Case Study- Train Signalling System

Tanya Shkinev

Overview

- Goal
  - Design and implement Train Signalling System
- Requirements
  - The system should implement the following components
    - Track segments of arbitrary length
    - Connections between segments
    - Signals which control traffic flow and can have two states GREEN or RED
    - Trains which initially placed at specific segment and can move both directions
    - The system should automatically find shortest route between start and destination points for each train
    - The system should provide facility to build a system by adding track segments, connections between tracks, trains, signals

Assumptions

- Tracks can be consisted of several segments and can be of different length
- Train length is equal to shortest segment length
- Trains run concurrently
- Trains move with the same speed and have the same priority

Choice of platform, programming language and framework

- C++
- Linux Ubuntu
- User Interface is implemented using XML files which can be modified by user
- These XML files serve as blueprints for the system builds

User Interface - configuration

The rail road model is described in XML file elements

- Track segments
- Connections for each segment
- Trains
- In current implementation signals created automatically – one per each segment step
- Train direction is set according to the aceding or descending order of segment IDs in the train path

Rail Road Model XML Schema (examples)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<RailRoad xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="RailRoad.xsd">

    <TrackSegments>
        <TrackSegment id="0" length="1"/>
        <TrackSegment id="1" length="1"/>
        <TrackSegment id="2" length="1"/>
        <TrackSegment id="3" length="1"/>
        <TrackSegment id="4" length="1"/>
        <TrackSegment id="5" length="1"/>
        <TrackSegment id="6" length="1"/>
    </TrackSegments>

    <Connections>
        <Connection source="0" target="1"/>
        <Connection source="0" target="2"/>
        <Connection source="1" target="3"/>
        <Connection source="2" target="4"/>
        <Connection source="3" target="5"/>
        <Connection source="4" target="5"/>
        <Connection source="5" target="6"/>
    </Connections>

    <Trains>
        <Train id="2" from="1" to="5"/>
        <Train id="3" from="6" to="1"/>
        <Train id="4" from="3" to="6"/>
    </Trains>

</RailRoad>
```

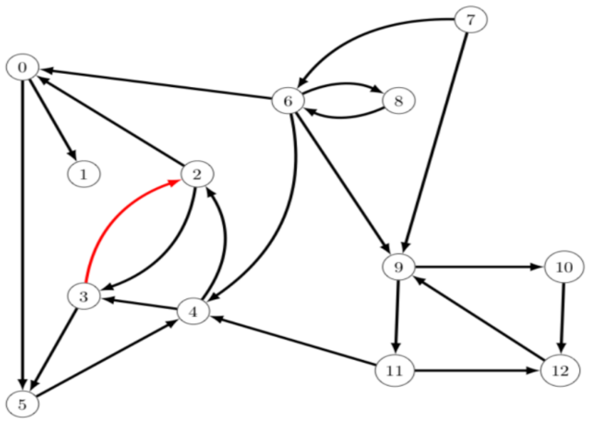
```xml
<?xml version="1.0" encoding="UTF-8"?>
<RailRoad xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="RailRoad.xsd">

    <TrackSegments>
        <TrackSegment id="0" length="1"/>
        <TrackSegment id="1" length="1"/>
        <TrackSegment id="2" length="10"/>
        <TrackSegment id="3" length="1"/>
        <TrackSegment id="4" length="1"/>
        <TrackSegment id="5" length="15"/>
        <TrackSegment id="6" length="1"/>
    </TrackSegments>

    <Connections>
        <Connection source="0" target="1"/>
        <Connection source="0" target="2"/>
        <Connection source="1" target="3"/>
        <Connection source="2" target="4"/>
        <Connection source="3" target="5"/>
        <Connection source="4" target="5"/>
        <Connection source="5" target="6"/>
    </Connections>

    <Trains>
        <Train id="12" from="5" to="1"/>
        <Train id="23" from="6" to="2"/>
    </Trains>

</RailRoad>
```

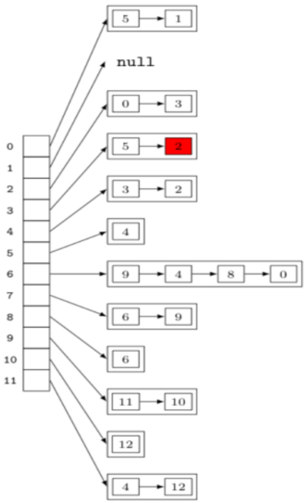
Main application functions

- Build the Rail Road – reads xml file, creates objects such Segments, Connections, Trains, Signals and build bidirectional graph representing complete rail segments model
- Print Rail Road map in adjacent list form
- Find Shortest Path – calculate shortest path for each train from source to destination point
- Start Train Engines (starts train threads)

Algorithms and Data Structures

- Complete rail road model is implemented by weighted bidirectional graph , where track segments represented as vertices. Connections between tracks segments represented as graph edges where the length of destination vertex is equal to segment weight
- Bidirectional weighted graph is implemented by STL map
  - Unorderedmap <<int>, list<int>> where map key is a segment id and element is list of segments connected to this segment from both ends





Algorithms and Data Structure - cont

- Two algorithms were implemented to find shortest path between train start and destination point
- BFS algorithm - Modified version of Breadth First Search Algorithm (BFS) to find and save distance from source to destination and list of segments included into this path
- Dijkstra algorithm - Modified version of Dijkstra Algorithm to find and save distance from source to destination and list of segments included into this path
- The Dijkstra algorithm was selected. It gives optimal solution for weighted graphs (see comparison table in next slide)
- BFS algorithm was implemented and tested as well. It is included into checked in code inside of ifdef USE_UNWEIGHTED_GRAPH

Algorithms and Data Structure – cont

	BFS	Dijkstra
Main Concept	Visit nodes level by level based on the closest to the source	In each step, visit the node with the lowest cost
Optimality	Gives an optimal solution for unweighted graphs or weighted ones with equal weights	Gives an optimal solution for both weighted and unweighted graphs
Queue Type	Simple queue	Priority queue
Time Complexity	$O(V+E)$	$O(V+E(logV))$

Classes

- CRailRoad
  - The class keep list of all objects in model – list of segments, connections, trains and signals
  - The class represents a complete map of railroad system
- CTrackSegment
  - The class represents a track segment , its length, connections and signals
- CTrain
  - The class represents a train and its source, destination and shortest path
- CTrainSignal
  - The class represents signal for each track sgement
- Utilities
  - Set of functions which implement various algorithms

Classes

CRailRoad()

```
m_TrainList;
m_tarckSegmentsMap;
```

```
void AddTrackSegment(int id, int len)
void AddTrain(int id, int from, int to)
void AddConnection (int src, int dest)
CTrackSegment*GetTrackSegment(int
segmentId)

get/set
const unorederd_map<int,
CTrain*>&GetTrainList();
uint GetTrainsNumber();
uint GetSegmentsNumber();
void PrintRailRoad();
```

CTrackSegmens

```
int segmentId;
int segmentLength;
m_signals;
m_trainRecords;
m_currentDirection;
```

```
int GetTrackSegmentsLen();
int GetTrackSegmentId();
GetTrackSegmentConnections();
AddConnections();
EMoveStatus MoveTrain(direction, trainId,
step);
bool ReleaseSection(intstopIndex);

void WaitForGreenSignal(int stepIndex);
bool TryToTakeGreenSignal(int stepIndex);
void ReleaseSignal(int stopIndex);
```

CTrainSignal

```
eSignal state;
mutex signal;
mutex cs;
thread_id thId;
```

```
ESiganlState GetState();
bool WaitForGreenSignalAndTake();
bool TryToTakeGreenSignal();
bool ReleaseSignal();
```

CTrain

```
int train_id;
int currentPoint;
int departurePoint;
int destinationPoint();
vector<int> trainPath;
list<CTrackSegments*> shortPathList;
std::thread trainThread;
std::mutex coutMutex;
```

```
int GetTrainId();
int GetTrainDeparturePoint();
GetTrainDestinationPoint();
const vector<int> GetTrainPat();
void SetTrainPath(vector<int> & vec);
int GetTrainPathLength();
void SetTrainShortPathLength();
void SetTrainShortPathList(list
<CTrackSegment *> sgList);
void RunRain()
void StartEngine();
void WaitForTrainArrival();
void PrintTrainPath();
```

Train Run Implementation

- Each train runs in its own thread
- All threads have the same priority
- Locking/signalling between threads implemented by mutexes
- The segment can be single step and multi step sizes
- Train passes a single step in one move. Segments with length > 1 are passed in several moves
- For multi step segments train locks a step when it enters it and releases when it leaves
- This multi step locking allows several trains to enter the same segment in the same direction
- When train enters step, the step signal turned RED, when train leaves, signal turned GREEN
- Before train attempts moving to the next segment, it checks the signal of the next segment
- If signal is RED, train waits for it to switch GREEN, otherwise it proceeds to next segment
- If train moving on a long segment and has not reached its end yet, it continues to move till it reaches next connection point to next segment

Simulation Output Format

```
Rail Road Connections Map
Segment 8: Connected to:7
Segment 7: Connected to:4 8
Segment 6: Connected to:5
Segment 5: Connected to:4 6
Segment 4: Connected to:1 3 5 7
Segment 3: Connected to:2 4
Segment 2: Connected to:3
Segment 1: Connected to:0 4
Segment 0: Connected to:1

Calculating path for train 2 dept 2 dest 6
Shortest path weight is : 8 for src : 2 to dest : 6
The shortest path contains : 5 segments
2 3 4 5 6


Calculating path for train 1 dept 0 dest 8
Shortest path weight is : 8 for src : 0 to dest : 8
The shortest path contains : 5 segments
0 1 4 7 8
```

```
Running train 2 path: 2 3 4 5 6
train: 2 at 2
train: 2 moving to 3
Running train 1 path: 0 1 4 7 8
train: 1 at 0
train: 1 moving to 1
train: 2 at 3
train: 2 moving to 4
train: 1 at 1
train: 1 waiting on RED for 4
train: 2 at 4
train: 1 waiting on RED for 4
train: 2 continues at 4
train: 1 waiting on RED for 4
train: 2 continues at 4
train: 1 waiting on RED for 4
train: 2 continues at 4
train: 1 waiting on RED for 4
train: 2 continues at 4
train: 2 moving to 5
train: 1 moving to 4
train: 2 at 5
train: 2 moving to 6
train: 1 at 4
train: 2 at 6
train: 1 continues at 4
train: 2 arrived
train: 1 continues at 4
train: 1 continues at 4
train: 1 continues at 4
train: 1 moving to 7
train: 1 at 7
train: 1 moving to 8
train: 1 at 8
train: 1 arrived
```

Test Cases run through simulation

- TC1: All segments with the same length which is equal to train length
- TC2: Segments have different length (the shortest length is equal to train length)
- TC3: Invalid XML file – missing elements such segments or/and connections
- TC4: Configuration with trains moving the same direction passing the same multi step segment
- TC5: Configuration with trains moving opposite direction passing the same multi step segment