午餐系統及點餐資料分析

[文件副標題]

[壹、研究動機 3](#_Toc530432357)

[貳、研究目的 3](#_Toc530432358)

[參、實驗器材及設備 4](#_Toc530432359)

[肆、研究過程及方式 5](#_Toc530432360)

[一、資料庫 5](#_Toc530432361)

[(一)、資料結構 5](#_Toc530432362)

[(二)、資料表 6](#_Toc530432363)

[1. Order資料表 6](#_Toc530432364)

[2. Dish資料表 6](#_Toc530432365)

[3. User資料表 6](#_Toc530432366)

[4. 待擴充的資料表 6](#_Toc530432367)

[(三)、預存程序 6](#_Toc530432368)

[(四)、查詢點單 6](#_Toc530432369)

[1. 主查詢 6](#_Toc530432370)

[2. 擴充查詢 6](#_Toc530432371)

[3. 篩選條件 6](#_Toc530432372)

[(五)、資料庫死結 7](#_Toc530432373)

[二、後端 7](#_Toc530432374)

[(一)、系統架構 7](#_Toc530432375)

[(二)、處理流程 9](#_Toc530432376)

[(三)、session阻塞 9](#_Toc530432377)

[(四)、後端安全性 9](#_Toc530432378)

[(五)、最大負荷量 10](#_Toc530432379)

[(六)、架設環境 10](#_Toc530432380)

[1. 使用xampp的問題 10](#_Toc530432381)

[2. 使用wamp 10](#_Toc530432382)

[三、前端 10](#_Toc530432383)

[(一)、iOS前端 10](#_Toc530432384)

[1.呼叫後端 11](#_Toc530432385)

[2.Layout圖 11](#_Toc530432386)

[3.錯誤分析 11](#_Toc530432387)

[4.推播功能 11](#_Toc530432388)

[(二)、網頁前端 11](#_Toc530432389)

[1. ajax 11](#_Toc530432390)

[2. 分類樹 11](#_Toc530432391)

[3. 半耦合式後端 11](#_Toc530432392)

[(三)、輔助外掛 11](#_Toc530432393)

[(四)、前後端交換介面 12](#_Toc530432394)

[1. 點單資料結構 12](#_Toc530432395)

[2. 菜單資料結構 12](#_Toc530432396)

[3. 編碼 12](#_Toc530432397)

[四、預測模型 12](#_Toc530432398)

[(一)、擷取資料 12](#_Toc530432399)

[1. 點餐序列 12](#_Toc530432400)

[2. 點餐圖 13](#_Toc530432401)

[(二)、點餐矩陣 13](#_Toc530432402)

[1. 填入矩陣 13](#_Toc530432403)

[2. 矩陣特徵 13](#_Toc530432404)

[(三)、預測演算法 14](#_Toc530432405)

[壹、Dijkstra漫遊演算法 14](#_Toc530432406)

[一、目的 14](#_Toc530432407)

[二、實作 14](#_Toc530432408)

[貳、矩陣快速冪 15](#_Toc530432409)

[一、目的 15](#_Toc530432410)

[二、數學性質 15](#_Toc530432411)

[三、實作 15](#_Toc530432412)

[參、反矩陣解穩定狀態 16](#_Toc530432413)

[一、目的 16](#_Toc530432414)

[二、數學性質 16](#_Toc530432415)

[三、虛擬碼 16](#_Toc530432416)

[四、實作 16](#_Toc530432417)

[肆、線性回歸 17](#_Toc530432418)

[一、公式 17](#_Toc530432419)

[二、實作結果 17](#_Toc530432420)

[(四)、各演算法比較 18](#_Toc530432421)

[伍、目前研究結果 19](#_Toc530432422)

[陸、參考資料及其他 20](#_Toc530432423)

[篩選條件 20](#_Toc530432424)

# 壹、研究動機

在學校買午餐時，常常發生問題，例如點餐單根本就不見了、點了餐結果忘了繳錢、或是忘了把點餐單給廠商，諸如此類的問題，造成了大家每天都吃不到便當。

「每天遇到這種事，我想大家都不會開心吧」，我與朋友說道，「要是能夠有套系統，協助我們點餐，該有多好」，於是我就開始製作午餐系統了，午餐系統是一套專門拿來點餐用的系統，系統幫助學生、合作社、廠商整理了繁雜的點餐資料。

我在資料庫看到了一個現象，韓式拌飯的點餐人數總是特別多，大家總是特別愛吃韓式拌飯，我腦中突然閃過一個想法，「要是我能夠利用先前的點餐資料，去預測明天的點餐數量，這樣就能減少廚餘了」，於是我就開始進行資料分析，建立模型來預測明天的點餐數量。

# 貳、研究目的

一、預測未來之餐點需求量。

二、製作一套能夠讓全校同時使用的系統。

三、調查廠商、學生，尋找系統能夠改進的地方。

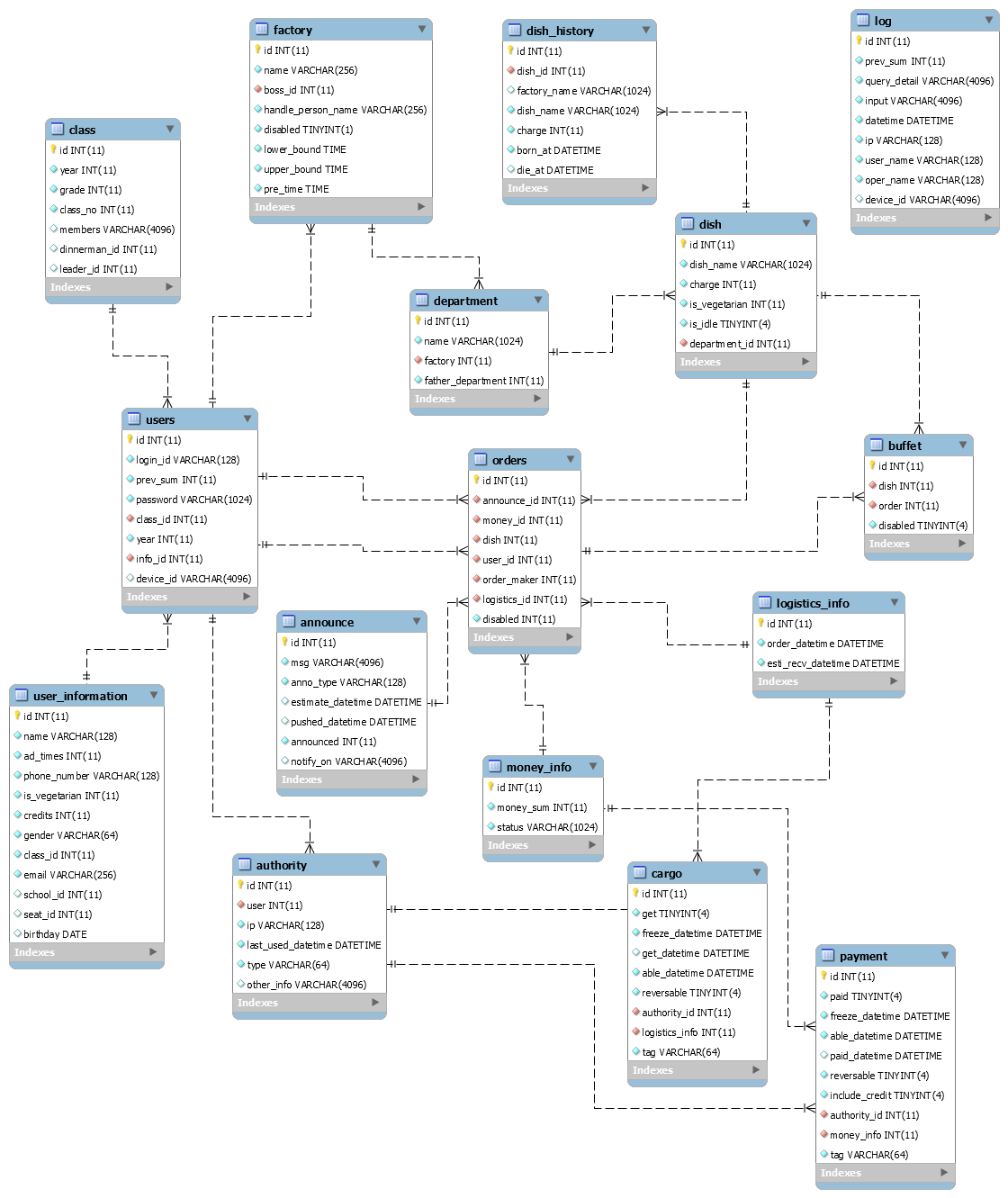
# 參、實驗器材及設備

|  |  |  |
| --- | --- | --- |
| 電腦類 | 行政類 | 其他類 |
|  |  |  |
| Firebase | 校務會議支持 |  |
| Dynamic DNS |  |  |
| 伺服器電腦 |  |  |
|  |  |  |
|  |  |  |

# 肆、研究過程及方式

## 一、資料庫

### (一)、資料結構



### (二)、資料表

#### 1. Order資料表

這個資料表為整套點餐系統的核心，這個資料表幾乎不儲存實際資料，除了Disabled欄位之外，全部都是ForeignKey，MariaDB根據OrderID排序的時候能夠直接依據本資料表的PrimaryKey排序，不必執行O(n log n)的排序演算法。

#### 2. Dish資料表

這個資料表負責處理關於餐點的資料，該餐點是否不再供應，該餐點的價格、菜名、供應廠商…等，皆由此資料表紀錄。

#### 3. User資料表

這個資料表負責處理關於使用者的資料，包括登入的帳號密碼、裝置ID、所屬班級、權限大小。

#### 4. 待擴充的資料表

像是cargo、buffet

### (三)、預存程序

以make order為例

### (四)、查詢點單

#### 1. 主查詢

丟出sql表格，順便炫耀效能

#### 2. 擴充查詢

使用in加速

#### 3. 篩選條件

|  |  |
| --- | --- |
| 時間下界 | AND (? < LO.esti\_recv\_datetime) |
| 時間上界 | AND (? > LO.esti\_recv\_datetime) |
| 針對特定使用者搜尋 | AND (? = O.user\_id) |
| 針對某間廠商進行搜尋 | AND (? = DP.factory) |
| 針對特定列查詢 | AND (? = O.id) |
| 針對班級查詢 | AND ((SELECT U.class\_id FROM users AS U WHERE U.id = ?) = U.class\_id) |

LO.esti\_recv\_datetime，此欄位表示該餐點應大約於何時送達，此欄位有加上索引，在進行時間上下界查詢時，可以利用二分搜進行搜尋。

針對特定使用者進行篩選，如果則不顯示該筆資料。

O.user\_id，此欄位記錄餐點應該要送給誰，且O.user\_id連結到User資料表中的主鍵(Primary Key)，可以執行最高速的const篩選。

針對某一間廠商進行篩選，如果提供該餐點的廠商不符合條件，則不顯示該筆資料。

DP.factory，此欄位記錄餐點的製作廠商，且DP.factory連結到Factory資料表中的主鍵(Primary Key)，可以執行最高速的const篩選。

含有不會變動輸出結果的子查詢，Mysql會自動將其最佳化，故該程式碼在執行時實際上如同下列程式碼。

U.class\_id，此欄位記錄餐點應該要送給哪一班，且U.class\_id連結到class資料表中的主鍵(Primary Key)，可以執行最高速的const篩選。

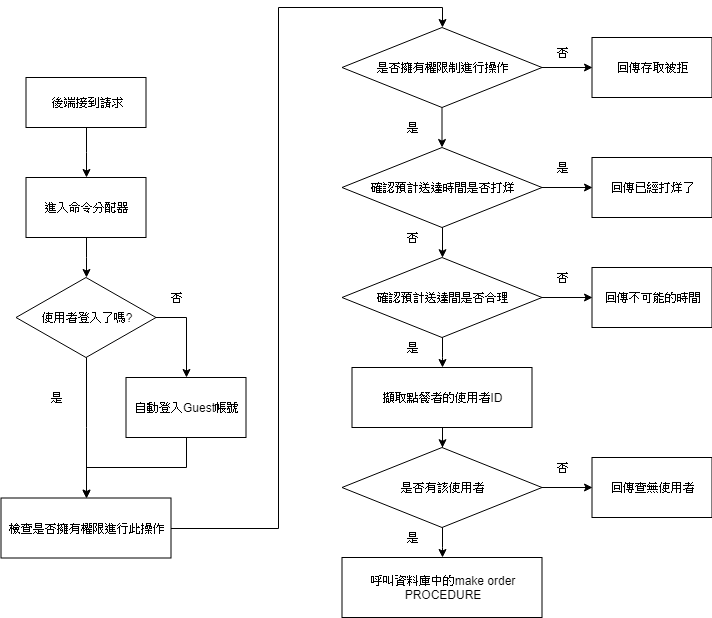
### (五)、資料庫死結

## 二、後端

### (一)、系統架構

後端連結了資料庫、前端，後端是整套系統的核心，以下為整套系統的架構圖。其中後端扮演著核心角色，只要後端無法運作，不論前端或是資料庫都無法繼續運作。

### (二)、處理流程

****

由流程圖可見，後台經過了嚴密的審查之後，才將命令交給資料庫去處理，將資料寫入資料庫時，因為要連續執行多次SQL語句，所以將多個語句包裝成一個PROCEDURE，減少多次與資料庫伺服器連線的負擔。

### (三)、session阻塞

必須要處理掉，才能使用ajax

flock

### (四)、後端安全性

密碼安全性

SQL注入

XSS

### (五)、最大負荷量

將所需資料先載入記憶體，就不需要一天到晚跟資料庫要資料了。

### (六)、架設環境

#### 1. 使用xampp的問題

#### 2. 使用wamp

## 三、前端

### (一)、iOS前端

iOS前端擁有一個特別的功能，就是推撥功能，推播功能會提醒使用者今天已經有訂餐了，不要再下去買午餐了。

以下為iOS前端的點餐畫面。



我們採用物件導向架構

#### 1.呼叫後端

#### 2.Layout圖

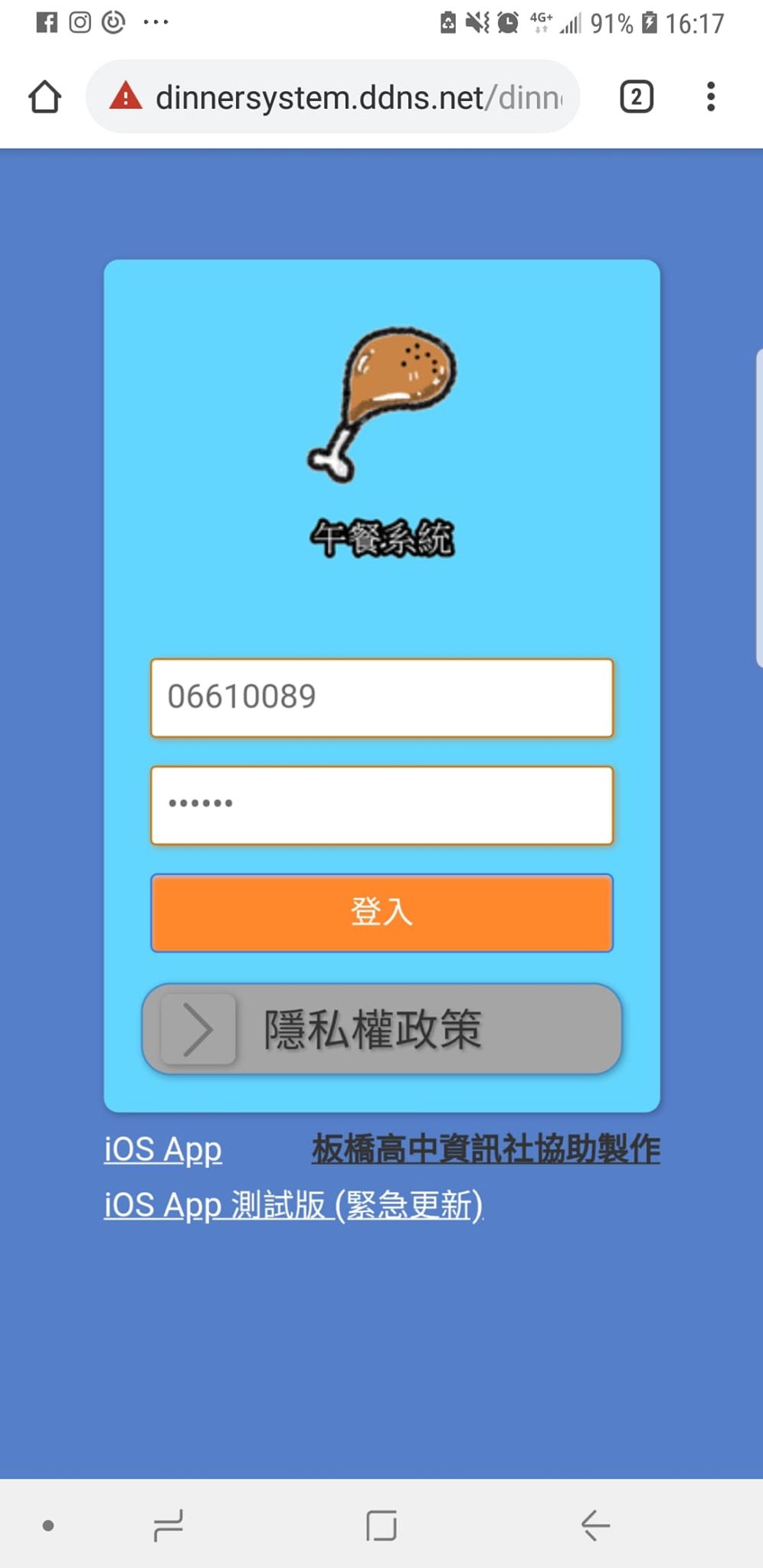
#### 3.錯誤分析

#### 4.推播功能

### (二)、網頁前端

網頁版前端會先讓伺服器使用分類樹進行分類，再將分類後的資料傳給前端，前端就不需要進行大量的運算了。

以下為網頁版的登入畫面。



#### 1. ajax

#### 2. 分類樹

#### 3. 半耦合式後端

### (三)、輔助外掛

輸出月份資料，輸入新年級學生資料

### (四)、前後端交換介面

#### 1. 點單資料結構

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | |  |  | | --- | --- | | **id** | 22410 | | **user** | |  |  | | --- | --- | | **id** | 11184 | | **name** | 白翔云 | | **seat\_no** | 20904 | | | **dish** | |  |  | | --- | --- | |  | | | **dish\_name** | 卡拉雞 雞塊 三副菜 | | **dish\_id** | 1 | | **dish\_cost** | 55 | | | **recv\_date** | 2018-11-11 12:00:00 | |

#### 2. 菜單資料結構

#### 3. 編碼

## 四、預測模型

我們想出了三個方法來預測明天的點餐趨勢，分別為以下四種方法。

(一)、Dijkstra漫遊演算法

(二)、矩陣快速冪

(三)、反矩陣解穩定狀態

(四)、線性回歸

針對以上幾種演算法，我們將探討這些演算法之間的優劣以及差異。

### (一)、擷取資料

#### 1. 點餐序列

整理成以使用者為索引的資料

#### 點餐圖

將點餐序列畫成一張圖

### (二)、點餐矩陣

#### 1. 填入矩陣

將點餐圖填入矩陣

#### 矩陣特徵

### (三)、預測演算法

#### 壹、Dijkstra漫遊演算法

##### 一、目的

使用*Dijkstra*在點餐圖上隨機走動，經過任何節點就將該節點之值加一，最後點餐圖上節點之值的比例將會接近實際點餐量的比例；移動的步數越多，越接近實際比例。

以下表格解釋執行*Dijkstra*演算法時，電腦進行的模擬操作與現實世界的代表意義。

|  |  |
| --- | --- |
| 電腦進行的操作 | 現實世界的意義 |
|  |  |
| 從點餐圖上的任意一個點出發 | 第一天先隨便點一道菜 |
| 經由任意一條邊 | 經過了一天 |
| 到達另外圖上的另外一個點 | 第二天點了另外一到餐 |
| 經過一個自環 | 經過了一天，決定要吃同一道菜 |
| 到了同一個點 | 第三天點了同一道菜 |

##### 二、實作

#### 貳、矩陣快速冪

##### 一、目的

使用漫遊演算法，進行N步移動，所需要的時間複雜度為；而移動一步可以視為矩陣相乘，我們可以使用矩陣快速冪將時間複雜度下降至*)*。

##### 二、數學性質

將每天的點餐比例視為一個矩陣，該矩陣稱呼為*D*；將點餐矩陣稱呼為T。

如果要求明天的資料，可以得到該方程式 ；如果要求N天後的資料，可以得到方程式。

根據矩陣的結合律，先求出再與D相乘，將獲得相同的結果。

再運用下列兩式，先使用上式建立數列*K*到第項，再運用下式遞迴出，即可在內求解。

##### 三、實作

#### 參、反矩陣解穩定狀態

##### 一、目的

上述兩種方法，由於*N*趨近於無限大，無法在有限時間內計算出穩定矩陣；如果將點餐矩陣視為一個線性函數，則可以得到一組聯立方程式，解聯立方程式即得穩定狀態。

##### 二、數學性質

穩定狀態為一機率矩陣，名為*S*，形狀如，且，如果定義一個形狀如同S的機率矩陣K，K值隨機，則。

對於任意點餐圖製作成的點餐矩陣，必定有下列性質。

已知，則我們可以得到聯立方程組，畫簡後得聯立方程組，其中矩陣，則我們可以得到係數矩陣Z，以及值矩陣。

穩定狀態。

##### 三、虛擬碼

##### 四、實作

#### 肆、線性回歸

##### 一、公式

##### 二、實作結果

### (四)、各演算法比較

# 伍、目前研究結果

# 陸、參考資料及其他

前後端資料交換介面

前後端命令界面

前後台原始程式碼 github

壓力測試數據

漫遊演算法虛擬碼

矩陣快速冪虛擬碼

資料表詳細描述

點餐序列

點餐圖

點餐矩陣

馬可夫轉移矩陣

方程式矩陣、反矩陣

平衡狀態矩陣

線性回歸方程式

伺服器環境參數

矩陣相乘演算法

並查集

有向圖、入度、出度、完全圖

反矩陣得解之證明

線性回歸公式之證明

系統權限等級

二分搜

合併排序、快速排序