For office use only	Team Control Number	For office use only
T1 _____	**1907110**	F1 _____
T2 _____		F2 _____
T3 _____	Problem Chosen	F3 _____
T4 _____	**D**	F4 _____

2019
MCM/ICM
Summary Sheet

Optimal Evacuation Plan of Louvre Based on the Graph Theory

Summary

Nowadays, the terrorist attacks in France have risen public concern. As one of the most popular tourist attractions in the world, Louvre is in urgent need of an excellent emergency evacuation plan.

First, we introduce the concept of directed weighted graphs in graph theory. We abstract hundreds of exhibition halls in Louvre into 24 nodes, the four main exits into four export nodes and the escape channel connecting two nodes into a directed edge. As a result, we boil down the problem of solving the optimal escape route to find the lowest cost between two nodes in the directed weighted graph.

Second, we select transit time as the weight of each edge. To calculate the time, we start from both the actual distance between two adjacent nodes and evacuation speed and summarize 3 types of road and 6 types of people with different moving speed. Next, under the conditions of fixed distance and flow velocity, we use Dijkstra algorithm to solve the optimal path from one node to exit node.

Third, we also need to consider that during the actual evacuation process, changes in flow density and exit waiting time have an impact on the calculation of evacuation time. In response to the above two problems, we first find the function of flow density versus flow velocity by consulting the literature, and then we introduce the concept of a super-exit to ensure that the exit waiting time is added to the calculation of the optimal path.

Last but not least, we set a series of security thresholds to determine the possible security bottlenecks in the current emergency evacuation model and discuss when to open hidden exits and how to avoid possible security incidents. In the end, we propose 3 practicable suggestions on how to implement our model to Louvre and other large-scale public places.

Keywords: Evacuation;Directed weighted graph;Dijkstra;Super Exit

Contents

1 Introduction

1.1 Problem Background

From students being killed in Toulouse to a soldier attacked outside the Louvre museum in Paris, there have been at least 12 major terror-related incidents in France since 2012. The increasing number of terror attacks in France requires a review of the emergency evacuation plans at many popular destinations. As the world's largest art museum and the most famous tourist destination in France, the Louvre has become the preferred target of terrorists. In 2017, the Louvre attracted more than 8.1 million visitors from all over the world, which provided major challenges for regular movement with the museum. Additionally, the diversity of visitors - multilingual, group travel and disabled travellers - makes evacuation in emergencies more challenging.

In view of these problems, it's extremely significant for us to propose evacuation plans which help all the occupants leave the Louvre as quickly and safely as possible. We need to consider the specific architectural structure of the Louvre and the potential bottlenecks to design the best evacuation path for the Louvre.

1.2 Problem Restatement

After careful analysis of the problem, we list the issues that the problem requires.

1. Design a model which allows visitors to evacuate from the Louvre as soon as possible when an emergency occurs.
2. The model should allow emergency personnel to enter the building as quickly as possible when an emergency occurs.
3. Consider when and how any additional exits might be utilized.
4. Our model should be adaptable to potential bottlenecks.
5. Propose policy and recommendations for emergency management of Louvre.

2 Model abstraction and simplification

Because the internal structure of the Louvre is complex and there are too many exhibition sites, we first abstract the evacuation of the Louvre.

We define ***connected*** as: A node can reach another node by corridor,elevator,stairs or disabled access.

- We abstract the exhibition halls which are close and interconnected with each other and four exits as geometric centers and represent them by nodes (the Louvre is abstracted as 28 nodes in total).
- If two nodes are connected, we draw an edge between them.
- The weight of the edge represents the average transit time between nodes.
- The edges are directed which can be unidirectional or bidirectional.

3 Model I: Ideal Louvre Evacuation Model

3.1 Assumptions

Considering the complicated internal structure of the Louvre and the excessive number of exhibition points, we make the following basic assumptions[1] in order to simplify the problem. Each of our assumptions is justified and is consistent with the basic fact.

- **There are no obstacles in the corridor or on the stairs.** Evacuation of all types of people is unimpeded.
- **Personnel are not allowed to circle during the evacuation process.** Because ideally, the personnel will always move in the direction of the exit.
- **The personnel is divided into six different types and the evacuation speed among the same type of personnel is mostly the same.**
- **We assume that the elevator is out of service during the evacuation process.** Personnel can only move by upstairs,downstairs or corridor.

3.2 Notations

Here we list the symbols and notations used in this ideal louvre evacuation model, as shown in Table 1.

3.3 Model Implementation

3.3.1 Model construction

Directed multi-distance weighted graph Firstly, we divided the passages in Louvre into three parts,upstairs,downstairs and corridor. We explain the symbol

Abbreviation	Description
$G(V, E)$	The evacuation network
V	The set of nodes, $V = \{D, N\}$
$D = \{D_1, D_2, ..., D_k\}$	The set of exit points
$N = \{N_1, N_2, ..., N_r\}$	The set of exhibition hall node
$E = \{e_{ij}\}$	The set of edge, where e_{ij} represents the edge between i and $j(i, j \in V)$
L_{ij}	The distance between node i and node j
$W_{ij}^{(m)}$	The weighting coefficient of average transit time from node i to node j w.r.t. the m^{th} type of personnel
$t(i, j, m)$	The average transit time from node i to node j w.r.t. the m^{th} type of personnel
$P_l(i, m)$	The path available for the m^{th} type of personnel at node i to be evacuated, where $l = 1, 2, ..., i = 1, 2, ...28$
$T_{P_l}(i, k, m)$	The time required for the m^{th} type of personnel at node i to reach the k^{th} exit point through the path P_l. $T_{P_l} = \sum_{e_{ij} \in P_l}(t(i, j, m))$
$T_{P_l}(k, i, m)$	The time required for the m^{th} type of personnel to reach node i from the k^{th} entrance

Table 1: Notations

L_{ij} in the above notation table in detail. L_{ij} represents the distance between node i and node j. The distance can be represented by a three-dimensional vector.

$$L_{ij} = \begin{bmatrix} L1_{ij} \\ L2_{ij} \\ L3_{ij} \end{bmatrix} \tag{1}$$

where

- $L1_{ij}$ represents *the upstairs distance* from node i to j
- $L2_{ij}$ represents *the downstairs distance* from node i to j
- $L3_{ij}$ represents *the corridor distance* from node i to j

Therefore, we can use these three-dimensional vectors as elements for our directed multi-distance weighting graph.

Directed time weighted graph After consulting relevant literature[2], we got the relationship between personnel density and personnel flow velocity, which is shown as Table 2.

According to Table 2, in the ideal case, the speed of walking downstairs, upstairs and in the corridor is 1m/s, 0.8m/s and 1.4m/s respectively.We use q_1, q_2, q_3 to represent them accordingly.

Then we explain the symbol $W^{(m)}$ in the above notation table in detail.$W^{(m)}$ represents the weighting coefficient of the average transit speed of the m^{th} type of personnel. The values we set for the weighting coefficients are shown in Table 3.

We can calculate the speed $v(m,i)$ at which the m^{th} type of personnel advances in the i^{th} way.

$$v(m,i) = W^{(m)} * q_i \tag{2}$$

Definition of crowded conditions	Personnel density $(person/m^2)$	Personnel flow velocity v(m/s)			Personnel flow volume $(person/(m \cdot s))$		
		Stairs(down)	Stairs(up)	Corridor(doorway)	Stairs(down)	Stairs(up)	Corridor(doorway)
Low	<1.9	1.00	0.80	1.40	0.54	0.43	0.76
Good	1.9∼2.7	0.50	0.40	0.70	0.94	0.75	1.30
Medium	2.7∼3.2	0.28	0.22	0.39	0.77	0.62	1.12
Crowded	>3.2	0.13	0.10	0.18	0.42	0.32	0.55

Table 2: The relationship between personnel density and personnel flow velocity

Type	m	Weighting coefficient
Normal visitors	1	1
Disable visitors	2	0.5
Groups traveling visitors	3	1.2
Elderly visitors	4	0.7
Emergency personnel	5	1.6
Other language speakers	6	0.85

Table 3: Weighting coefficient of average transit speed for different types of personnel

Then we combine with the *Directed multi-distance weighted graph* to get m different *Directed time weighted graph* $G_m(V, E)$ for different types of personnel. Each element of the adjacency matrix in *Directed time weighted graph* can be calculated by formula 3.

$$t(i, j, m) = \sum_{k=1}^{3} \frac{L_k(i, j)}{v(i, j, m, k)} \tag{3}$$

Evacuation Personnel Based on the above-defined symbols and variables, the problem of optimizing the evacuation path for different types of people in each node becomes the optimization problem. The optimization objective function is shown as formula 4.

$$minT_{P_l}(i, k, m) = min \sum_{e_{ij} \in P_l} (t(i, j, m)) \tag{4}$$

Emergency Personnel For emergency personnel, the optimal path from the k^{th} exit to node j is also an optimal problem. The optimization objective function is shown as formula 5.

$$minT_{P_l}(k, i, m) = min \sum_{e_{kj} \in P_l} (t(k, i, m)) \tag{5}$$

The above two optimization problems are equivalent to the problem of finding the minimum weight path between two points of the *Directed weighted graph*. We use the Dijkstra algorithm to solve the problem.

3.3.2 Algorithm display

The specific operation of Dijkstra algorithm is shown in Figure 1.

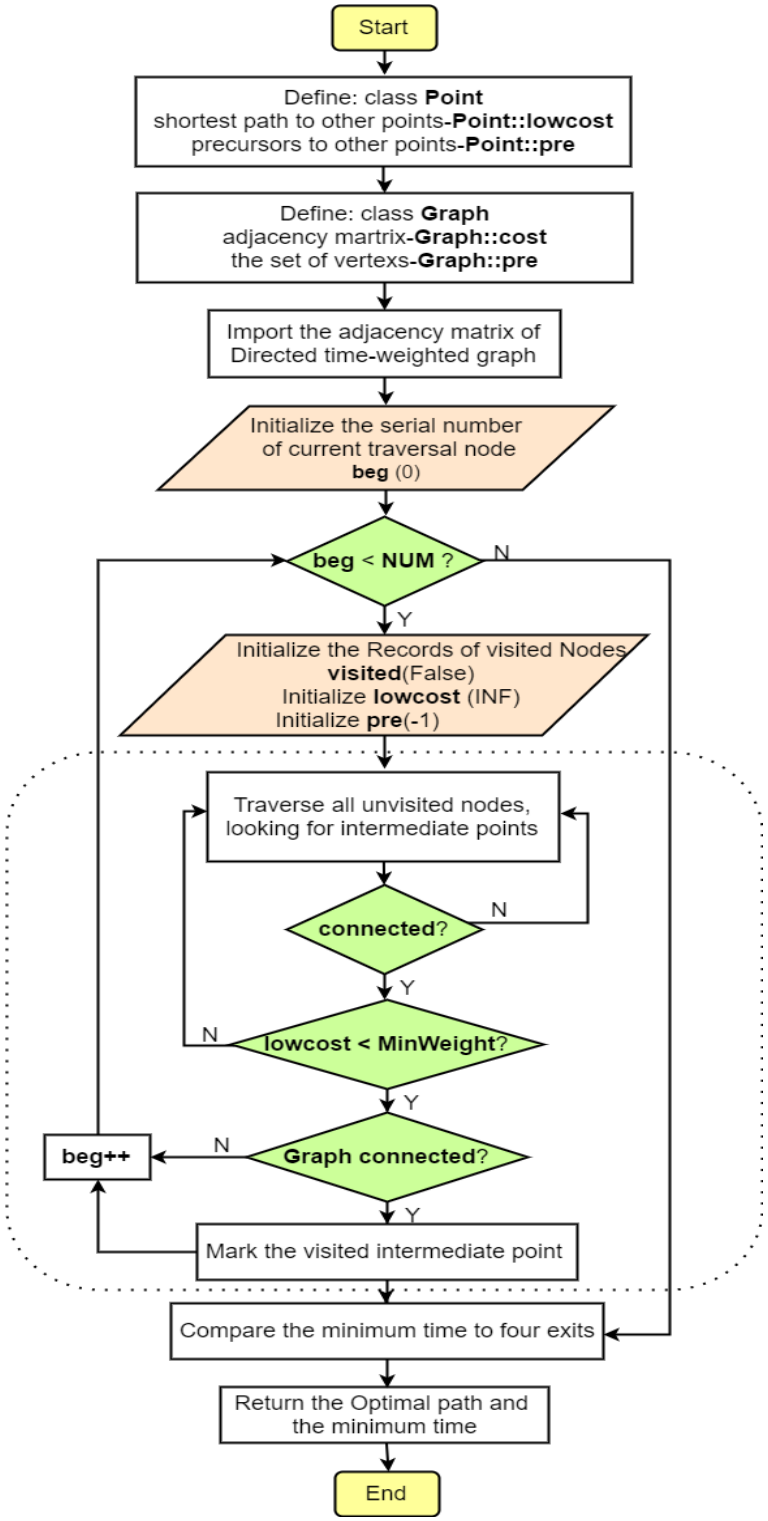

Figure 1: Dijkstra algorithm in simulating Model I.

3.3.3 Result Analysis

Result Display We implement the model by C++ programming and find the the optimal evacuation path and evacuation time for the normal personnel at each node under ideal conditions. The results are shown in Table 4 and Figure 2.

Potential bottlenecks ***The bottleneck zone*** is one of the high-incidence areas of crowded stampede events. When people crowd from spacious spaces to narrow bridges, entrances or stairways, many people will squeeze in from both sides, hindering the normal crowd. Due to the increased cluster density, an arched crowd will form at the entrance and exit ,where everyone squeeze together and can not pass. Without timely evacuation guidance, most people were squeezed down due to the sudden loss of balance and they were sneaked to death by latecomers who were eager to go out.
As a matter of experience and convenience, the bottleneck zone should ensure

Point	Out	Best Time(s)	Path
2	1	150	$2 \to 1$
3	1	120	$3 \to 1$
4	1	195	$4 \to 3 \to 1$
5	1	180	$5 \to 1$
6	1	200	$6 \to 1$
7	14	125	$7 \to 14$
8	13	160	$8 \to 9 \to 13$
9	13	100	$9 \to 13$
10	13	175	$10 \to 9 \to 13$
11	16	150	$11 \to 15 \to 16$
12	16	150	$12 \to 15 \to 16$
15	16	75	$15 \to 16$
17	14	325	$17 \to 7 \to 14$
18	14	385	$18 \to 17 \to 7 \to 14$
19	13	260	$19 \to 9 \to 13$
20	13	270	$20 \to 10 \to 9 \to 13$
21	16	315	$21 \to 11 \to 15 \to 16$
22	16	350	$22 \to 11 \to 15 \to 16$
23	16	320	$23 \to 24 \to 15 \to 16$
24	16	195	$24 \to 15 \to 16$
25	14	455	$25 \to 17 \to 7 \to 14$
26	14	460	$26 \to 17 \to 7 \to 14$
27	13	390	$27 \to 19 \to 9 \to 13$
28	16	395	$28 \to 21 \to 11 \to 15 \to 16$

Table 4: Ideal optimal evacuation path and evacuation time for normal visitors

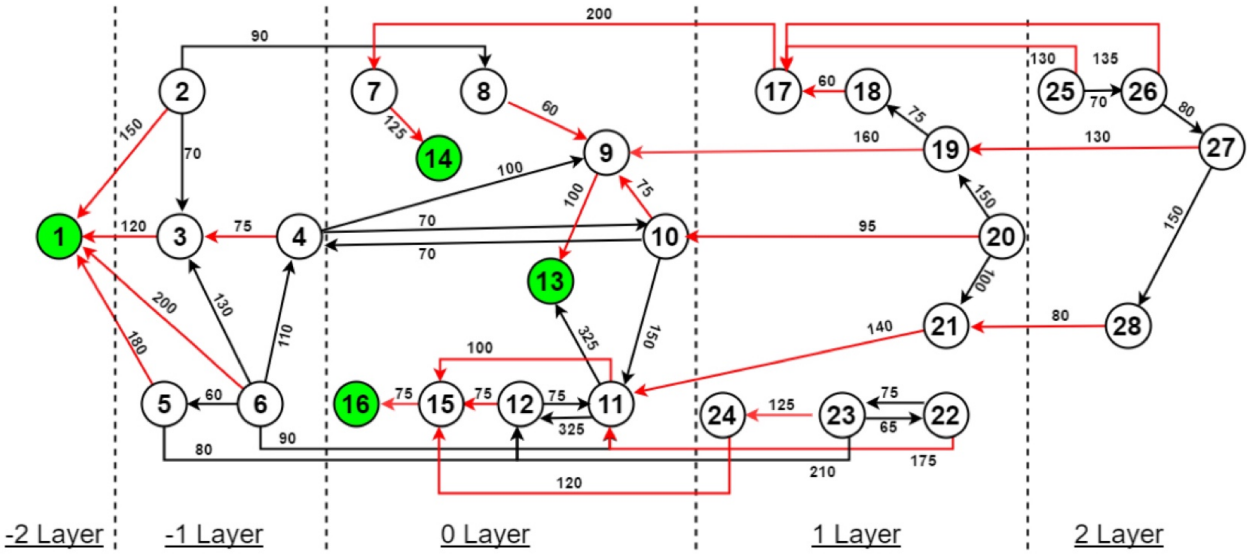

Figure 2: Ideal optimal evacuation path for normal visitors

that personnel density is no more than 3 $persons/m^2$. In our model, we select the number of optimal paths passing through one node to determine whether the node is a bottleneck zone or not. Considering that most of visitors are normal people with free mobility, when judging the potential bottleneck under ideal conditions, we only consider the flow of normal types of individuals.

- For each node i, we set a bottleneck coefficient z_1. z_1 equals to the number of optimal paths through node i. Besides, we set a threshold of 3. When z_1 is greater than or equals to 3, the node i is more likely to be a bottleneck node.
- For paths, we set another bottleneck coefficient

$$z_2 = \frac{L1_{ij} + L2_{ij}}{L_{ij}} * 100\%, \tag{6}$$

which means the percentage of the length of upstairs and downstairs in the length of total path. Besides, we set another threshold of 0.4 for paths. When z_2 is greater than or equals to 0.4, the path is more likely to be a bottleneck path.

As shown in Table 4, z_1 of node N_9, N_{17}, N_{15} is greater than or equals to 3 so they may be potential bottlenecks hindering movements to the optimal exit. At

the same time, we find that z_1 of exit D_1 is equal to 4 so D_1 is very likely to occur congestion and there are potential security risks, so we recommend to strengthen security forces and add more emergency personnel at 1^{th} exit point.

4 Model II: Modified Louvre Evacuation Model

The actual emergency evacuation situation need to consider more influence factors. On the one hand, high flow density will cause inevitable environmental resistance, which has a significant influence on evacuation speed. On the other hand, to calculate the waiting time at the exit queue, we has to consider the maximum number of people allowed to exit in a unit of time and potential bottleneck. Based on Model I, Model II gives a practical solution to the above two problems.

4.1 Additional Assumptions

- Under the same flow density, the evacuation speed of the same type of people is the same and the different type of people have the same rate of decline in speed.
- High flow density will cause crowding effect, which will reduce the evacuation speed of the crowd.
- The maximum number of people allowed to exit per unit time is known.
- At all exit points, everyone is queued in order and first come first in first out.

4.2 Additional Notations

In addition to the variables listed in Model I, Model II adds some new variables which are shown as Table 5.

4.3 Model Implementation

4.3.1 Model construction

Changes in evacuation speed of people under different flow densities. Combined with our common sense of life, in a specific evacuation site, the population density is higher, the average passenger flow speed is slower, that is to say, there is a certain inverse relationship between the escape speed and flow density. By consulting the literature, we get the relationship between the flow density and the evacuation speed of normal people, as shown in Table 2.

Abbreviation	Description
$v(p,m,q)$	The evacuation speed of the m^{th} type of personnel when the flow density is p and the evacuation mode is q
$t(i,j,p,m)$	The average transit time of the m^{th} type of personnel from node i to node j when the flow density is p
Q_k	Maximum evacuation volume of the k^{th} exit point per unit time
X_k	Number of people waiting to leave the Louvre at the k^{th} exit point
T_k	Time to wait at the k^{th} exit point

Table 5: Additional Notations

Optimal evacuation route Based on the defined notations in Table 5, the mathematical model of Model II is as follows: In case that the flow density of each hall of Louvre can be detected in real time, to find the optimal evacuation route of node i, we need to calculate the optimization objective function which is shown as formula 7.

$$minT = min(T_{P_l}(i,k,m) + T_k) \tag{7}$$

4.3.2 Algorithm display

In case of known flow density and the calculation method between flow density and evacuation speed:

$$v(i,j,p,m,q) = W(m,q) * v(i,j,p,q) \tag{8}$$

Refer to *Directed multi-distance weighted graph*:

$$t(i,j,p,m) = \sum_{k=1}^{3} \frac{L_k(i,j)}{v(i,j,p,m,k)} \tag{9}$$

We can get new *Directed time weighted graph.* As for the other problem, because the exit waiting time is considered, when T is minimum, $T_{Pl}(i,k,m)$ is not necessarily the smallest, that is to say, we can use the algorithm in Model I to directly calculate the new optimal path.
In order to calculate $minT$, we introduce the method of extended graph $G^*(V,E)$ hence we can transform the current problem into a model suitable for the algorithm in Model I.
We set up a **super exit**: this super exit has unlimited capacity and personnel doesn't need to queue up at this super exit at anytime.[3] We define the time weight of the edge between the super exit and the exit which is one of the four exit points as T_k(the waiting time of D_k)[4]. The specific operation is shown in Figure 3.

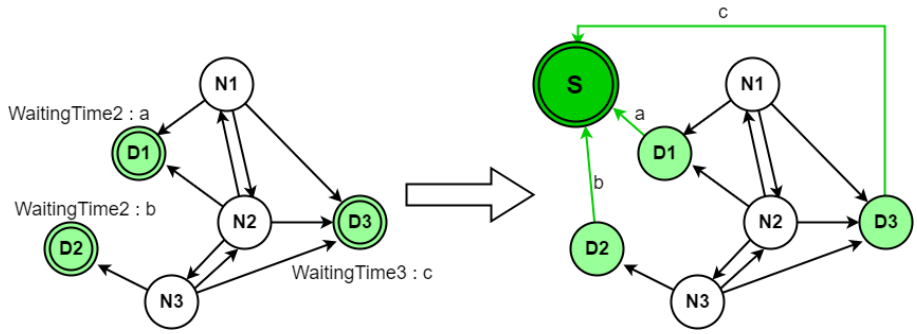

Figure 3: Super Exit.

Through the above steps, the problem of calculating $minT$ becomes the problem of calculating the minimum weighted path from node i to node k in the extended graph $G^*(V,E)$.

The specific operation of modified Dijkstra algorithm is shown in Figure 4.

4.3.3 Result Analysis

Result Display In Model II, the best escape route for the m^{th} type of individual at node i equals to the minimum weighted path$(node\ i \rightarrow exit\ k)$ in the extended graph $G^*(V,E)$. On the contrary, the best entry path for emergency personnel equals to the minimum weighted path$(exit\ k \rightarrow node\ i)$ in the extended graph $G^*(V,E)$. This two solutions are exactly the same. Finally, we find the optimal evacuation path and evacuation time for each node combined with the real-time monitoring of the flow density, as well as the optimal entry path and entry time for emergency personnel to arrive at the accident location. The optimal evacuation path and evacuation time for the normal person are shown in Figure 5 and Table 6.

Note that the yellow portion of Table 6 represents the node in Model II where the optimal evacuation path changes compared to Model I.

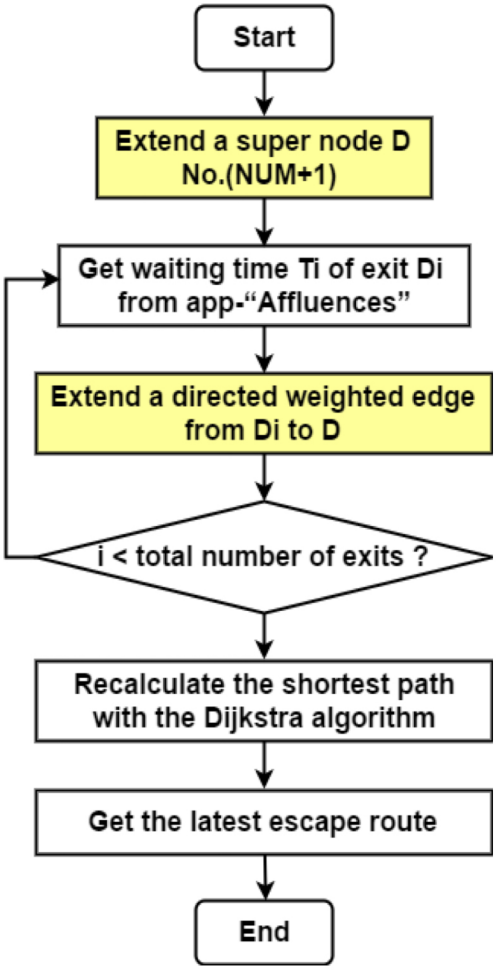

Figure 4: Dijkstra algorithm in simulating Model II

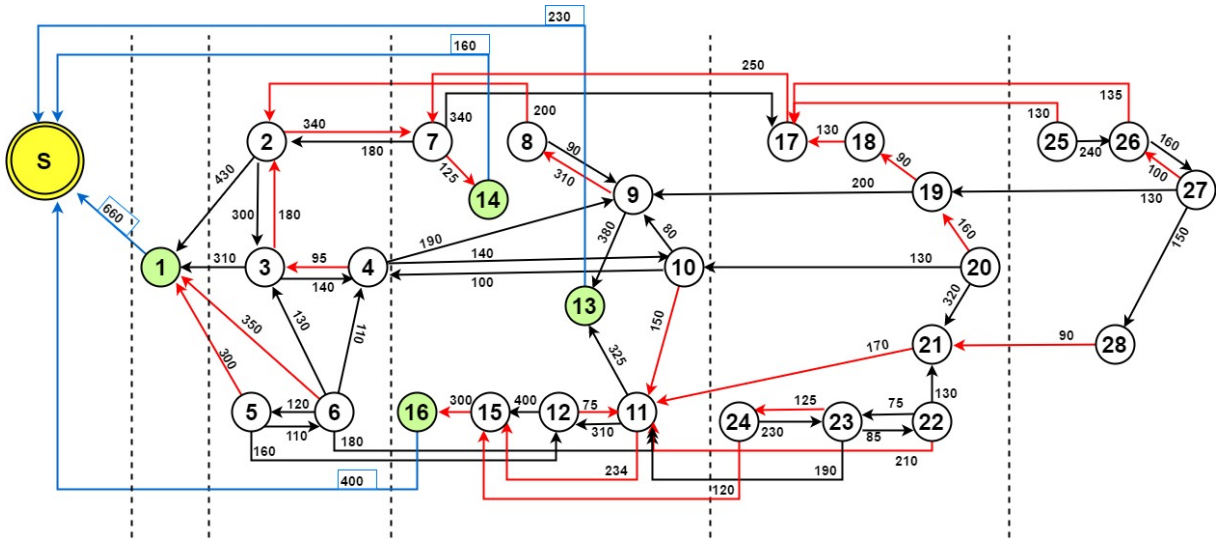

Figure 5: Modified optimal evacuation path and evacuation time for normal visitors

Point	Out	Best Time(s)	Path
2	14	740	$2 \to 7 \to 14$
3	14	920	$3 \to 2 \to 7 \to 14$
4	14	1015	$4 \to 3 \to 2 \to 7 \to 14$
5	14	960	$5 \to 1$
6	1	1010	$6 \to 1$
7	14	400	$7 \to 14$
8	14	940	$8 \to 2 \to 7 \to 14$
9	14	1250	$9 \to 8 \to 2 \to 7 \to 14$
10	16	1084	$10 \to 11 \to 15 \to 16$
11	16	934	$11 \to 15 \to 16$
12	16	1009	$12 \to 15 \to 16$
15	16	700	$15 \to 16$
17	14	650	$17 \to 7 \to 14$
18	14	780	$18 \to 17 \to 7 \to 14$
19	14	870	$19 \to 18 \to 17 \to 7 \to 14$
20	14	1030	$20 \to 19 \to 18 \to 17 \to 7 \to 14$
21	14	920	$21 \to 11 \to 15 \to 16$
22	16	1020	$22 \to 11 \to 15 \to 16$
23	16	945	$23 \to 24 \to 15 \to 16$
24	16	820	$24 \to 15 \to 16$
25	14	780	$25 \to 17 \to 7 \to 14$
26	14	785	$26 \to 17 \to 7 \to 14$
27	14	885	$27 \to 26 \to 17 \to 7 \to 14$
28	14	1010	$28 \to 21 \to 18 \to 17 \to 7 \to 14$

Table 6: Modified optimal evacuation path and evacuation time for normal visitors.

Potential Bottleneck The judgement metrics of potential bottleneck in Model II is the same as that of Model I.

- Under the premise of considering flow density and exit waiting time, we find the best escape route for individuals at each exhibition hall and if the number of optimal evacuation paths passing through node j is greater than 3, we regard node j as potential bottleneck.
- The way to determine the potential bottleneck of the path is exactly same as Model I.

5 Specific implementation plan for evacuation

According to the requirements of the problem, our model needs to be highly adaptable. We will discuss once different kinds of emergencies occur, how to

validate our model and how the Louvre would implement it.

5.1 Model adjustments for different emergencies

We regard the place where accident is happening as a ***breakpoint***.

- When the breakpoint is on the edge, then $e(i, j)$ become disconnected so we change the weight of $e(i, j)$ to *INF*.
- When the breakpoint is node i, then node i cannot accommodate visitors and visitors cannot pass through node i, so we delete both node i and all edges connected to node i, that means we delete $Row(i-1)$ and $Column(i-1)$ in adjacency matrix.

Then we can recalculate the best evacuation route based on the new *Directed time weighted graph* which excludes breakpoints.

5.2 The specific implementation for Louvre

5.2.1 Crowd management and control procedures

We recommend setting up monitoring points for flow density at each entrance to the Louvre and all the entrances and exits of each node. By importing the detected data of flow density into Model II, a *Directed time weighted graph* can be generated to calculate the minimum evacuation time of each node. We call the node whose shortest evacuation time still exceeds the safe time as a ***timeout node***. We set a security threshold of 3.

- When the number of timeout nodes in the graph is less than 3, we suggest that the museum managers send staff to the entrances to the timeout node to implement the crowd management measures for exhibition hall, which means the exhibition hall is only available to leave while visitors cannot enter. After implementing the crowd management measures in the exhibition halls, the flow density and the minimum evacuation time of the timeout node will decrease with time. We will not reopen the exhibition halls until the latest new minimum evacuation time is less than three quarters of the safe time.
- When the number of timeout nodes in the figure is greater than or equals to 3, this situation indicates that the total number of visitors exceeds the safety load. Therefore, we propose to implement the overall crowd management measures at the entrances of the Louvre while implementing the local crowd management measures in the exhibition halls. The overall crowd management measures means the entire Louvre only allow visitors to leave

but not allow to enter. Similarly, we will not reopen the entrances to the Louvre until the number of timeout nodes in the graph is less than 2.

5.2.2 App Function Extension

We also advise Louvre administrators to expand the functionality of the App ***"Affluence"***.

1. ***Multi-language version***.
2. ***The function of finding the optimal route***.Visitors can choose the exhibition hall they are visiting and the type of people they belong to(aged people/ disabled people/member of tour group) to get real-time optimal route.Moreover, for the reason that our model uses the *Dijkstra algorithm* to solve the shortest distance, we can obtain the optimal routes from one node to all other nodes. Accordingly, this function can also be extended to choose the current node and the target node to get the optimal route between them, which can not only improve the evacuation speed in the emergency state, but also give the visitors a better experience during normal sightseeing.

5.2.3 Plans for opening the additional exit points

We assume that there are three situations in which additional exits(Other available exit points except the four main exits) need to be opened:

1. The evacuation time of one node exceeds the safe time.
2. The current optimal path has many potential bottlenecks which we have discussed in Model I and Model II.
3. Emergency personnel need to reach the accident location as soon as possible.

In addition, the potential safety hazards that may exist in additional exits are not negligible. For instance, insufficient lighting equipment is prone to stamping accidents; the exit may collapse due to the lack of maintenance and excessive load; the cultural heritage may be damaged, etc.

For the first two cases, we set a safe time threshold c. If evacuation time is greater than c, we need to consider open other exits nearby to achieve crowd diversion. In view of the complexity of potential safety hazards in additional exits, we assume that an additional exit is safe only when it has no waiting time. When the museum leaders decide to open an additional exit, the emergency personnel responsible for opening this additional exit need to control the flow rate of this exit within a certain range to ensure that this additional exit will not block. In

the new graph with an additional super exit added, the edge weights of all the extra exits to the super exit are set to 0.

As for the third case, we can exhaust all the situations, find the minimum entry time for emergency personnel when going to different nodes, and choose a solution that can open fewer additional exits and ensure emergency personnel reach the node where the accident happened quickly.

6 The Simulation

We use the software ***AnyLogic*** to simulate our Louvre evacuation model. AnyLogic is a widely used tool for modeling and simulating discrete, system dynamics, multi-agent and hybrid systems. We use the pedestrian library in AnyLogic to simulate the individual behavior of tourists in the case of emergency evacuation. The key to implement the simulation of individual behaviors is to design the influence function of flow density and flow velocity. From the relevant literature[5], we know that under normal circumstances, the flow velocity in the horizontal channel is:

$$V = 112D^4 - 380D^3 + 434D^2 - 217D + 57 \tag{10}$$

Under emergency, the flow velocity in the horizontal channel becomes:

$$V_e = V \times (1.49 - 0.36D) \tag{11}$$

where

- V represents flow velocity
- D represents flow density. $D \leqq 0.92$.

When the flow density reaches or exceeds this value, the flow of people will become crowded or clogged.

For the convenience of simulation, we assume that before evacuation begins, tourists are distributed in different nodes with different density values, and there are no pedestrian on the channel. When the simulation starts, the flow from different nodes moves to the exit according to the pre-specified optimal route. The instant screenshot of the dynamic simulation process is shown as figure 6.

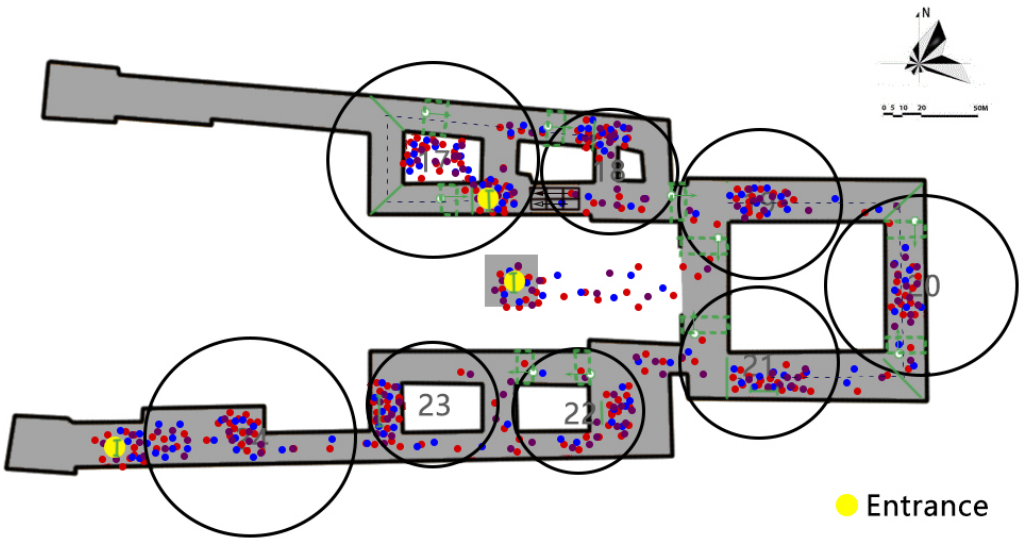

Figure 6: Layer 0 plane simulation diagram

The simulation results show that the evacuation time of the simulation is a little longer than that obtained by our proposed model. The main reason for this result is that during the evacuation process, the escalators, the entrances of the stairway and the corner of the evacuation passage are narrow, which become the bottlenecks of evacuation. During the actual evacuation process, pedestrians will stay in the bottleneck area for a certain period of time, so the simulated evacuation time is longer.

7 Conclusion

7.1 Conclusion of the Problem

We establish a *Directed multi-distance weighted graph* and a *Directed time weighted graph* to build our model. At the same time, we take many factors which may affect our model into consideration such as personnel density, the type of evacuation personnel, and the waiting time at exit points. We implement our model by C++ programming and use the *Anylogic* software for simulation verification.

According to the results shown by our model, N_9, N_{15}, N_{17}, D_1 are potential bottleneck nodes. $e_{27}, e_{28}, e_{49}, e_{4'10}, e_{11'23}, e_{17'25}$ are potential bottleneck path. The Louvre management should add staff to these bottlenecks to increase the width of the passage and improve safety. At the same time, due to the long evacuation time of special personnel such as the disabled, it is necessary to increase the convenient passage for evacuation of the disabled, such as automatic escalators. When the total evacuation time exceeds the safe value, additional exits should be opened and additional maintenance personnel should be deployed at the addi-

tional exit to prevent additional exits being blocked.For the entry of emergency personnel, it is necessary to avoid the path of large-scale evacuation of people, and can consider the opening of additional exit points. When there are special situations such as breakpoint, the model parameters need to be updated in real time to realize real-time scheduling.

For the case of changes in human flow density, the results of our models show that as the flow density increases, the evacuation time increases significantly. It is therefore necessary to control the number of people entering the Louvre and entering the various exhibition halls in real time, so that the maximum evacuation time is within the safe value.It is of great importance to detect the human flow density. Consequently,it is recommended that the Louvre management set up a flow density monitor in both the route and the exhibition hall so that the best evacuation plan can be updated in real time.

7.2 Strengths and Weaknesses

7.2.1 Strengths

Our model take time, space, flow density into consideration.Based on the analysis above, we list strengths of our model as follows:

- **Our model takes many factors which may affect the evacuation speed into consideration.**
  After consulting relevant literature, we have relative knowledge of different evacuation speed w.r.t different types of personnel and different types of evacuation path.We calculate the corresponding evacuation speed for each different type of combination. So our model is comprehensive.

- **Our model is simple but universal.**
  The main strength of our model is that as long as we have enough data to set precise inputs,such as the distance between different nodes through different ways and flow density,then we can get the optimal evacuation path as the output. And the models can still be well applied in other large,crowded structures and we just need to change our input parameters.

- **We cleverly added a super node to the model.**
  We add a super node with no capacity limit, which makes the model still usable even when the problem conditions become more complicated(taking into account the wait time of the exit).

- **Our model is practical and reliable.**
  We do simulation to test the accuracy of our model. Based on the simulation results, we safely draw the conclusion that the simulation in our model fits the actual simulations.

- **Our model has good scalability.**
  The Dijkstra algorithm can find the shortest path between any two nodes. If the app can be implemented, visitors can set the location of the current exhibition hall and the location of the target exhibition hall to get the best tour route.

7.2.2 Weaknesses

- **The Dijkstra algorithm has a large computational complexity.**
  The algorithm finds the shortest path from the starting point to each of the remaining nodes and it is necessary to traverse all the paths and nodes,which leads to a large computational complexity.

- **There may be deviations in the optimal evacuation time.**
  Because we don't get a specific Louvre plan, some data deviated from the real value which may have an effect on the optimal evacuation time and optimal evacuation path, such as the distance between the exhibition halls.However, if we can get the precise parameters as inputs, the performance of our model would be better.

- **Our model is a little macro.**
  We mainly consider the evacuation of personnel from different type of groups in an exhibition hall and our model is not very descriptive for microscopic behavior which may have a certain impact on macro scheduling.

7.3 Future Applications of Models

Our proposed model can also be used for other large,crowded places such as railway stations, airports, large venues when an emergency(fire, poison gas,terrorist attack) occurs.

The core of our model is to construct a *Directed multi-distance weighted graph* and a *Directed time weighted graph*.We can convert the spatial information for any large,crowded structures into a *Directed multi-distance weighted graph*.It is worth noting that we need to change the dimension of our *Directed time weighted graph* according to the actual available evacuation ways.For example, if the accident occur on a flat surface, there is no way to pass through the stairs,then we can set the dimension of matrix in *Directed time weighted graph* to 1.Meanwhile, as long as we place many flow density detectors in these crowded places,we can get the personnel flow velocity accordingly. Then we combine the *Directed time weighted graph* with the personnel flow velocity to generate the *Directed time weighted graph*. Then we can get the optimal evacuation path and evacuation time for the given crowded place when an emergency occurs.

References

[1] Yang Jianfang,Gao Yan,Li Lihua.Emergency Evacuation Model and Algorithm for Multi-Exit Buildings[J].Systems Engineering ąłTheory & Practice,2011,31(S1):147-153.

[2] Xu Hui, Tian Wei, Wang Yong. Simulation study on emergency evacuation of dense passenger flow in rail transit transfer station [J/OL]. Journal of System Simulation: 1-9[2019-01-29].http://kns.cnki.net/kcms/detail/11.3092.V.20190111.0919.024.html.

[3] Han Litao,Guo Huan,Zhang Haisi.A Multi-Exit Indoor Emergency Evacuation Path Planning Algorithm[J].Science and Mapping Science,2018,43(12):105-110.

[4] STEVENS M, CHOI J. CAD data conversion to a node-relation structure for 3D sub-unit topological rep-resentation[J]. Journal of the Korean Geographical Society. 2006.41(2):188-194.

[5] John L Bryan.Human Behavior and Fire[M]čňFire Protection HandBook, 18th Edition 1997, Notional Fire Protection Association, Section 8, Chapter 1.

[6] Jin Nanjiang.Study on the evacuation path model of multi-story buildings in fire environment[J].Journal of Armed Police College,2018,34(10):13-17.

[7] Liu Yang,Chen Juan,Ma Jian.The Method of Export Distribution of Evacuated Personnel Based on User Equilibrium Theory[J].China Safety Science Journal,2018,28(10):44-49.

Appendices

List of Figures

List of Tables

Appendix A first appendix

some more text **Input C++ source:**

```
ï¿£
//Dijkstra.cpp
#include <iostream>
#include <vector>
#include <cstddef>
#include <algorithm>
using namespace std;

const int Out1 = 0;   //1-1
const int Out2 = 12;  //13-1
const int Out3 = 13;  //14-1
const int Out4 = 15;  //16-1
const int ERROR = -1;
const int pNUM = 29;      //Number of nodes
const int INF = 0x3f3f3f; //satisfying INF + INF =INF

class Point
{
private:
    int num; //Node number
public:
    vector<int> lowcost; //lowest cost to other nodes
    vector<int> pre;  //Predecessors of each node
    Point();
    void put_num(int n) { (n >= 0 && n < pNUM) ? num = n : num = ERROR; }
    int get_num() { return num; }
    int get_lowcost(int i) { return lowcost[i]; }
    int get_pre(int i) { return pre[i]; }
    friend ostream& operator<<(ostream& out, const Point& p)
    {
        out << p.num;
        return out;
    }
};

Point::Point() :num(0) {
    vector<int> temp(pNUM, 0);
    lowcost = temp;
    pre = temp;
}

class Graph
{
public:
    vector<vector<int>> cost;//the adjacency matrix
    vector<Point> vertex;//the vertex family
    Graph();
    void dijkstra();
    void bestpath(int beg);
};
```

```
Graph::Graph()
{
    //Initialize the adjacency matrix
    vector<vector<int>> matrix(pNUM, vector<int>(pNUM, INF));
    matrix[0][1] = 700; matrix[0][2] = 430; matrix[0][4] = 360; matrix[0][5] = 400;
    matrix[0][28] = 660;
    matrix[1][0] = 430; matrix[1][2] = 300; matrix[1][6] = 340; matrix[1][7] = 410;
    matrix[2][0] = 310; matrix[2][1] = 180; matrix[2][3] = 140; matrix[2][5] = 260;
    matrix[3][2] = 95;  matrix[3][5] = 210; matrix[3][8] = 190; matrix[3][9] = 140;
    matrix[4][0] = 300; matrix[4][5] = 110; matrix[4][11] = 160;
    matrix[5][0] = 350; matrix[5][2] = 130; matrix[5][3] = 110; matrix[5][4] = 120;
     matrix[5][10] = 180;
    matrix[6][1] = 180; matrix[6][13] = 240; matrix[6][16] = 340;
    matrix[7][1] = 200; matrix[7][8] = 90;
    matrix[8][3] = 300; matrix[8][7] = 310; matrix[8][9] = 270; matrix[8][12] = 380;
    matrix[8][19] = 410;
    matrix[9][3] = 100; matrix[9][8] = 80; matrix[9][10] = 150; matrix[9][19] = 200;
    matrix[10][5]=210;matrix[10][9]=200;matrix[10][11]=310;matrix[10][12]=320;
    matrix[10][14]=234;matrix[10][20]=216;matrix[10][21]=270;matrix[10][22]=280;
    matrix[11][4] = 180; matrix[11][10] = 75;matrix[11][14]=400;matrix[11][22]=200;
    matrix[12][8] = 180; matrix[12][10] = 325; matrix[12][10] = 230;
    matrix[13][6] = 200; matrix[13][28] = 160;
    matrix[14][11] = 260; matrix[14][15] = 300; matrix[14][23] = 310;
    matrix[15][14] = 100; matrix[15][28] = 400;
    matrix[16][6] = 250; matrix[16][17] = 330;matrix[16][24]=320;matrix[16][25]=340;
    matrix[17][16] = 130; matrix[17][18] = 260;
    matrix[18][8] = 200; matrix[18][17] = 90;matrix[18][19] = 175;matrix[18][26]=160;
    matrix[19][9] = 130; matrix[19][18] = 160; matrix[19][20] = 320;
    matrix[20][10] = 170; matrix[20][19] = 210;matrix[20][21]=200;matrix[20][17]=140;
    matrix[21][10] = 210; matrix[21][20] = 130; matrix[21][22] = 75;
    matrix[22][10] = 190; matrix[22][11] = 170;matrix[22][21]=85;matrix[22][23]=125;
    matrix[23][14] = 120; matrix[23][22] = 230;
    matrix[24][16] = 130; matrix[24][25] = 240;
    matrix[25][16] = 135; matrix[25][24] = 170; matrix[25][26] = 160;
    matrix[26][18] = 130; matrix[26][25] = 100; matrix[26][27] = 150;
    matrix[27][20] = 90; matrix[27][26] = 150;

    cost = matrix;

    //Initialize the vertex family
    vector<Point> temp(pNUM);
    vertex = temp;
    for (int i = 0; i < pNUM; i++) {
        vertex[i].put_num(i + 1);
    }
}

void Graph::dijkstra()
{
    for (int beg = 0; beg < pNUM; beg++) {
        size_t N = cost.size();
        vector<bool> visited(N, false);
        for (size_t i = 0; i < N; ++i) {//Initialize lowcost and pre
            vertex[beg].lowcost[i] = INF;
            vertex[beg].pre[i] = -1;
```

```
        }

        vertex[beg].lowcost[beg] = 0;//The shortest path from the starting point to
                                     // its own is 0.

        for (size_t i = 0; i < N; ++i) {
            int newNode = -1;//The number of the next point to be selected,
                             //initialized to -1
            int MinWeight = INF;//Minimum weight of the edge adjacent to the current
    /*The first for, traverses all the nodes that have not been visited,
    and selects the inter-point (the shortest distance from the starting point
    except the selected node)*/
            for (size_t j = 0; j < N; ++j) {
                if (!visited[j] && vertex[beg].lowcost[j] < MinWeight) {
                    MinWeight = vertex[beg].lowcost[j];
                    newNode = j;
                }
            }
            if (-1 == newNode)
                break;//Exit when the graph is not connected
            visited[newNode] = true;//Mark intermediate points as accessed
            for (size_t j = 0; j < N; ++j) {
                if (!visited[j] && vertex[beg].lowcost[newNode] + cost[newNode][j]
                        < vertex[beg].lowcost[j]) {
                    vertex[beg].lowcost[j] = vertex[beg].lowcost[newNode]
                                                + cost[newNode][j];
                    vertex[beg].pre[j] = newNode;
                }
            }
        }
    }

}

void Graph::bestpath(int beg)
{
    dijkstra();

    cout << "Point" << '\t' << "Out" << '\t' << "bestTime" << endl;
    cout << beg + 1 << '\t' << vertex[beg].pre[pNUM - 1] + 1 << '\t'
                            << vertex[beg].lowcost[pNUM - 1] << endl;
    cout << "The predecessor of each node:" << endl;
    for (size_t i = 0; i < vertex.size(); ++i) {
        cout<<((vertex[beg].pre[i] == -1)?NULL:vertex[vertex[beg].pre[i]].get_num())
         << ":" << vertex[i] << endl;
    }
}


int main()
{
    Graph g;
    g.bestpath(27); //[1,pNUM-1],except 0,12,13,15
    return 0;
}
```

Appendix B Second appendix

Part I According to the connection between the Louvre's internal exhibition halls, we abstracted these exhibition halls into 28 nodes which is shown as Table 7.

Floor	Node number	Exhibition number
-2	1	The Carrousel du Louvre Entrance
-1	2	100,101,102,104,105
	3	130,131
	4	132,133,134,135,137,338
	5	160,164,169
	6	170,173,174,179,180,181,183,186
0	7	200,201,206,212,218
	8	219,226,229,230,231,236,305
	9	300,308,312,316
	10	317,321,323,328,337,338
	11	339,345,347,348,405,406, 408,410,413,417,418,419,423
	12	400,402,403
	13	The Pyramid Entrance
	14	The Passage Richelieu entrance
	15	424,427,433
	16	The Portes Des Lions Entrance
1	17	500,501,503,505,535, 544,548,552,557,558,564
	18	507,516,517,519,526,531
	19	600,601,605,606,617,632
	20	633,635,636,640
	21	641,645,650,651,656,660,661,663
	22	702,703,705,706,708,709,710
	23	700,701,711,712,715
	24	716,717,718,719,720,726,727,734
2	25	800,801,802,836,843,845,848,855,864
	26	803,811,818,823,825,835,830
	27	902,909,912,913,917,924
	28	900,931,936,940,944,952

Table 7: Division of the Louvre exhibition hall nodes

Part II Use an exhaustive approach to discuss how to open other exports.

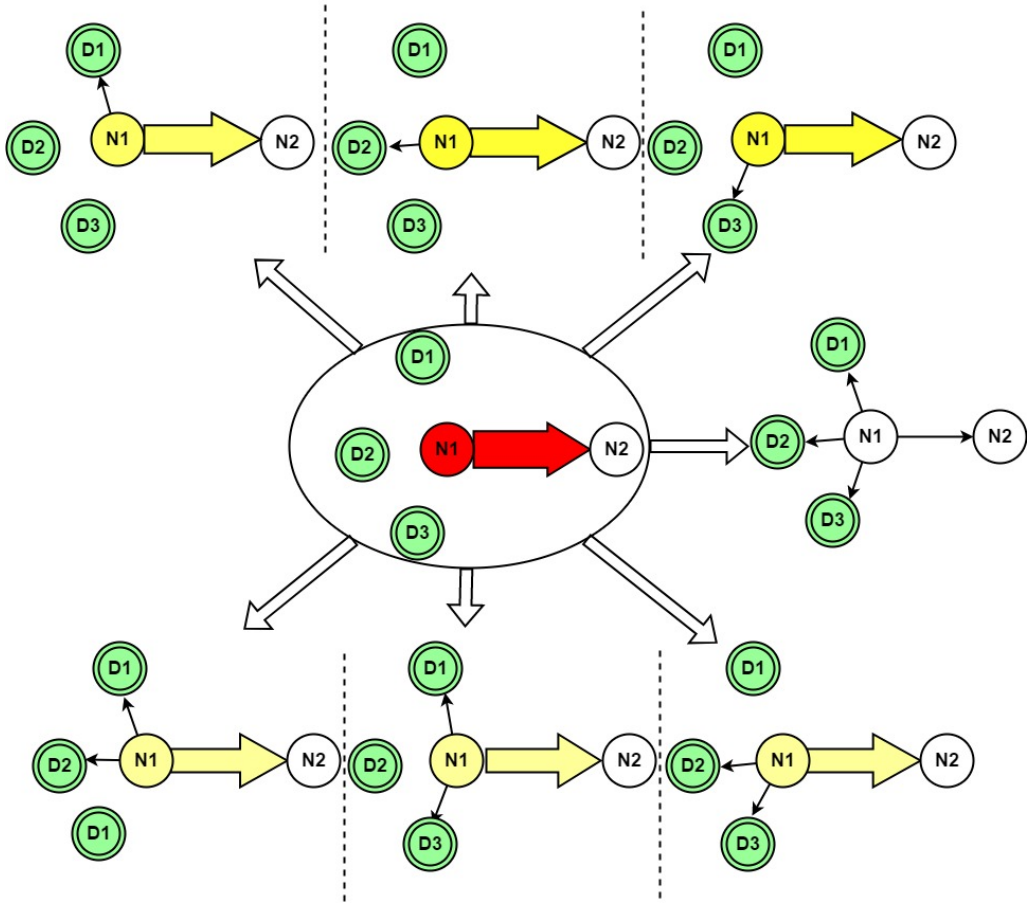

Figure 7: Exhaustive solution for timeout nodes

Part III The following figure shows the adjacency matrix of our estimated distance. The triples (a, b, c): a, b and c respectively represents the uphill distance, the downhill distance, and the horizontal channel distance.

	0	1	2	3	4	5	6	7	8	9	10
0	0	(131,0,32)	(45,0,34)	INF	(90,0,27)	(112,0,14)	INF	INF	INF	INF	INF
1	(0,131,32)	0	(56,0,64)	INF	INF	INF	(0,0,114)	(70,0,20)	INF	INF	INF
2	(0,45,34)	(0,56,64	0	(18,0,90)	INF	(0,69,14)	INF	INF	INF	INF	INF
3	INF	INF	(0,19,90)	0	INF	(0,0,78.6)	INF	INF	(33,0,10)	(31,0,15)	INF
4	(0,90,27)	INF	INF	INF	0	(9,11,17)	INF	INF	INF	INF	INF
5	(0,113,14)	INF	(69,0,14)	(0,0,78.6)	(11,9,17)	0	INF	INF	INF	INF	(0,28.6)
6	INF	(0,0,114)	INF	INF	INF	INF	0	INF	INF	INF	INF
7	INF	(0,70,20)	INF	INF	INF	INF	INF	0	(3,3,14)	INF	INF
8	INF	INF	INF	(0,33,10)	INF	INF	INF	(3,3,14)	0	(28,0,21)	INF
9	INF	INF	INF	(0,31,15)	INF	INF	INF	INF	(0,28,21)	0	(142,0,69)
10	INF	INF	INF	INF	INF	(29,0,95)	INF	INF	INF	(0,142,70)	0
11	INF	INF	INF	INF	(17,32,61)	INF	INF	INF	INF	INF	(0,9,43)
12	INF	INF	INF	INF	INF	INF	INF	INF	(28,0,39)	INF	(0,132,81)
13	INF	INF	INF	INF	INF	INF	(32,39,60)	INF	INF	INF	INF
14	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
15	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
16	INF	INF	INF	INF	INF	INF	(50,13,57)	INF	INF	INF	INF
17	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
18	INF	INF	INF	INF	INF	INF	INF	INF	(0,60,46)	INF	INF
19	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,44,17)	INF
20	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,61,30)
21	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,77,38)
22	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,79,60)
23	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
24	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
25	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
26	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
27	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF

Figure 8: The adjacency matrix of distance 1

	0	1	2	3	4	5	6	7	8	9	10
0	0	(131,0,32)	(45,0,34)	INF	(90,0,27)	(112,0,14)	INF	INF	INF	INF	INF
1	(0,131,32)	0	(56,0,64)	INF	INF	INF	(0,0,114)	(70,0,20)	INF	INF	INF
2	(0,45,34)	(0,56,64	0	(18,0,90)	INF	(0,69,14)	INF	INF	INF	INF	INF
3	INF	INF	(0,19,90)	0	INF	(0,0,78.6)	INF	INF	(33,0,10)	(31,0,15)	INF
4	(0,90,27)	INF	INF	INF	0	(9,11,17)	INF	INF	INF	INF	INF
5	(0,113,14)	INF	(69,0,14)	(0,0,78.6)	(11,9,17)	0	INF	INF	INF	INF	(0,28.6)
6	INF	(0,0,114)	INF	INF	INF	INF	0	INF	INF	INF	INF
7	INF	(0,70,20)	INF	INF	INF	INF	INF	0	(3,3,14)	INF	INF
8	INF	INF	INF	(0,33,10)	INF	INF	INF	(3,3,14)	0	(28,0,21)	INF
9	INF	INF	INF	(0,31,15)	INF	INF	INF	INF	(0,28,21)	0	(142,0,69)
10	INF	INF	INF	INF	INF	(29,0,95)	INF	INF	INF	(0,142,70)	0
11	INF	INF	INF	INF	(17,32,61)	INF	INF	INF	INF	INF	(0,9,43)
12	INF	INF	INF	INF	INF	INF	INF	INF	(28,0,39)	INF	(0,132,81)
13	INF	INF	INF	INF	INF	INF	(32,39,60)	INF	INF	INF	INF
14	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
15	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
16	INF	INF	INF	INF	INF	INF	(50,13,57)	INF	INF	INF	INF
17	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
18	INF	INF	INF	INF	INF	INF	INF	INF	(0,60,46)	INF	INF
19	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,44,17)	INF
20	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,61,30)
21	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,77,38)
22	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	(0,79,60)
23	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
24	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
25	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
26	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
27	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF

Figure 9: The adjacency matrix of distance 2

21	22	23	24	25	26	27
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
(76, 0, 38)	(79, 0, 60)	INF	INF	INF	INF	INF
INF	(81, 0, 29)	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	(22, 6, 51)	INF	INF	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	(12, 10, 65)	(8, 7, 77)	INF	INF
INF	INF	INF	INF	INF	INF	INF
INF	INF	INF	INF	INF	(0, 0, 93)	INF
INF	INF	INF	INF	INF	INF	INF
(15, 12, 51)	INF	INF	INF	INF	INF	(10, 8, 34)
0	(4, 3, 37)	INF	INF	INF	INF	INF
(3, 4, 37)	0	(87, 0, 42)	INF	INF	INF	INF
INF	(0, 88, 43)	0	INF	INF	INF	INF
INF	INF	INF	0	(48, 0, 24)	INF	INF
INF	INF	INF	(0, 48, 24)	0	(4, 3, 34)	INF
INF	INF	INF	INF	(3, 3, 34)	0	(14, 11, 75)
INF	INF	INF	INF	INF	(11, 14, 75)	0

Figure 10: The adjacency matrix of distance 3

Part IV The adjacency matrix in the following figure is generated by the directed time-weighted graph we imported in Model II.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
0	0	700	430	INF	360	400	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	660
1	430	0	300	INF	INF	INF	340	410	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
2	310	180	0	140	INF	260	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
3	INF	INF	95	0	INF	210	INF	INF	190	140	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
4	300	INF	INF	INF	0	110	INF	INF	INF	INF	INF	160	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
5	350	INF	130	110	120	0	INF	INF	INF	INF	180	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
6	INF	180	INF	INF	INF	INF	0	INF	INF	INF	INF	INF	INF	240	INF	INF	340	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
7	INF	200	INF	INF	INF	INF	INF	0	90	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
8	INF	INF	INF	300	INF	INF	INF	310	0	270	INF	INF	380	INF	INF	INF	INF	INF	410	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
9	INF	INF	INF	100	INF	INF	INF	INF	80	0	150	INF	INF	INF	INF	INF	INF	INF	INF	200	INF	INF	INF	INF	INF	INF	INF	INF	INF
10	INF	INF	INF	INF	INF	210	INF	INF	INF	200	0	310	320	INF	234	INF	INF	INF	INF	INF	216	270	280	INF	INF	INF	INF	INF	INF
11	INF	INF	INF	INF	180	INF	INF	INF	INF	INF	75	0	INF	INF	400	INF	INF	INF	INF	INF	INF	INF	200	INF	INF	INF	INF	INF	INF
12	INF	INF	INF	INF	INF	INF	INF	INF	180	INF	325	INF	0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	230
13	INF	INF	INF	INF	INF	INF	200	INF	INF	INF	INF	INF	INF	0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	160
14	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	260	INF	INF	0	300	INF	INF	INF	INF	INF	INF	INF	310	INF	INF	INF	INF	INF
15	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	100	0	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	400
16	INF	INF	INF	INF	INF	INF	250	INF	INF	INF	INF	INF	INF	INF	INF	INF	0	330	INF	INF	INF	INF	INF	INF	320	340	INF	INF	INF
17	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	130	0	260	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF
18	INF	INF	INF	INF	INF	INF	INF	INF	200	INF	INF	INF	INF	INF	INF	INF	INF	90	0	175	INF	INF	INF	INF	INF	INF	160	INF	INF
19	INF	INF	INF	INF	INF	INF	INF	INF	INF	130	INF	INF	INF	INF	INF	INF	INF	INF	160	0	320	INF	INF	INF	INF	INF	INF	INF	INF
20	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	170	INF	INF	INF	INF	INF	INF	INF	INF	210	0	200	INF	INF	INF	INF	INF	140	INF
21	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	210	INF	INF	INF	INF	INF	INF	INF	INF	INF	130	0	75	INF	INF	INF	INF	INF	INF
22	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	190	170	INF	INF	INF	INF	INF	INF	INF	INF	INF	85	0	125	INF	INF	INF	INF	INF
23	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	120	INF	INF	INF	INF	INF	INF	INF	230	0	INF	INF	INF	INF	INF
24	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	130	INF	INF	INF	INF	INF	INF	INF	0	240	INF	INF	INF
25	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	135	INF	INF	INF	INF	INF	INF	INF	170	0	160	INF	INF
26	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	130	INF	INF	INF	INF	INF	INF	100	0	150	INF
27	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	90	INF	INF	INF	INF	INF	150	0	INF
28	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	INF	0

Figure 11: The adjacency matrix of optimal evacuation time