Computer Science 150 – 01
Spring 2014
Prof. C.W. Liew

Ivan Evtimov

Project 1

1. **Set-up of the project**

The project has been implemented in 9 class files and 5 test classes.
The general organization of the simulation is as follows:
The Person class provides the functionality for simulating the people who are in the building. Every Person gets assigned a time to arrive in the run of the simulation. Two classes, Tenant and Visitor, are inherited from the Person class and take up all of its functionality. On each simulation run, the ratio of tenants to visitors is 3 to 1.
The Floor class holds the Person objects at the beginning of the simulation. It also provides the implementation that simulates the normal distribution of the times to arrive of the people. It uses an ArrayList container because quick random access by index is required.
The WaitingLine class simulates the waiting lines that pile up in the waiting area for entry to the staircases when the fire alaram goes off. Since not everybody can enter the staircase right away, people wait in the waiting line until there is an opening in the staircase. The WaitingLine class is an implementation of a queue and has been adapted from a previous lab assignment.
The Staircase class provides the functionality needed for moving the people objects outside of the building. It resembles a queue with multiple entry points in its implementation since people can only enter at their floor and can only exit at the bottom of the staircase. Again, quick random access by index and more importantly, constant size is needed, so an ArrayList is used in this class's implementation.
The Building class holds the containers for the multiple instances of the Staircase, the Floor, and the WaitingLine classes. Data is accessed through an iterator for the purposes of object-oriented data hiding, but the underlying data structure is an ArrayList once again for similar reasons as for the other classes.
The Simulation class ties everything together and runs the simulation providing information on time steps needed for completion. In the final version that has been submitted, it reads data from a *param.txt* file and uses it to start the simulation. However, for conducting the experiment, the class was temporarily modified to perform the simulation with different seeds and varying numbers of floors, etc., as per assignment.
For more details on the implementation, please consult the appropriate JavaDoc.

2. **Varying number of floors with a fixed amount of staircases**

For this experiment, I construct the building so that its staircases always have a capacity of 10. The number of tenants per floor is normally distributed and has a mean of 50 and a deviation of 10. The time to arrive for each tenant at each floor has also been normally distributed and has a mean of 30 and a standard deviation of 10. These variables are held constant for the purposes of this experiment.
The goal is to establish a pattern when we hold the number of staircases constant and vary the number of floors. Hence, I fixed the number of staircases at 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 100, 150, 163. At each number of staircases I run the simulation with 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 25, 50, 100, 150, and 163 floors. The number 163 was chosen as upper bound because that's the number of floors of the

currently tallest building in the world.
Out of all the generated data, I choose to consider the situation with fixed staircases numbering 1, 10, 100, 150. Since each simulation run was performed with 6 different seeds, I look at the average of all of them.
The results can be found in these tables:

Table 1: Average number of steps taken to complete evacuation of building with varying number of floors.

Number of Floors	Staircases: 1	Staircases: 10	Staircases: 10	Staircases: 15
1	62	17	8	8
2	130	23	10	5
3	212	31	11	9
4	294	39	11	9
5	352	45	13	11
6	433	53	15	11
7	509	60	15	13
8	574	67	16	13
9	656	75	17	15
10	716	81	18	15
25	1680	178	27	22
50	3349	344	44	33
75	4987	507	59	43
100	6642	673	76	54
150	9986	1007	109	76
163	10932	1102	119	

The data is also plotted here:

Figure 1

Average time steps needed for completion with fixed number of staircases

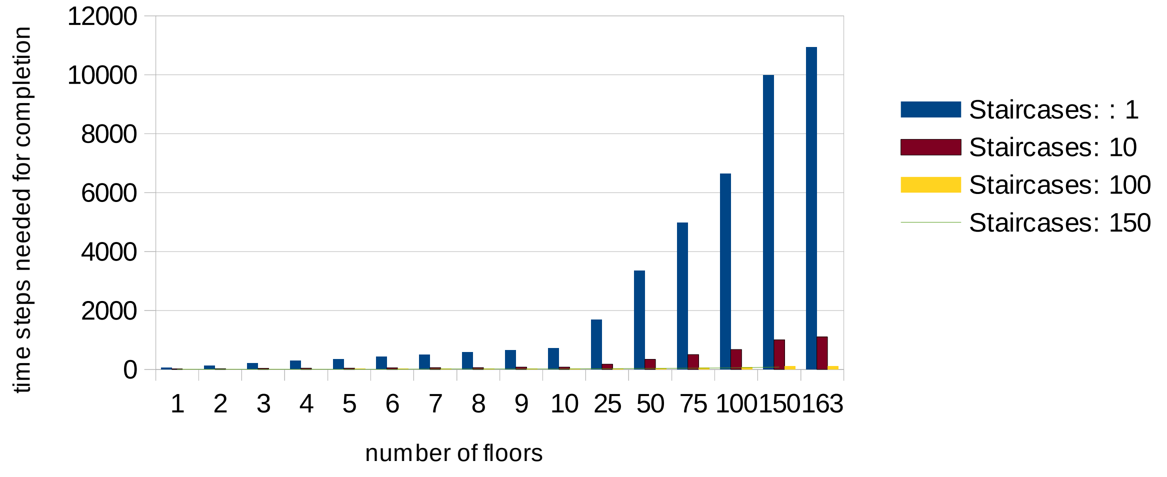
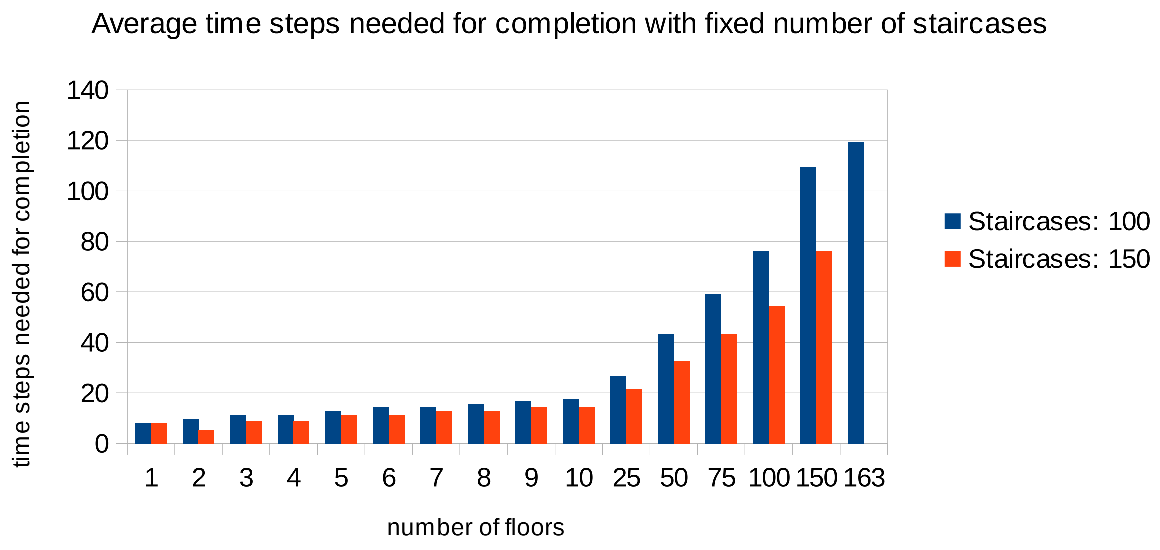


Figure 2



It is evident, especially from Figure 2, that as the number of floors grows, so does the time required to evacuate the building with a fixed amount of staircases. This corresponds to the situation in the real world. For a faster evacuation, you will need more staircases.

3. **Fixed required time for evacuation**

The experiment is set up, so that the simulation doesn't complete until a satisfactory number of staircases (that evacuates the building under the given amount of time) has been found. Based on the output, I could see that no matter how many times the simulation was run with different seeds, the number of staircases was all the same. The fixed timestep limit chosen was 300 time steps. The same other variabales are held constant as in 2.

The data is given in the following table:

Table 2: Number of staircases needed for evacuation within 300 timesteps

Number of floors	Staircases needed for evacuation within 300 timesteps
1	15
11	15
21	15
31	15
41	15
51	20
61	20
71	25
81	25
91	30
101	30
111	30
121	35
131	35
141	40

It is evident from this table that the number of staircases needed for evacuation within a preset time grows as the number of floors grows. However, it does not grow as fast as the timesteps needed for evacuation of a building as the number of floors grows with a fixed number of staircases. Again, this situation is analogous to the real world.