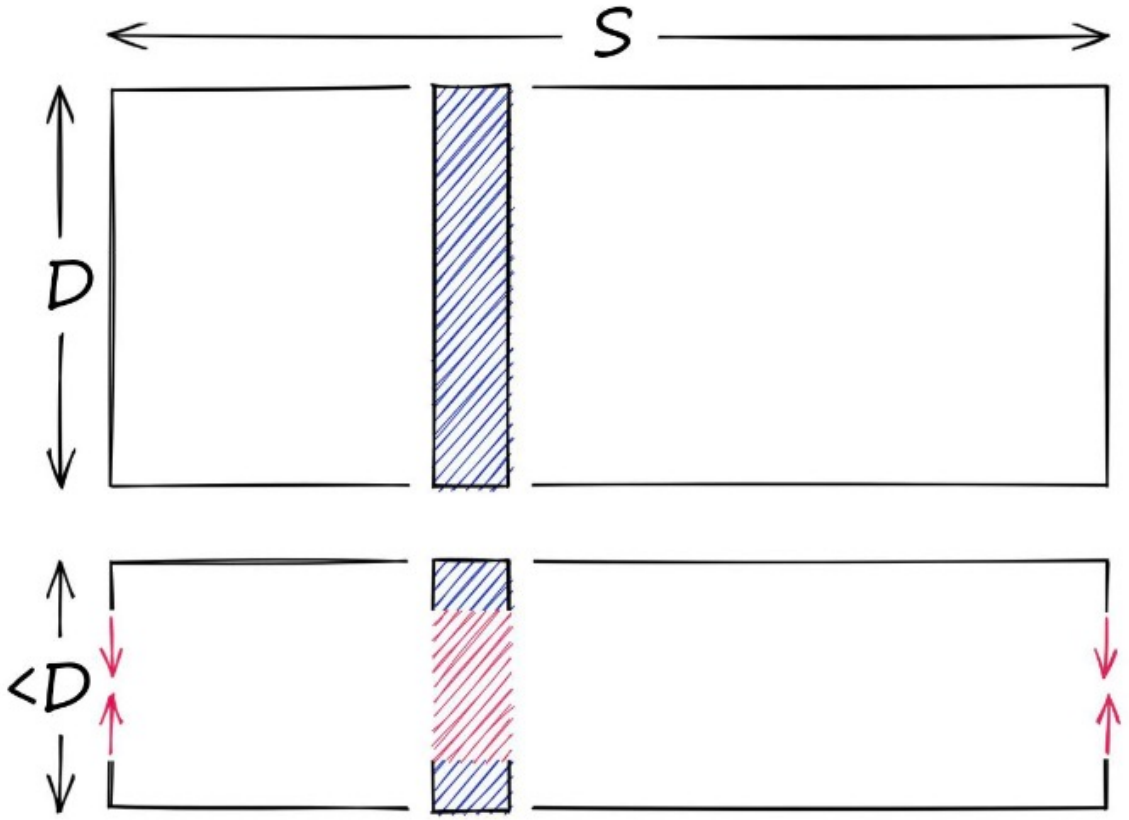
- Product Quantization
  - 고차원 벡터를 압축하는 방법 중 하나
  - 압축의 장점?
    - 적은 메모리
    - 빠른 액세스



Product Quantization

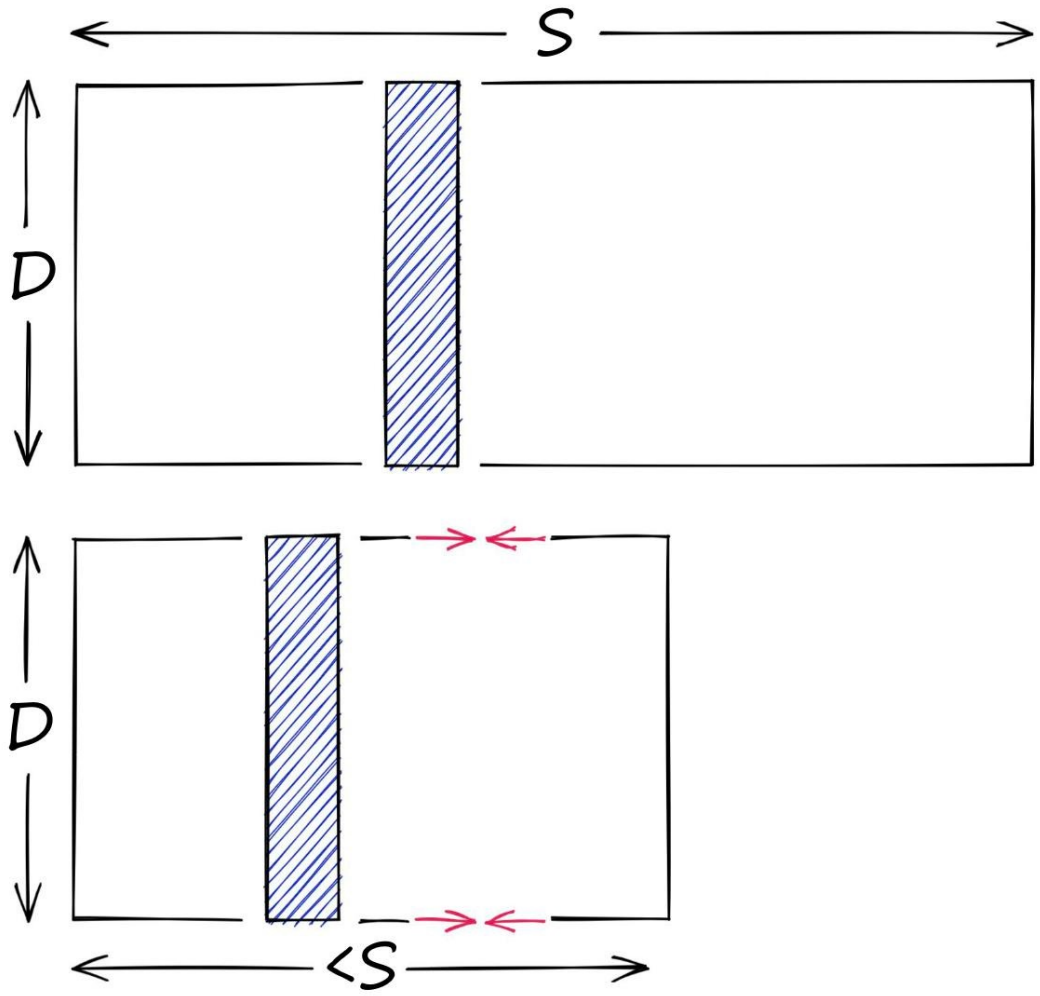
- 기존 차원 축소 방법
  - 범위가 아닌

    벡터의 차원 축소에 초점
  - 예) PCA, SVD ...



Product Quantization

- Product Quantization
  - 차원이 아니라 탐색할 scope를 줄이는 방법이라고 볼 수 있음



Product Quantization

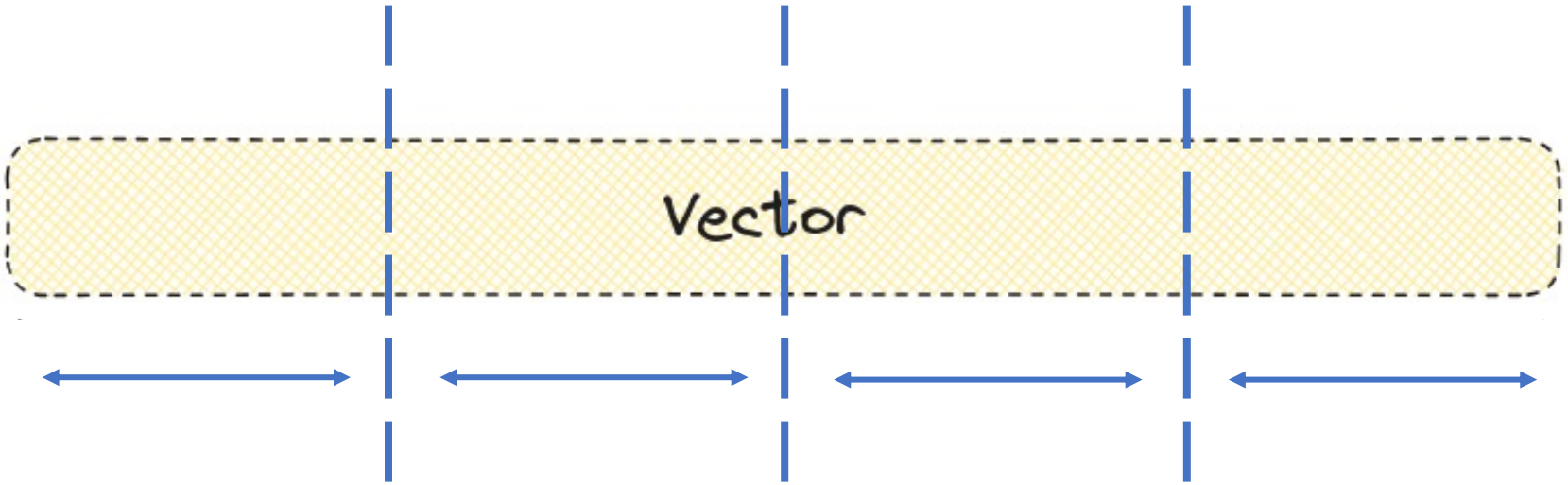
- Indexing

D (벡터의 길이)

Ex) D = 12

Product Quantization

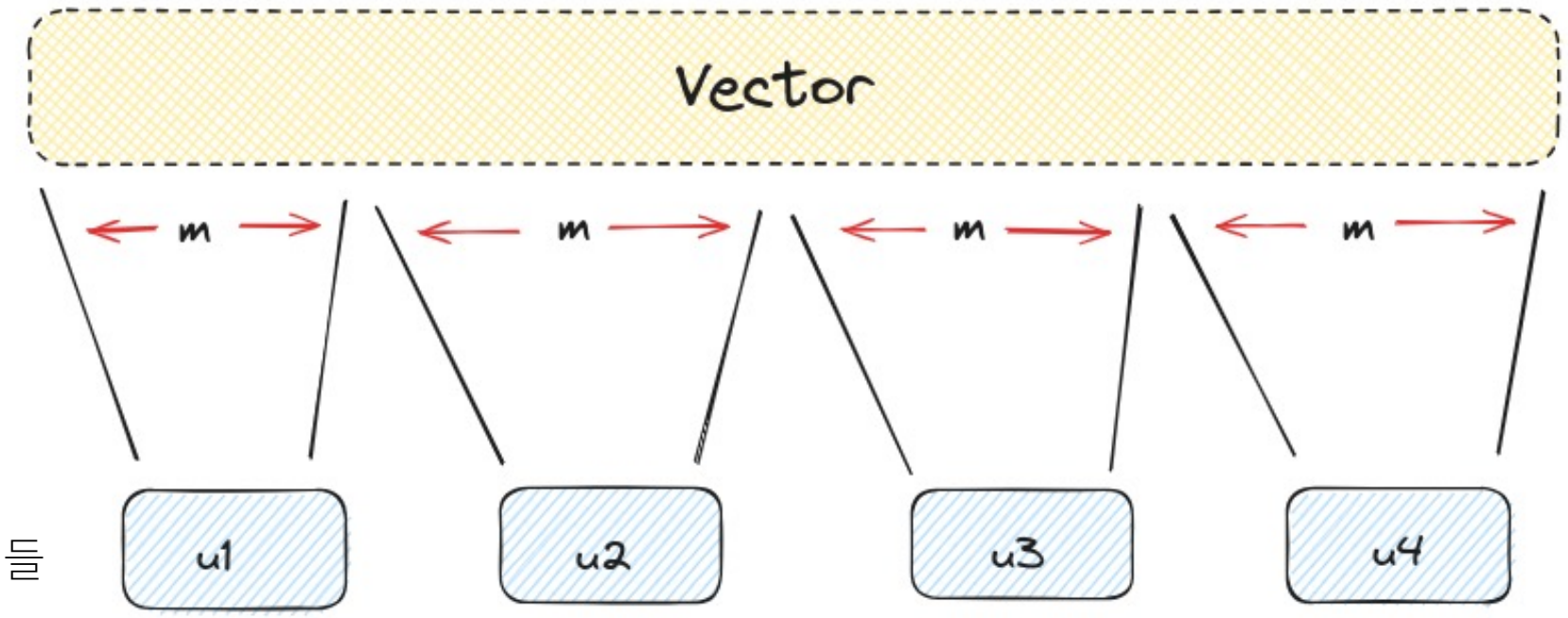
- Subvector : 벡터를 몇 개로 나눠서 indexing할지 정하는 파라미터



m (서브벡터 개수)

Ex) m = 4

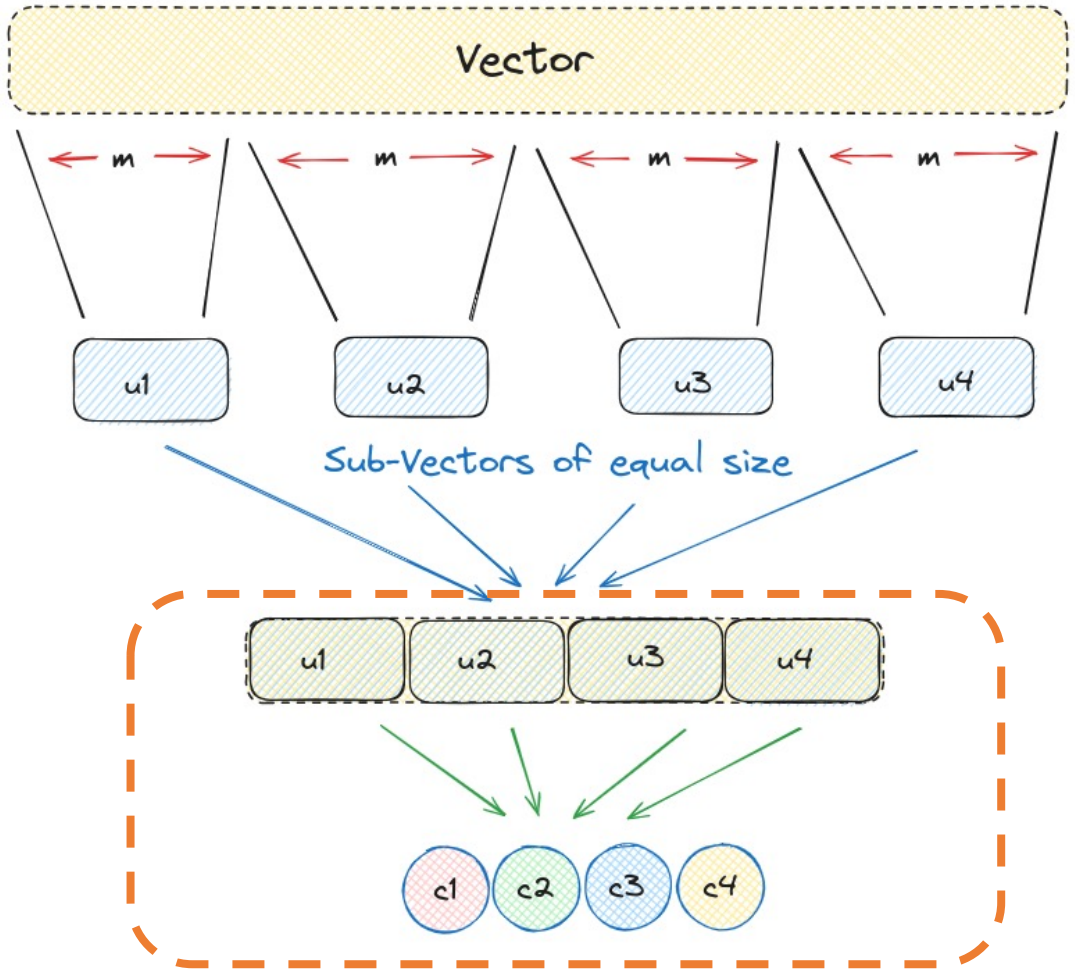
Product Quantization



D / m = D*

12 / 4 하면 나눠 떨어져야함

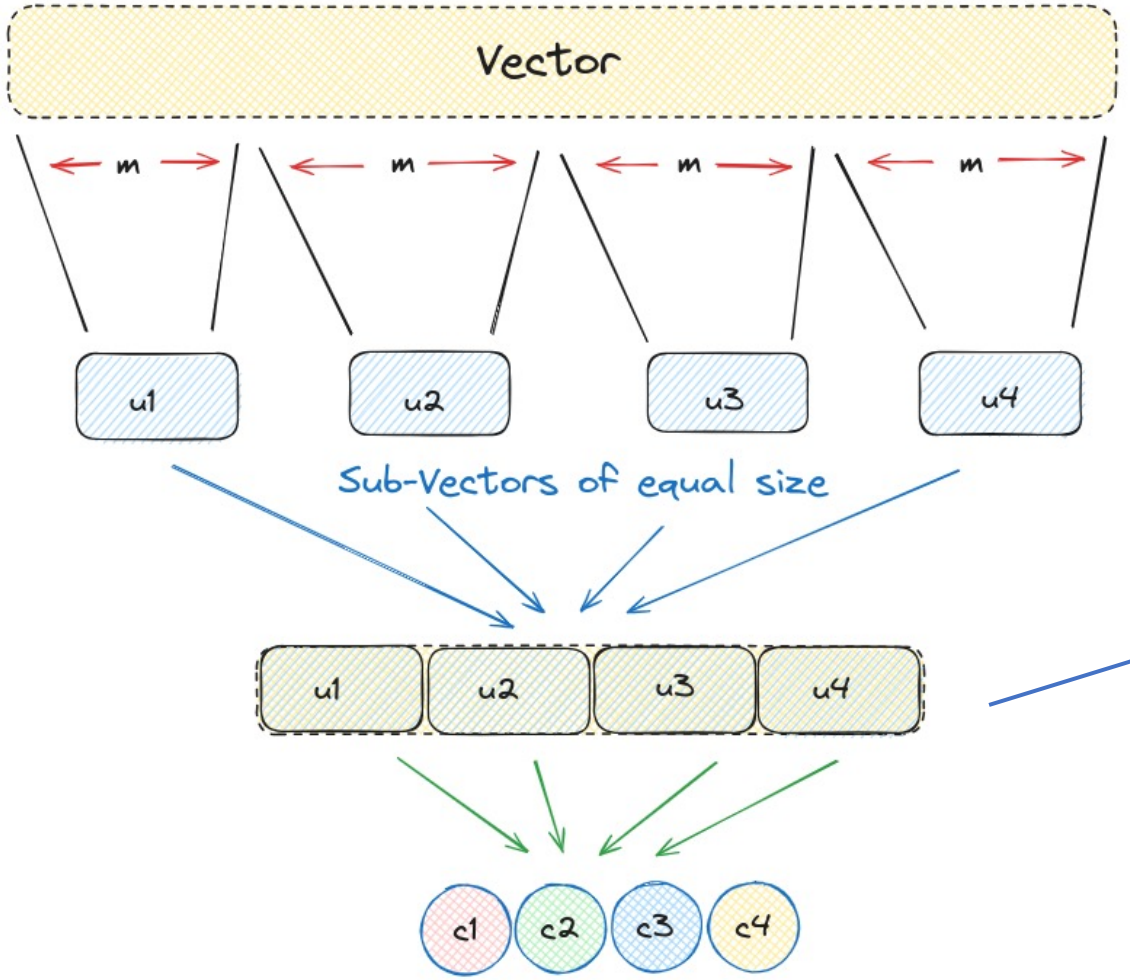
Product Quantization



- 동일한 크기로 나눠진 서브벡터 각각에서 k-means 클러스터링
- 벡터를 이 때 생성된 centroid number로 표현 가능
- 센트로이드 개수는 파라미터

  Ex) k = 3

Product Quantization



Ex) 2, 4, 7, 10

서브벡터 개수

Ex) m = 4

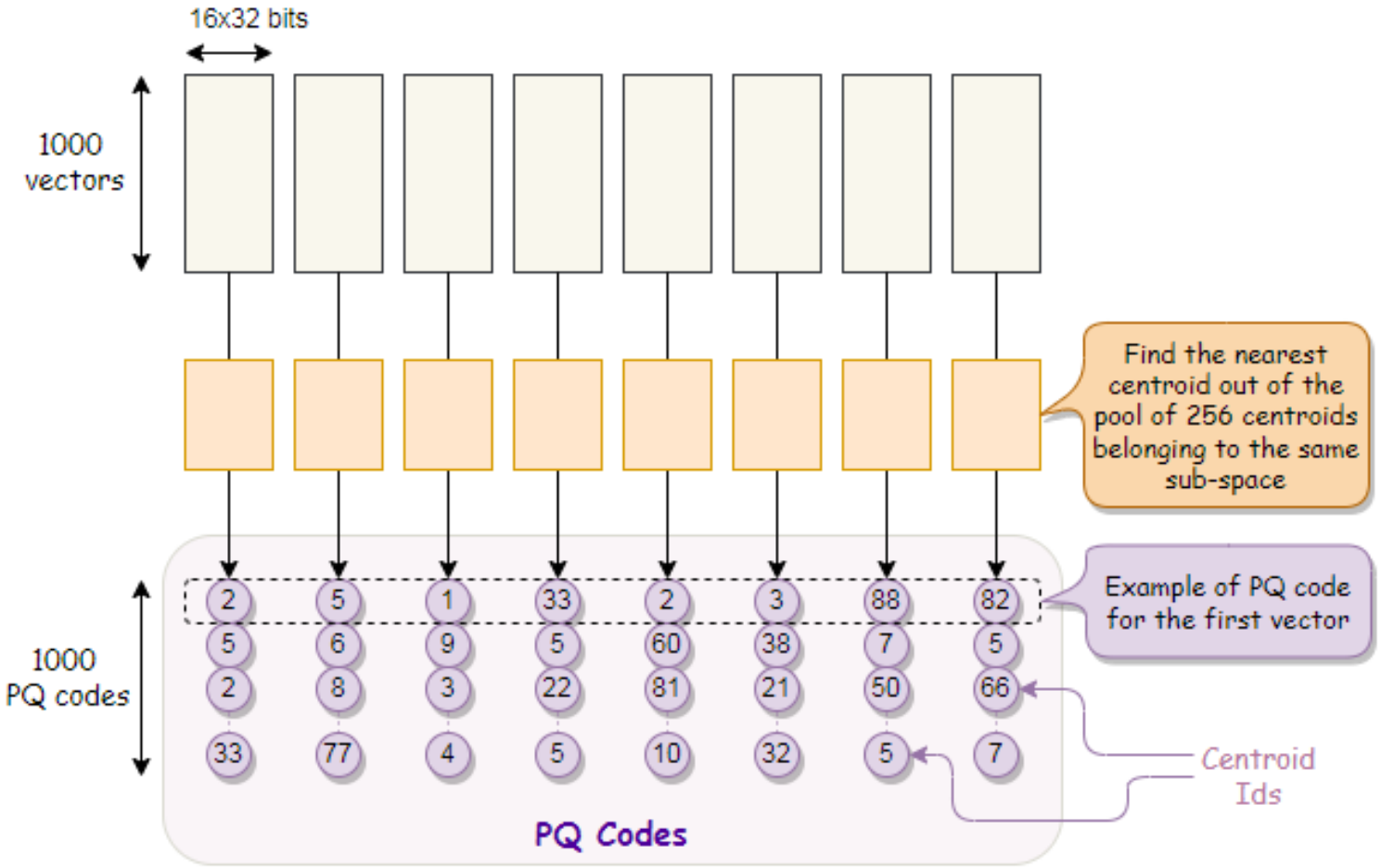
	u1	u2	u3	u4
k1	0	3	6	9
k2	1	4	7	10
k3	2	5	8	11

센트로이드 개수

Ex) k = 4

서브벡터마다 코드북이 생성됨

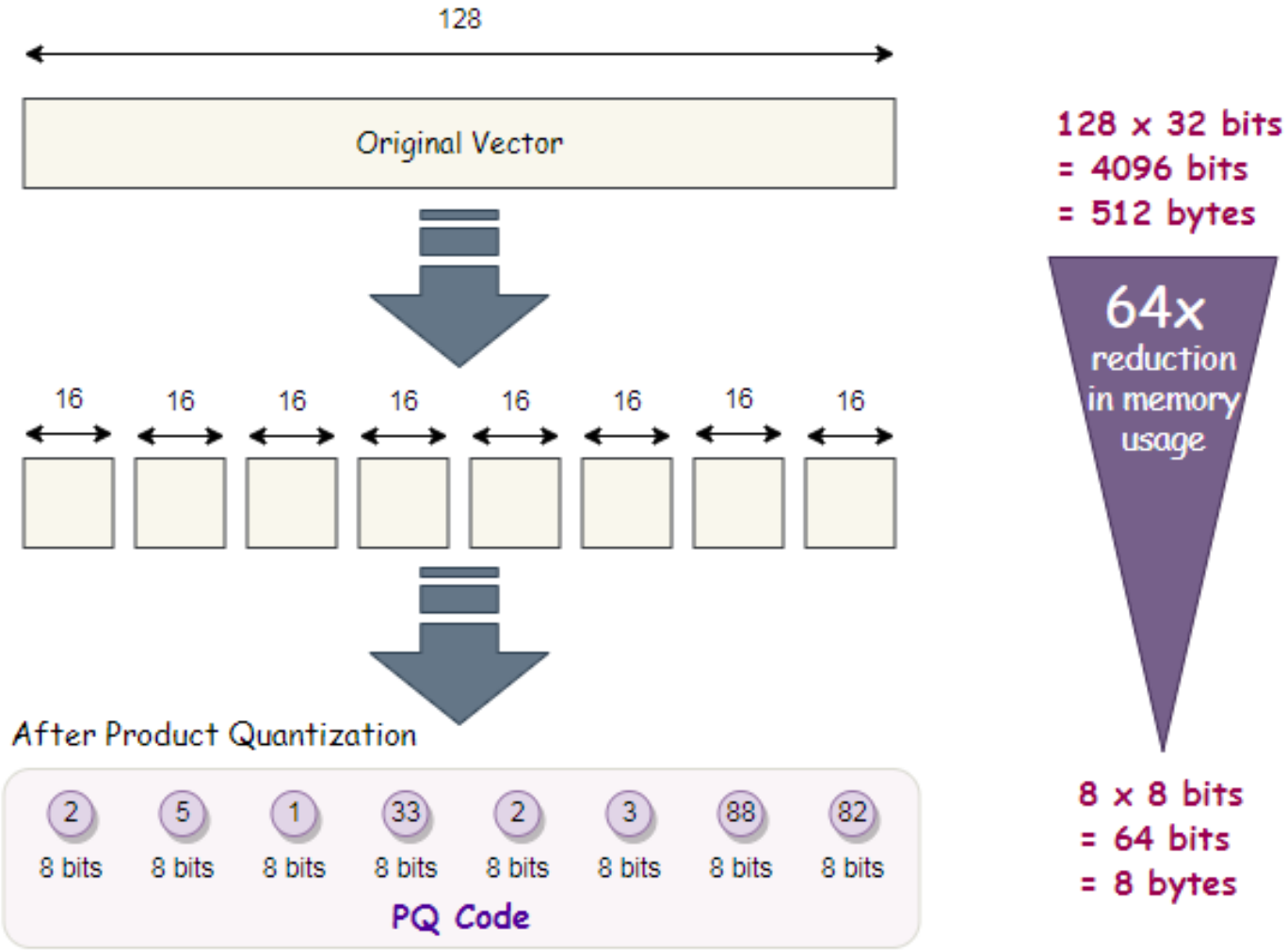
Product Quantization



- 1000개의 벡터가 있었다면,

  1000개의 PQ화된 코드 생성됨

Product Quantization



- 메모리 감소 효과
  - 생성된 코드북을 사용하여

    적은 숫자로도 벡터 표현 가능
  - 기존 차원 128,

    서브벡터 개수 8,

    centroid 256 일 때,

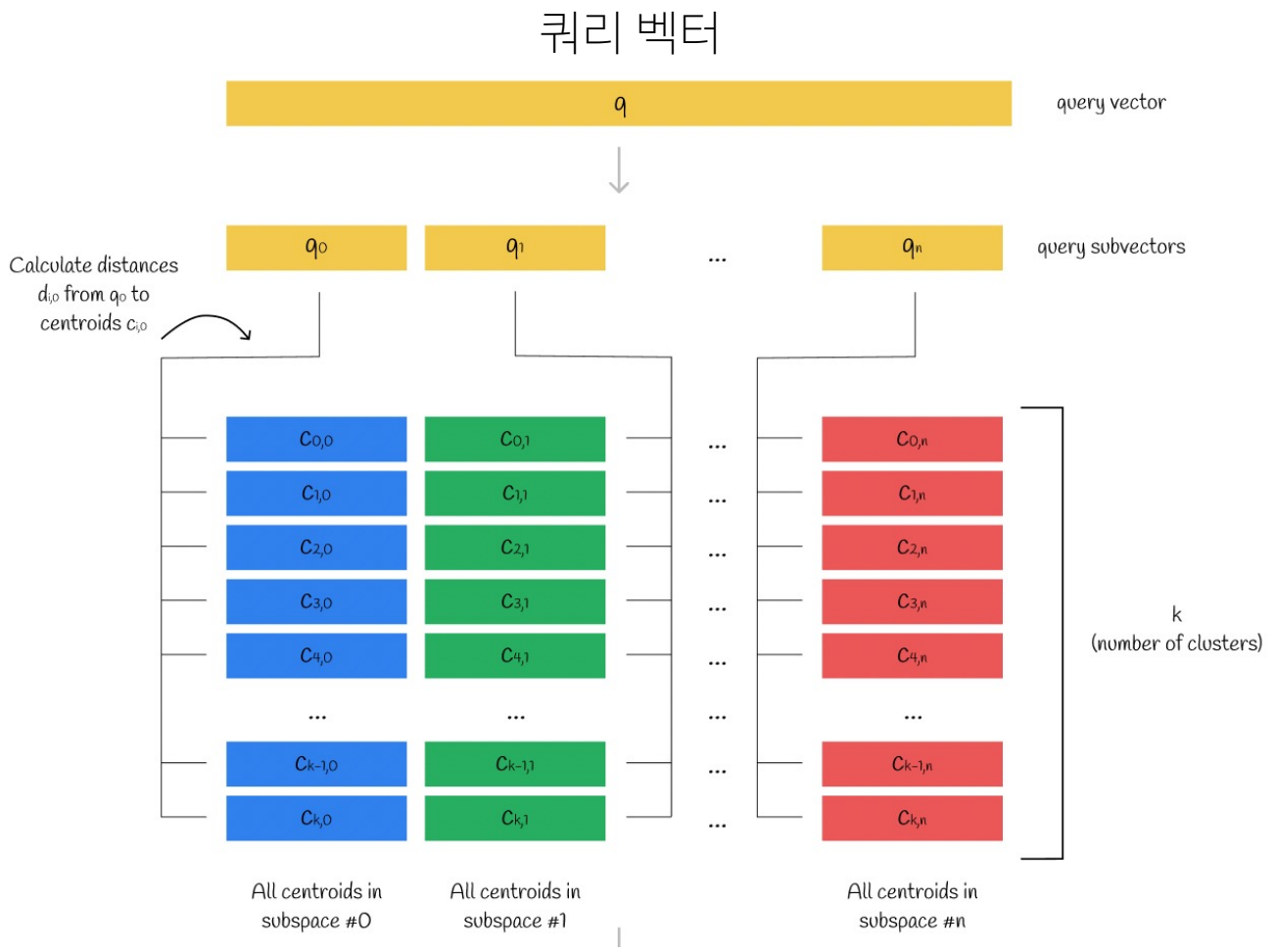
    512 -> 8 bytes

Product Quantization

- Product Quantization for nearest search
  - 쿼리 벡터가 주어지면,

    서브벡터들로 나눠서 각 서브벡터 별

    센트로이드와의 거리 계산



Product Quantization

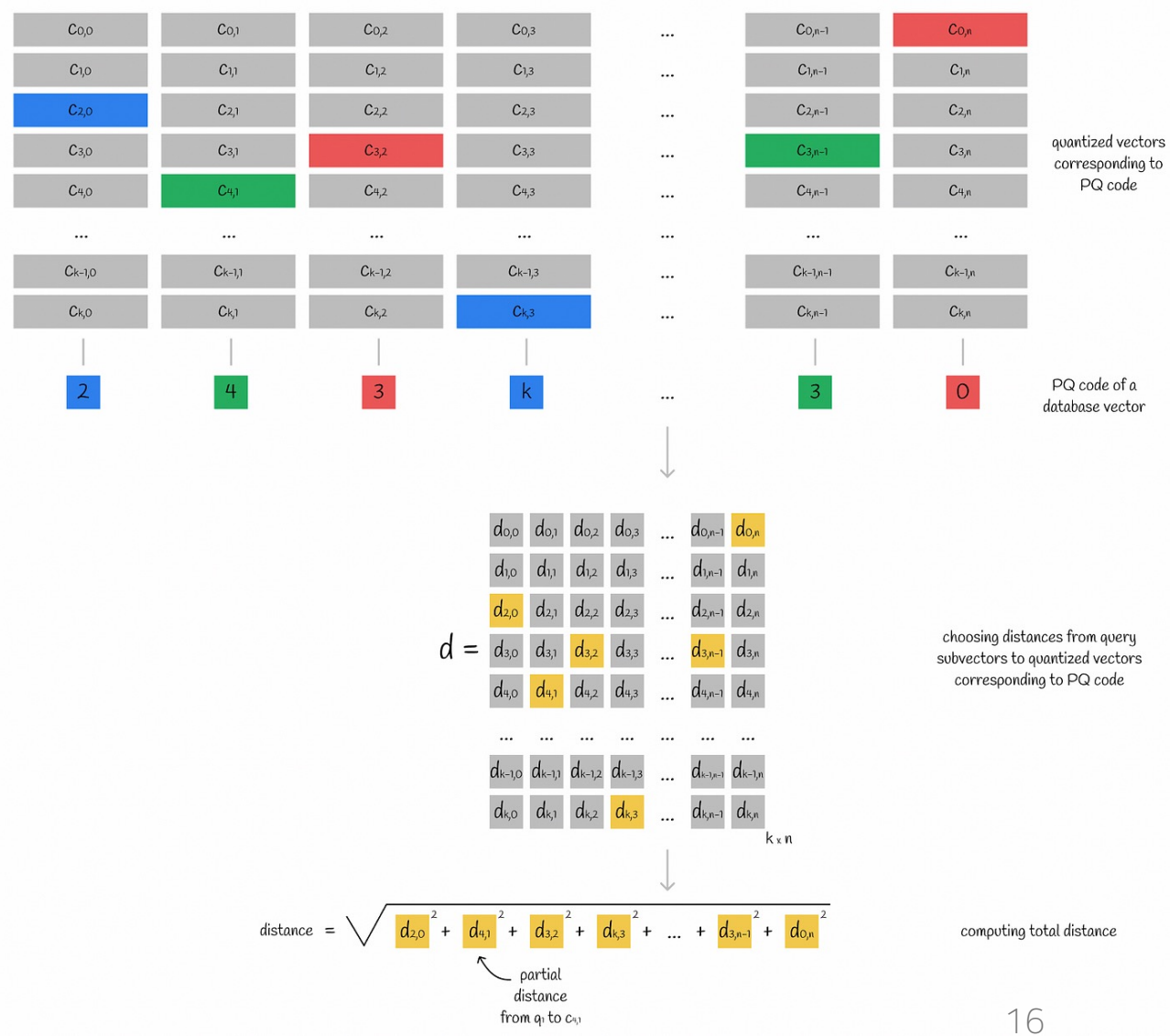
- Product Quantization for nearest search
  - 서브벡터 별 가까운 센트로이드 선정
  - 쿼리 벡터와 해당 센트로이드들과

    total 거리 계산
  - 쿼리 시, 가까운 센트로이드 idx와

    total 거리 얻을 수 있음
  - 모든 벡터 대신,

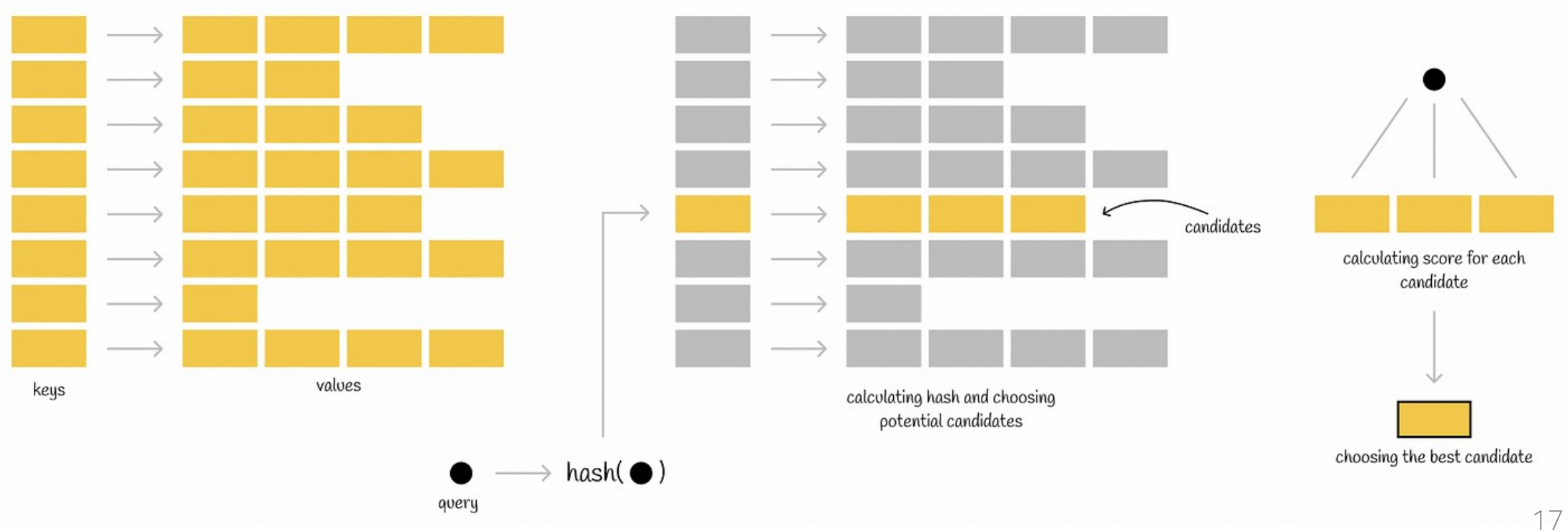
    센트로이드들만 탐색하면 됨

-> 탐색 범위 줄어드는 효과



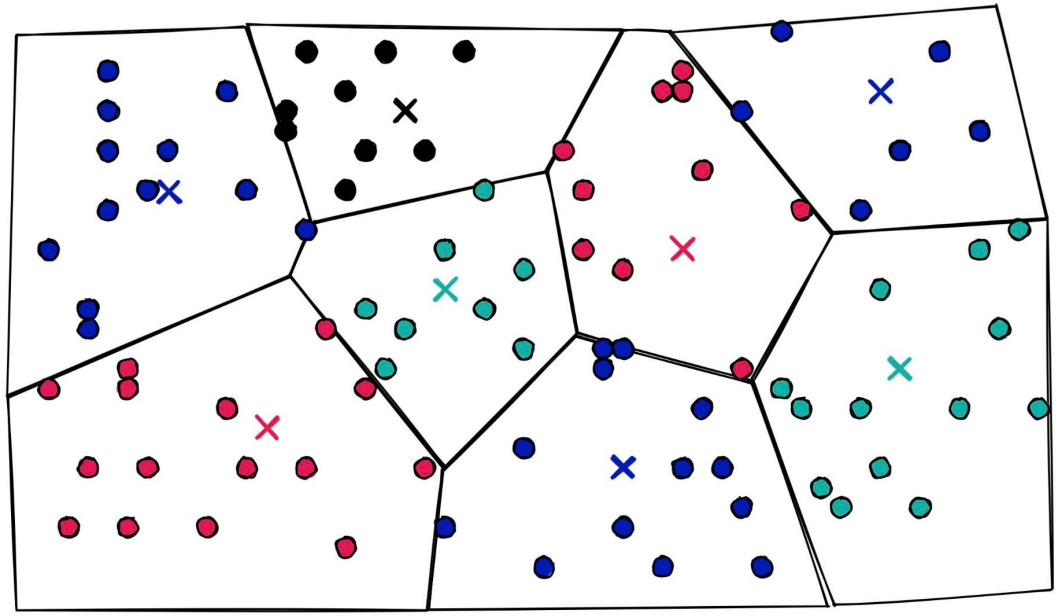
IVF

- Inverted File (Inverted index)
  - 단어나 숫자와 같은 콘텐츠에서 테이블, 집합 등의 해당 위치로 매핑하는 데이터베이스 인덱스



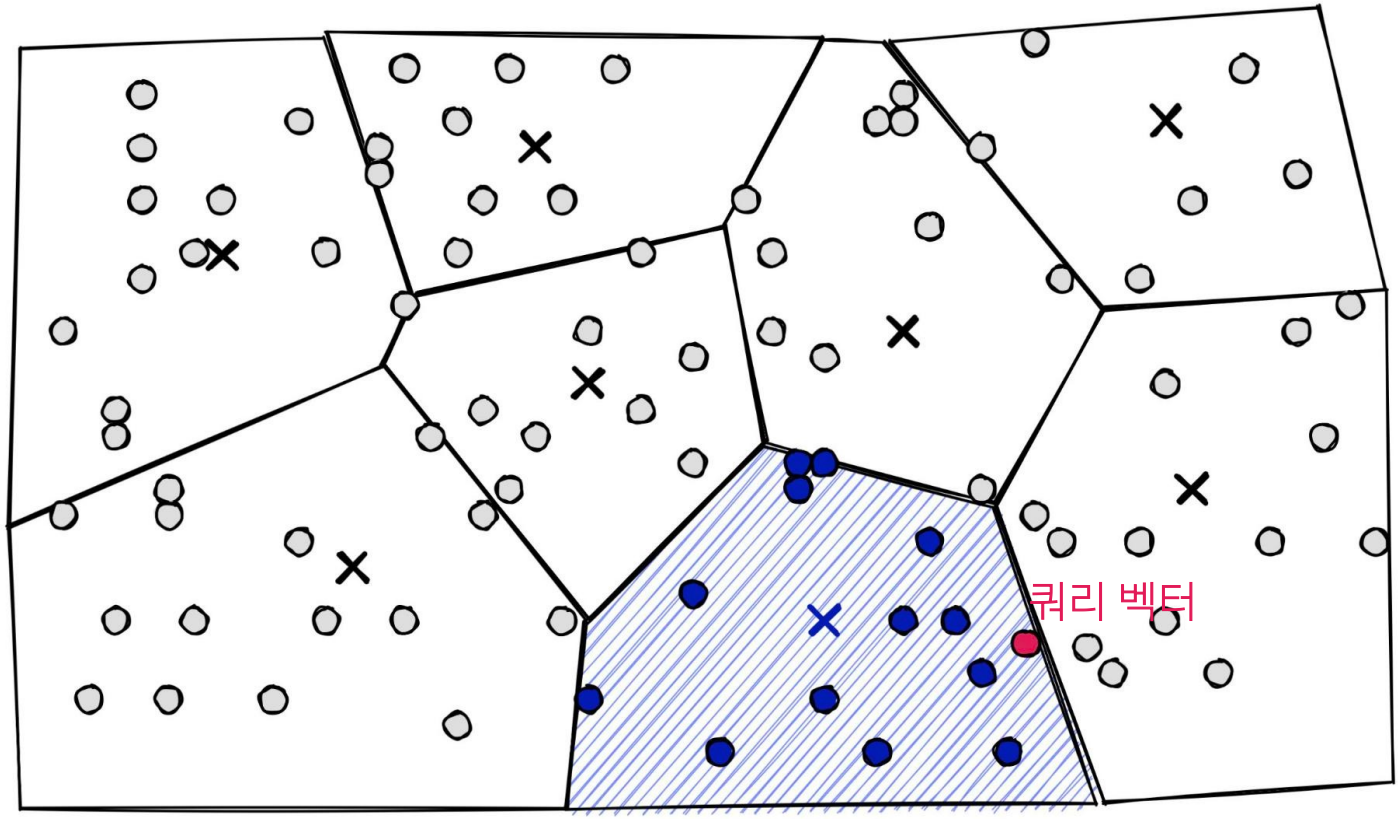
IVF

- Inverted File (Inverted index)
  - Voronoi diagram
    - 중심에서 해당 영역의 특정 지점까지의 거리가 해당 지점에서 다른 중심까지의 거리보다 작다는 특성을 가짐



IVF

- Inverted File for nearest search
  - 쿼리 벡터와 가장 가까운 센트로이드를 계산하여, 그 영역 내 벡터와 거리 비교



IVF

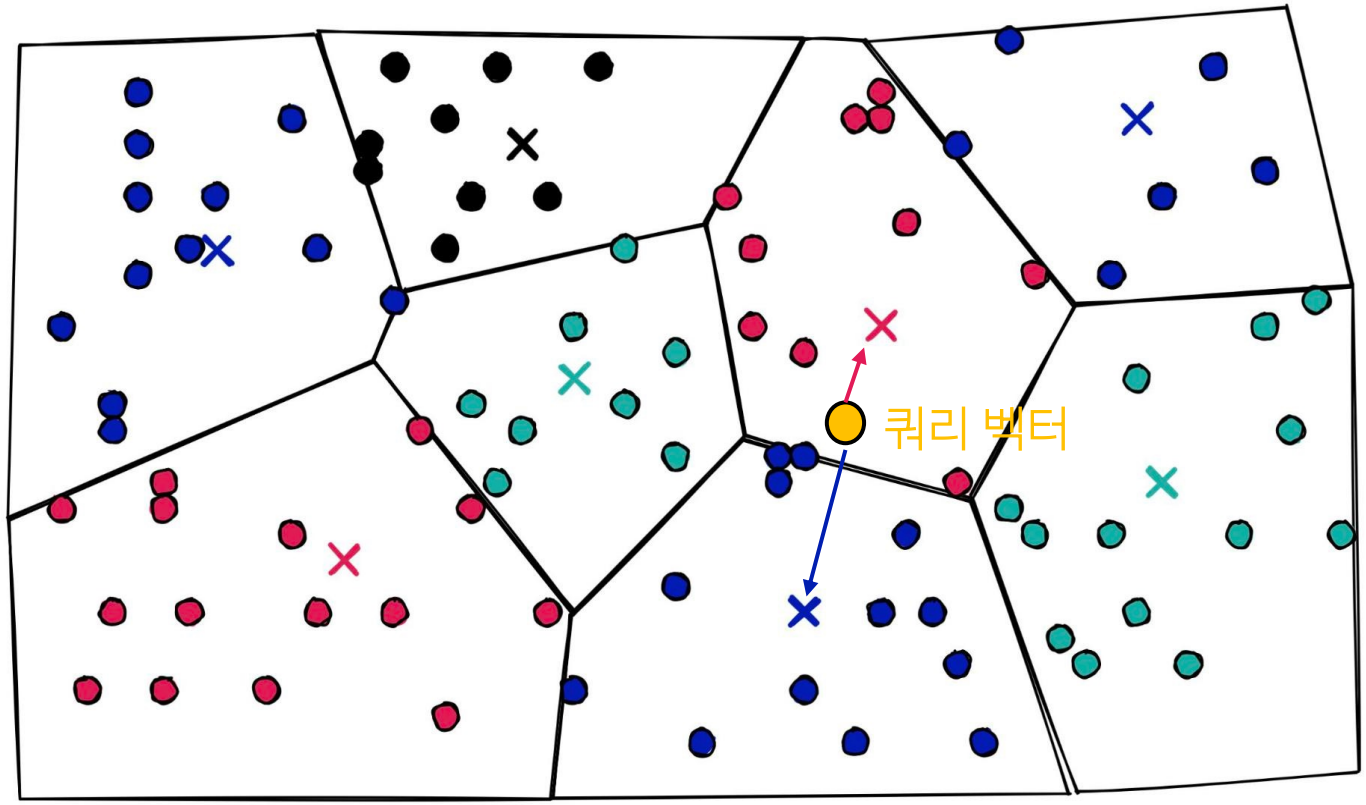
- Inverted File for nearest search
  - Edge problem 발생 가능
  - 파란색 영역의 점들이

    쿼리 벡터와 더 가깝지만,

    센트로이드와의 거리로 인해

    빨간색 영역이 선택되어

    정확한 결과값 반환이 어려움

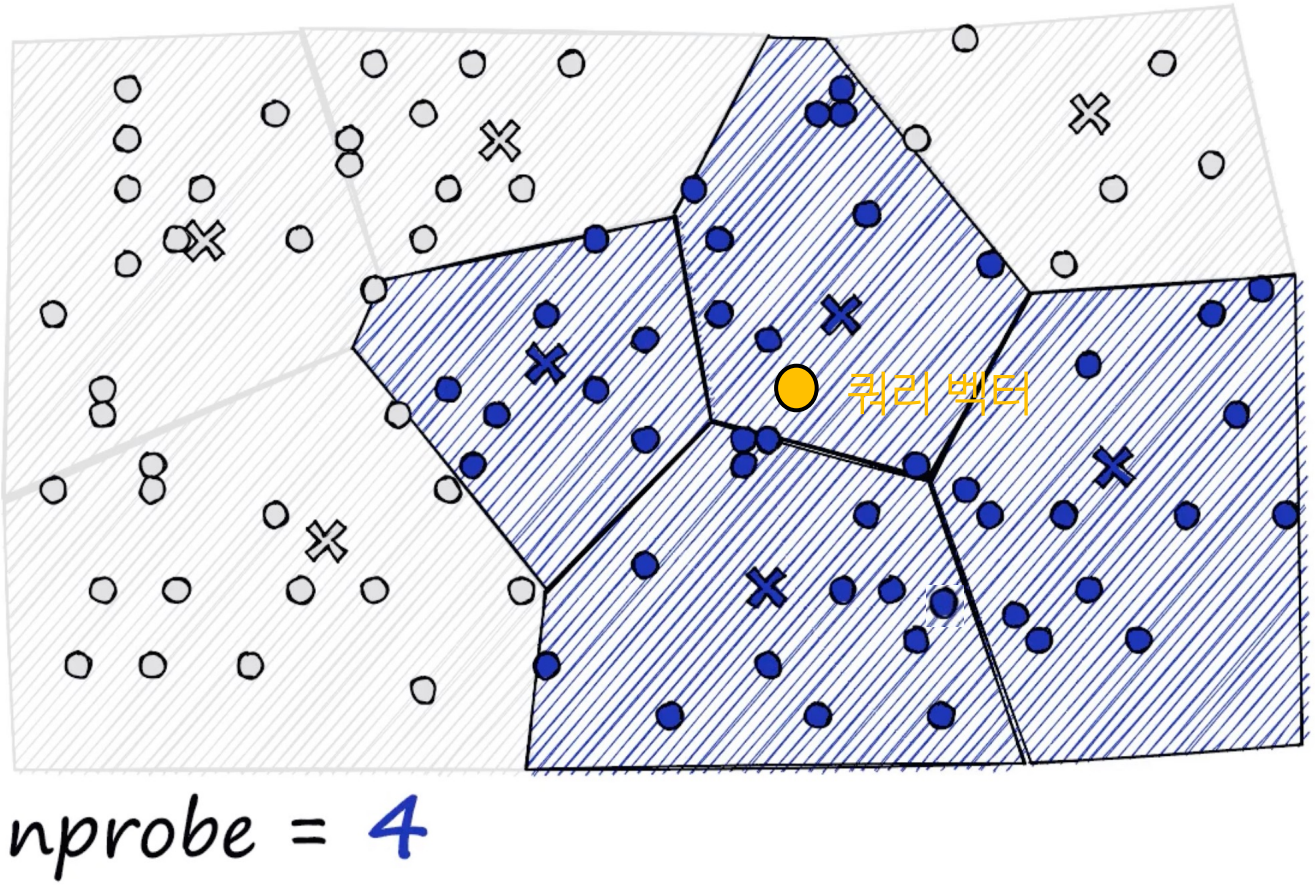


IVF

- Inverted File for nearest search
  - N_probe
  - 이 파라미터를 사용하여

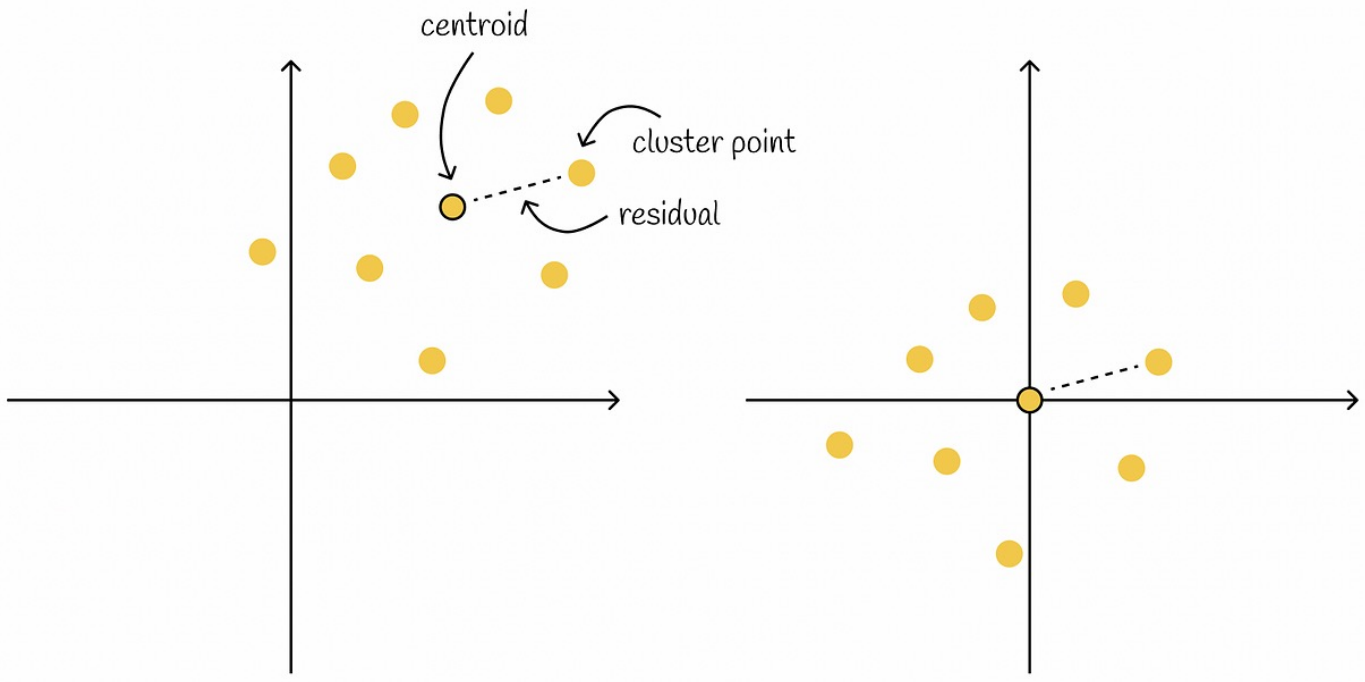
    확인할 영역을 늘리면

    Edge problem 어느정도 해결



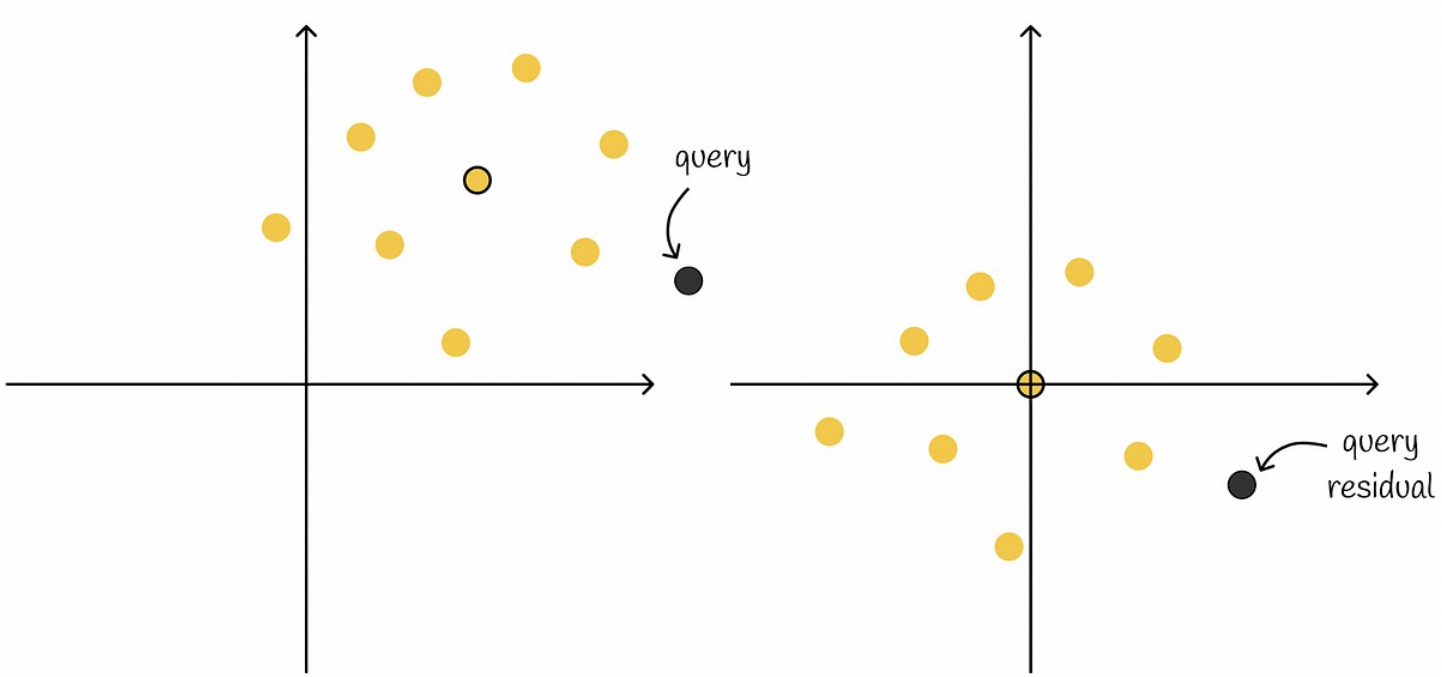
IVFPQ

- IVF + PQ
  - 잔차 : 중심으로부터 벡터의 오프셋
  - K-means를 한 클러스터의 센트로이드는 해당 클러스터에 속한 모든 점들의 평균
  - 모든 벡터에서 클러스터 평균값을 빼면 점들이 0을 중심으로 배치됨



IVFPQ

- IVF + PQ
  - 원래 벡터를 잔차로 대체해도 벡터들의 상대 위치는 변하지 않음
  - 쿼리에서 클러스터 평균을 빼고 난 후, 잔차 중에서 search를 하는 것이 가능함



IVFPQ

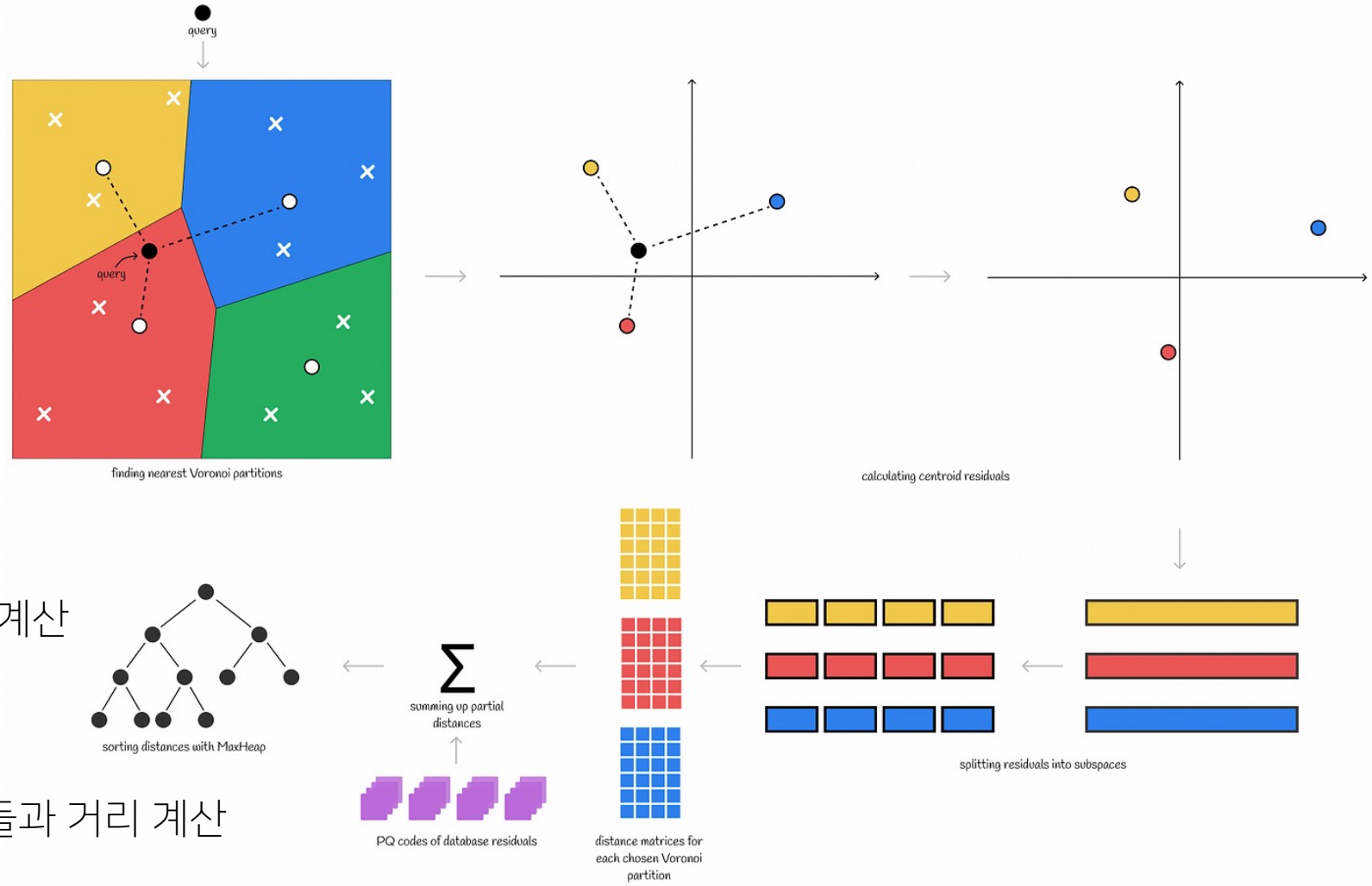
- IVF+PQ
  - Clustering해서 IVF 생성
  - 각 클러스터 내에서 offset 구하기
  - 구한 offset을 Product Quantization
  - IVF에 코드 넣기

IVFPQ

- IVF+PQ for nearest search
  - 쿼리 시,
    - 가까운 영역 선택
    - 영역별 잔차 계산
    - 서브벡터로 나눠

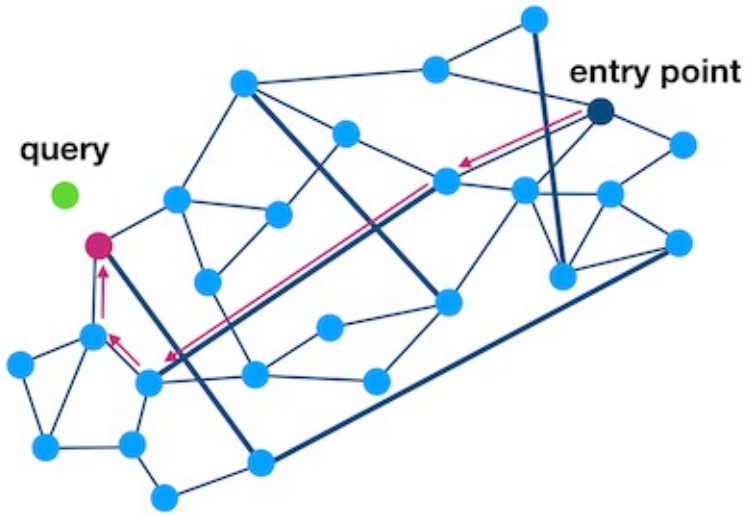
      PQ 센트로이드와의 거리 계산
    - IVF를 사용하여

      해당 센트로이드 내 벡터들과 거리 계산



Graph + PQ

- Graph + PQ
  - Clustering하여 centroid 생성 및 IVF 생성
  - 생성한 centroid로 Nearest Neighbor 그래프 만들기
  - 쿼리 시, 가까운 센트로이드를 선정한 후 Top-k에 대해 구역 내 전역 탐색
  - Centroid 개수 많이 늘려야 함



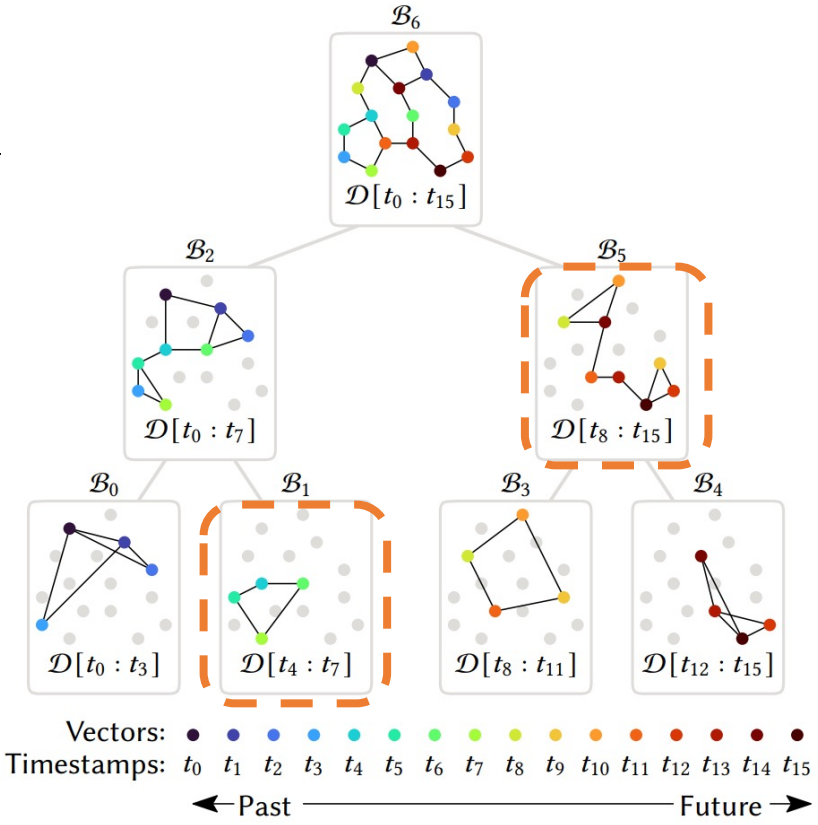
TKNN

- Time-Restricted KNN Search
  - 주어진 시간 구간 내에 있는 가장 유사한 k개 벡터를 효율적으로 찾는 문제

Q. 2000-2023년 사이 '주토피아'와 가장 유사한 영화 3개는?

TKNN

- MBI (Multi Block Indexing)
  - 시간 구간에 따라 데이터를 여러 블럭으로 계층적으로 나누고 그래프를 만들어 indexing
  - 검색 시,
    - 시간 구간에 해당하는 블럭에서 KNN 질의
  - 질의하는 시간 구간의 길이에 상관없이, 뛰어난 성능을 보임

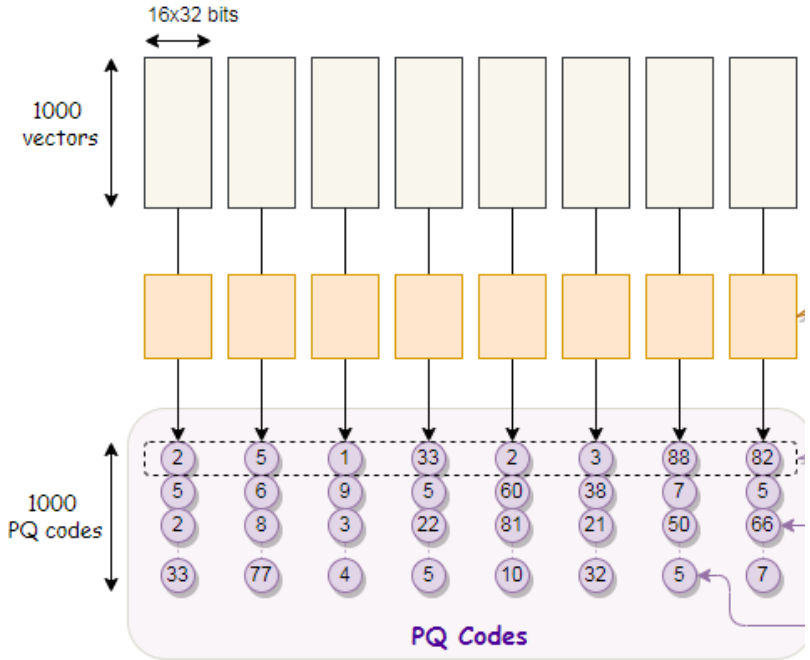


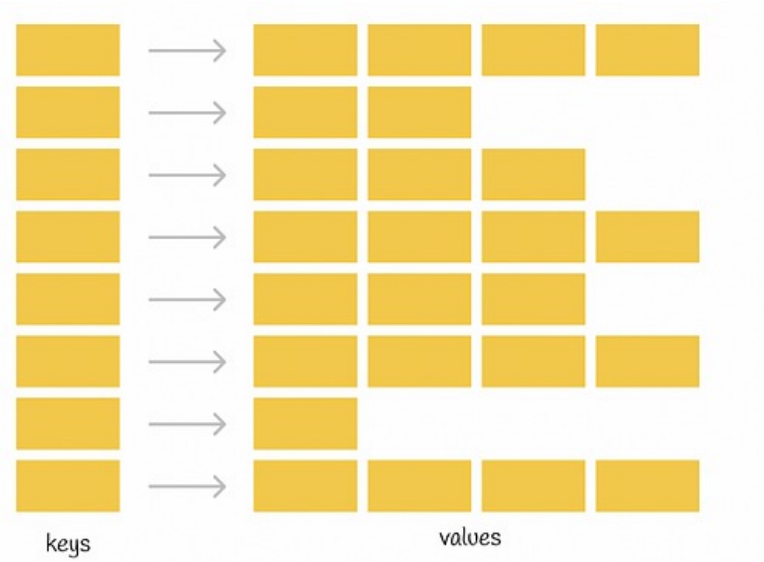
TKNN

- MBI (Multi Block Indexing) 단점
  - 계층적으로 그래프를 생성하므로 indexing에 많은 메모리 사용
  - 새로운 데이터를 append만 할 수 있으며 중간 구간에 삽입 및 삭제가 안됨

수지, 창훈의 IVFPQ

- 간단한 구현이 가능한 indexing 방법
  - 기존 벡터들을 Product Quantization
  - 각 서브벡터 내 센트로이드들을 key로 하여 벡터 index의 IVF 생성





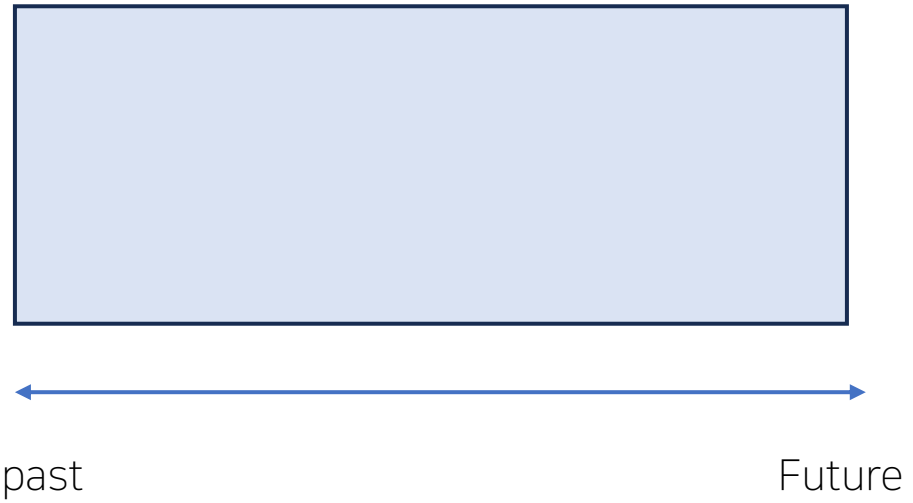
센트로이드 인덱스 실제 벡터의 인덱스

수지, 창훈의 IVFPQ

- IVFPQ for nearest search
  - 쿼리 벡터 PQ
  - 서브벡터별 PQ한 결과가 비슷한 센트로이드들 찾기 (n_probe만큼)
  - 해당 센트로이드에 속한 벡터들과 모두 거리 계산하여 k개만큼 결과 반환

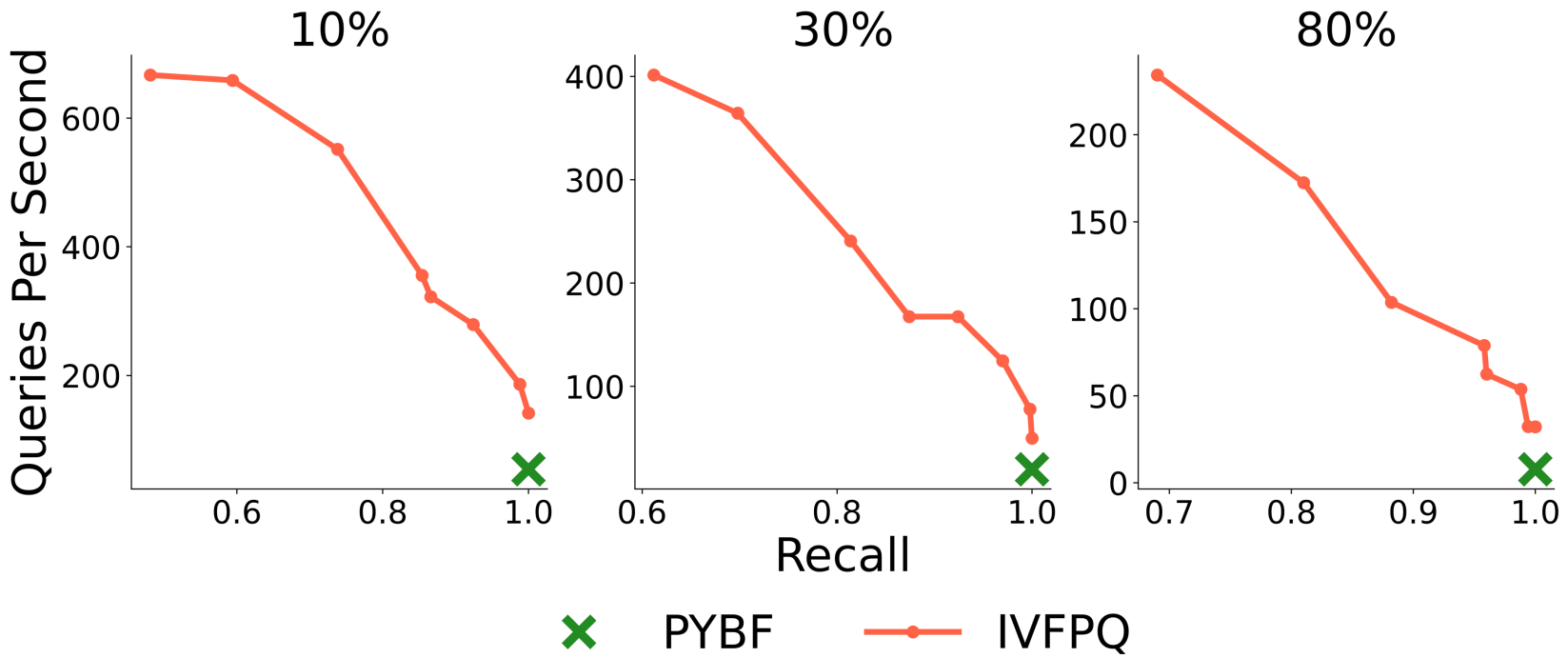
IVFPQ + TKNN

- 아이디어 1
  - 시간 순으로 벡터 정렬 후 한번에 IVFPQ indexing
  - IVF 내 벡터들은 시간순으로 정렬
  - 검색 시,
    - 쿼리 벡터와 유사한 벡터들을 IVF를 이용해 탐색
    - 시간 단위로 이분 탐색하여 필터링된 벡터만 반환
  - 장점 / 단점
    - 기존 MBI (+SF)보다 index 사이즈 감소
    - 삽입 및 삭제 어려움



IVFPQ + TKNN

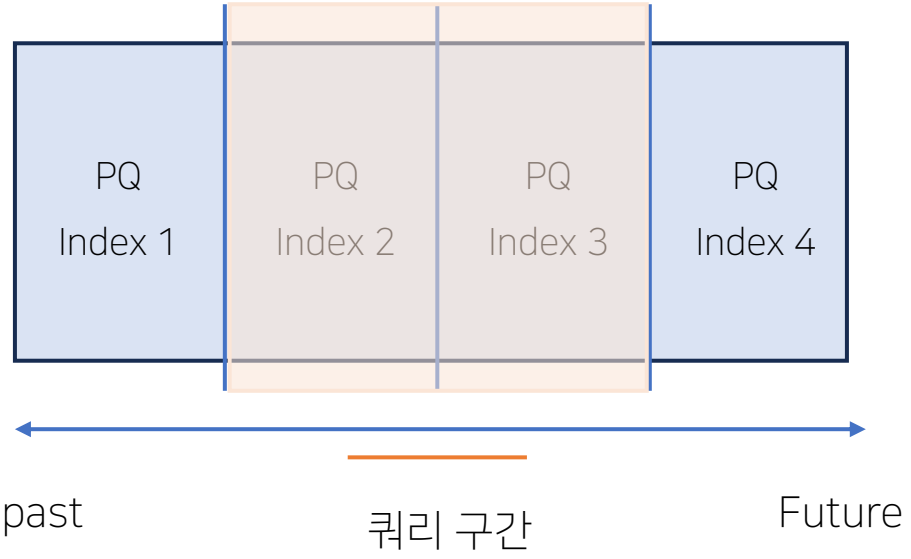
- 현재 실험 결과
  - SIFT 데이터 32000개 잘라서 사용한 결과, 모든 시간 구간에서 Bruteforce 방법보다 성능 좋음
  - 인덱스 크기 약 1MB (MBI 한 층 : 7MB)



IVFPQ + TKNN

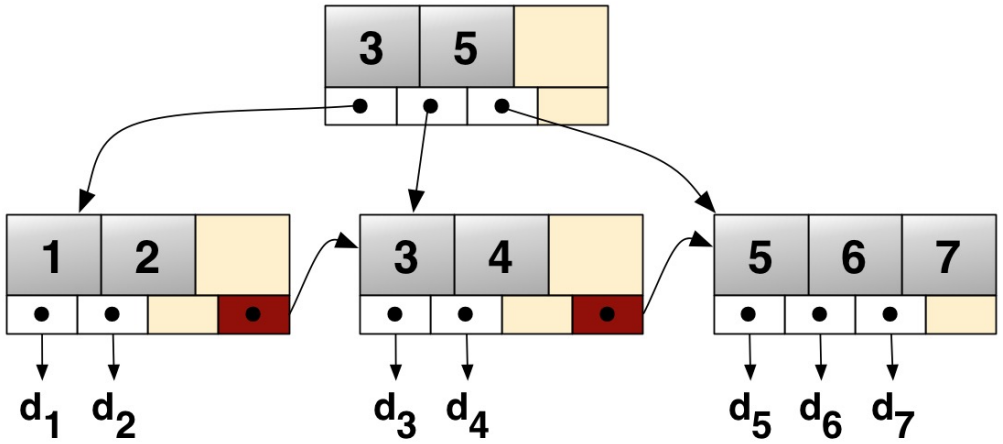
- 아이디어 2
  - 시간 순으로 벡터 정렬 후 구간을 나눠서 따로 indexing
  - 검색 시,
    - 이분 탐색으로 시간 시작점 및 끝점 탐색
    - 해당 구간에 속하는 부분만 탐색하여 반환
  - 장점 / 단점
    - 삽입 및 삭제가 비교적 쉬움

      (해당 시간 구간만 다시 인덱싱하면 됨)



IVFPQ + TKNN

- 아이디어 3
  - B-tree, B+ tree에서 아이디어를 차용하여 구조화
  - 장점 / 단점
    - 이분 탐색보다 빠르게 시간 구간 찾기 가능
    - 삽입 및 삭제 쉬움



B+ tree 예시

감사합니다