DBMS Implementation Project Phase 3: Graph Database Operations

Group Members:

Elan Markov

Simarpreet Kaur

Prachi Sharma

Jayanth Kumar Melinavolagerehalli Jayaramaiah

Harshdeep Singh Sandhu

Priyekant Aghi

Shalmali Bhoir

CSE510: Database Management System Implementation

Abstract: This report details the implementation of Phase 3 of the CSE 510 project, in which the graph database created in Phase 2 is extended in order to allow for further graph-based queries. Specifically, this phase involves the implementation of path expression queries, which give pathways between nodes within the graph structure, and triangle queries, which give sets of three-node cycles ("triangles") within the graph. Each of these queries find matches under specified conditions (including values by label, node descriptors, or edge weights) and give a list of all results within the graph that satisfy the query conditions. In order to implement each of these queries, the join operation was needed, which was implemented as with index nested loop join and sort-merge join algorithms, each of which is most effective under certain conditions to be evaluated. A path expression operator was implemented in order to allow the path expression queries to be able to parse the graph and determine whether a given path matches the desired query structure. Finally, persistence functionality was added to preserve data between runs. Each of these components was added to the previously developed Phase 2 graph DBMS built atop the Minibase relational DBMS.

Keywords: Minibase, database management system, join operation, path expression query, triangle query

1 Introduction

This section details the background information of Phase 3 of the Database Management System Implementation (CSE 510) final project. This section details terminology used throughout the document, the problem specification, and core assumptions made in the design of the database management system.

1.1 Terminology

DBMS - Database Management System

ID - Identification number

I/O - Input/Output

Label - A string that represents the name of a given object

Node - A structure that consists of a label, id, and descriptor. Fundamental unit of a graph. Also known as a vertex.

Edge - A structure that consists of a label, id, descriptor, source node, and destination node. Connects two nodes. For the graph consisted here, the edges will be directed - this means that the source node and the destination node are not interchangeable for a given edge; an edge from Node A to Node B is distinct from an edge going from Node B to Node A.

Graph - A structure that consists of a set of nodes and a set of edges connecting nodes. Since edges in this graph are directed edges, the graph in this database is a directed graph.

Query - an operation performed on the database to add, remove, or retrieve data within the database. The terms “node query” and “edge query” use the more standard definition of query, i.e. accessing the database to retrieve entries in a readable format.

Graph Database - A database which stores data in the form of a graph - i.e. in the form of a set of nodes connected by a set of edges.

B-Tree - A tree-based data structure which is an extension of the binary tree, used for read/write operations on a large set of data.

Z-Curve - A structure which orders objects based on temporal locality.

N-D - N-dimensional data, where N is an integer describing the number of inputs to a data vector.

Triangle - Three nodes A,B,C connected by edges such that there is an edge from A to B, B to C, and C to A; i.e. a three-node cycle.

Path - a set of nodes connected by edges; e.g. A, B, C is a path if A connects to B and B connects to C.

Persistence - Data within the DBMS is saved when the program running the DBMS is closed so that it can be used in a future run of the DBMS.

1.2 Problem Specification

In Phase 1, the Minibase relational DBMS framework was explored and tested; in Phase 2, this database was extended to the case of a graph DBMS, where the data is in the form of node points and edges which connect points. In Phase 3, the initial graph DBMS is further extended in order to incorporate join operators and operators and queries that are based on join operations. Index-nested loop join and sort-merge operations are implemented, along with new path expression operators based on those joins and queries which use each of these operators, including path expression and triangle queries. Since join operations are one of the most expensive features of a DBMS (in terms of both runtime and memory usage), it is important to effectively implement joins and join-based operations in order to effectively use this new graph DBMS and to ensure that the queries complete within a reasonable amount of time. In addition, persistence of data between runs of the DBMS program was implemented in order to use a DBMS previously created or to use a DBMS after crash recovery. These features complement the initial graph DBMS functionality created in Phase 2 in order to create a more complete graph DBMS for storing and querying graph-based data.

1.3 Assumptions

Each of the assumptions from Phase 2 are preserved: the graph will be a directed graph, nodes and edges are tuples as described in Section 1.1, input will be correct as described in each respective subsection of Section 2 of this document, and nodes will be inserted before edges between those nodes. In addition, sorting on node and edge labels will be in alphanumerical order even if the label is a number; e.g. “1120” will come before “846” in sorting order since the former value is alphanumerically first despite being numerically second.

2 Description of Proposed Solution

This section describes the implementation of the new graph DBMS features added to the graph DBMS created for Milestone 2.

2.1 Persistence (Task 0)

This task aims at making the graph database persistent between different runs. This is achieved by making sure that all the scans during the queries are properly closed so that the pages can be flushed after each query. Later on if a query needs to access an existing database that database is opened. The graph is saved to a file using the existing DB class file save features, and opened upon future runs with the DB class's file open features.

2.2. Index Nested Loop Join (Task 1)

For the queries in this phase, join operations will need to be performed in order to obtain tuples corresponding to the path and triangle expressions. Two forms of joins will be implemented for this purpose: index-nested loop joins, and sort-merge joins. Sort-merge joins are described in Section 2.3. The general algorithm of an index nested loop join between two sets R and S, denoted as:

R JOIN (RCondition, SCondition) S

is given as:

```
for r in R
  if RCondition(r) is true
        for s in S where SCondition(s) is true
              add (r,s) to join heap
```

While the RCondition is checked on every iteration of the loop, the SCondition is resolved by iterating over an index of terms in S which corresponds to elements of S that satisfy SCondition. This improves over the standard nested loop join, which iterates over both sets, by reducing the space of elements which have to be checked.

For the case of graph databases, four join operations are considered:

Node JOIN(Source, EdgeCondition) Edge

Node JOIN(Destination, EdgeCondition) Edge

Edge JOIN(Source, NodeCondition) Node

Edge JOIN(Destination, NodeCondition) Node

In addition, compound joins will be performed between previous joins and further edges or nodes, with a Node or Edge condition on the additional element.

For the specific queries performed here, not all elements of a node or edge are needed for the joined operation; in general, only one part of a node or edge - such as a node label or descriptor, or an edge weight or label - are needed. As a result, in addition to the general node/edge join operations as given above, another index nested join which only provides the relevant information was implemented, not storing the excess information for improved access speed. Indices on labels, descriptors, and edge weights help to improve access times for this reduced version.

The implementation of the index nested loop join follows the design of the Iterator class of the initial Minibase implementation. To avoid storing the joined data in main memory, the joined tuples are stored within a heap file and accessed using a get_next() method which returns the next tuple element within the join. Similarly, following the Iterator design, the index nested loop provides a close() option for closing and saving all data within the join for future use.

According to Ramakrishnan, et.al, while the index nested loop join is relatively slow for conditions with large numbers of elements satisfying them, due to the relatively large I/O costs, if the index is relatively small then the index-nested loop join will be very effective [1]. Unlike the sort-merge join, the index-nested loop join does not perform any sorting on the tuples, which makes it more effective where sorting is unnecessary, but ineffective where sorting is needed. Depending on the specific conditions tested, the sort-merge join or the index-nested loop join may be more effective. As a general rule, the sort-merge join would be more effective for queries involving range conditions, whereas the index-nested loop join is more effective for equality and point conditions. Both are used for the queries described below, depending on which is best for the specific form of condition being evaluated within the join.

2.3. Sort-Merge Join (Task 2)

A new sort-merge join operator was created, which performs a sort merge join operation between edge and edge relations and provides support for performing sort merge joins in multiple ways. There are 3 constructors; the first one takes a label and then that label is used to set up the condition expression for the sort merge join, the second constructor does not take any parameter and sets condition expression for the equality of destination node of first edge relation to source node of second edge relation, and the third constructor takes the condition expression from the caller and performs sort merge on the two edge relations based on that condition expression.

As noted in section 2.2, the sort-merge join operation sorts the two sets (on the column of the element being evaluated in the condition) and then uses the sorted structure in order to more effectively evaluate the condition expression. For this specific implementation, the join operation intended to be performed by the sort-merge join is:

Edge JOIN(dest_node = source_node) Edge

i.e., the join will create an (Edge, Edge) tuple where the destination node of the first edge is the source node of the second edge. This will be used in order to construct a path of nodes that will be relevant for the queries considered in this implementation. In addition, the sort-merge join will be able to merge a previous joined tuple with another edge, allowing for a longer path. In combination with the index nested loop join, the sort-merge join is used by the queries in order to find paths and triangles within the graph database.

2.4 Path Expression Operators (Tasks 3-5)

All path expressions are implemented using an index nested loop join on source labels. This is implemented using the left-deep query plan which minimizes number of tuples joined on every iteration. As per Ramakrishnan, et. al, while this may not yield the optimal query plan but it is the best that can be reasonably searched while maintaining other desirable features of the database structure, such as joining only singular elements to previous joined tuples [1].

The PathExpression class in the diskmgr package provides two API's that are useful for this task:

API 1: public Iterator evaluatePathExpression(ExpType[] inputAttrTypes, String[] values);

API 2: public Iterator evaluateBoundPathExpression(ExpType inputAttrType, String values,ExpType boundType, int bound);

The first API can be used to execute type 1 and type 2 queries. In addition, the implementation of this operator is robust enough to handle queries which are a combination of a Type 1 and Typ1 2 queries. The grammar of the query types that can be execute using the PathExpression API 1 is given below:

* PE1: NN/(NN/)*/NN
* PE2: NID /EN(/EN)*
* PE12Hybrid: NN/[(NN/*)(EN/*)]/[(NN)(EN)]
* where NN <- (Node_label | Node_descriptor),
* where EN <- (Edge_label | max edge weight)

The second API can be used to execute the path expressions of type 3:

* PE3: NID //Bound
* where Bound <- (number_of_Edges | Max_total_weight)

Implementation:

API 1: This path expression API takes expression types and the corresponding values as input. Expression types can be node label, node descriptor, edge label, or edge weight.

For expression of type- PE1:NN/(NN/)*/NN -

Step 1: Depending on the start expression label/Descriptor, either open an index search on either label index file or descriptor index file with value condition.

Step 2: For the next condition, a source node index file on edgeHeapfile's details and projection list is created (the index search condition is created in nested index loop join, as the source label condition changes for every outer tuple).

Step 3: The index details are sent to NestedIndexLoopJoin class, the optimized implementation of the index nested loop join, which takes outer iterator and index file details, condition and projection list on index files along with few other details for the join. A right filter is also created for every join,

which will be an OR Condition on all possible destination labels (example: if next condition is descriptor, an or condition on all labels whose descriptor values is given label).

Step 4: The output iterator of the first join is used as the outer relation of the second join.

Step 5: Step 2-4 is repeated for all expressions in the path expression.

Expression type:

PE2: NID / EN (/ EN)*

In order to return an iterator over the IDs of the nodes in the tail of each path that satisfies the path expression, the optimal method is to establish joins between nodes and edges. To implement the joins, the nested index loop join was modified accordingly to accommodate edges and nodes.

To return IDs of the nodes we will first open a label index or descriptor index file depending on the type of input according to the value condition. The index is created for the source nodes. The source node index file is used to search elements from the edge heap file. The condition for index search is based in nested index loop join because for every outer tuple the condition for source label is different. So, in this way we have a centralized method of indexing. Next, the nested index loop join class is invoked which takes the iterator, index details, condition and projection list on the index file. The final output will be obtained according to the conditions mentioned in projection. Also, to keep a check on the edge labels and edge weight conditions, a right filter is created. The iterator which we get from the first join is passed as an outer relation for the second join. For the next expression in the path expression, again a source node index file is created and similar steps are followed.

API 2: This path expression takes start expression type, value, bound type and bound values as input. Depending on the start expression index search on the node labels or node description file is opened which will be used as outer iterator in index nested loop join.

For Expressions with number of edges bound:

A new index nested loop join is opened for each edge with index on source label without any right filter on it. The output of join is written into two files: a results file, and intermediate file which will be used for the next join. The output of the previous join is used as the outer element in each case.

This is repeated as many time as mentioned in the bound. An iterator on the outer file is passed as the output of the API.

For expressions with Maximum total weight as bound:

A new sum expression was created for this specific task. It takes two field offset, one from inner and one from outer and adds the values and projects it in the output tuple. 'FldSpec' and 'Projections' class has been changed to achieve this.

An index search on source label is used for the inner relation. From each join the results are compared and written into the "results" and 'intermediate' file. This file is used for the next join. The joins are repeated until a join doesn't return any results. The sumfilter check is passed at each step to make sure the total max weight of the output tuple never crosses the given bound.

2.5 Path Expression Queries (Tasks 6-8)

2.5.1 Query Type 1 (Task 6)

The query type 1 takes as input a path expression of the form NN/(NN/)*/NN where NN is either a Node Label or Node Descriptor. The expression is then evaluated for the conditions on the label or the descriptor. Each condition prefixed by ND/NL is the value for the condition for the required path. It allows any number of NNs to be specified but requires it to start with a least one NN and end with another NN. It has three ways of getting the results, which depend on user choice. Option a outputs all the paths in the graph which contain specified nodes satisfying the given path. Option b outputs the same paths in alphanumeric sorted order while option c eliminates the repeated paths and outputs only the distinct source and tail labels of the paths. An example query can be given as follows:

PQ1 graphdb1 1000 a ND2,3,,4,5,6/NL50/NL30

The method pathQuery1a() from PathQueryHandler.java takes as input the path expression given in above query. It separates the string on the delimiter and obtains value of each NN in an array. Depending on the prefix of the value which consists of initial two letters before the actual value (NL for Node label, ND for node descriptor), a sequential array is maintained for the values in the path

expression and another sequential array is maintained for the type of the value (One of Node label, node descriptor). To maintain these type of expressions, we create an enumeration class called expType. The two arrays are passed to the method PathExpression.evaluatePathExpression(). The query operates by performing multiple joins. First it finds tuples on Node Label/ Node Descriptor Index File which satisfies the given value, then joins are performed on Source label Index file with source condition and node label Condition (node_label = next_node_label_in_PE) if the NN is a Node Label and performs join on Source label Index file with source condition and Node Descriptor Condition if NN is the Node Descriptor. It returns an iterator over a file containing the source and tail labels. We iterate over this file to print the required fields i.e. only the source and destination labels as output.

The method pathQuery1b() follows the same procedure as pathQuery1a(String exp) until passing of the arrays to PathExpression.evaluatePathExpression() but first sorts the file using iterator.sort() method and then prints the required fields. The sorting is performed on the destination node label. The method pathQuery1c() follows the same procedure and prints only the distinct pairs of source and tail labels. For finding distinct pairs, we use an inbuilt function from Minibase, DuplElim(). This method uses the first field as the sort type. Hence, we concatenate the source and tail label strings as first field and pass it to the DuplElim() method which then sorts the file and removes duplicate entries. The end result of pathQuery1c() is distinct pairs of source and tail labels satisfying the path expression.

2.5.2 Query Type 2 (Task 7)

The path expression for Path Expression Query, Type 2 is of the form below:

NN / EN (/ EN)*

Here, NN represents Node label or Node Descriptor and EN represents Edge Label or Maximum Edge Weight. It allows any number of ENs to be specified but requires it to start with a NN followed by at least one EN. It has three ways of getting the results which depend on user choice. Option a outputs all the paths in the graph which contain specified nodes and edges. Option b outputs the

same paths in alphanumeric sorted order while option c eliminates the repeated paths and outputs only the distinct source and tail labels of the paths. An example query can be given as follows:

PQ2 graphdb1 1000 a NL23/MW50/MW30

The method pathQuery2a() from PathQueryHandler.java takes input of the only the path expression given in above query. It separates the string on the delimiter and obtains value of each NN and EN in an array. Depending on the prefix of the value which consists of initial two letters before the actual value (NL for Node label, ND for node descriptor, EL for edge label and MW for Maximum Edge Weight), a sequential array is maintained for the values in the path expression and another sequential array is maintained for the type of the value (One of Node label, node descriptor, edge label and maximum edge weight). To maintain these type of expressions, we create an enumeration class called expType. The two arrays are passed to the method PathExpression.evaluatePathExpression(). The query operates by performing multiple joins. First it finds tuples on Node Label/ Node Descriptor Index File which satisfies Given value then joins are performed on Source label Index file with source condition and edge label Condition (edge_label = next_edge_label_in_PE) if the EN is an Edge Label and performs join on Source label Index file with source condition and edge weight Condition (edge_weight <= next_edge_weight_in_PE) if EN is Maximum Edge Weight. It returns an iterator over a file containing the source and tail labels. We iterate over this file to print the required fields i.e. only the source and tail labels as output.

The method pathQuery2b() follows the same procedure as pathQuery2a(String exp) until passing of the arrays to PathExpression.evaluatePathExpression() but first sorts the file using iterator.sort() method and then prints the required fields. The method pathQuery2c() follows the same procedure and prints only the distinct pairs of source and tail labels. For finding distinct pairs, we use an inbuilt function from Minibase, DuplElim(). This method uses the first field as the sort type. Hence, we concatenate the source and tail label strings as first field and pass it to the DuplElim() method which then sorts the file and removes duplicate entries. The end result of pathQuery2c() is distinct pairs of source and tail labels satisfying the path expression.

2.5.3 Query Type 3 (Task 8)

The path expression for Path Expression Query, Type 2 is of the form below:

NN / Bound

Here, NN represents Node label or Node Descriptor and Bound represents Maximum number of edges or Maximum Total Edge Weight. It has three ways of getting the results which depend on user choice. Option a outputs all the paths in the graph which contain specified nodes and edges. Option b outputs the same paths in alphanumeric sorted order while option c eliminates the repeated paths and outputs only the distinct source and tail labels of the paths. An example query can be given as follows:

PQ3 graphdb1 1000 a NL23/ME2

The method pathQuery3a() from PathQueryHandler.java takes input of the only the path expression given in above query. It separates the string on the delimiter and obtains value of each NN and Bound in an array. Depending on the prefix of the value which consists of initial two letters before the actual value (NL for Node label, ND for node descriptor), a sequential array is maintained for the values in the path expression and another sequential array is maintained for the type of the value (One of Node label, node descriptor, maximum number of edges and maximum total edge weight). Variables for bound type and bound value are obtained from the prefixes and passed along with the array to the method PathExpression.evaluateBoundPathExpression (). The query operates by performing multiple joins. First it finds tuples on Node Label/ Node Descriptor Index File which satisfies Given condition then joins are performed on Source label Index file with source condition) if the Bound is Maximum number of edges and performs join on Source label Index file with source condition and totalWeight <= given_total_weight if Bound is Maximum Total Edge Weight. It returns an iterator over a file containing the source and tail labels. We iterate over this file to print the required fields i.e. only the source and tail labels as output.

The method pathQuery3b() follows the same procedure as pathQuery3a(String exp) until passing of the arrays to PathExpression.evaluateBoundPathExpression() but first sorts the file using iterator.sort() method and then prints the required fields. The method pathQuery3c() follows the same procedure and prints only the distinct pairs of source and tail labels. For finding distinct pairs, we use an inbuilt function from Minibase, DuplElim(). This method uses the first field as the sort

type. Hence, we concatenate the source and tail label strings as first field and pass it to the DuplElim() method which then sorts the file and removes duplicate entries. The end result of pathQuery3c() is distinct pairs of source and tail labels satisfying the path expression.

2.6 Triangle Query (Task 9)

The triangle Query takes as input , three expressions EN;EN;EN such that EN is either Edge Label or Maximum Edge weight. A triangle is defined as a set of 3 interconnected edges. An edge is considered to be connected if the source label of an edge is the same as the destination label of another edge and vice versa. Each condition prefixed by EL or MW is the value for condition on that edge in the triangle.

The query is implemented using 2 join operations. The first join is between an Indexscan iterator (labeled IS1) and the Source Node Label Index. IS1 was given the condition for first edge while initiating it and projection contains the destination and source labels of the edges which satisfies the given condition. The first join is:

Node JOIN(Source Node Label Index, Edge Condition) Edge;

and is performed on the Destination Labels obtained from IS1 and the destination label column of edges. The Join is provided the condition for second edge as its right expression. The projection from this join has 4 columns (SourceNode1,DestNode1,SourceNode2,DestNode2), in which DestNode1 and SourceNode2 are identical and both the edges from which these Nodes are taken are satisfying the condition 1 and 2 respectively. The second join is performed on the output from first join and the edge source label index with the final edge condition provided during query, then a select is performed on the Join's output such that Destination Label of the edge is equivalent to the Column 1 (SourceNode 1) of the first join. A projection on Source Node 1, Dest Node1 and Dest Node 2 columns of the output from first join is then performed to give the final unsorted result.

For obtaining a sorted result, a sort is performed on the first column of the triplets. For obtaining unique results, circular right rotate operations are performed on a single triplet until the NodeLabel with least value is at position 1. Once this is complete, all the 3 node labels in triplet are written on a string and this string and the output tuples are written on a temporary heap file. A DuplElim class iterator is initialized on a scan of the temporary heap file. This removes the duplicate triangles while

making sure that two triangles with same node labels as vertices but of opposite direction are not eliminated.

3 Graph DB Interface Specifications

The interface for the graph database will be the same command line interface that was used for the tests performed for the relational DBMS tests for the original Minibase functionality. Invocation of the six tests is described in Section 2.5 and 2.6; each is invoked by a specified input command that is also shown in the menu displayed by the command line interface. Similarly, type “menu” to reprint the menu, or “exit” to exit the tests.

4 System Requirements and Execution Instructions

Similar to the original Minibase code, the instructions to execute the code are as follows:

1. Download and uncompress the file.
2. Modify the makefiles to reflect your own directory structure.
3. In the src directory, run the command “make db” to compile the database.
4. In the src directory, run the command “make test” to run all tests (including the relational DBMS tests) and “make graphtest” to run just the tests added to the original Minibase implementation. Input format is described in the menu or in Section 3 and the referenced sections, above.
5. To record the output, use a typescript; i.e. use the command “script” before running the Graph DB tests, and use the command “exit” to end the typescript and receive a file containing the output.

As with the Minibase program, this is best run on a UNIX machine for access to bash functionality. Java (version 1.7 or newer) and Java Development Kit (JDK) need to be installed on your system in order to compile and run this program.

5 Conclusion

This phase of the CSE 510 final project involved the implementation of two forms of queries, a path expression query and a triangle query, and the operators necessary for their function. These

operators included a join operation, which was implemented using both index nested loop and sort-merge join algorithms, and path expression operators, which parsed the graph for elements that satisfied the conditions stated within the query. Each of these features were added to the initial graph DBMS developed within Phase 2, which were built atop the Minibase implementation of a relational DBMS, providing further functionality relevant to graph-based input data.

These new features provide further functionality for a graph-based DBMS that start to highlight the circumstances under which it may be useful to store data in a graph database, as these queries allow for analysis of the data based on paths formed by the nodes and edges contained within the graph, both for arbitrary paths and for paths that form a three-node triangle. These features are beyond the scope of a relational DBMS but can be performed by a graph DBMS, providing functionality within the graph DBMS that was not originally available.

As these queries are each based on join operations, this represents a query task that is relatively computationally expensive and often memory-intensive, requiring effective use of the database structure in order to be effective at scale. The heap structures reduce dependence upon memory, while the indices developed in Phase 2 allow for more rapid access of the relevant data as needed. Each of the operators were developed with these structures in mind, allowing for more effective use of the DBMS that would be effective where significantly larger databases are involved.

Bibliography

1. Ramakrishnan, Raghu, and Johannes Gehrke. *Database Management Systems*. 1st ed. Boston: VGM Career Books, 2003. Print.

Appendix A: Contributions by Team Member

Elan Markov - Completed index nested loop join code and wrote documentation and report sections for index nested loop joins. Designed test cases to evaluate performance of DBMS under positive (query returns some results) and negative (no matches for the query) queries for each of the four queries considered.

Simarpreet Kaur - Completed the implementation of Path query 1 and a portion of Path query 2. Wrote the documentation and report sections for the above completed tasks.

Prachi Sharma - Implemented path query operator 2 and wrote the documentation and report sections for that task.

Jayanth Kumar Melinavolagerehalli Jayaramaiah - Implemented path query operators 1 and 3, added sum expression and resolved integration issues. Tested all components. Added Javadoc for these components and wrote documentation.

Harshdeep Singh Sandhu - Implemented the Triangle Queries including sorting and duplicate removal of the triplets for the part B and C of the query. Wrote the documentation and report sections for the implementation of the triangle queries.

Priyekant Aghi - Completed sort-merge join operator and wrote the code for making the database persistent over multiple runs. Wrote the documentation and report sections for implementing sort merge operator and making the database persistent. Also fixed some of the tasks which were not working during the phase 2 of the project.

Shalmali Bhoir - Completed the implementation of part of path query 2 and all of path query 3. Wrote the documentation and report sections for the above completed tasks.

Appendix B: Text of Typescript Output

```
Script started on Wed 26 Apr 2017 09:15:17 AM MST
[01;32muser@user-Linux[01;34m ~/Documents/CSE510/minjava/javaminibase $[00m cd src;
make db; make   graphtest
make -C global
make[1]:                              Entering                                directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/global'
/usr/lib/jvm/default-java/bin/javac                                          -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]:                               Leaving                               directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/global'
make -C chainexception
make[1]:                              Entering                                directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/chainexception'
/usr/lib/jvm/default-java/bin/javac                                         -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]:                               Leaving                               directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/chainexception'
make -C btree
make[1]:                              Entering                                directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/btree'
/usr/lib/jvm/default-java/bin/javac                                         -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]:                               Leaving                               directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/btree'
make -C bufmgr
make[1]:                              Entering                                directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/bufmgr'
/usr/lib/jvm/default-java/bin/javac                                         -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]:                               Leaving                               directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/bufmgr'
make -C diskmgr
make[1]:                              Entering                                directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/diskmgr'
/usr/lib/jvm/default-java/bin/javac                                         -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
graphDB.java:5: warning: ArrayType is internal proprietary API and may be removed in
a future release
of a graph database.
        ^
1 warning
```

```
make[1]: Leaving directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/diskmgr'
make -C heap
make[1]: Entering directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/heap'
/usr/lib/jvm/default-java/bin/javac -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]: Leaving directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/heap'
make -C index
make[1]: Entering directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/index'
/usr/lib/jvm/default-java/bin/javac -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]: Leaving directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/index'
make -C iterator
make[1]: Entering directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/iterator'
/usr/lib/jvm/default-java/bin/javac -classpath
/usr/lib/jvm/default-java/lib/classes.zip:.:..:/home/user/Documents/CSE510/minjava/j
avaminibase/construction *.java
make[1]: Leaving directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/iterator'
cd tests; make graph
make[1]: Entering directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/tests'
/usr/lib/jvm/default-java/bin/javac -classpath .:.. testcases.java
/usr/lib/jvm/default-java/bin/java  -classpath .:.. tests.testcases
Graph Database Test Cases
Format of command line input:

============= Phase 2 Tests =============
Batch Node Insert (Task 10 Query):
batchnodeinsert NODEFILENAME GRAPHDBNAME
NODEFILENAME should be a file in tests folder

Batch Edge Insert (Task 11 Query):
batchedgeinsert EDGEFILENAME GRAPHDBNAME
EDGEFILENAME should be a file in tests folder

Batch Node Delete (Task 12 Query):
batchnodedelete NODEFILENAME GRAPHDBNAME
NODEFILENAME should be a file in tests folder
```

```
Batch Edge Delete (Task 13 Query):
batchnodeinsert EDGEFILENAME GRAPHDBNAME
EDGEFILENAME should be a file in tests folder

Simple Node Query (Task 14 Query):
nodequery GRAPHDBNAME NUMBUF QTYPE INDEX [QUERYOPTIONS]

Simple Edge Query (Task 15 Query):
edgequery GRAPHDBNAME NUMBUF QTYPE INDEX [QUERYOPTIONS]


============= Phase 3 Tests =============
NL = Node Label
ND = Node Descriptor
EL = Edge Label
MW = Max Edge Weight
ME = Max Num Edges
TW = Max Total Edge Weight

Path Expression Query 1 (Task 6 Query):
PQ1 GRAPHDBNAME NUMBUF a/b/c ARGS
ARGS = NN/(NN/)*/NN
NN = (NL|ND)
Example: PQ1 graph 1000 a NL1/ND1,2,3,4,5

Path Expression Query 2 (Task 7 Query):
PQ2 GRAPHDBNAME NUMBUF a/b/c ARGS
ARGS = NN/EN(/EN)*
NN = (NL|ND)
EN = (EL|MW)
Example: PQ2 graph 1000 a NL1/EL2

Path Expression Query 3 (Task 8 Query):
PQ3 GRAPHDBNAME NUMBUF a/b/c ARGS
ARGS = NN/Bound
NN = (NL|ND)
Bound = (ME|TW)
Example: PQ3 graph 1000 a NL1/ME5

Triangle Query (Task 9 Query):
TQA/TQB/TQC GRAPHDBNAME NUMBUF ARGS
ARGS = EN;EN;EN
EN = (EL|MW)
Example: TQA graph 1000 EL1;EL2;EL3


Enter menu to print the menu, exit to exit, or a command line input to execute:
```

```
batchnodeinsert NodeInsertData.txt graphdb1
Replacer: Clock

buffers : 5000 unpinned : 4993

Running Batch Node Insert tests....

Node Count after batch insertion on graph database: 1056
Edge Count after batch insertion on graph database: 0

... Batch Node Insert tests completed successfully..


No. of pages read : 3
No. of pages write : 252
buffers : 5000 unpinned : 5000
Enter menu to print the menu, exit to exit, or a command line input to execute:
batchedgeinsert EdgeInsertData.txt graphdb1
Replacer: Clock

buffers : 5000 unpinned : 4993

Running Batch Edge Insert tests....

Node Count after batch insertion on graph database: 1056
Edge Count after batch insertion on graph database: 3198

... Batch Edge Insert tests completed successfully..


No. of pages read : 48
No. of pages write : 592
buffers : 5000 unpinned : 5000
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a ND7,1,44,22,12/NL248/NL384/NL514
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[0, 514]
PathExpression.close() reads : [7, 95, 46]
```

```
PathExpression.close() write : [0, 0, 0]

No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b ND7,1,44,22,12/NL248/NL384/NL514
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[0, 514]
PathExpression.close() reads : [8, 95, 46]
PathExpression.close() write : [0, 0, 0]

No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c ND7,1,44,22,12/NL248/NL384/NL514
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[0, 514]
PathExpression.close() reads : [8, 95, 46]
PathExpression.close() write : [0, 0, 0]

No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a NL384/NL514
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[384, 514]
PathExpression.close() reads : [158]
PathExpression.close() write : [0]

No. of pages read : 170
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b NL384/NL514
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[384, 514]
PathExpression.close() reads : [159]
PathExpression.close() write : [0]

No. of pages read : 170
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c NL384/NL514
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[384, 514]
PathExpression.close() reads : [159]
PathExpression.close() write : [0]

No. of pages read : 170
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a ND24,42,37,8,44/ND42,2,47,47,33
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
```

```
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
[64, 261]
PathExpression.close() reads : [30]
PathExpression.close() write : [0]

No. of pages read : 58
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b ND24,42,37,8,44/ND42,2,47,47,33
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
[64, 261]
PathExpression.close() reads : [31]
PathExpression.close() write : [0]

No. of pages read : 58
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c ND24,42,37,8,44/ND42,2,47,47,33
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
[64, 261]
PathExpression.close() reads : [31]
PathExpression.close() write : [0]

No. of pages read : 58
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a NL23/ND20,14,37,12,23/NL727
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
```

```
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[23, 727]
PathExpression.close() reads : [14, 140]
PathExpression.close() write : [0, 0]

No. of pages read : 186
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b NL23/ND20,14,37,12,23/NL727
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[23, 727]
PathExpression.close() reads : [15, 140]
PathExpression.close() write : [0, 0]

No. of pages read : 186
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c NL23/ND20,14,37,12,23/NL727
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_Descriptor's_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
[23, 727]
PathExpression.close() reads : [15, 140]
PathExpression.close() write : [0, 0]

No. of pages read : 186
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL23/MW100/MW150
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[23, 887]
[23, 960]
[23, 842]
[23, 727]
[23, 979]
PathExpression.close() reads : [16, 223]
PathExpression.close() write : [0, 0]

No. of pages read : 251
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL23/MW100/MW150
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[23, 727]
[23, 842]
[23, 887]
[23, 960]
[23, 979]
PathExpression.close() reads : [17, 223]
PathExpression.close() write : [0, 0]

No. of pages read : 251
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL23/MW100/MW150
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
```

```
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[23, 727]
[23, 842]
[23, 887]
[23, 960]
[23, 979]
PathExpression.close() reads : [17, 223]
PathExpression.close() write : [0, 0]

No. of pages read : 251
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/EL0_248
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
[0, 248]
PathExpression.close() reads : [7]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/EL0_248
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
[0, 248]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/EL0_248
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
[0, 248]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/EL0_248/MW140
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
[0, 616]
[0, 649]
[0, 467]
[0, 885]
PathExpression.close() reads : [7, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/EL0_248/MW140
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
```

```
[0, 467]
[0, 616]
[0, 649]
[0, 885]
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/EL0_248/MW140
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
[0, 467]
[0, 616]
[0, 649]
[0, 885]
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/EL0_248/MW20
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
[0, 885]
PathExpression.close() reads : [7, 95]
PathExpression.close() write : [0, 0]
```

```
No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/EL0_248/MW20
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
[0, 885]
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/EL0_248/MW20
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 384]
[0, 422]
[0, 885]
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/MW20
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 1023]
PathExpression.close() reads : [7]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/MW20
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 1023]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/MW20
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 1023]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/MW40
Replacer: Clock

buffers : 1000 unpinned : 993
```

```
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
PathExpression.close() reads : [7]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/MW40
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 1000]
[0, 1023]
[0, 248]
[0, 850]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/MW40
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
[0, 1000]
[0, 1023]
[0, 248]
[0, 850]
PathExpression.close() reads : [8]
PathExpression.close() write : [0]
```

```
No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL45/ME4
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[45, 577]
[45, 453]
[45, 457]
[45, 620]
[45, 435]
[45, 918]
[45, 791]
[45, 989]
[45, 739]
[45, 911]
[45, 818]
[45, 852]
[45, 922]
[45, 735]
[45, 516]
[45, 842]
[45, 525]
[45, 702]
[45, 728]
[45, 472]
[45, 486]
[45, 975]
[45, 788]
[45, 596]
[45, 924]
[45, 982]
[45, 824]
[45, 1000]
[45, 963]
[45, 844]
[45, 1017]
[45, 891]
[45, 964]
[45, 845]
```

[45, 745]
[45, 1002]
[45, 797]
[45, 1053]
[45, 885]
[45, 683]
[45, 798]
[45, 781]
[45, 886]
[45, 975]
[45, 795]
[45, 989]
[45, 862]
[45, 782]
[45, 1024]
[45, 705]
[45, 806]
[45, 879]
[45, 914]
[45, 918]
[45, 967]
[45, 933]
[45, 981]
[45, 966]
[45, 988]
[45, 887]
[45, 921]
[45, 801]
[45, 874]
[45, 1042]
[45, 1018]
[45, 1011]
[45, 861]
[45, 1029]
[45, 946]
[45, 900]
[45, 905]
[45, 1035]
[45, 1042]
[45, 873]
[45, 1037]
[45, 974]
[45, 872]
[45, 91]
[45, 991]
[45, 1035]
[45, 918]

```
[45, 991]
[45, 941]
PathExpression.close() reads : [30, 208, 22, 12]
PathExpression.close() write : [0, 0, 0, 0]

No. of pages read : 280
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL45/ME4
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[45, 1000]
[45, 1002]
[45, 1011]
[45, 1017]
[45, 1018]
[45, 1024]
[45, 1029]
[45, 1035]
[45, 1035]
[45, 1037]
[45, 1042]
[45, 1042]
[45, 1053]
[45, 435]
[45, 453]
[45, 457]
[45, 472]
[45, 486]
[45, 516]
[45, 525]
[45, 577]
[45, 596]
[45, 620]
[45, 683]
[45, 702]
[45, 705]
[45, 728]
[45, 735]
[45, 739]
```

[45, 745]
[45, 781]
[45, 782]
[45, 788]
[45, 791]
[45, 795]
[45, 797]
[45, 798]
[45, 801]
[45, 806]
[45, 818]
[45, 824]
[45, 842]
[45, 844]
[45, 845]
[45, 852]
[45, 861]
[45, 862]
[45, 872]
[45, 873]
[45, 874]
[45, 879]
[45, 885]
[45, 886]
[45, 887]
[45, 891]
[45, 900]
[45, 905]
[45, 91]
[45, 911]
[45, 914]
[45, 918]
[45, 918]
[45, 918]
[45, 921]
[45, 922]
[45, 924]
[45, 933]
[45, 941]
[45, 946]
[45, 963]
[45, 964]
[45, 966]
[45, 967]
[45, 974]
[45, 975]
[45, 975]

```
[45, 981]
[45, 982]
[45, 988]
[45, 989]
[45, 989]
[45, 991]
[45, 991]
PathExpression.close() reads : [30, 208, 22, 12]
PathExpression.close() write : [0, 0, 0, 0]

No. of pages read : 280
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL45/ME4
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[45, 1000]
[45, 1002]
[45, 1011]
[45, 1017]
[45, 1018]
[45, 1024]
[45, 1029]
[45, 1035]
[45, 1037]
[45, 1042]
[45, 1053]
[45, 435]
[45, 453]
[45, 457]
[45, 472]
[45, 486]
[45, 516]
[45, 525]
[45, 577]
[45, 596]
[45, 620]
[45, 683]
[45, 702]
[45, 705]
```

[45, 728]
[45, 735]
[45, 739]
[45, 745]
[45, 781]
[45, 782]
[45, 788]
[45, 791]
[45, 795]
[45, 797]
[45, 798]
[45, 801]
[45, 806]
[45, 818]
[45, 824]
[45, 842]
[45, 844]
[45, 845]
[45, 852]
[45, 861]
[45, 862]
[45, 872]
[45, 873]
[45, 874]
[45, 879]
[45, 885]
[45, 886]
[45, 887]
[45, 891]
[45, 900]
[45, 905]
[45, 91]
[45, 911]
[45, 914]
[45, 918]
[45, 921]
[45, 922]
[45, 924]
[45, 933]
[45, 941]
[45, 946]
[45, 963]
[45, 964]
[45, 966]
[45, 967]
[45, 974]
[45, 975]

```
[45, 981]
[45, 982]
[45, 988]
[45, 989]
[45, 991]
PathExpression.close() reads : [30, 208, 22, 12]
PathExpression.close() write : [0, 0, 0, 0]

No. of pages read : 280
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a ND7,1,44,22,12/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
[0, 384]
[0, 422]
[0, 885]
[0, 514]
[0, 811]
[0, 443]
[0, 438]
[0, 938]
[0, 962]
[0, 1032]
[0, 974]
[0, 827]
[0, 992]
PathExpression.close() reads : [14, 226, 3, 4, 9, 0]
```

```
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 264
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b ND7,1,44,22,12/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
[0, 1000]
[0, 1023]
[0, 1032]
[0, 248]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 514]
[0, 811]
[0, 827]
[0, 850]
[0, 885]
[0, 938]
[0, 962]
[0, 974]
[0, 992]
PathExpression.close() reads : [14, 226, 3, 4, 9, 0]
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 264
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
```

```
PQ3 graphdb1 1000 c ND7,1,44,22,12/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
[0, 1000]
[0, 1023]
[0, 1032]
[0, 248]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 514]
[0, 811]
[0, 827]
[0, 850]
[0, 885]
[0, 938]
[0, 962]
[0, 974]
[0, 992]
PathExpression.close() reads : [14, 226, 3, 4, 9, 0]
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 264
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL0/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
```

```
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
[0, 384]
[0, 422]
[0, 885]
[0, 514]
[0, 811]
[0, 443]
[0, 438]
[0, 938]
[0, 962]
[0, 1032]
[0, 974]
[0, 827]
[0, 992]
PathExpression.close() reads : [11, 226, 3, 4, 9, 0]
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 261
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL0/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
  Join  On  (  Source  label  Index  file  with  source  condition  and  totalWeight  <=
given_total_weight)
```

```
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
[0, 1000]
[0, 1023]
[0, 1032]
[0, 248]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 514]
[0, 811]
[0, 827]
[0, 850]
[0, 885]
[0, 938]
[0, 962]
[0, 974]
[0, 992]
PathExpression.close() reads : [11, 226, 3, 4, 9, 0]
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 261
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL0/TW45
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
```

```
[0, 1000]
[0, 1023]
[0, 1032]
[0, 248]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 514]
[0, 811]
[0, 827]
[0, 850]
[0, 885]
[0, 938]
[0, 962]
[0, 974]
[0, 992]
PathExpression.close() reads : [11, 226, 3, 4, 9, 0]
PathExpression.close() write : [0, 0, 0, 0, 0, 0]

No. of pages read : 261
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
```

```
[0, 1000]
[0, 1023]
[0, 248]
[0, 850]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[0, 1000]
[0, 1023]
[0, 248]
[0, 850]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL0/ME5
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
[0, 384]
[0, 422]
[0, 616]
[0, 649]
[0, 467]
```

[0, 885]
[0, 1005]
[0, 514]
[0, 811]
[0, 939]
[0, 706]
[0, 593]
[0, 443]
[0, 542]
[0, 438]
[0, 887]
[0, 880]
[0, 910]
[0, 624]
[0, 791]
[0, 734]
[0, 938]
[0, 869]
[0, 935]
[0, 744]
[0, 962]
[0, 1032]
[0, 974]
[0, 652]
[0, 791]
[0, 911]
[0, 727]
[0, 827]
[0, 941]
[0, 531]
[0, 1032]
[0, 947]
[0, 705]
[0, 868]
[0, 940]
[0, 764]
[0, 924]
[0, 982]
[0, 824]
[0, 800]
[0, 856]
[0, 231]
[0, 923]
[0, 943]
[0, 901]
[0, 791]
[0, 855]

```
[0, 996]
[0, 992]
[0, 940]
[0, 924]
[0, 982]
[0, 824]
[0, 1017]
[0, 1045]
[0, 966]
[0, 990]
[0, 616]
[0, 759]
[0, 631]
[0, 999]
[0, 993]
[0, 940]
[0, 933]
[0, 981]
[0, 910]
[0, 880]
[0, 941]
[0, 965]
[0, 458]
[0, 913]
[0, 261]
PathExpression.close() reads : [11, 226, 6, 17, 10]
PathExpression.close() write : [0, 0, 0, 0, 0]

No. of pages read : 278
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL0/ME5
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[0, 1000]
[0, 1005]
[0, 1017]
[0, 1023]
[0, 1032]
```

[0, 1032]
[0, 1045]
[0, 231]
[0, 248]
[0, 261]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 458]
[0, 467]
[0, 514]
[0, 531]
[0, 542]
[0, 593]
[0, 616]
[0, 616]
[0, 624]
[0, 631]
[0, 649]
[0, 652]
[0, 705]
[0, 706]
[0, 727]
[0, 734]
[0, 744]
[0, 759]
[0, 764]
[0, 791]
[0, 791]
[0, 791]
[0, 800]
[0, 811]
[0, 824]
[0, 824]
[0, 827]
[0, 850]
[0, 855]
[0, 856]
[0, 868]
[0, 869]
[0, 880]
[0, 880]
[0, 885]
[0, 887]
[0, 901]
[0, 910]

```
[0, 910]
[0, 911]
[0, 913]
[0, 923]
[0, 924]
[0, 924]
[0, 933]
[0, 935]
[0, 938]
[0, 939]
[0, 940]
[0, 940]
[0, 940]
[0, 941]
[0, 941]
[0, 943]
[0, 947]
[0, 962]
[0, 965]
[0, 966]
[0, 974]
[0, 981]
[0, 982]
[0, 982]
[0, 990]
[0, 992]
[0, 993]
[0, 996]
[0, 999]
PathExpression.close() reads : [11, 226, 6, 17, 10]
PathExpression.close() write : [0, 0, 0, 0, 0]

No. of pages read : 278
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL0/ME5
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
 Join On (Source label Index file with source condition)
[0, 1000]
```

[0, 1005]
[0, 1017]
[0, 1023]
[0, 1032]
[0, 1045]
[0, 231]
[0, 248]
[0, 261]
[0, 384]
[0, 422]
[0, 438]
[0, 443]
[0, 458]
[0, 467]
[0, 514]
[0, 531]
[0, 542]
[0, 593]
[0, 616]
[0, 624]
[0, 631]
[0, 649]
[0, 652]
[0, 705]
[0, 706]
[0, 727]
[0, 734]
[0, 744]
[0, 759]
[0, 764]
[0, 791]
[0, 800]
[0, 811]
[0, 824]
[0, 827]
[0, 850]
[0, 855]
[0, 856]
[0, 868]
[0, 869]
[0, 880]
[0, 885]
[0, 887]
[0, 901]
[0, 910]
[0, 911]
[0, 913]

```
[0, 923]
[0, 924]
[0, 933]
[0, 935]
[0, 938]
[0, 939]
[0, 940]
[0, 941]
[0, 943]
[0, 947]
[0, 962]
[0, 965]
[0, 966]
[0, 974]
[0, 981]
[0, 982]
[0, 990]
[0, 992]
[0, 993]
[0, 996]
[0, 999]
PathExpression.close() reads : [11, 226, 6, 17, 10]
PathExpression.close() write : [0, 0, 0, 0, 0]

No. of pages read : 278
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQA graphdb1 1000 EL1;EL2;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first
expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with no sorting are:
[771, 68, 255]
[1024, 91, 163]
[813, 129, 629]
[512, 221, 236]
[887, 531, 616]

No. of pages read : 260
```

```
No. of pages write : 10
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQB graphdb1 1000 EL1;EL2;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with ascending sorting done on the first Node Label are:
Sort on Column 1
[1024, 91, 163]
[512, 221, 236]
[771, 68, 255]
[813, 129, 629]
[887, 531, 616]

No. of pages read : 261
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQC graphdb1 1000 EL1;EL2;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Heap Created with Concatenated String of Node Labels as first field.
Duplicate Removal Object Initiated.
[1024, 91, 163]
[129, 629, 813]
[221, 236, 512]
[255, 771, 68]
[531, 616, 887]

No. of pages read : 261
```

```
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQA graphdb1 1000 EL1;MW10;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with no sorting are:
No Triangles Found.

No. of pages read : 221
No. of pages write : 10
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQA graphdb1 1000 EL1;MW30;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with no sorting are:
[771, 68, 255]
[512, 221, 236]

No. of pages read : 269
No. of pages write : 11
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQB graphdb1 1000 EL1;MW30;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
```

```
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with ascending sorting done on the first Node Label
are:
Sort on Column 1
[512, 221, 236]
[771, 68, 255]

No. of pages read : 270
No. of pages write : 13
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQC graphdb1 1000 EL1;MW30;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first
expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Heap Created with Concatenated String of Node Labels as first field.
Duplicate Removal Object Initiated.
[221, 236, 512]
[255, 771, 68]

No. of pages read : 270
No. of pages write : 13
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a ND7,1,44,22,12/NL248/NL384/NL0
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [7, 95, 46]
PathExpression.close() write : [0, 0, 0]
```

```
No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b ND7,1,44,22,12/NL248/NL384/NL0
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [8, 95, 46]
PathExpression.close() write : [0, 0, 0]

No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c ND7,1,44,22,12/NL248/NL384/NL0
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [8, 95, 46]
PathExpression.close() write : [0, 0, 0]

No. of pages read : 163
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 a NL64/NL91
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
```

```
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [34]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 b NL64/NL91
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [35]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ1 graphdb1 1000 c NL64/NL91
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition anddestination Condition
(dest_label = next_label_in_PE))
PathExpression.close() reads : [35]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/MW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [7]
PathExpression.close() write : [0]
```

```
No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/MW2
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/MW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [8]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL64/EL0_248
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
PathExpression.close() reads : [34]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
```

```
PQ2 graphdb1 1000 b NL64/EL0_248
Replacer: Clock

buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
PathExpression.close() reads : [35]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL64/EL0_248
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
PathExpression.close() reads : [35]
PathExpression.close() write : [0]

No. of pages read : 46
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 a NL0/EL0_248/MW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [7, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 b NL0/EL0_248/MW2
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
PathQuery.runTests() 3
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ2 graphdb1 1000 c NL0/EL0_248/MW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On ( Source label Index file with source condition andedge label Condition
(edge_label = next_edge_label_in_PE))
 Join On ( Source label Index file with source condition andedge weight Condition
(edge_weight <= next_edge_weight_in_PE))
PathExpression.close() reads : [8, 95]
PathExpression.close() write : [0, 0]

No. of pages read : 114
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a ND7,1,4,22,12/TW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on node Descriptor Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
PathExpression.close() reads : [5]
PathExpression.close() write : [0]

No. of pages read : 14
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL7,1,4,22,12/TW2
Replacer: Clock
```

```
buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
PathExpression.close() reads : [4]
PathExpression.close() write : [0]

No. of pages read : 13
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL7,1,4,22,12/TW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
PathExpression.close() reads : [4]
PathExpression.close() write : [0]

No. of pages read : 13
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL0/TW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
PathExpression.close() reads : [10]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL0/TW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
```

```
PathExpression.close() reads : [10]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL0/TW2
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
  Join On ( Source label Index file with source condition and totalWeight <=
given_total_weight)
PathExpression.close() reads : [10]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[0, 248]
[0, 1000]
[0, 850]
[0, 1023]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 b NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[0, 1000]
[0, 1023]
[0, 248]
```

```
[0, 850]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 c NL0/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[0, 1000]
[0, 1023]
[0, 248]
[0, 850]
PathExpression.close() reads : [11]
PathExpression.close() write : [0]

No. of pages read : 19
No. of pages write : 0
buffers : 1000 unpinned : 983
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQA graphdb1 1000 EL1;EL2;EL1
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first
expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with no sorting are:
No Triangles Found.

No. of pages read : 260
No. of pages write : 10
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQB graphdb1 1000 EL1;EL2;EL1
Replacer: Clock

buffers : 1000 unpinned : 993
```

```
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with ascending sorting done on the first Node Label are:
Sort on Column 1
No Triangles Found.

No. of pages read : 261
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQC graphdb1 1000 EL1;EL2;EL1
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Heap Created with Concatenated String of Node Labels as first field.
Duplicate Removal Object Initiated.

No. of pages read : 261
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQA graphdb1 1000 EL1;MW0;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with no sorting are:
No Triangles Found.
```

```
No. of pages read : 219
No. of pages write : 10
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQB graphdb1 1000 EL1;MW0;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first
expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Triangles with duplications, with ascending sorting done on the first Node Label
are:
Sort on Column 1
No Triangles Found.

No. of pages read : 220
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
TQC graphdb1 1000 EL1;MW0;EL3
Replacer: Clock

buffers : 1000 unpinned : 993
PROJECTION of Source Node and Destination Node Label of edge(SELECT with first
expression as edge condition).
Join On(Edge Source Node Label Index with second expression as edge condition)
Projection of [SourceNode1,DestNode1,SourceNode2,DestNode2]
Join on (Edge Source Node Label Index with third expression as edge condition and
DestNode1 = DestNodeLabel of inner relation)
Projection of [SourceNode1,DestNode1,DestNode2]
Heap Created with Concatenated String of Node Labels as first field.
Duplicate Removal Object Initiated.

No. of pages read : 220
No. of pages write : 12
buffers : 1000 unpinned : 1000
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL4/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
```

```
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[4, 736]
[4, 231]
[4, 616]
[4, 41]
[4, 395]
[4, 889]
[4, 476]
[4, 865]
PathExpression.close() reads : [13]
PathExpression.close() write : [0]

No. of pages read : 21
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL40/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[40, 387]
[40, 805]
[40, 870]
[40, 601]
[40, 474]
[40, 410]
PathExpression.close() reads : [28]
PathExpression.close() write : [0]

No. of pages read : 36
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
PQ3 graphdb1 1000 a NL55/ME1
Replacer: Clock

buffers : 1000 unpinned : 993
(Find tuples on NodeLabel Index File which satisfies Given condition)
 Join On (Source label Index file with source condition)
[55, 172]
[55, 694]
[55, 263]
PathExpression.close() reads : [33]
PathExpression.close() write : [0]
```

```
No. of pages read : 41
No. of pages write : 0
buffers : 1000 unpinned : 993
Enter menu to print the menu, exit to exit, or a command line input to execute:
exit
Replacer: Clock

buffers : 1000 unpinned : 993

No. of pages read : 8
No. of pages write : 0
buffers : 1000 unpinned : 993
make[1]:                              Leaving                                  directory
`/home/user/Documents/CSE510/minjava/javaminibase/src/tests'
[01;32muser@user-Linux[01;34m ~/Documents/CSE510/minjava/javaminibase/src $[00m exit
exit

Script done on Wed 26 Apr 2017 09:27:40 AM MST
```