Final set of constraints for data, query and output formats

General (FAQ)

- For read dataset:
  - the alphabet size is 5
  - the dataset consists of ~ 15.000.000 strings
- For geonames dataset:
  - each string is at least of length 5 (in order to avoid a large result set for self join with larger k)
  - each string is at most of length 64
  - the alphabet size is smaller than 255
  - the dataset consists of ~ 1.000.000 strings

Track I (search)

Input format of queries

- Each line corresponds to one query
- Format: ID:query,k
  - ID is a **unique** identifier of the query in the file, starting from 1
  - Query is a string, e.g. Berlin
  - k depends on the dataset (reads={0,4,8,12,16}, geonames={0,1,2,3,4})
  - In total we will have 100.000 queries for each dataset (reads/geonames)
- Each line is terminated by a line break
- Please note that there are no whitespaces between ":" and ","
- The query file is terminated with the line break of the last query only; no further line breaks
- Example input (syntactical example only; no relation to the real data):

  1:Berlin,2
  2:Paris,1
  3:Beijing,3

Output format of results

- Each line corresponds to a (partial) query answer
- Format: ID:matchID1,matchID2,…,matchIDn
  - ID is the identifier from the query file; matchIDi is the ID of one matching string
- Each line is terminated by a line break
- Please note that there are no whitespaces between ":" and ","
- The result file is terminated with the line break of the answer only; no further line breaks
- Example output (syntactical example only; no relation to the real data):

  1:5,11
  2:14,10
  1:11,12
- In the above example there is no result for query 3 (in this case no line with query ID 3 exists). For query 1 (Berlin) the results are: 5,11, and 12.

Constraints on the results

- There are no constraints on the order of results
- There are no uniqueness constraints, i.e. a match for a query can occur several times in the result file
- We allow partial matches, i.e. the result file may contain several lines with the same query ID. For evaluating correctness, we will take the union of all the lines with the same ID
- The union of results for each query ID should be correct and complete, i.e. missing matches or wrong matches will lead to disqualification of your program

Communication with the test environment

1. Your program will be executed with a path to the data file (first parameter) and a link to the query file (second parameter),
   e.g. ./program /competition/track1/reads.txt /competition/track1/queries.txt
2. Initially, when starting your program, the query file does not yet exist, but only the data file
3. After your program is being started, you should create your index for the input data
4. Once your program is finished with the index construction, it should create an empty file at /competition/track1/index_generation_finished.txt
   a. The time between calling your program and the creation of the file will be measured by us and taken as the index creation time.
5. Now your program should wait until a file exists at /competition/track1/queries_available.txt (by polling, or some other means in your programming language). We will copy the actual query file and create the file queries_available.txt on completion of the copying process
6. Once the file queries_available.txt exists, you can start processing the queries
   a. The time between creation of the queries_available.txt and the termination of your program will be measured as query answering time for Track I
   b. The output (=answer to the queries) should go to the file “result_track1.out” in your local directory, i.e. the same directory as your executable

Evaluation

Competing programs will be evaluated based on query answering times. Index creation times will be only shown for information purposes. Please note that we stop your program, if the index creation is taking longer than **three** hours.

Track II (join)

Input format of queries

- The input exists of a single value for k

Output format of results

- Each line corresponds to a (partial) query answer
- Format: stringID:matchID1,matchID2,…,matchIDn
  - stringID is a string identifier from the data file; matchIDi is the ID of one matching string
- Each line is terminated by a line break
- Please note that there are no whitespaces between “:” and “,”
- The result file is terminated with the line break of the answer only; no further line breaks

- Example output(syntactical example only; no relation to the real data):

  10:5,11
  11:10,5
  10:9,5
  …
- In the above example string 10 has k-approximate matches 5, 11, and 9. String 11 has k-approximate matches 5 and 10.

Constraints on the results

- There are no constraints on the order of results
- There are no uniqueness constraints, i.e. an element of the join can occur several times in the result file
- We allow partial matches, i.e. the result file may contain several lines with the same string ID. For evaluating correctness, we will take the union of all the lines with the same ID
- Only programs which compute a correct and complete self-join will be considered as a winner

Communication with the test environment

1. Your program will be executed with a path to the data file (first parameter) and a value for k (second parameter),
   e.g. ./program /competition/track2/reads.txt 1
2. After your program is being started, you should create your index for the k-approximate self-join if necessary, and serialize the join result
   a. The time between starting your program and the termination of your program will be measured as query answering time for Track II
   b. The output (=result of the join) should go to the file “result_track2.out” in your local directory, i.e. the same directory as your executable

Evaluation

Competing programs will be evaluated based on query answering times for the self-join. Index creation time is included.