Database Assignment #1

Top-k Query Processing

PPT source - Sanjay Kulhari

Objects, Attributes and Scores

- **Each object X_i has m scores $(r_{i1},r_{i2},\ldots,r_{im})$, one for each of m attributes.**
- **Objects are listed, for each attribute sorted by score.**
- **Each object is assigned an overall score by combining the attribute score using aggregate function or combining rule.**
- **Aim: Determine *k* objects with the highest overall score.**

	R_1	R_2	R_3
X_1	1	0.3	0.2
X_2	0.8	0.8	0
X_3	0.5	0.7	0.6
X_4	0.3	0.2	0.8
X_5	0.1	0.1	0.1

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Querying Fuzzy Data - Example

- Given the following relational structure

	R_1	R_2	R_3
X_1	1	0.3	0.2
X_2	0.8	0.8	0
X_3	0.5	0.7	0.6
X_4	0.3	0.2	0.8
X_5	0.1	0.1	0.1

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

- Query: Select top-2 for the sum aggregate function
  Monotonicity property: An aggregation function t is monotone
  - if $t(x_1, \ldots, x_m) \leq t(x'_1, \ldots, x'_m)$ whenever $x_i \leq x'_i$ for every i.

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_1	1.5

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_1	1.5
X_2	1.6

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_1	1.5
X_2	1.6
X_3	1.8

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_1	1.5
X_2	1.6
X_3	1.8
X_4	1.3

Naïve Algorithm

- 1. Compute overall score for every object by looking into each sorted list.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_1	1.5
X_2	1.6
X_3	1.8
X_4	1.3
X_5	0.3

Naïve Algorithm

- 2. Return *k* objects with the highest overall score.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3
X_5	0.3

Return top-2 objects

Fagin’s Algorithm

Fagin's Algorithm

- 1. Sequentially access all the sorted lists in parallel until there are k objects that have been seen in all lists.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

- 1. Sequentially access all the sorted lists in parallel until there are k objects that have been seen in all lists.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

- 1. Sequentially access all the sorted lists in parallel until there are k objects that have been seen in all lists.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

- **1. Sequentially access all the sorted lists in parallel until there are k objects that have been seen in all lists.**

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Since k = 2, and X_1 and X_3 have been seen in all the 3 lists

Fagin’s Algorithm

- 2. Perform random accesses to obtain the scores of all seen objects

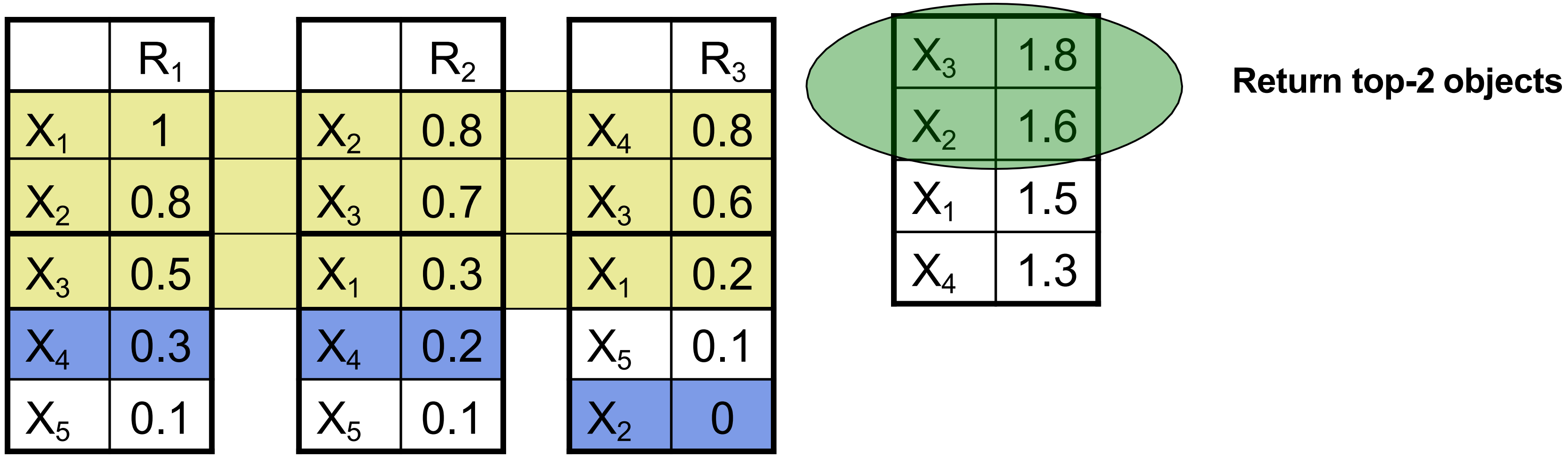
	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Fagin’s Algorithm

- 3. Compute score for all objects and return the top-k



	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

X_3	1.8
X_2	1.6
X_1	1.5
X_4	1.3

Return top-2 objects

Threshold Algorithm

Threshold Algorithm

- 1. Access the elements sequentially

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

Threshold Algorithm

- **At each sequential access**
  - (a) Set the threshold t to be the aggregate of the scores seen in this access.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

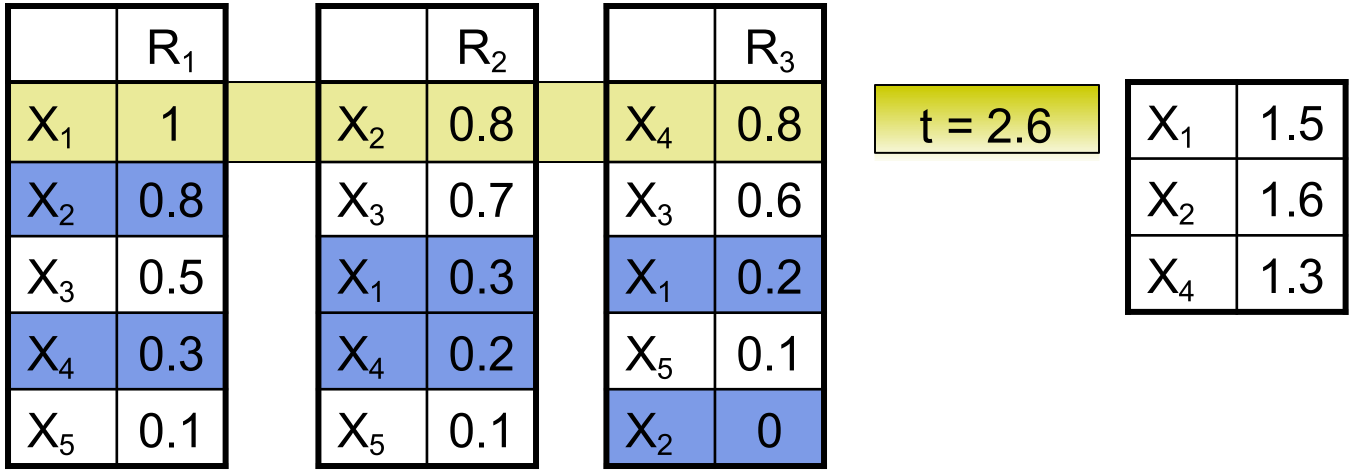
	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 2.6

Threshold Algorithm

- **At each sequential access**
  - (b) Do random accesses and compute the scores of the seen objects.



	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 2.6

X_1	1.5
X_2	1.6
X_4	1.3

Threshold Algorithm

- **At each sequential access**
  - (c) Maintain a list of top-k objects seen so far

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 2.6

X_2	1.6
X_1	1.5

Threshold Algorithm

- **At each sequential access**
  - (d) Stop, when the scores of the top-k are greater or equal to the threshold.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 2.1$

X_3	1.8
X_2	1.6

Threshold Algorithm

- **At each sequential access**
  - (d) Stop, when the scores of the top-k are greater or equal to the threshold.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

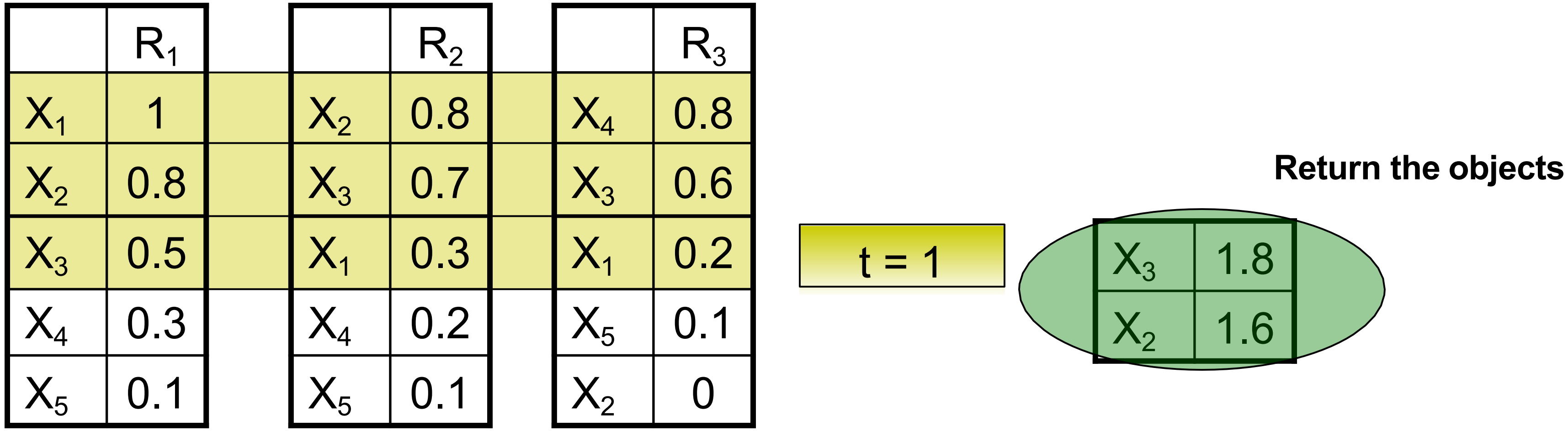
	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

t = 1

X_3	1.8
X_2	1.6

Threshold Algorithm

- 2. Return the top-k seen so far



	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

$t = 1$

Return the objects

X_3	1.8
X_2	1.6

No Random Access Algorithm

No Random Access Algorithm

- 1. Access sequentially all lists in parallel until there are k objects for which the lower bound is higher than the upper bound of all other objects.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

	LB	UB
X_1	1	2.6
X_2	.8	2.6
X_4	.8	2.6

No Random Access Algorithm

- 1. Access sequentially all lists in parallel until there are k objects for which the lower bound is higher than the upper bound of all other objects.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

	LB	UB
X_2	1.6	2.2
X_3	1.3	2.1
X_1	1	2.3
X_4	0.8	2.3

No Random Access Algorithm

- 1. Access sequentially all lists in parallel until there are k objects for which the lower bound is higher than the upper bound of all other objects.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

	LB	UB
X_3	1.8	1.8
X_2	1.6	1.8
X_1	1.5	1.5
X_4	0.8	1.6

No Random Access Algorithm

- 2. Return top-k objects for which the lower bound is higher than the upper bound of all other objects.

	R_1
X_1	1
X_2	0.8
X_3	0.5
X_4	0.3
X_5	0.1

	R_2
X_2	0.8
X_3	0.7
X_1	0.3
X_4	0.2
X_5	0.1

	R_3
X_4	0.8
X_3	0.6
X_1	0.2
X_5	0.1
X_2	0

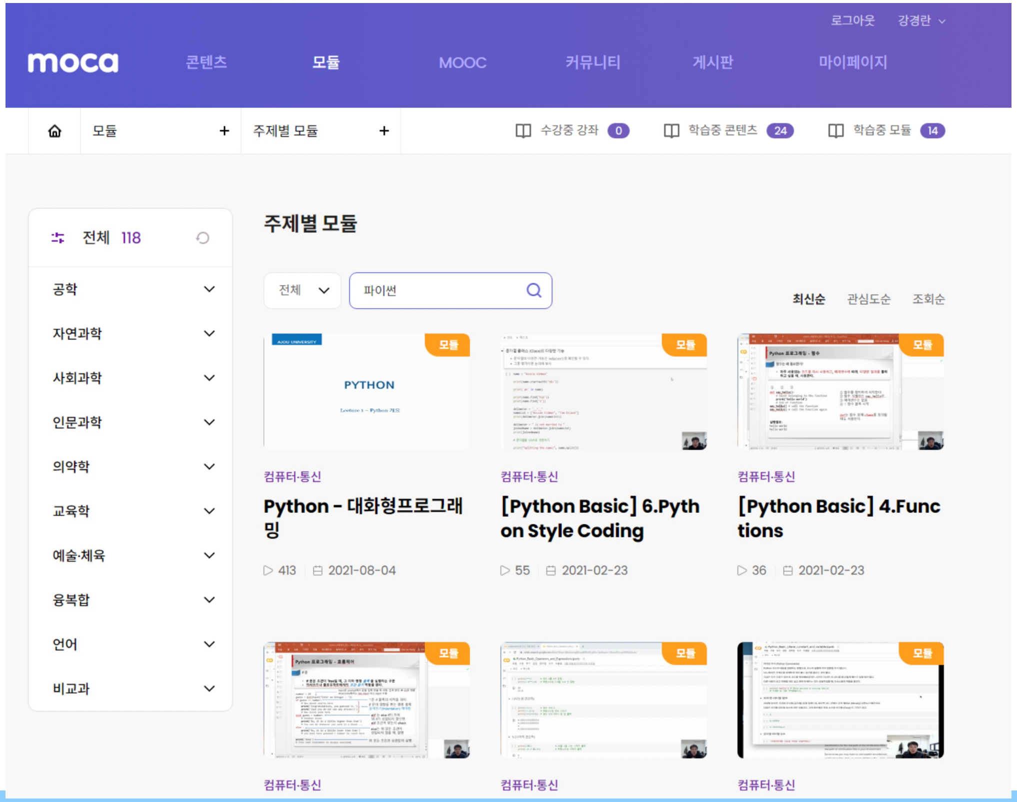
	LB	UB
X_3	1.8	1.8
X_2	1.6	1.8
X_1	1.5	1.5
X_4	0.8	1.6

Return top-2 objects

Tips for Assignment #2

Python

- https://moca.ajou.ac.kr
- 신승훈 교수님 & 최재영 교수님의 python 강의 동영상



주제별 모듈

전체 | 파이썬

컴퓨터·통신
Python - 대화형프로그래밍
▷ 413 2021-08-04

컴퓨터·통신
[Python Basic] 6.Python Style Coding
▷ 55 2021-02-23

컴퓨터·통신
[Python Basic] 4.Functions
▷ 36 2021-02-23

컴퓨터·통신

컴퓨터·통신

컴퓨터·통신

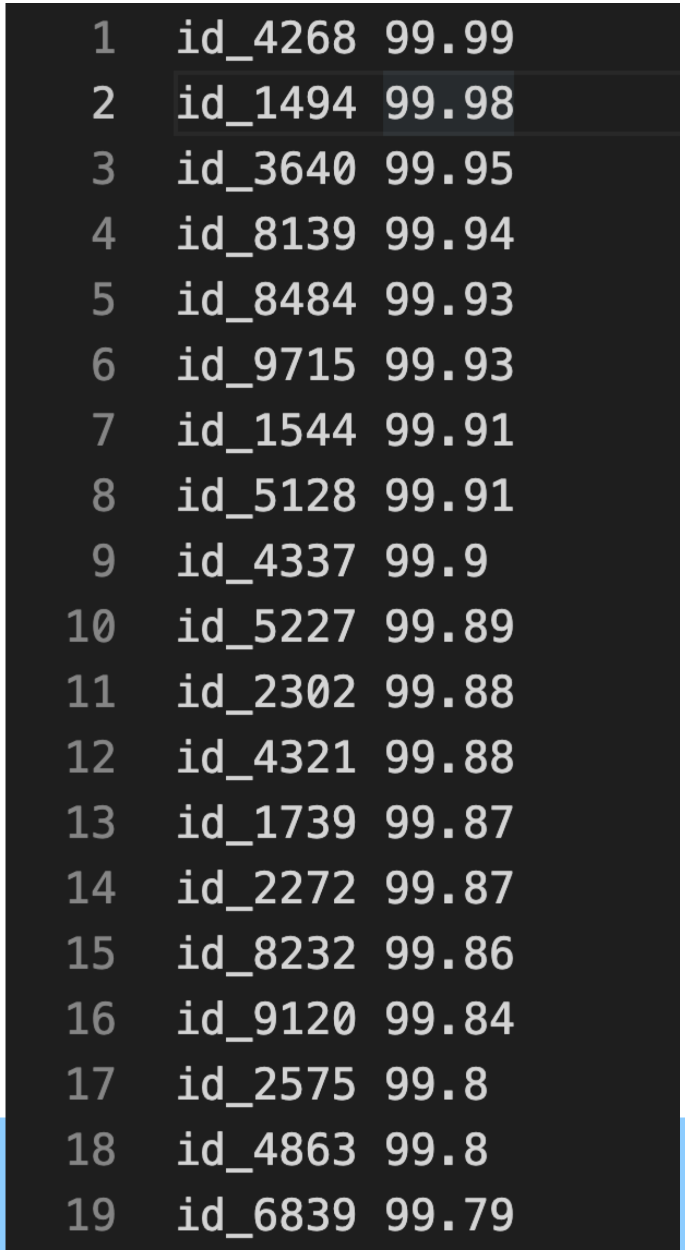
Structure

- **202312345**
    - **compare_alg.py**

    - sorted_score_0.txt
    - …
    - sorted_score_9.txt

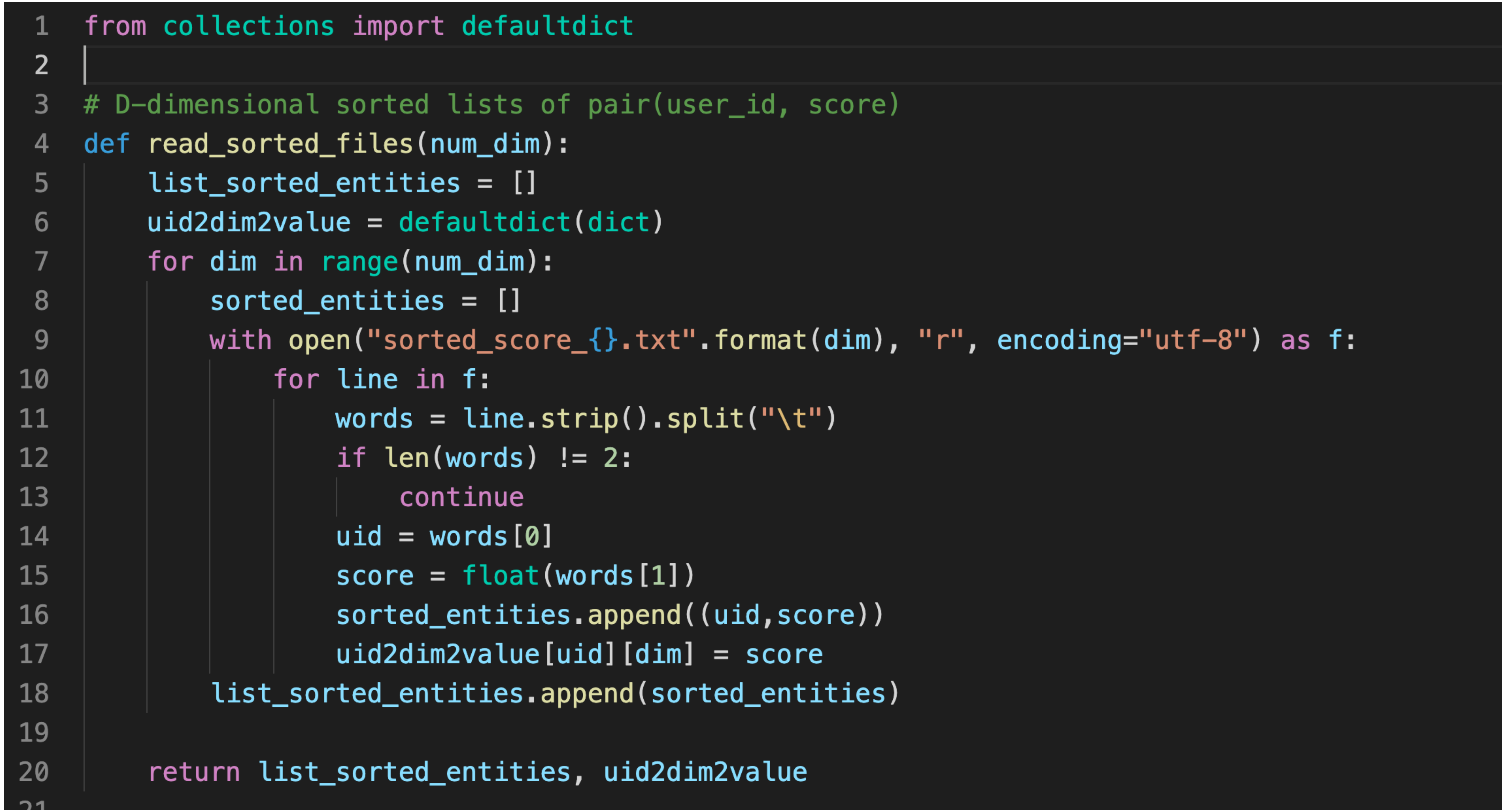
    - **topk.py**

Raw Data File (sorted_score_0.txt)



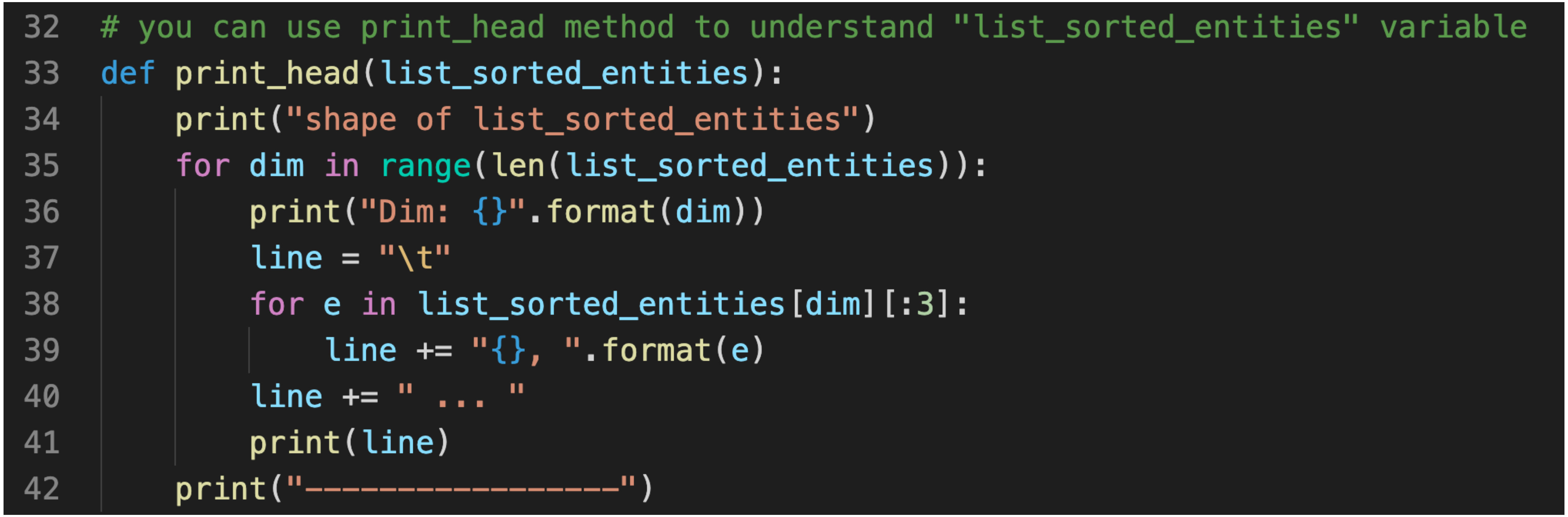
```
id_4268 99.99
id_1494 99.98
id_3640 99.95
id_8139 99.94
id_8484 99.93
id_9715 99.93
id_1544 99.91
id_5128 99.91
id_4337 99.9
id_5227 99.89
id_2302 99.88
id_4321 99.88
id_1739 99.87
id_2272 99.87
id_8232 99.86
id_9120 99.84
id_2575 99.8
id_4863 99.8
id_6839 99.79
```

compare_alg.py



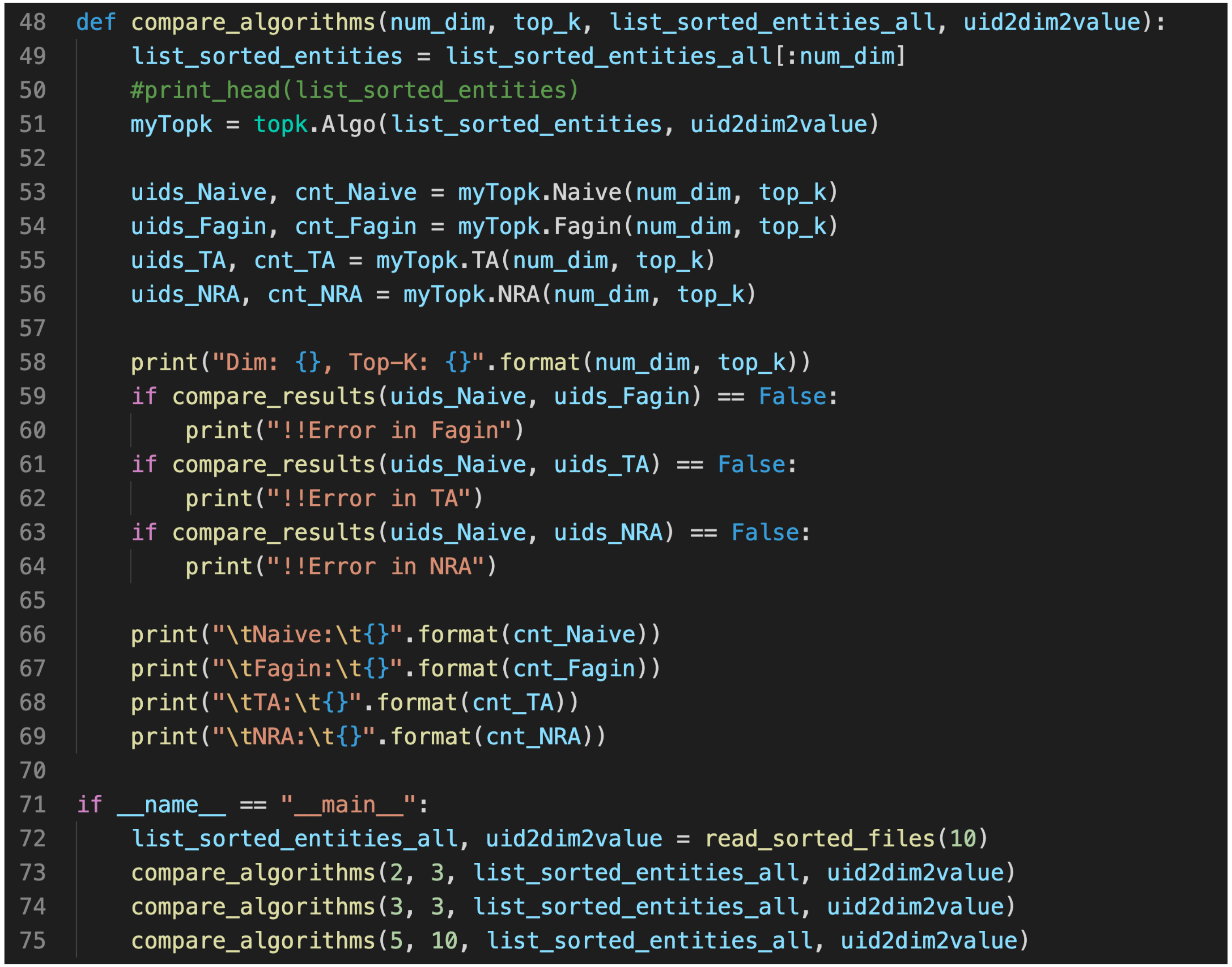
```python
from collections import defaultdict

# D-dimensional sorted lists of pair(user_id, score)
def read_sorted_files(num_dim):
    list_sorted_entities = []
    uid2dim2value = defaultdict(dict)
    for dim in range(num_dim):
        sorted_entities = []
        with open("sorted_score_{}.txt".format(dim), "r", encoding="utf-8") as f:
            for line in f:
                words = line.strip().split("\t")
                if len(words) != 2:
                    continue
                uid = words[0]
                score = float(words[1])
                sorted_entities.append((uid,score))
                uid2dim2value[uid][dim] = score
        list_sorted_entities.append(sorted_entities)

    return list_sorted_entities, uid2dim2value
```

compare_alg.py



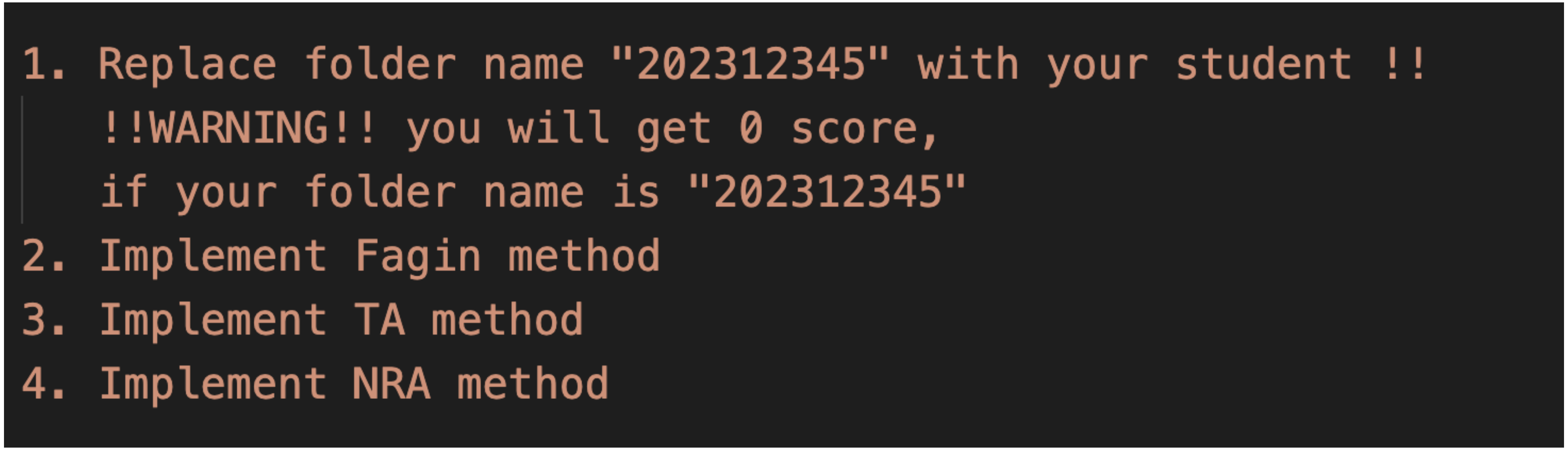
```python
# you can use print_head method to understand "list_sorted_entities" variable
def print_head(list_sorted_entities):
    print("shape of list_sorted_entities")
    for dim in range(len(list_sorted_entities)):
        print("Dim: {}".format(dim))
        line = "\t"
        for e in list_sorted_entities[dim][:3]:
            line += "{}, ".format(e)
        line += " ... "
        print(line)
    print("-----------------")
```

compare_alg.py



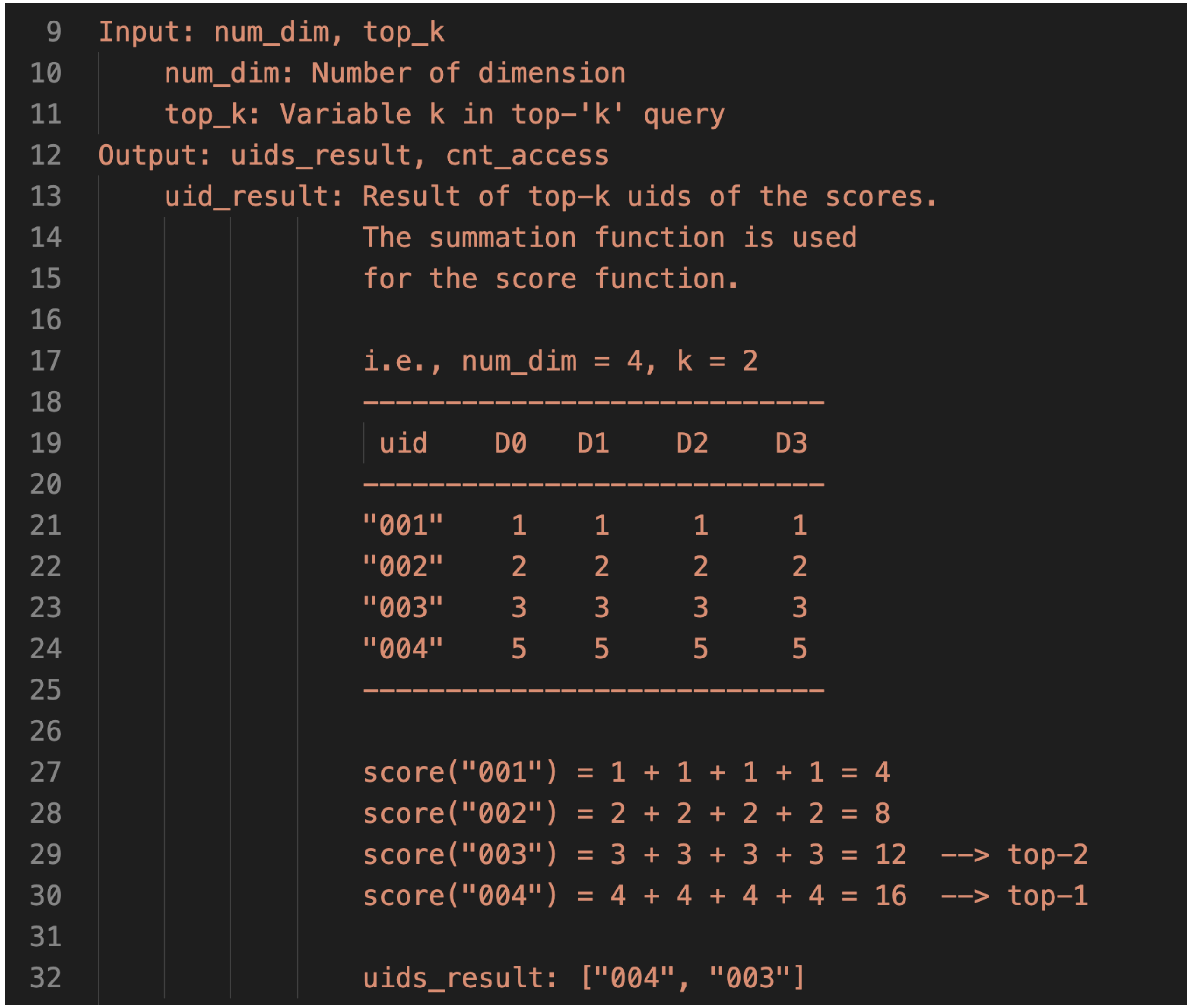
```python
def compare_algorithms(num_dim, top_k, list_sorted_entities_all, uid2dim2value):
    list_sorted_entities = list_sorted_entities_all[:num_dim]
    #print_head(list_sorted_entities)
    myTopk = topk.Algo(list_sorted_entities, uid2dim2value)

    uids_Naive, cnt_Naive = myTopk.Naive(num_dim, top_k)
    uids_Fagin, cnt_Fagin = myTopk.Fagin(num_dim, top_k)
    uids_TA, cnt_TA = myTopk.TA(num_dim, top_k)
    uids_NRA, cnt_NRA = myTopk.NRA(num_dim, top_k)

    print("Dim: {}, Top-K: {}".format(num_dim, top_k))
    if compare_results(uids_Naive, uids_Fagin) == False:
        print("!!Error in Fagin")
    if compare_results(uids_Naive, uids_TA) == False:
        print("!!Error in TA")
    if compare_results(uids_Naive, uids_NRA) == False:
        print("!!Error in NRA")

    print("\tNaive:\t{}".format(cnt_Naive))
    print("\tFagin:\t{}".format(cnt_Fagin))
    print("\tTA:\t{}".format(cnt_TA))
    print("\tNRA:\t{}".format(cnt_NRA))

if __name__ == "__main__":
    list_sorted_entities_all, uid2dim2value = read_sorted_files(10)
    compare_algorithms(2, 3, list_sorted_entities_all, uid2dim2value)
    compare_algorithms(3, 3, list_sorted_entities_all, uid2dim2value)
    compare_algorithms(5, 10, list_sorted_entities_all, uid2dim2value)
```

topk.py



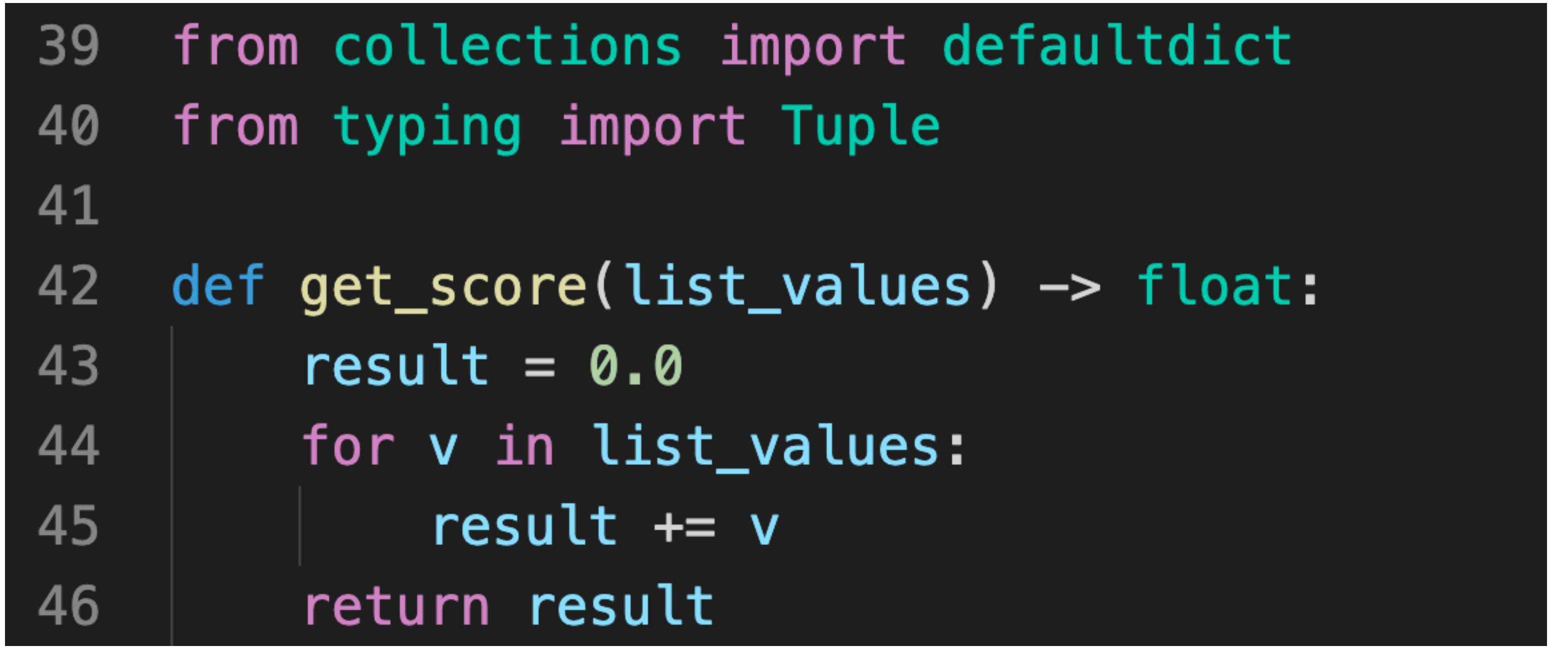
```
1. Replace folder name "202312345" with your student !!
   !!WARNING!! you will get 0 score,
   if your folder name is "202312345"
2. Implement Fagin method
3. Implement TA method
4. Implement NRA method
```

topk.py



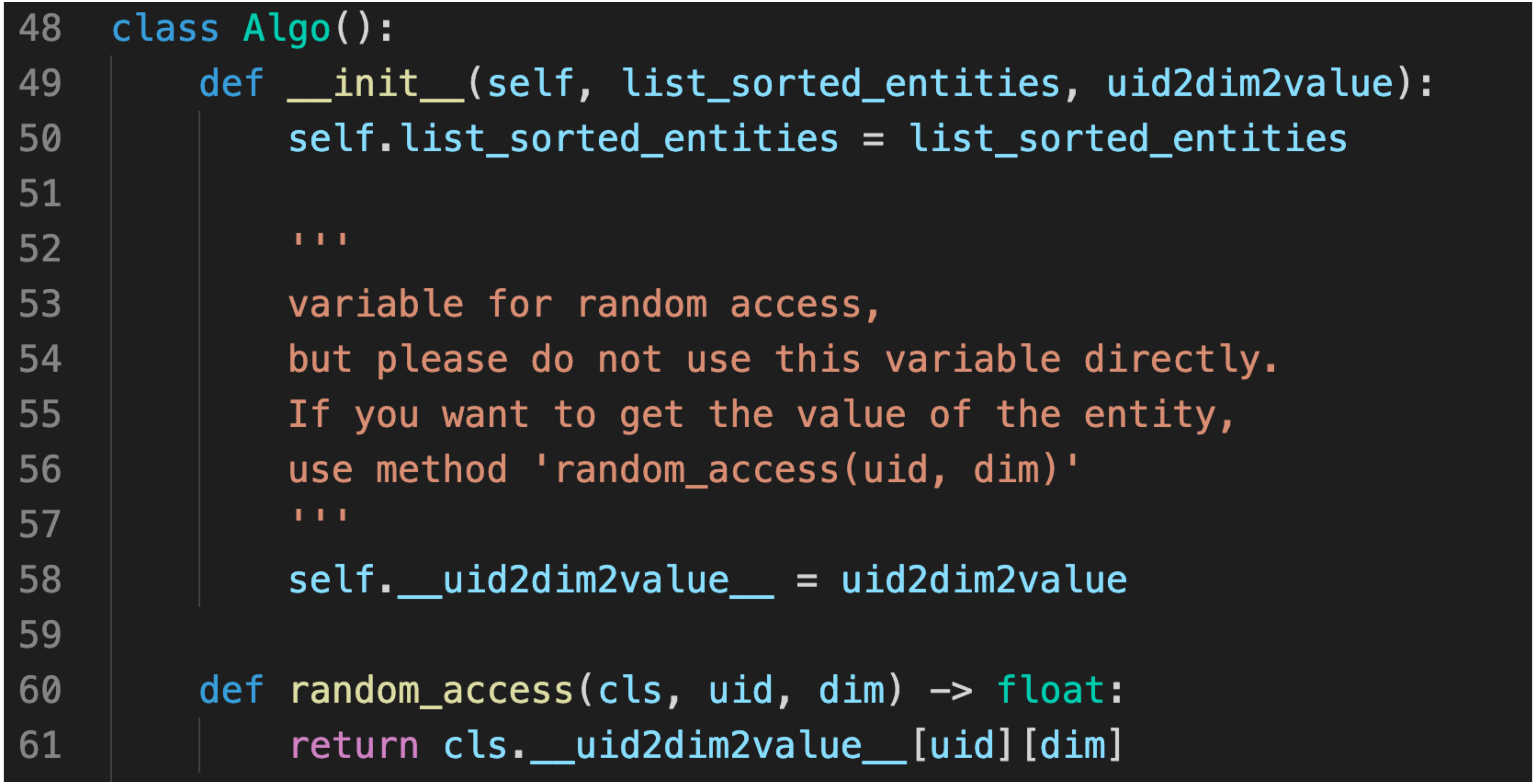
```
Input: num_dim, top_k
    num_dim: Number of dimension
    top_k: Variable k in top-'k' query
Output: uids_result, cnt_access
    uid_result: Result of top-k uids of the scores.
                The summation function is used
                for the score function.

                i.e., num_dim = 4, k = 2
                ----------------------------
                 uid    D0   D1    D2    D3
                ----------------------------
                "001"    1    1     1     1
                "002"    2    2     2     2
                "003"    3    3     3     3
                "004"    5    5     5     5
                ----------------------------

                score("001") = 1 + 1 + 1 + 1 = 4
                score("002") = 2 + 2 + 2 + 2 = 8
                score("003") = 3 + 3 + 3 + 3 = 12  --> top-2
                score("004") = 4 + 4 + 4 + 4 = 16  --> top-1

                uids_result: ["004", "003"]
```

topk.py [get_score]



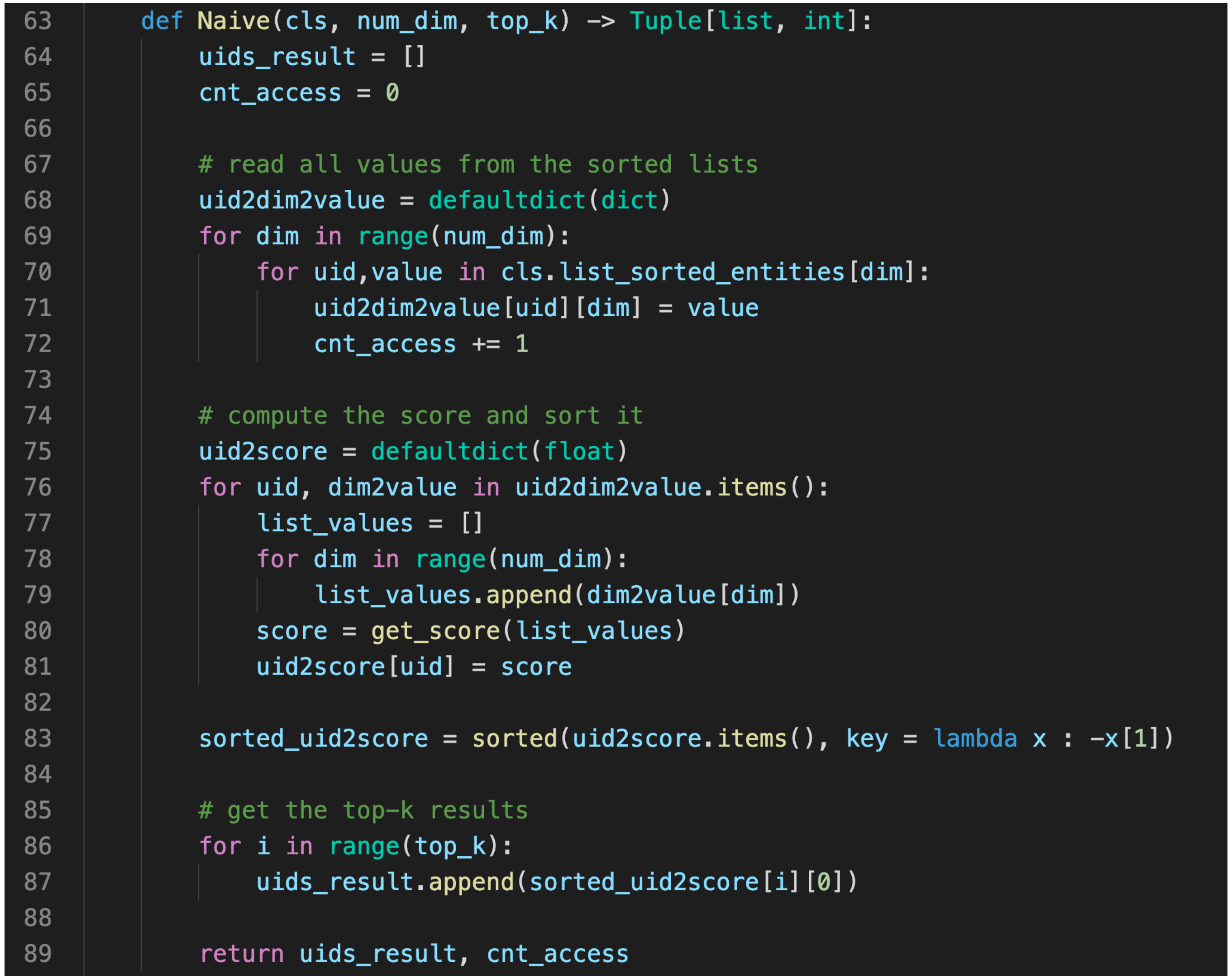
```python
from collections import defaultdict
from typing import Tuple

def get_score(list_values) -> float:
    result = 0.0
    for v in list_values:
        result += v
    return result
```

topk.py [random_access]



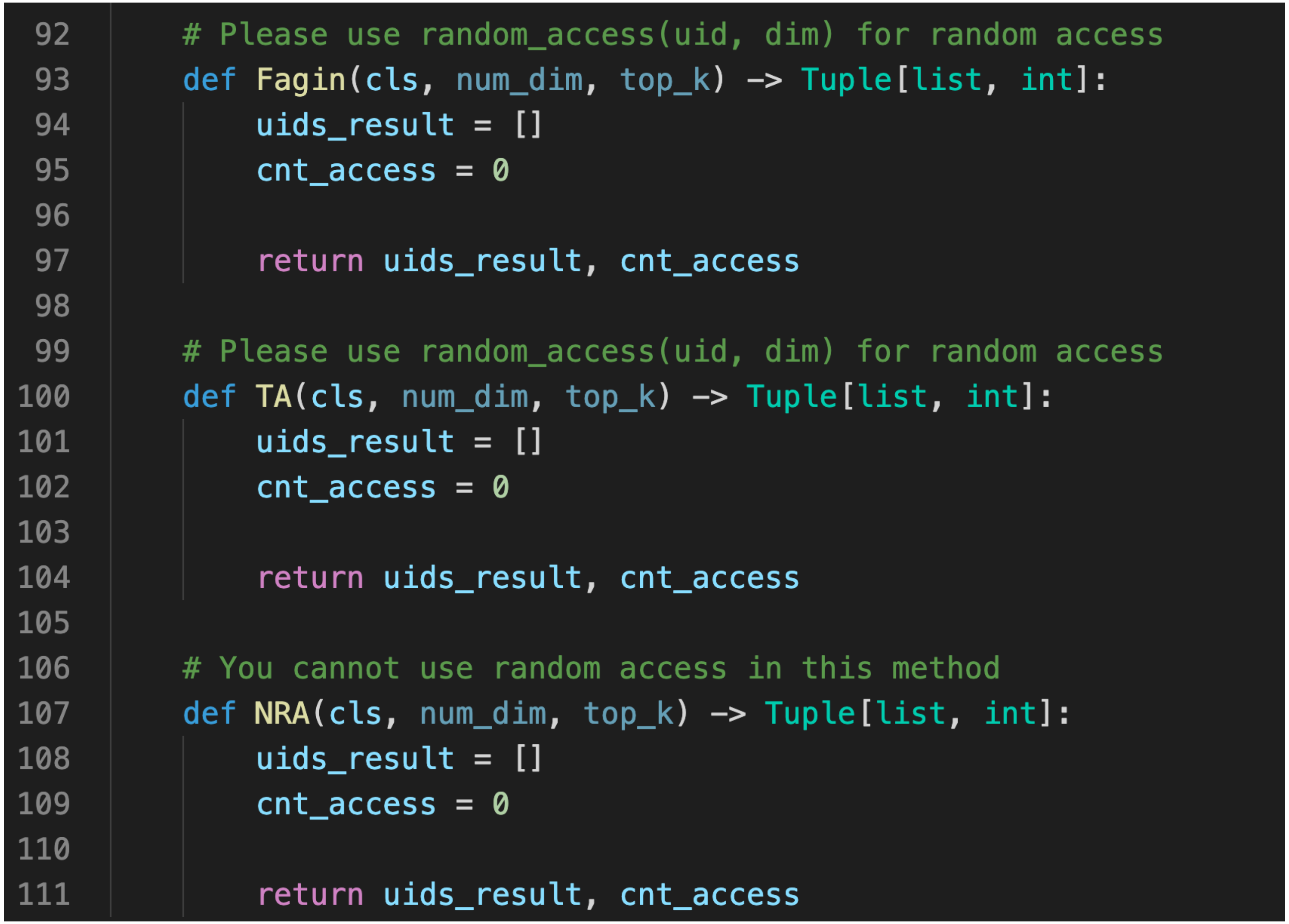
```python
class Algo():
    def __init__(self, list_sorted_entities, uid2dim2value):
        self.list_sorted_entities = list_sorted_entities

        '''
        variable for random access,
        but please do not use this variable directly.
        If you want to get the value of the entity,
        use method 'random_access(uid, dim)'
        '''
        self.__uid2dim2value__ = uid2dim2value

    def random_access(cls, uid, dim) -> float:
        return cls.__uid2dim2value__[uid][dim]
```

topk.py [Naive: gift]



```python
    def Naive(cls, num_dim, top_k) -> Tuple[list, int]:
        uids_result = []
        cnt_access = 0

        # read all values from the sorted lists
        uid2dim2value = defaultdict(dict)
        for dim in range(num_dim):
            for uid,value in cls.list_sorted_entities[dim]:
                uid2dim2value[uid][dim] = value
                cnt_access += 1

        # compute the score and sort it
        uid2score = defaultdict(float)
        for uid, dim2value in uid2dim2value.items():
            list_values = []
            for dim in range(num_dim):
                list_values.append(dim2value[dim])
            score = get_score(list_values)
            uid2score[uid] = score

        sorted_uid2score = sorted(uid2score.items(), key = lambda x : -x[1])

        # get the top-k results
        for i in range(top_k):
            uids_result.append(sorted_uid2score[i][0])

        return uids_result, cnt_access
```

topk.py [To Do]



```python
    # Please use random_access(uid, dim) for random access
    def Fagin(cls, num_dim, top_k) -> Tuple[list, int]:
        uids_result = []
        cnt_access = 0

        return uids_result, cnt_access

    # Please use random_access(uid, dim) for random access
    def TA(cls, num_dim, top_k) -> Tuple[list, int]:
        uids_result = []
        cnt_access = 0

        return uids_result, cnt_access

    # You cannot use random access in this method
    def NRA(cls, num_dim, top_k) -> Tuple[list, int]:
        uids_result = []
        cnt_access = 0

        return uids_result, cnt_access
```