Shanghai High School International Division Campus Navigation

June 2023

**Table of Contents**

[1. Introduction 1](#_Toc137246198)

[2. Program Design 2](#_Toc137246199)

[3. Implementation 5](#_Toc137246201)

[4. Difficulties & Solution 9](#_Toc137246202)

[5. Strengths and Weaknesses 10](#_Toc137246203)

[6. Future Improvements 11](#_Toc137246204)

[7. Conclusion 12](#_Toc137246205)

[8. Appendix 13](#_Toc137246206)

# **Introduction**

Navigating through large campuses can be an overwhelming task, especially for parents that are entering the campus to attend parent-teacher conferences or talks. The Shanghai High School International Division (SHSID) campus covers an area larger than 200 thousand square meters, and this enormous size can cause frustrations for parents that are not familiar with the campus. Many parents struggle to find the route to the building they want to go to or cannot find the exit gate afterwards. Baidu Maps is a widely used navigation system in Shanghai, but it fails to provide specific routes inside the campus, due to its insufficient data about the specific buildings in SHSID.

To address this issue, this project aims to develop an SHSID campus navigator specifically for SHSID parents, which can assist them in finding their way through the campus. The user will be able to choose a starting point (a school gate or a school building) and an end point (another school gate or school building), and the program will show the most effective route to move from the starting point to the ending point. A 2D map will be shown, and the path will be traced on the map. Along with the traced map, there will be route details shown, which are the specific path directions, e.g., “turn right after 30 meters”. This program is created in the form of a WeChat Mini Program, thus accessible for users both on iPhone and Android without the need of installing an additional app.

The program functions by recording each building’s neighbor building and the distance to each neighboring building. Since there is no traffic inside the campus, the time needed is directly proportional to the total distance. By using the Bellman-Ford algorithm, it is possible to find the shortest path from the starting point to the ending point. which is the most efficient route.

# **Program Design**

* 1. **Prototype** The program is in the form of a WeChat Mini Program, which is accessible for iOS and Android users. The user interface consists of a picker selection for starting location and ending location. The choices are categorized for parents to find the locations more conveniently, into Gates, Teaching Buildings, Parking Lots, Cafeterias, Sport Grounds, and Others. It is achieved by using the Wei Xin Markup Language (WXML), Wei Xin Style Sheets (WXSS) to design the user interface, like how Hyper Text Markup Language (HTML) and Cascading Style Sheets (CSS) can be used to design websites. WXML is for coding the elements of the Mini Program, such as the picker, while WXSS is for adding styles to the Mini Program. Figure 1 shows the interface of the Mini Program. View Appendix 1 for home A screenshot of a phone

     Description automatically generated with medium confidencepage WXML code, Appendix 2 for route page WXML code.

A screen shot of a map

Description automatically generated with medium confidenceA screen shot of a map

Description automatically generated with low confidence

Figure 1: The Interface of the Mini Program

* 1. **Algorithm**

Bellman Ford algorithm is an algorithm to calculate the shortest path from a source node to all other nodes in a weighted and directed graph. It takes in a graph, represented through an edge adjacency list, and can help to find the shortest path from one node to all other nodes in the graph. This program uses the Bellman Ford algorithm to calculate the shortest path from one node to another.   
The algorithm works by inputting an adjacency list of all distances between any two nodes, represented in the form of:

[starting node, ending node, weight]

The program first sets the distance from source node to all other nodes as infinity, and predecessor of all vertices to null. It then iterates through all the edges in the graph, which is [u, v, w]. u is the starting node, v is the ending node, and w is the weight, which is the distance from starting node to ending node. It will check if the distance from source node to node d can be minimized by going through node u, and minimize the distance if possible, and set predecessor of vertex v to u. Suppose the total number of edges are V. The distance from the original node to all other nodes can only be changed V-1 times because there are only V-1 other nodes in the graph. Therefore, iterate through the edges list V-1 times.

\*View Appendix 3 for Bellman Ford Algorithm used in the Campus Navigation\*

* 1. **Data Structures**

The program utilizes varieties of data structures in JavaScript. As mentioned in the above section, the Bellman Ford algorithm requires an input of an edge adjacency list. The adjacency list is stored as an array of arrays, where multiple arrays that store the starting node, ending node, and weight are stored in another array. The benefits of using arrays to store the adjacency list are that it reduces the memory required when storing the arrays, and it is easy to reference the adjacency list and its respective child elements.

Another data structure used is dictionary. This program uses two dictionaries to store data. One dictionary stores each building’s nodes, and the other stores coordinates for each node. Using a dictionary to store data can store two pieces of information into one group, a key-value pair. The key for the dictionary that stores building’s nodes is the building’s name, and the value is its specific node index.

* 1. **Naming Rules**

In this program, the variables are named using the camel case naming convention.



# **Implementation**

* 1. **Planning**
     1. **Framework**

A screenshot of a map

Description automatically generated with low confidenceBefore starting to code the framework of the Mini Program, planning was necessary to set a goal. Using the paint software, I drew the basic outline of the Mini Program framework, as shown in Figure 2.

Figure 2. Planning of Mini Program framework

The drawing on the left shows the initial page when the user enters the Mini Program, prompting the user to enter a starting location, and an ending location, given a map of SHSID campus. The drawing on the right shows the page after user has entered the locations, which shows the traced map and the directions.

* + 1. **Nodes**

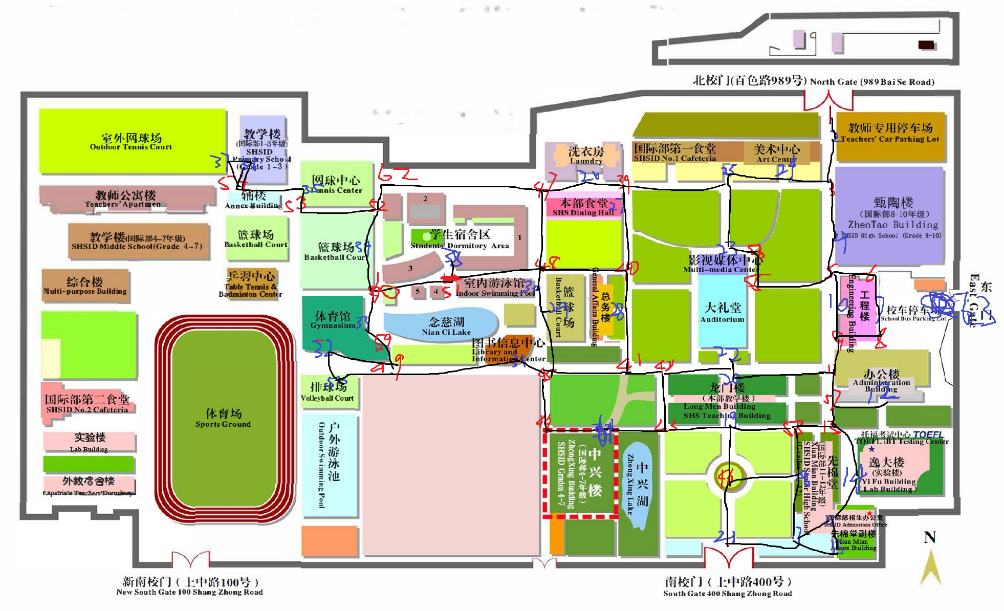
To create the edge adjacency list required for the Bellman Ford algorithm, a visualization of the graph of nodes would be ideal. I used the paint software to label and connect the nodes, which are all buildings and junctions, as shown in Figure 3.

Figure 3. SHSID Map labelled with nodes

In the figure above, nodes are marked starting from 1 to 64, the total number of nodes. Nodes labelled in blue are buildings, while nodes labelled in red are junctions. This is only for visual purposes. By sketching out all the nodes, the building’s respective node index, and adding this image into GeoGebra, an online geometry software that can display relative distances between any two points, the edge adjacency list can be formed. Then, I added this image into paint again, and was able to find each node’s relative pixels. With the relative pixels of each node, the program was able to draw the shortest path by drawing from one pixel location to another during the Bellman Ford algorithm.

* 1. **Coding**

The coding section can be separated into two sections: framework and algorithm & logic.

* + 1. **Framework**

Using WXML, the program uses <picker>, a tool that allows user to select one from many choices, to choose starting location and ending location. The text is represented by <text>, and an icon is placed beside by <icon>. The SHSID Campus map is placed by <image>, along with a <button> that switches page when clicked. On top of the campus map is a <canvas> that allows drawing, specifically to trace the shortest route for the user. Figure 1 shows the display page with these elements.

Using WXSS, the program adds style to image to let it stretch across the entire screen meanwhile keeping aspect ratio for clarity, and adds a round border for the picker, along with some padding for a better user visual experience. By mentioning the width, height, and positions of the image and canvas, the program places the canvas on top of the image directly, allowing a direct markup on the image.

* + 1. **Algorithm & Logic**

The algorithm entirely uses JavaScript to code.

* + - 1. **Map Zooming**

The map can be zoomed in and out and dragged side to side. When zooming, the program will catch the distance between user’s two fingers and get the difference in distance when the fingers move apart or closer. By multiplying the distance difference by a scale multiplier, the image will be enlarged or reduced in size based on the final scale.

* + - 1. **Button Clicks**

The button on the home page is used to submit user’s initial and final location to be processed by the algorithm. The button is linked with a function that toggles the next page to appear and sends the initial and final location to the next page to be processed.

* + - 1. **Bellman Ford algorithm**

As mentioned above, the bellman ford algorithm is a method to find the shortest path from a source vertex to all other vertices and is used in this program to find the shortest route from one location to another location. The directions are then stored in an array in a format like: “Walk for \_\_\_ meters, turn \_\_\_\_”, so it can be directly shown in the Mini program page by referring to the array.

* 1. **Debugging**
     1. **Purpose**

Debugging is a crucial step for app development because it allows developers to identify and resolve bugs, meanwhile testing the program for further areas of improvement. This improves the overall functionality and user experience for the program.

* + 1. **Techniques**

In the debugging phase of app development, the developer thoroughly tests every area of the code, trying to find loopholes or malfunctions. Once the developer identifies a malfunction, the developer can log every process of the malfunctioning code and try to identify which section has errors. By analyzing the logged data, the developer can easily track success and failures to find errors. Another method is unit testing, which is used for early stages of debugging. The developer will run areas of the code individually and compare the expected behavior with actual behavior to identify any unexpected discrepancies.

* + 1. **Examples of errors**

In the SHSID Campus Navigation program, many errors occurred as a surprise, mainly due to careless mistakes. For example, the expected result of moving two fingers apart on an image would be a zoom in on the image, but the program zooms out. This was caused due to an accidental negative sign added to the scaling parameter. However, there were also errors not due to accidents. One example is the misplacement of traced route on different phone dimensions. The program was tested on an iPhone 7 simulator, and the trace drawings were done by exact calculations. When the program was tested on other phone dimensions, such as an iPhone X simulator, the traced route deviates from the actual route. To fix this problem, the calculations had to be referenced to the phone dimensions, then performing ratio calculations. This ultimately allowed the route tracing to be correct in all dimensions.

# **Difficulties & Solution**

The first difficulty that was faced was tracing the route on the map. There were so many routes in total, so pre-drawing the routes was not an option. It was also impossible to markup images because image manipulation APIs were not available. However, the canvas element allowed drawing, so another option was setting the canvas background as the map and drawing on the canvas to trace the route. However, when the canvas background is an image, the image map would be very blurry, so this is not an option as well. Finally, I thought of putting the canvas element on top of the image element, which I had to style the positions using WXSS so the canvas would always be on top of the image despite image zooming and moving.

# **Strengths and Weaknesses**

This program possesses various strengths and weaknesses that distinguish it from other similar products. The SHSID Campus image map in the program can be zoomed in or out and can be moved around. This allows the user to see the map more clearly, considering users with eyesight problems that may not be able to see the small image. Moreover, the selection for the starting and final locations are categorized into several groups: gates, teaching buildings, parking lots, cafeterias, sport grounds, and other places. Categorizing the campus buildings let users that are not familiar with the SHSID Campus to be able to easily locate their desired building based on the general categories.

However, the program also has many weaknesses. This navigation does not support live location, so the program will only produce a one-time route for the user from its initial location to the final location. This means that if the user moves away from the shortest route, and goes to another route, the program will not be able to guide the user until the user reselects a new initial location.

In addition, the program only provides one possible route based on the shortest location. It doesn’t provide an extra one or two routes in case the shortest route is blocked. This also means the user will not be able to find their way if the shortest route is blocked.

Third, the program calculates the distances through a relative length ratio from the school map provided and converts it to real-life distance by multiplying a conversion factor that is achieved through comparing with Baidu Maps. Because the distances are relative distances ratio from the school map provided, and not actual distances measured by GPS or a ruler, there will be inaccuracies that cause the distance to be wrong.

Lastly, the program only supports English. This causes users that are not familiar with English to suffer, and it will decrease user experience for them.

# **Future Improvements**

Due to the numerous weaknesses of the program that hindered its functionality, the program can have a few improvements for the future.

First, the program needs to support more languages, such as Chinese, Japanese, and Korean. This can allow more users to be able to use this program and increase user experience.

Second, the program needs to provide more routes, and let the user choose a route based on their preferences, such as time, turns required, or type of road. This allows the user to choose their own preferences based on their needs, and boosts user experience.

Third, the program will incorporate the user’s live location into the map and will be able to detect if user has gone off the suggested route, and automatically suggest a new route from the wrong location. This can also allow the program to automatically set starting location for the user, which also boosts the user experience.

Lastly, the program will add floor plans for each building. After the user successfully navigated through the campus to the desired building, the user might not be able to navigate to the specific room. By adding the floor plan feature, the user will be able to navigate to the specific room.

# **Conclusion**

In this project, I was tasked with creating a program that solves a real-life problem. After consulting with my mom, she suggested a problem that I might be able to solve parents struggled to get to the school building, due to their unfamiliarity with the school campus. I decided to create a navigation specifically for the SHSID Campus, to resolve parents’ issues. To do this, I gathered data about SHSID buildings, their respective locations, their distances between each other, and used WXML to create the front-end interface, WXSS to style the interface, and JS to control the page. I used the Bellman Ford algorithm to calculate the shortest distance between any two buildings and traced the shortest route onto the map by using canvas element to draw. Ultimately, this project reinforced my planning skill, sharpened my debugging skill, and nevertheless improved my coding skills. Despite the efforts, there were improvements to be made, which are to include floor plans for each multi-story building, support more languages, add more possible routes from one building to another, and integrate live location into this program. By completing this program as a project, I was able to learn coding a WeChat Mini Program, meanwhile enjoying the app development process, which I conclude to be a very meaningful project for me!

# **Appendix**

Appendix 1 (home.wxml):

<view class="container">

  <view class="section">

    <picker mode="multiSelector" bindchange="bindStartMultiPickerChange" bindcolumnchange="bindStartMultiPickerColumnChange" value="{{startMultiIndex}}" range="{{startMultiArray}}">

    <view class="picker">

      <icon type="search" color="black" size="18px"></icon>

      Starting Location: {{startMultiArray[1][startMultiIndex[1]]}}

    </view>

  </picker>

  </view>

  <view class="section">

    <picker mode="multiSelector" bindchange="bindEndMultiPickerChange" bindcolumnchange="bindEndMultiPickerColumnChange" value="{{endMultiIndex}}" range="{{endMultiArray}}">

    <view class="picker">

      <icon type="search" color="black" size="18px"></icon>

      Ending Location: {{endMultiArray[1][endMultiIndex[1]]}}

    </view>

  </picker>

  </view>

  <view class="map\_view" catchtouchstart="touchstartCallback" catchtouchmove="touchmoveCallback" catchtouchend="touchendCallback">

    <image style="transform:translate({{stv.offsetX}}px, {{stv.offsetY}}px) scale({{stv.scale}});" src="../../media/images/SHSID\_Map.png" mode="aspectFit"></image>

  </view>

  <view class="done">

    <button type="primary" data-path="routes/index" class="done" bindtap="toRoute">Find Route</button>

  </view>

</view>

Appendix 2 (routes.wxml):

<view class="container">

  <view class="map\_view\_out" catchtouchstart="touchstartCallback" catchtouchmove="touchmoveCallback" catchtouchend="touchendCallback">

  <view class="map\_view\_in">

    <image style="transform:translate({{stv.offsetX}}px, {{stv.offsetY}}px) scale({{stv.scale}}); transform-origin: center;" src="../../media/images/SHSID\_Map.png" mode="aspectFit" class="coveredImage"></image>

    <canvas style="transform:translate({{stv.offsetX}}px, {{stv.offsetY}}px) scale({{stv.scale}}); transform-origin: center;" canvas-id="canvas" class="coveringCanvas" ></canvas>

  </view>

  </view>

  <view class="routes">

  From {{startingLocation}} to {{endingLocation}}

  </view>

  <view class="info-row">

      <view class="info">

        {{shortestDistance}} meters

      </view>

      <view class="info">

        {{shortestTime}} minutes

      </view>

  </view>

  <view wx:for="{{directions}}" class="routes">

    <view class="routes-text">

      <text>{{item[0]}}</text>

    </view>

    <view class="routes-image" >

      <image wx:if="{{item[1]}}" src="../../media/images/{{item[1]}}.png" mode="aspectFit" style="width: 30px; height: 30px;"></image>

    </view>

    <text>\n</text>

  </view>

</view>

Appendix 3 (Bellman Ford Algorithm):

bellmanFord: function(){

    let edges = this.data.edges;

    let N = 64; // there are N nodes

    let predecessor = [];

    this.data.dist.length = 0;

    for (let i = 0; i <= N; i++){ // there are N nodes

      this.data.dist.push(100000);

      predecessor.push(null);

    }

    this.data.dist[this.data.startIndex[0][0]] = 0;

    for (let k = 1; k <= N-1; k++) {

      for (let i = 0; i < this.data.edges.length; i++) {

          let a = edges[i][0];

          let b = edges[i][1];

          let w = edges[i][2];

          if (this.data.dist[b] > this.data.dist[a] + w){

            this.data.dist[b] = this.data.dist[a] + w;

            predecessor[b] = a;

          }

      }

    }

    let route = []

    let directions = [[-1, 0, this.data.startIndex[1][0]]];

    let current = this.data.endIndex[0][0];

    while (!(current === null)){

      route.splice(0, 0, current);

      current = predecessor[current];

    }

    for (let i = 1; i < route.length; i++){

      let startFrom = route[i-1];

      let endAt = route[i];

      for (let e = 0; e < this.data.edges.length; e++){

        if (this.data.edges[e][0] == startFrom && this.data.edges[e][1] == endAt){

          directions.push([endAt, this.data.edges[e][2], this.data.edges[e][3]]);

        }

      }

    }

    directions.push([-1, 0, this.data.endIndex[1][0]]);

    this.setData({

      shortestDistance: Math.round(this.data.dist[this.data.endIndex[0][0]] \* this.data.multiplier),

      shortestTime: Math.ceil(Math.round(this.data.dist[this.data.endIndex[0][0]] \* this.data.multiplier) / 60),

      route: directions,

    })

  },

Appendix 4 (home.js):

// home.js

const app = getApp()

Page({

  data: {

    userInfo: {},

    hasUserInfo: false,

    canIUse: wx.canIUse('button.open-type.getUserInfo'),

    canIUseGetUserProfile: false,

    canIUseOpenData: wx.canIUse('open-data.type.userAvatarUrl') && wx.canIUse('open-data.type.userNickName'), // 如需尝试获取用户信息可改为false

    stv: {

      offsetX: 0,

      offsetY: 0,

      zoom: false,

      distance: 0,

      scale: 1,

    },

    startMultiArray: [

      ["Gates", "Teaching Buildings", "Parking Lots", "Cafeterias", "Sport Grounds", "Other Places"],

      ["North Gate: Baise Road", "South Gate: Shang Zhong Road", "East Gate: North Long Chuan Road"],

    ],

    endMultiArray: [

      ["Gates", "Teaching Buildings", "Parking Lots", "Cafeterias", "Sport Grounds", "Other Places"],

      ["North Gate: Baise Road", "South Gate: Shang Zhong Road", "East Gate: North Long Chuan Road"],

    ],

    objectMultiArray: [

      [

        {

          id: 0,

          name: 'Gates'

        },

        {

          id: 1,

          name: 'Teaching Buildings'

        },

        {

          id: 2,

          name: 'Parking Lots'

        },

        {

          id: 3,

          name: 'Cafeterias'

        },

        {

          id: 4,

          name: 'Sport Grounds'

        },

        {

          id: 5,

          name: 'Other places'

        }

      ],

      [

        {

          id: 0,

          name: 'North Gate: Baise Road'

        },

        {

          id: 1,

          name: 'South Gate: Shang Zhong Road'

        },

        {

          id: 2,

          name: 'East Gate: North Long Chuan Road'

        }

      ]

    ],

    startMultiIndex: [0,0],

    endMultiIndex: [0,0],

  },

  bindStartMultiPickerChange: function(e){

    console.log("Picker value changed into ", e.detail.value);

    this.setData({

      startMultiIndex: e.detail.value,

    })

  },

  bindEndMultiPickerChange: function(e){

    console.log("Picker value changed into ", e.detail.value);

    this.setData({

      endMultiIndex: e.detail.value,

    })

  },

  bindStartMultiPickerColumnChange: function (e) {

    console.log('修改的列为', e.detail.column, '，值为', e.detail.value);

    var data = {

      startMultiArray: this.data.startMultiArray,

      startMultiIndex: this.data.startMultiIndex

    };

    data.startMultiIndex[e.detail.column] = e.detail.value;

    switch (e.detail.column) {

      case 0:

        switch (data.startMultiIndex[0]) {

          case 0:

            data.startMultiArray[1] = ["North Gate: Baise Road", "South Gate: Shang Zhong Road", "East Gate: North Long Chuan Road"];

            break;

          case 1:

            data.startMultiArray[1] = ["Primary School Building - Grades 1-3", "ZhongXing Building - Grades 4-7", "ZhenTao Building - Grades 8-10", "XianMian Building - Grades 11-12", "YiFu Building", "LongMen Building - Local School", "Engineering Building"];

            break;

          case 2:

            data.startMultiArray[1] = ["Teacher's Car Parking Lot", "School Bus Parking Lot"];

            break;

          case 3:

            data.startMultiArray[1] = ["SHSID Cafeteria", "SHS Cafeteria"];

            break;

          case 4:

            data.startMultiArray[1] = ["Tennis Center", "Outdoor Tennis Court", "Gymnasium", "Basketball Court", "Volleyball Court", "Soccer Field", "Indoor Swimming Pool"];

            break;

          case 5:

            data.startMultiArray[1] = ["Auditorium", "Administration Building", "XianMian Annex Building", "SHSID Admissions Office", "Multi-media Center", "Art Center", "General Affairs Building", "Library and Information Center", "Laundry", "Lawson"];

            break;

        }

        data.startMultiIndex[1] = 0;

        break;

    }

    this.setData(data);

  },

  bindEndMultiPickerColumnChange: function (e) {

    console.log('修改的列为', e.detail.column, '，值为', e.detail.value);

    var data = {

      endMultiArray: this.data.endMultiArray,

      endMultiIndex: this.data.endMultiIndex

    };

    data.endMultiIndex[e.detail.column] = e.detail.value;

    switch (e.detail.column) {

      case 0:

        switch (data.endMultiIndex[0]) {

          case 0:

            data.endMultiArray[1] = ["North Gate: Baise Road", "South Gate: Shang Zhong Road", "East Gate: North Long Chuan Road"];

            break;

          case 1:

            data.endMultiArray[1] = ["Primary School Building - Grades 1-3", "ZhongXing Building - Grades 4-7", "ZhenTao Building - Grades 8-10", "XianMian Building - Grades 11-12", "YiFu Building", "LongMen Building - Local School", "Engineering Building"];

            break;

          case 2:

            data.endMultiArray[1] = ["Teacher's Car Parking Lot", "School Bus Parking Lot"];

            break;

          case 3:

            data.endMultiArray[1] = ["SHSID Cafeteria", "SHS Cafeteria"];

            break;

          case 4:

            data.endMultiArray[1] = ["Tennis Center", "Outdoor Tennis Court", "Gymnasium", "Basketball Court", "Volleyball Court", "Soccer Field", "Indoor Swimming Pool"];

            break;

          case 5:

            data.endMultiArray[1] = ["Auditorium", "Administration Building", "XianMian Annex Building", "SHSID Admissions Office", "Multi-media Center", "Art Center", "General Affairs Building", "Library and Information Center", "Laundry", "Lawson"];

            break;

        }

        data.endMultiIndex[1] = 0;

        break;

    }

    this.setData(data);

  },

  // 事件处理函数

  onLoad() {

    if (wx.getUserProfile) {

      this.setData({

        canIUseGetUserProfile: true

      })

    }

  },

  getUserProfile(e) {

    // 推荐使用wx.getUserProfile获取用户信息，开发者每次通过该接口获取用户个人信息均需用户确认，开发者妥善保管用户快速填写的头像昵称，避免重复弹窗

    wx.getUserProfile({

      desc: '展示用户信息', // 声明获取用户个人信息后的用途，后续会展示在弹窗中，请谨慎填写

      success: (res) => {

        console.log(res)

        this.setData({

          userInfo: res.userInfo,

          hasUserInfo: true

        })

      }

    })

  },

  getUserInfo(e) {

    // 不推荐使用getUserInfo获取用户信息，预计自2021年4月13日起，getUserInfo将不再弹出弹窗，并直接返回匿名的用户个人信息

    console.log(e)

    this.setData({

      userInfo: e.detail.userInfo,

      hasUserInfo: true

    })

  },

  touchstartCallback: function(e) {

    console.log(e);

    console.log("touchstartCallback");

    if (e.touches.length === 1){

      let {clientX, clientY} = e.touches[0];

      this.startX = clientX;

      this.startY = clientY;

      this.touchStartEvent = e.touches;

    } else {

      let xMove = e.touches[1].clientX - e.touches[0].clientX;

      let yMove = e.touches[1].clientY - e.touches[0].clientY;

      let distance = Math.sqrt(xMove\*xMove + yMove\*yMove);

      this.setData({

        "stv.distance": distance,

        "stv.zoom": true,

      })

    }

  },

  touchmoveCallback: function(e) {

    if (e.touches.length === 1){

      // single finger movement

      if (this.data.stv.zoom) {

        return;

      }

      let {clientX, clientY} = e.touches[0];

      let offsetX = clientX - this.startX;

      let offsetY = clientY - this.startY;

      this.startX = clientX;

      this.startY = clientY;

      let {stv} = this.data;

      stv.offsetX += offsetX;

      stv.offsetY += offsetY;

      stv.offsetLeftX = -stv.offsetX;

      stv.offsetLeftY = -stv.offsetLeftY;

      this.setData({

        stv: stv

      })

    } else {

      // double finger zooming

      let xMove = e.touches[1].clientX - e.touches[0].clientX;

      let yMove = e.touches[1].clientY - e.touches[0].clientY;

      let distance = Math.sqrt(xMove\*xMove + yMove\*yMove);

      let distanceDiff = distance - this.data.stv.distance;

      let newScale = this.data.stv.scale + 0.005 \* distanceDiff;

      this.setData({

        'stv.distance': distance,

        'stv.scale': newScale,

      })

    }

  },

  touchendCallback: function(e){

    // end of touch

    if (e.touches.length === 0){

      this.setData({

        'stv.zoom': false,

      })

    }

  },

  toRoute: function(e) {

    console.log(e);

    console.log(this.data.startMultiArray[1][this.data.startMultiIndex[1]]);

    console.log(this.data.endMultiArray[1][this.data.endMultiIndex[1]]);

    var path = e.currentTarget.dataset.path;

    wx.navigateTo({

      url: '../' + path,

    })

  }

})

Appendix 5 (routes.js):

const app = getApp()

Page({

  data: {

    startingLocation: '',

    startIndex: -1,

    endingLocation: '',

    endIndex: -1,

    multiplier: 50,

    originalX: Math.round(wx.getSystemInfoSync().windowWidth - wx.getSystemInfoSync().windowWidth / 18.75),

    originalY: Math.round(wx.getSystemInfoSync().windowWidth) \* 800 \* 0.5434 / 750,

    xMultiplier: 1962,

    yMultiplier: 1194,

    xM: Math.round(wx.getSystemInfoSync().windowWidth - wx.getSystemInfoSync().windowWidth / 18.75) / 1962,

    yM: Math.round(wx.getSystemInfoSync().windowWidth) \* 800 \* 0.5434 / 750 / 1194,

    stv: {

      offsetX: 0,

      offsetY: 0,

      zoom: false,

      distance: 0,

      scale: 1,

    },

    dist: [],

    shortestDistance: 0,

    shortestTime: 0,

    route: [],

    directions: [],

    context: '',

    edges: [

      [1, 2, 0.45012, "S"],

      [2, 1, 0.45012, "N"],

      [2, 3, 0.76993, "S"],

      [3, 2, 0.76993, "N"],

      [3, 24, 0.74726, "W"],

      [24, 3, 0.74726, "E"],

      [3, 4, 1.00651, "S"],

      [4, 3, 1.00651, "N"],

      [4, 5, 0.42027, "S"],

      [5, 4, 0.42027, "N"],

      [5, 6, 0.60121, "E"],

      [6, 5, 0.60121, "W"],

      [6, 7, 0.66421, "S"],

      [7, 6, 0.66421, "N"],

      [7, 8, 0.44817, "S"],

      [8, 7, 0.44817, "N"],

      [8, 9, 0.69004, "W"],

      [9, 8, 0.69004, "E"],

      [5, 10, 0.6017, "S"],

      [10, 5, 0.6017, "N"],

      [10, 9, 0.57463, "S"],

      [9, 10, 0.57463, "N"],

      [9, 11, 0.4553, 'S'],

      [11, 9, 0.4553, 'N'],

      [5, 55, 1.42527, 'W'],

      [55, 5, 1.42527, 'E'],

      [24, 25, 0.90523, 'W'],

      [25, 24, 0.90523, 'E'],

      [25, 23, 0.96448, 'S'],

      [23, 25, 0.96448, 'N'],

      [23, 56, 0.37766, 'E'],

      [56, 23, 0.37766, 'W'],

      [56, 55, 0.51794, 'S'],

      [55, 56, 0.51794, 'N'],

      [11, 57, 0.40536, 'S'],

      [57, 11, 0.40536, 'N'],

      [57, 12, 0.79785, 'E'],

      [12, 57, 0.79785, 'W'],

      [57, 13, 0.42143, 'S'],

      [13, 57, 0.42143, 'N'],

      [13, 64, 0.38272, 'E'],

      [64, 13, 0.38272, 'W'],

      [14, 64, 0.66, 'N'],

      [64, 14, 0.66, 'S'],

      [13, 58, 0.57734, 'W'],

      [58, 13, 0.57734, 'E'],

      [58, 18, 0.7693, 'S'],

      [18, 58, 0.7693, 'N'],

      [18, 17, 0.91125, 'S'],

      [17, 18, 0.91125, 'N'],

      [17, 15, 0.61025, 'E'],

      [15, 17, 0.61025, 'W'],

      [17, 16, 0.56347, 'E'],

      [16, 17, 0.56347, 'W'],

      [16, 15, 0.43691, 'S'],

      [15, 16, 0.43691, 'N'],

      [21, 17, 1.16017, 'E'],

      [17, 21, 1.16017, 'W'],

      [21, 46, 0.89731, 'N'],

      [46, 21, 0.89731, 'S'],

      [46, 18, 1.02049, 'E'],

      [18, 46, 1.02049, 'W'],

      [46, 19, 0.75403, 'N'],

      [19, 46, 0.75403, 'S'],

      [19, 58, 1.02433, 'E'],

      [58, 19, 1.02433, 'W'],

      [19, 43, 1.06036, 'W'],

      [43, 19, 1.06036, 'E'],

      [43, 42, 0.9506, 'N'],

      [42, 43, 0.9506, 'S'],

      [43, 61, 0.70391, 'W'],

      [61, 43, 0.70391, 'E'],

      [61, 44, 1.07775, 'W'],

      [44, 61, 1.07775, 'E'],

      [44, 45, 0.89851, 'N'],

      [45, 44, 0.89851, 'S'],

      [45, 41, 1.31994, 'E'],

      [41, 45, 1.31994, 'W'],

      [41, 42, 0.56069, 'E'],

      [42, 41, 0.56069, 'W'],

      [42, 20, 1.03446, 'E'],

      [20, 42, 1.03446, 'W'],

      [42, 22, 1.07412, 'E'],

      [22, 42, 1.07412, 'W'],

      [20, 22, 0.34793, 'N'],

      [22, 20, 0.34793, 'S'],

      [20, 11, 1.65088, 'E'],

      [11, 20, 1.65088, 'W'],

      [45, 31, 0.48199, 'W'],

      [31, 45, 0.48199, 'E'],

      [31, 49, 1.94339, 'W'],

      [49, 31, 1.94339, 'E'],

      [49, 50, 1.00223, 'W'],

      [50, 49, 1.00223, 'E'],

      [50, 32, 0.36584, 'N'],

      [32, 50, 0.36584, 'S'],

      [32, 49, 1.05827, 'E'],

      [49, 32, 1.05827, 'W'],

      [49, 59, 0.41957, 'N'],

      [59, 49, 0.41957, 'S'],

      [59, 33, 0.5202, 'W'],

      [33, 59, 0.5202, 'E'],

      [33, 60, 0.57105, 'N'],

      [60, 33, 0.57105, 'S'],

      [60, 34, 0.61646, 'N'],

      [34, 60, 0.61646, 'S'],

      [34, 52, 0.64435, 'N'],

      [52, 34, 0.64435, 'S'],

      [52, 53, 1.30183, 'W'],

      [53, 52, 1.30183, 'E'],

      [53, 35, 0.35637, 'N'],

      [35, 53, 0.35637, 'S'],

      [35, 54, 1.21608, 'W'],

      [54, 35, 1.21608, 'E'],

      [54, 36, 0.48199, 'N'],

      [36, 54, 0.48199, 'S'],

      [54, 37, 0.46275, 'N'],

      [37, 54, 0.46275, 'S'],

      [37, 36, 0.43332, 'E'],

      [36, 37, 0.43332, 'W'],

      [60, 51, 1.21051, 'E'],

      [51, 60, 1.21051, 'W'],

      [51, 38, 0.38375, 'N'],

      [38, 51, 0.38375, 'S'],

      [51, 48, 1.42593, 'E'],

      [48, 51, 1.42593, 'W'],

      [48, 30, 0.31875, 'S'],

      [30, 48, 0.31875, 'N'],

      [30, 29, 0.40824, 'S'],

      [29, 30, 0.40824, 'N'],

      [29, 45, 1.04844, 'S'],

      [45, 29, 1.04844, 'N'],

      [48, 40, 1.37162, 'E'],

      [40, 48, 1.37162, 'W'],

      [40, 28, 0.57378, 'S'],

      [28, 40, 0.57378, 'N'],

      [28, 41, 0.85277, 'S'],

      [41, 28, 0.85277, 'N'],

      [47, 48, 1.34305, 'S'],

      [48, 47, 1.34305, 'N'],

      [47, 26, 0.82517, 'E'],

      [26, 47, 0.82517, 'W'],

      [26, 39, 0.62402, 'E'],

      [39, 26, 0.62402, 'W'],

      [39, 27, 0.32271, 'S'],

      [27, 39, 0.32271, 'W'],

      [27, 40, 1.096, 'S'],

      [40, 27, 1.096, 'N'],

      [39, 25, 1.58006, 'E'],

      [25, 39, 1.58006, 'W'],

      [62, 47, 2.5522, 'E'],

      [47, 62, 2.5522, 'W'],

      [62, 52, 0.37307, 'S'],

      [52, 62, 0.37307, 'N'],

      [63, 7, 1.05636, 'W'],

      [7, 63, 1.05636, 'E'],

    ],

    coords: {

      1: [1624, 988],

      2: [1623, 929],

      3: [1619, 837],

      4: [1631, 728],

      5: [1619, 667],

      6: [1721, 667],

      7: [1722, 597],

      8: [1727, 523],

      9: [1621, 523],

      10: [1631, 595],

      11: [1615, 479],

      12: [1721, 409],

      13: [1619, 361],

      14: [1661, 261],

      15: [1631, 129],

      16: [1621, 179],

      17: [1553, 137],

      18: [1559, 269],

      19: [1423, 363],

      20: [1423, 463],

      21: [1423, 133],

      22: [1421, 489],

      23: [1418, 711],

      24: [1536, 837],

      25: [1417, 833],

      26: [1151, 841],

      27: [1221, 793],

      28: [1226, 587],

      29: [1059, 587],

      30: [1047, 633],

      31: [1019, 487],

      32: [638, 505],

      33: [714, 557],

      34: [732, 707],

      35: [590, 827],

      36: [469, 871],

      37: [440, 880],

      38: [880, 687],

      39: [1229, 829],

      40: [1229, 667],

      41: [1229, 473],

      42: [1295, 473],

      43: [1295, 357],

      44: [1067, 357],

      45: [1067, 475],

      46: [1425, 265],

      47: [1045, 831],

      48: [1057, 665],

      49: [763, 475],

      50: [664, 469],

      51: [880, 653],

      52: [741, 785],

      53: [577, 785],

      54: [455, 831],

      55: [1469, 639],

      56: [1469, 705],

      57: [1619, 407],

      58: [1549, 357],

      59: [763, 531],

      60: [739, 639],

      61: [1185, 357],

      62: [741, 830],

      63: [1857, 607],

      64: [517, 423],

    },

    key: {

      "North Gate: Baise Road": [[1], ['S']],

      "South Gate: Shang Zhong Road": [[21], ['N']],

      "East Gate: North Long Chuan Road": [[63], ['W']],

      "Primary School Building - Grades 1-3": [[36], ['S']],

      "ZhongXing Building - Grades 4-7": [[61], ['N']],

      "ZhenTao Building - Grades 8-10": [[4], ['W']],

      "XianMian Building - Grades 11-12": [[18],  ['W']],

      "YiFu Building": [[14], ['W']],

      "LongMen Building - Local School": [[20, 19], ['N', 'S']],

      "Engineering Building": [[10], ['W']],

      "Teacher's Car Parking Lot": [[2], ['W']],

      "School Bus Parking Lot": [[7], ['W']],

      "SHSID Cafeteria": [[25], ['S']],

      "SHS Cafeteria": [[27], ['E']],

      "Tennis Center": [[35], ['W']],

      "Outdoor Tennis Court": [[37], ['E']],

      "Gymnasium": [[32, 33], ['S', 'E']],

      "Basketball Court": [[34], ['E']],

      "Volleyball Court": [[50], ['N']],

      "Soccer Field": [[29], ['W']],

      "Indoor Swimming Pool": [[30], ['E']],

      "Auditorium": [[22], ['S']],

      "Administration Building": [[12], ['S']],

      "XianMian Annex Building": [[15], ['W']],

      "SHSID Admissions Office": [[16], ['W']],

      "Multi-media Center": [[23], ['N']],

      "Art Center": [[24], ['S']],

      "General Affairs Building": [[28], ['E']],

      "Library and Information Center": [[31], ['S']],

      "Laundry": [[26], ['S']],

      "Lawson": [[17], ['N']],

    }

  },

  drawLine (ind) {

    let coords = this.data.coords[ind];

    this.data.context.lineTo(coords[0]\*this.data.xM, this.data.originalY - coords[1]\*this.data.yM);

    this.data.context.moveTo(coords[0]\*this.data.xM, this.data.originalY - coords[1]\*this.data.yM);

  },

  bellmanFord: function(){

    let edges = this.data.edges;

    let N = 64; // there are N nodes

    let predecessor = [];

    this.data.dist.length = 0;

    for (let i = 0; i <= N; i++){ // there are N nodes

      this.data.dist.push(100000);

      predecessor.push(null);

    }

    this.data.dist[this.data.startIndex[0][0]] = 0;

    for (let k = 1; k <= N-1; k++) {

      for (let i = 0; i < this.data.edges.length; i++) {

          let a = edges[i][0];

          let b = edges[i][1];

          let w = edges[i][2];

          if (this.data.dist[b] > this.data.dist[a] + w){

            this.data.dist[b] = this.data.dist[a] + w;

            predecessor[b] = a;

          }

      }

    }

    let route = []

    let directions = [[-1, 0, this.data.startIndex[1][0]]];

    let current = this.data.endIndex[0][0];

    while (!(current === null)){

      route.splice(0, 0, current);

      current = predecessor[current];

    }

    for (let i = 1; i < route.length; i++){

      let startFrom = route[i-1];

      let endAt = route[i];

      for (let e = 0; e < this.data.edges.length; e++){

        if (this.data.edges[e][0] == startFrom && this.data.edges[e][1] == endAt){

          directions.push([endAt, this.data.edges[e][2], this.data.edges[e][3]]);

        }

      }

    }

    directions.push([-1, 0, this.data.endIndex[1][0]]);

    this.setData({

      shortestDistance: Math.round(this.data.dist[this.data.endIndex[0][0]] \* this.data.multiplier),

      shortestTime: Math.ceil(Math.round(this.data.dist[this.data.endIndex[0][0]] \* this.data.multiplier) / 60),

      route: directions,

    })

  },

  getDirections: function(){

    let key = {"N": 1, "E": 2, "S": 3, "W": 4};

    let dirkey = {"1": "right", "3": "left", "2": "in front of you"};

    let route = this.data.route;

    let length = 0;

    let direction = "";

    let directions = [["Start at " + this.data.startingLocation]];

    for (let i = 1; i < route.length - 1; i++){

      if (route[i][2] == route[i-1][2]){

        length += route[i][1];

      } else {

        if (i != 1){

          directions.push([" Walk for " + Math.round(length \* this.data.multiplier) + " meters"]);

        }

        direction = dirkey[((key[route[i][2]] - key[route[i-1][2]]+4)%4).toString()];

        directions[directions.length - 1][0] += ", Turn " + direction;

        directions[directions.length - 1][1] = direction;

        length = route[i][1];

      }

    }

    if (length != 0){

      directions.push([" Walk for " + Math.round(length \* this.data.multiplier) + " meters"]);

      direction = dirkey[((key[route[route.length - 2][2]] - key[route[route.length - 1][2]] + 4)%4).toString()];

      if (direction == "left" || direction == "right" ){

        directions.push([" " + this.data.endingLocation + " is on your " + direction]);

      } else {

        directions.push([" " + this.data.endingLocation + " is " + direction]);

      }

    }

    if (directions.length == 1){

      directions.push([" You are already at " + this.data.endingLocation]);

    }

    this.setData({

      directions: directions,

    })

  },

  onLoad: function(){

    const currentPages = getCurrentPages();

    let startingLocation = currentPages[0].\_\_data\_\_.startMultiArray[1][currentPages[0].\_\_data\_\_.startMultiIndex[1]];

    let endingLocation = currentPages[0].\_\_data\_\_.endMultiArray[1][currentPages[0].\_\_data\_\_.endMultiIndex[1]];

    this.setData({

      startingLocation: startingLocation,

      endingLocation: endingLocation,

      startIndex: this.data.key[startingLocation],

      endIndex: this.data.key[endingLocation],

    })

    this.bellmanFord();

    this.getDirections();

    var context = wx.createContext();

    context.setStrokeStyle('red');

    context.lineWidth = 3;

    context.moveTo(this.data.coords[this.data.startIndex[0][0]][0]\*this.data.xM, this.data.originalY - this.data.coords[this.data.startIndex[0][0]][1]\*this.data.yM);

    context.drawImage("../../media/images/placeholder.png", this.data.coords[this.data.startIndex[0][0]][0]\*this.data.xM - 15, this.data.originalY - this.data.coords[this.data.startIndex[0][0]][1]\*this.data.yM - 30, 30, 30);

    this.setData({

      context: context,

    })

    for (let i = 1; i < this.data.route.length - 1; i++){

      this.drawLine(this.data.route[i][0]);

    }

    this.data.context.fill();

    this.data.context.stroke();

    wx.drawCanvas({

      canvasId: 'canvas',

      actions: this.data.context.getActions()

    })

  },

  touchstartCallback: function(e) {

    if (e.touches.length === 1){

      let {clientX, clientY} = e.touches[0];

      this.startX = clientX;

      this.startY = clientY;

      this.touchStartEvent = e.touches;

    } else {

      let xMove = e.touches[1].clientX - e.touches[0].clientX;

      let yMove = e.touches[1].clientY - e.touches[0].clientY;

      let distance = Math.sqrt(xMove\*xMove + yMove\*yMove);

      this.setData({

        "stv.distance": distance,

        "stv.zoom": true,

      })

    }

  },

  touchmoveCallback: function(e) {

    if (e.touches.length === 1){

      // single finger movement

      if (this.data.stv.zoom) {

        return;

      }

      let {clientX, clientY} = e.touches[0];

      let offsetX = clientX - this.startX;

      let offsetY = clientY - this.startY;

      this.startX = clientX;

      this.startY = clientY;

      let {stv} = this.data;

      stv.offsetX += offsetX;

      stv.offsetY += offsetY;

      stv.offsetLeftX = -stv.offsetX;

      stv.offsetLeftY = -stv.offsetLeftY;

      this.setData({

        stv: stv

      })

    } else {

      // double finger zooming

      let xMove = e.touches[1].clientX - e.touches[0].clientX;

      let yMove = e.touches[1].clientY - e.touches[0].clientY;

      let distance = Math.sqrt(xMove\*xMove + yMove\*yMove);

      let distanceDiff = distance - this.data.stv.distance;

      let newScale = this.data.stv.scale + 0.005 \* distanceDiff;

      this.setData({

        'stv.distance': distance,

        'stv.scale': newScale,

      })

    }

  },

  touchendCallback: function(e){

    // end of touch

    if (e.touches.length === 0){

      this.setData({

        'stv.zoom': false,

      })

    }

  },

})

Appendix 6 (home.wxss):

/\*\*home.wxss\*\*/

.section {

  border: 1px solid black;

  border-radius: 3px;

  padding: 5px;

  margin-top: 5px;

}

.map\_view{

  width: 100%;

  height: 800rpx;

  overflow: hidden;

  margin-top: 10px;

  margin-bottom: 10px;

}

.map\_view image {

  width: 100%;

  height: 64%

}

Appendix 7 (routes.wxss):

.map\_view\_out{

  width: 100%;

  height: 500rpx;

  overflow: hidden;

  margin-top: 10px;

  margin-bottom: 10px;

}

.map\_view\_in {

  width: 100%;

  height: 100%;

  position: relative;

}

.coveredImage {

  width: 100%;

  height: 86.936%;

  position: absolute;

  top: 0px;

  left: 0px;

}

.coveringCanvas {

  width: 100%;

  height: 86.936%;

  position: absolute;

  top: 0px;

  left: 0px;

}

.routes {

  padding: 10px;

  font-size: 14px;

  border: 1px solid black;

}

.routes-text {

  width: 70%;

  float: left;

}

.routes-image {

  width: 30%;

  float: left;

  align-self:center

}

.info-row {

  padding: 10px;

  font-size: 14px;

  border: 1px solid black

}

.info {

  width: 49.5%;

  float: left;

  text-align: center;

}

.info:first-child {

  border-right: 1px solid black

}