Real-World Application

A regularly-updated IMDB database file listing the known filming locations for every movie can be found at
ftp://ftp.fu-berlin.de/pub/misc/movies/database/locations.list.gz
This file, when truncated to include only movies starting with the word 'A' looks like:

```
A Split Personality (2013)              Winchester, Virginia, USA
A Split Personality (2013)              Culpeper, Virginia, USA
A Split Personality (2013)              Washington, District of Columbia, USA
A Split Personality (2013)              Virginia, USA
A Split Personality (2013)              Manassas, Virginia, USA
A Split Personality (2013)              Germantown, Maryland, USA
A Split Personality (2013)              Atlantic City, New Jersey, USA
A Split Personality (2013)              Bowie, Maryland, USA
A Split Personality (2013)              Warrenton, Virginia, USA
A Split Personality (2013)              Toronto, Ontario, Canada
A Spoonful of Desi (2004)               Adelaide, South Australia, Australia
A Spoonful of Sugar (2003)              New York City, New York, USA
A Spoonful of Zanny (2012)              USA
A Sport Called Squash (2013)                Toronto Harbour, Toronto, Ontario, Canada   (city)
A Sport Called Squash (2013)                New City, New York, USA (Grand Central Station)
A Sport Called Squash (2013)                Vancouver, British Columbia, Canada (city)
A Sporting Chance (1945)                Republic Studios, Hollywood, Los Angeles, California, USA   (studio)
A Sporting Chance (2002)                Heidelberg, Victoria, Australia (location)
```

Because most movies are filmed in multiple locations, 2 random movies can often be linked together by a path of movies linked by common filming locations. We wanted to find the shortest such path between 2 movies.
This data set works well because these movies have an average of 6 listed filming locations, striking a good balance between graph size and ability to find a path.

The dataset was converted into a digraph by a separate Java program (included in the submission). The program reads the raw file line-by-line and uses tabs to differentiate movies and locations. It creates a hashmap from a location to a list of movies filmed in that location. It lastly iterates the hashmap and writes to movies-vertex.txt and movies-edge.txt (seen below).

```
A Walk with Nigel (2010)
A Year in the Life (2009) }
1   London, England, UK
A Walk with Nigel (2010)
A Year of Your Love (2009)
1   London, England, UK
A Walk with Nigel (2010)
A doppia faccia (1969)
1   London, England, UK
A Walk with Nigel (2010)
A mi me gusta (2008)
1   London, England, UK
A Walk with Nigel (2010)
A to Z: The Life Cycle of a Car (1997)
1   London, England, UK
```

Another goal was to display exactly which location linked each movie when a path is found. To accomplish this, the main program was modified to optionally read a tab-separated “edge description” (in this case, location name) after the edge weight in the edge file. This description is then stored as a field in Edge.java, and is displayed upon finding a shortest path, if present:

```
Vertices size: 6516
Edges size: 501442
Start vertex? A Christmas Story (1983)
Destination vertex? A Day in the Warsaw Ghetto: A Birthday Trip in Hell (1991)
Path is [A Christmas Story (1983),
        <<<connected by Cleveland, Ohio, USA>>>
A Film About Races (2009),
        <<<connected by Los Angeles, California, USA>>>
A Brief Sketch (2002) (V),
        <<<connected by Hollywood, Los Angeles, California, USA>>>
A Recipe for Love (2010),
        <<<connected by Warsaw, Mazowieckie, Poland>>>
A Day in the Warsaw Ghetto: A Birthday Trip in Hell (1991)]
Weight is 4
```

While this dataset works well with our program, it has some inherit flaws. One, the problem is actually an unweighted shortest path problem. Treating it as a weighted problem surely adds some computing overhead. Also, this dataset could be more efficiently represented as an undirected graph, as (movie1, movie2) implies the inverse. To accommodate this however, the program would have to be mostly rewritten.