|  |
| --- |
| 第一问 |
| DataAnaly.m |
| clear,home  %% 导入数据  %[~,~,data] = xlsread('附件一.xls');  [data] = readtable('Train.csv');  [n,m] = size(data);  for i=1:n  data.A\_Time{i}=data.A\_Time{i}(12:end);  data.D\_Time{i}=data.D\_Time{i}(12:end);  end  save data.mat  %% 转换时间  [n] = size(data,1);  for i=1:n  index = strfind(data.A\_Time{i},':');  h = str2double(data.A\_Time{i}(1:index(1)-1));  m = str2double(data.A\_Time{i}(index(1)+1:index(2)-1));  totalM = h\*60 + m;  data.A\_Time\_CurrentMinute(i) = totalM;  data.A\_Time\_TotalMinute(i) = totalM + (data.Day(i)-1)\*24\*60;  end  for i=1:n  index = strfind(data.D\_Time{i},':');  h = str2double(data.D\_Time{i}(1:index(1)-1));  m = str2double(data.D\_Time{i}(index(1)+1:index(2)-1));  totalM = h\*60 + m;  data.D\_Time\_CurrentMinute(i) = totalM;  data.D\_Time\_TotalMinute(i) = totalM + (data.Day(i)-1)\*24\*60;  end  save data2.mat  %% 将每一列车的站点信息放入一行，保存成CVS格式  [n,m] = size(data);  fileID = fopen('trainLine.txt','w');  for i=1:n  if data.S\_No(i)==1  fprintf(fileID,'\n%s,%s',data.ID{i},data.Type{i});  end  fprintf(fileID,',%s',data.Station{i});  end  fclose(fileID);  %% 列车类型与列车编号  trainType = unique(data.Type);  n = size(trainType,1);  typeIndex = cell(n,1);  for i = 1:n  typeIndex{i} = find(strcmp(data.Type,trainType(i)));  end  trainNo = unique(data.ID);  n = size(data,1);  for i=1:n  [~,No] = intersect(trainNo,data.ID{i});  data.NumNo(i) = No;  end  %% 站点编号  station = unique(data.Station);  n = size(data,1);  for i=1:n  [~,No] = intersect(station,data.Station{i});  data.StationNo(i) = No;  end  %% 建立连接矩阵  m = length(station);  startStation = find(data.S\_No==1);  adjMat = zeros(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  % [~,s\_No] = intersect(station,data.Station{i});  for j = i+1:endStation  % [~,d\_No] = intersect(station,data.Station{j});  adjMat(data.StationNo(i),data.StationNo(j)) = 1;  end  end  save data3  %% 任意两站点之间的车次  m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  trainList{data.StationNo(i),data.StationNo(j)} = [trainList{data.StationNo(i),data.StationNo(j)},data.NumNo(i)];  end  end  save data4.mat |
| directSearchFun1.m |
| function [ path,pathTime,pathCost ] = directSearchFun1( data,trainList,adjMat,station,s\_name,d\_name )  [~,s\_No] = intersect(station,s\_name);  [~,d\_No] = intersect(station,d\_name);  %% 直达  if isempty(trainList{s\_No,d\_No})  path = [];  else  path = trainList{s\_No,d\_No}';  end  path = [ones(size(path,1),1)\*s\_No path ones(size(path,1),1)\*d\_No];  %% 计算时间  pathTime = zeros(size(path,1),1);  for i=1:size(path,1)  pathTime(i) = timeFun( data,path(i,:) );  end  %% 计算费用  pathCost = zeros(size(path,1),1);  for i=1:size(path,1)  [pathCost(i),~] = pathCostFun( data,path(i,:) );  end  end |
| directSearchFun2.m |
| function [ path,pathTime,pathCost,stopTime ] = directSearchFun2( data,trainList,adjMat,station,s\_name,d\_name )  [~,s\_No] = intersect(station,s\_name);  [~,d\_No] = intersect(station,d\_name);  %% 一次转乘  path=[];  s\_set = find(adjMat(s\_No,:)==1);  d\_set = find(adjMat(:,d\_No)==1);  stop = intersect(s\_set, d\_set);  for i=1:length(stop)  firstarc = trainList{s\_No,stop(i)};  for j = 1:length(firstarc)  secondarc = trainList{stop(i),d\_No};  for k = 1:length(secondarc)  path = [path;[firstarc(j) stop(i) secondarc(k)]];  end  end  end  path = [ones(length(path),1)\*s\_No path ones(length(path),1)\*d\_No];  %% 计算时间  pathTime = zeros(size(path,1),1);  stopTime = cell(size(path,1),1);  for i=1:size(path,1)  [pathTime(i),stopTime{i}] = timeFun( data,path(i,:) );  end  %% 计算费用  pathCost = zeros(size(path,1),1);  for i=1:size(path,1)  [pathCost(i),~] = pathCostFun( data,path(i,:) );  end  end |
| costFun.m |
| function [ cost ] = costFun( data,s\_No,d\_No,trainID )  %costFun 计算两个站点之间某车次的票价  % data 为列车数据库  % s\_No 为出发点编号  % d\_No 为目的地编号  % trainNo 为车次编号  % cost 为票价  % s\_No = 933; d\_No = 1065; trainID = 1154;  s = find(data.NumNo==trainID&data.StationNo==s\_No);  d = find(data.NumNo==trainID&data.StationNo==d\_No);  if data.S\_No(s)==1&&(~isempty(data.P1(d)))&&(~isnan(data.P1(d)))  cost = data.P1(d);  return  end  if data.S\_No(s)==1&&(~isempty(data.P3(d)))&&(~isnan(data.P3(d)))  cost = data.P3(d);  return  end  x = data.Distance(d) - ...  data.Distance(s);  index = find(data.NumNo==trainID,1);  if strcmp(data.Type{index},'城际高速')  cost = min(table2array(data(d,9:12))) - min(table2array(data(s,9:12)));  cost =round(cost);  return  end  if strcmp(data.Type{index},'动车组')  cost = 0.3085\*x;  cost =round(cost);  return  end  if strcmp(data.Type{index},'高速动车')  cost = 0.485\*x;  cost =round(cost);  return  end  if strcmp(data.Type{index},'新空直达')    cost = min(table2array(data(d,9:12))) - min(table2array(data(s,9:12)));  cost =round(cost);  return  end  sigma = 1;  if strcmp(data.Type{index},'普客')  sigma = sigma+0.02;  end  if strcmp(data.Type{index},'普快')  sigma = sigma+0.02+0.2;  end  if strcmp(data.Type{index},'新空普快')  sigma = sigma+0.02+0.2+0.25;  end  if strcmp(data.Type{index},'新空普客')  sigma = sigma+0.02+0.25;  end  if strcmp(data.Type{index},'快速')  sigma = sigma+0.02+0.4;  end  if strcmp(data.Type{index},'新空快速')  sigma = sigma+0.02+0.4+0.25;  end  if strcmp(data.Type{index},'新空特快')  sigma = sigma+0.02+0.4+0.25;  end  eta = 0.05861;  if 0<x&&x>200  cost = eta\*x\*sigma+1;  elseif 200>x&&x<=500  cost = eta\*(200+(x-200)\*0.9)\*sigma+1;  elseif 500>x&&x<=1000  cost = eta\*(200+300\*0.9+(x-500)\*0.8)\*sigma+1;  elseif 1000>x&&x<=1500  cost = eta\*(200+300\*0.9+500\*0.8+(x-1000)\*0.7)\*sigma+1;  elseif 1500>x&&x<=2500  cost = eta\*(200+300\*0.9+500\*0.8+500\*0.7+(x-1500)\*0.6)\*sigma+1;  else  cost = eta\*(200+300\*0.9+500\*0.8+500\*0.7+1000\*0.6+(x-2500)\*0.5)\*sigma+1;  end  cost =round(cost);  end |
| timeFun.m |
| function [ totalTime,stopTime,startTime ] = timeFun( data,sigpath )  %timeFun 计算路径方案的时间  % data 列车数据  % path 路径，格式为起点-车次-中转站-车次-终点  % totalTime 方案的总时间  % stopTime 方案的中间停留时间  % startTime 方案的出发时间  n = (length(sigpath)-1)/2;  section = zeros(n,2);  sectionTime = zeros(n,1);  stopTime = zeros(n-1,1);  for i=1:n  temp = sigpath(i\*2-1:i\*2+1);  m = find(data.NumNo==temp(2)&data.StationNo==temp(1));  n = find(data.NumNo==temp(2)&data.StationNo==temp(3));  section(i,:) = [m n];  sectionTime(i) = data.A\_Time\_TotalMinute(n)-data.D\_Time\_TotalMinute(m);  end  if n>1  n = (length(sigpath)-1)/2;  for i=1:n-1  s\_stop = data.A\_Time\_CurrentMinute(section(i,2));  d\_stop = data.D\_Time\_CurrentMinute(section(i+1,1));  if d\_stop<s\_stop  stopTime(i) = d\_stop + 24\*60 - s\_stop;  else  stopTime(i) = d\_stop - s\_stop;  end  end  else  stopTime = 0;  end  totalTime = sum(sectionTime) + sum(stopTime);  startTime = data.D\_Time(section(1,1));  end |
| bestPathFun.m |
| function [ bestIndex ] = bestPathFun( pathCost,pathTime,type )  %bestPathFun 将方案按照一定的优先级进行排序  % path 所有方案  % type 优先级类型：1为时间优先，2为费用优先  % bestPath 方案的排序  if type==1  [~,bestIndex]=sort(pathTime);  end  if type==2  [~,bestIndex]=sort(pathCost);  end  end |
| displayFun.m |
| function displayFun( path,pathCost,pathTime,station,trainNo )  %displayFun 输出方案  for i=1:length(path)  if mod(i,2)==1  fprintf('%s-',station{path(i)})  else  fprintf('%s-',trainNo{path(i)})  end  end  fprintf('%d元-',pathCost)  h = floor(pathTime/60);  m = mod(pathTime,60);  fprintf('%d小时%d分钟\n',h,m)  end |
| main.m |
| % 主程序  % 输出欢迎界面  fprintf('欢迎使用铁路查询系统！\n请输入起始站与终点站\n')  % 用户输入数据  s\_name = input('起始站：','s');  while ~sum(strcmp(station,s\_name))  fprintf('Sorry，没有找到此站点，请重新输入！\n')  s\_name = input('起始站：','s');  end  d\_name = input('终点站：','s');  while ~sum(strcmp(station,d\_name))  fprintf('Sorry，没有找到此站点，请重新输入！\n')  d\_name = input('终点站：','s');  end  type = input('输入优先级，时间优先输入1，费用优先输入2：');  path = cell(2,1);  pathTime = cell(2,1);  pathCost = cell(2,1);  % 直达方案  [ path{1},pathTime{1},pathCost{1} ] = directSearchFun1(...  data,trainList,adjMat,station,s\_name,d\_name );  goOn = 0;  if isempty(path{1})  fprintf('%s到%s ',s\_name,d\_name)  fprintf('没有直达车次 为您推荐如下转乘方案：\n')  goOn = 1;  else  [ bestIndex{1} ] = bestPathFun( pathCost{1},pathTime{1},type );  for i=1:length(path{1})  n = bestIndex{1}(i);  displayFun( path{1}(n,:),pathCost{1}(n),pathTime{1}(n),station,trainNo )  end  end  % 转乘一次  goOn = input('是否继续搜索一次转乘方案，若是输入1，否则输入0：');  while goOn~=1&&goOn~=0  goOn = input('是否继续搜索一次转乘方案，若是输入1，否则输入0：');  end  if goOn == 1  [ path{2},pathTime{2},pathCost{2},stopTime ] = directSearchFun2(...  data,trainList,adjMat,station,s\_name,d\_name );  if isempty(path{2})  fprintf('%s到%s ',s\_name,d\_name)  fprintf('没有一次转乘方案，无法搜索，Sorry\n')  else  [ bestIndex{2} ] = bestPathFun( pathCost{2},pathTime{2},type );  for i=1:length(path{2})  n = bestIndex{2}(i);  displayFun( path{2}(n,:),pathCost{2}(n),pathTime{2}(n),station,trainNo )  h = floor(stopTime{i}./60);  m = mod(stopTime{i},60);  fprintf('中转站等待时间：%d小时%d分钟\n',h,m)  end  end  end |
| oneBestPathFun.m |
| function [ bestPath,cost,time ] = oneBestPathFun( data,trainList,adjMat,station,s\_name,d\_name,type )  %oneBestPathFun 输出一条最优路线  [ path,pathTime,pathCost ] = directSearchFun1( data,trainList,adjMat,station,s\_name,d\_name );  if type==1  [directMin,directIndex] = min(pathTime);  bestPath = path(directIndex,:);  time = pathTime(directIndex);  cost = pathCost(directIndex);  else  [directMin,directIndex] = min(pathCost);  bestPath = path(directIndex,:);  time = pathTime(directIndex);  cost = pathCost(directIndex);  end  end |
| pathCostFun.m |
| function [ totalCost,cost ] = pathCostFun( data,sigpath )  %pathCostFun 计算路径方案的总费用  % path为路径方案  % totalCost为总费用  % cost为各车次票价  n = (length(sigpath)-1)/2;  cost = zeros(n,1);  for i=1:n  cost(i) = costFun( data,sigpath(i\*2-1),sigpath(i\*2+1),sigpath(i\*2) );  end  totalCost = sum(cost);  end |
| startStationMFile.m |
| m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  n = size(data,1);  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  trainList{data.StationNo(i),data.StationNo(j)} = [trainList{data.StationNo(i),data.StationNo(j)},data.NumNo(i)];  end  end  save data4.mat |
| dijkstraMain.m |
| tic  s = find(strcmp(station,'上海'));  d = find(strcmp(station,'苏州'));  beginH = 8;  beginM = 0;  beginTime = beginH\*60 + beginM;  [ t,p,q ] = dijkstraTurnFun( adjMat,C,TS,s,d,beginTime,60 );  toc  p(s) = 0;  path = graphpred2path(p',d);  n = size(path,2)\*2-1;  fullpath = zeros(1,n);  % fullpath(1) = path(1);  % fullpath(2) = trainList{path(1),path(2)}(q(path(2)));  % fullpath(3) = path(2);  % fullpath(4) = trainList{path(2),path(3)}(q(path(3)));  for i = 1:n  if mod(i,2)==1  fullpath(i) = path((i+1)/2);  else  fullpath(i) = trainList{path(i/2),path(i/2+1)}(q(path(i/2+1)));  end  end  pathCost = pathCostFun( data,fullpath );  [ totalTime,stopTime,startTime ] = timeFun( data,fullpath );  displayFun( fullpath,pathCost,totalTime,station,trainNo )  disp(stopTime)  % s\_No = 1579;  % d\_No = 933;  % station(s\_No)  % station(d\_No)  % train = trainNo(trainList{s\_No,d\_No}(q(d\_No))) |
| DataPossess.m |
| %% 任意两站点之间的车次  m = length(station);  startStation = find(data.S\_No==1);  trainList = cell(m,m);  C = cell(m,m);  TS = cell(m,m);  r = zeros(m,m);  n = size(data,1);  h = waitbar(0,'Please wait...');  for i = 1:n-1  endStation = startStation(find(startStation > i,1))-1;  for j = i+1:endStation  s\_No = data.StationNo(i);  d\_No = data.StationNo(j);  sigpath = data.NumNo(i);  trainList{s\_No,d\_No} = [trainList{s\_No,d\_No},sigpath];  %[ totalTime,~,startTime ] = timeFun( data,[s\_No,sigpath,d\_No] );  C{s\_No,d\_No} = [C{s\_No,d\_No},data.A\_Time\_TotalMinute(j)-data.D\_Time\_TotalMinute(i)];  TS{s\_No,d\_No} = [TS{s\_No,d\_No},data.D\_Time\_CurrentMinute(i)];  r(s\_No,d\_No) = r(s\_No,d\_No) + 1;  end  waitbar(i/n)  end  close(h)  save data6.mat  DG = sparse(adjMat);  minTurn = graphallshortestpaths(DG); |
| dijkstraFun.m |
| function [ t,p,q ] = dijkstraFun( adjMat,C,TS,s,d,beginTime,stopTime )  %dijkstraFun 改进的Dijkstra算法  % adjMat 邻接矩阵  % c 节点之间的行驶时间矩阵  % TS 出发时间  N = size(adjMat,1);  Node = 1:N; % 节点集合  t = zeros(N,1); % 到达节点u的最短时间  %dayT = zeros(N,1); % 到达节点u时的当天时间  p = zeros(N,1); % s到u最短路径上的u的前一个节点  q = zeros(N,1); % s到u最短路径上p(u)到u出发时间的索引  %S = [];  %s = 1; % 设置出发点  % 初始化  t(s) = beginTime;  %dayT(s) = 0;  p(s) = nan;  q(s) = nan;  S = s;  for v=setdiff(Node,s)  t(v) = inf;  p(v) = nan;  q(v) = nan;  end  while ~isempty(S)  [~,index] = min(t(S));  u = S(index);  S = setdiff(S,u);  for v = find(adjMat(u,:))    for i = 1:length(TS{u,v})  temp = TS{u,v}(i);  while temp<t(u)+stopTime;  temp = temp + 24\*60;  end  if (t(v)>temp+C{u,v}(i))  p(v) = u;  q(v) = i;  t(v) = temp + C{u,v}(i);  S = union(S,v);  end  end    end  if u == d  return  end  end  end |
| dijkstraTurnFun.m |
| function [ t,p,q,turn ] = dijkstraTurnFun( adjMat,C,TS,s,d,beginTime,stopTime )  %dijkstraFun 改进的Dijkstra算法  % adjMat 邻接矩阵  % c 节点之间的行驶时间矩阵  % TS 出发时间  N = size(adjMat,1);  Node = 1:N; % 节点集合  t = zeros(N,1); % 到达节点u的最短时间  turn = zeros(N,1); % 到达节点u的最小转乘次数  %dayT = zeros(N,1); % 到达节点u时的当天时间  p = zeros(N,1); % s到u最短路径上的u的前一个节点  q = zeros(N,1); % s到u最短路径上p(u)到u出发时间的索引  %S = [];  %s = 1; % 设置出发点  % 初始化  t(s) = beginTime;  turn(s) = 0;  %dayT(s) = 0;  p(s) = nan;  q(s) = nan;  S = s;  for v=setdiff(Node,s)  t(v) = inf;  turn(v) = inf;  p(v) = nan;  q(v) = nan;  end  while ~isempty(S)  tAndTurn = [t(S),turn(S)];  [~,index] = sortrows(tAndTurn,[2,1]);  index = index(1);  %[~,index] = min(t(S));  %[~,index] = min(turn(S));  u = S(index);  S = setdiff(S,u);  for v = find(adjMat(u,:))  for i = 1:length(TS{u,v})  temp = TS{u,v}(i);  while temp<t(u)+stopTime;  temp = temp + 24\*60;  end  if (turn(v)>turn(u)+1)||((turn(v)==turn(u)+1)&&(t(v)>temp+C{u,v}(i)))%(t(v)>temp+C{u,v}(i))  p(v) = u;  q(v) = i;  t(v) = temp + C{u,v}(i);  turn(v) = turn(u) + 1;  S = union(S,v);  end  end  end  if u == d  return  end  end  end |
| 第二问 |
| main2.m |
| travel = {'宜昌东','上海','南京','杭州','苏州','无锡'};  n = length(travel);  TimeFirstBestPath=cell(n,n);  TimeFirstCost = ones(n,n)\*inf;  TimeFirstTime = ones(n,n)\*inf;  type = 1;  for i=1:n  for j=1:n  if i==j  continue  end  [ TimeFirstBestPath{i,j},TimeFirstCost(i,j),TimeFirstTime(i,j) ] = oneBestPathFun( data,trainList,adjMat,station,travel{i},travel{j},type );  end  end  CostFirstBestPath=cell(n,n);  CostFirstCost = ones(n,n)\*inf;  CostFirstTime = ones(n,n)\*inf;  type = 2;  for i=1:n  for j=1:n  if i==j  continue  end  [ CostFirstBestPath{i,j},CostFirstCost(i,j),CostFirstTime(i,j) ] = oneBestPathFun( data,trainList,adjMat,station,travel{i},travel{j},type );  end  end |
| WENTI2.lg4 |
| MODEL:  SETS:  CITY / 1.. 6/: U; ! U( I) = sequence no. of city;  LINK( CITY, CITY):  DIST, ! The distance matrix;  X; ! X( I, J) = 1 if we use link I, J;  ENDSETS  DATA: !Distance matrix, it need not be symmetric;  DIST =  0 1064 767 1215 452 435  1016 0 99 90 25 42  774 99 0 228 71 55  1159 86 216 0 106 123  428 24 72 105 0 15  411 41 55 123 14 0;  ! 0 702 454 842 2396 1196  702 0 324 1093 2136 764  454 324 0 1137 2180 798  842 1093 1137 0 1616 1857  2396 2136 2180 1616 0 2900  1196 764 798 1857 2900 0;  ENDDATA  !The model:Ref. Desrochers & Laporte, OR Letters,  Feb. 91;  N = @SIZE( CITY);  MIN = @SUM( LINK: DIST \* X);  @FOR( CITY( K):  ! It must be entered;  @SUM( CITY( I)| I #NE# K: X( I, K)) = 1;  ! It must be departed;  @SUM( CITY( J)| J #NE# K: X( K, J)) = 1;  ! Weak form of the subtour breaking constraints;  ! These are not very powerful for large problems;  @FOR( CITY( J)| J #GT# 1 #AND# J #NE# K:  U( J) >= U( K) + X ( K, J) -  ( N - 2) \* ( 1 - X( K, J)) +  ( N - 3) \* X( J, K)  );  );  ! Make the X's 0/1;  @FOR( LINK: @BIN( X));  ! For the first and last stop we know...;  @FOR( CITY( K)| K #GT# 1:  U( K) <= N - 1 - ( N - 2) \* X( 1, K);  U( K) >= 1 + ( N - 2) \* X( K, 1)  );  END |