**“网络科学基础”-第三次上机报告**

班级： 物联网2303 姓名： 邱佳亮 学号： 3230611072

上机日期：2024.11.15，第十一周周五下七八节课

2024秋-网络科学基础（物联网23）-第三次上机报告提交

1. **上机题目**

Matlab 代码调用 Cpp 代码。

1. **上机目的**
2. 利用Cpp代码抽取Matlab矩阵的某些行、某些列上的元素，返回提取到 的矩阵数据到Matlab。
3. C++代码读取Matlab中的matrix类型数据。
4. C++代码读取struct类型数据。
5. 模仿3.1的例程，完成Matlab环境下调用C++代码（实现Dijkstra迪杰 斯特拉算法），计算网络的单源最短路径。
6. **功能描述、上机程序（含必要的注释）、上机调试运行结果**

### 利用 Cpp 代码抽取 Matlab 矩阵的某些行、某些列上的元素，返回提取到的矩阵数据到 Matlab：

新建一个 MtoCResizeArray.cpp 文件(MtoCResizeArray 将作为 MATLAB 环境中调用的函数名)，代码如下：

1. #include "mex.h"
2. *// 定义mexFunction函数，这是MEX文件中必须定义的函数，用于实现MATLAB和C/C++之间的接口*
3. void mexFunction(int nlhs, mxArray\* plhs[], int nrhs, const mxArray\* prhs[])
4. {
5. *// 检查输入参数的数量是否正确，应该有3个输入参数*
6. if (nrhs != 3)
7. {
8. mexErrMsgTxt("参数个数不正确!");
9. }
11. *// 获取第一个输入参数（矩阵）的行数和列数*
12. int rowNum = mxGetM(prhs[0]);
13. int colNum = mxGetN(prhs[0]);
15. *// 获取第一个输入参数（矩阵）的指针，用于访问矩阵元素*
16. double\* pArr = (double\*)mxGetPr(prhs[0]);
18. *// 获取选择的行和列的信息*
19. *// 无论是行向量还是列向量均支持*
20. double\* pSelRows = (double\*)mxGetPr(prhs[1]); *// 选择的行*
21. double\* pSelCols = (double\*)mxGetPr(prhs[2]); *// 选择的列*
23. *// 获取选择的行的维度信息*
24. int selRowsRowNum = mxGetM(prhs[1]);
25. int selRowsColNum = mxGetN(prhs[1]);
26. *// 检查选择的行是否为行向量或列向量*
27. if (selRowsRowNum != 1 && selRowsColNum != 1)
28. {
29. mexErrMsgTxt("行参数不正确！");
30. }
31. *// 计算选择的行的数量*
32. int selRowsNum = selRowsRowNum \* selRowsColNum;
34. *// 获取选择的列的维度信息*
35. int selColsRowNum = mxGetM(prhs[2]);
36. int selColsColNum = mxGetN(prhs[2]);
37. *// 检查选择的列是否为行向量或列向量*
38. if (selColsRowNum != 1 && selColsColNum != 1)
39. {
40. mexErrMsgTxt("列参数不正确！");
41. }
42. *// 计算选择的列的数量*
43. int selColsNum = selColsRowNum \* selColsColNum;
45. *// 创建输出参数1，用于返回选择的行和列的数量*
46. plhs[1] = mxCreateDoubleMatrix(2, 1, mxREAL);
47. double\* resizedDims = (double\*)mxGetPr(plhs[1]);
48. resizedDims[0] = selRowsNum; *// 选择的行数*
49. resizedDims[1] = selColsNum; *// 选择的列数*
51. *// 创建输出参数0，用于返回选择后的矩阵*
52. plhs[0] = mxCreateDoubleMatrix(selRowsNum, selColsNum, mxREAL);
53. double\* pResizedArr = (double\*)mxGetPr(plhs[0]);
55. *// 定义宏，用于简化数组访问，MATLAB中数据是按列优先存储的*
56. #define ARR(row,col) pArr[(col)\*rowNum+row]
57. #define RARR(row,col) pResizedArr[(col)\*selRowsNum+row]
59. *// 遍历选择的行和列，将原矩阵中对应的元素复制到新的矩阵中*
60. for (int ri = 0; ri < selRowsNum; ri++)
61. {
62. for (int ci = 0; ci < selColsNum; ci++)
63. {
64. RARR(ri, ci) = ARR((int)pSelRows[ri] - 1, (int)pSelCols[ci] - 1);
65. }
66. }
68. *// 打印信息，表示函数执行成功*
69. mexPrintf("OK!\n");
70. }

编译 C++函数为 MEX 函数：将 MtoCResizeArray.cpp 放在 MATLAB 当前目录中，在 MATLAB 中输入mex MtoCResizeArray.cpp，编译成功后将会生成 MtoCResizeArray.mexW32：

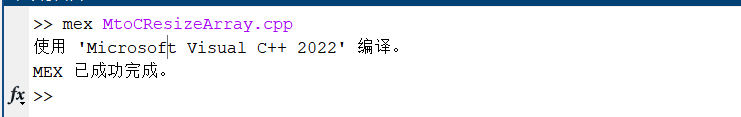


图 1 编译程序

在 Matlab 环境中调用函数实现预定的功能，理解观察程序运行：

1. arr=[11:19*;21:29;31:39;41:49;51:59;61:69]*
2. selRows=[1 3]*;*
3. selCols=[2:4 5 9]*;*
4. [rarr,rdims]=MtoCResizeArray(arr,selRows,selCols)

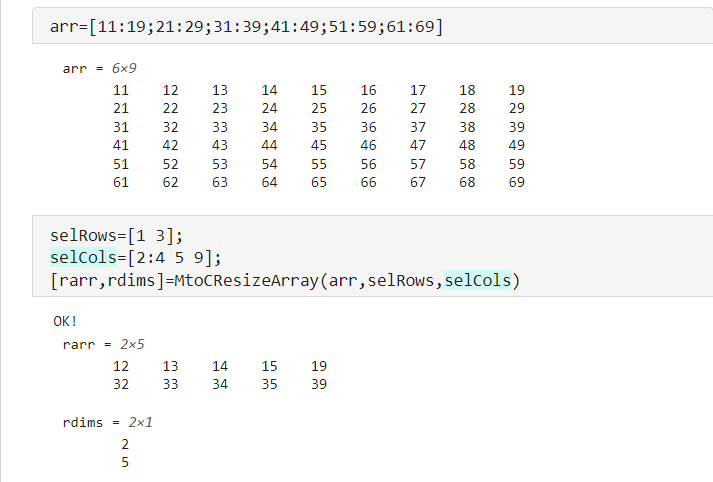


图 2 运行结果

C++代码读取 Matlab 中的 matrix 类型数据：利用 Matlab 产生一个 5\*3 矩阵，利用 c++读取其中位置为(row,col)的值，同时 c++向 Matlab传递一个新矩阵，代码如下：

1. #include "mex.h"
2. #include <cstdio> *// 包含标准输入输出库，用于printf函数*
3. *// 定义mexFunction函数，这是MEX文件中必须定义的函数，用于实现MATLAB和C/C++之间的接口*
4. void mexFunction(int nlhs, mxArray\* plhs[], int nrhs, mxArray\* prhs[]) {
5. *// 获取第一个输入参数（矩阵）的指针，用于访问矩阵元素*
6. double\* input = mxGetPr(prhs[0]);
8. *// 获取第一个输入参数（矩阵）的行数和列数*
9. size\_t m = mxGetM(prhs[0]); *// 行数*
10. size\_t n = mxGetN(prhs[0]); *// 列数*
12. *// 打印矩阵的行数和列数*
13. printf("the row of matrix is %d\n", m);
14. printf("the column of matrix is %d\n", n);
16. *// 获取第二个和第三个输入参数（行和列索引），这里假设它们是标量*
17. size\_t row = mxGetScalar(prhs[1]); *// 行索引*
18. size\_t col = mxGetScalar(prhs[2]); *// 列索引*
20. *// 打印指定位置的元素值*
21. *// 注意：MATLAB索引从1开始，C/C++索引从0开始，所以需要减1*
22. printf("the data of row %d column %d is: %f\n", row, col, \*(input + (col - 1) \* m + row - 1));
24. *// 创建一个与输入矩阵同样大小的输出矩阵*
25. plhs[0] = mxCreateDoubleMatrix((mwSize)m, (mwSize)n, mxREAL);
27. *// 获取输出矩阵的指针，用于设置矩阵元素*
28. double\* c = mxGetPr(plhs[0]);
30. *// 设置输出矩阵的特定元素值*
31. c[0] = 5; *// 设置第一个元素的值*
32. c[1] = 6; *// 设置第二个元素的值*
33. c[10] = 33; *// 设置第11个元素的值（索引从0开始）*
34. }

在 matlab 控制台运行 mex readMatrix.cpp：

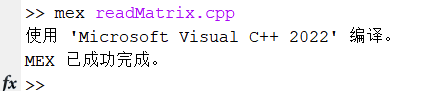


图 3 编译程序

在 Matlab 中的操作，理解观察程序运行：

1. A=rand(5,3)
2. row=4*;col=2;*
3. ans=readMatrix(A,row,col)

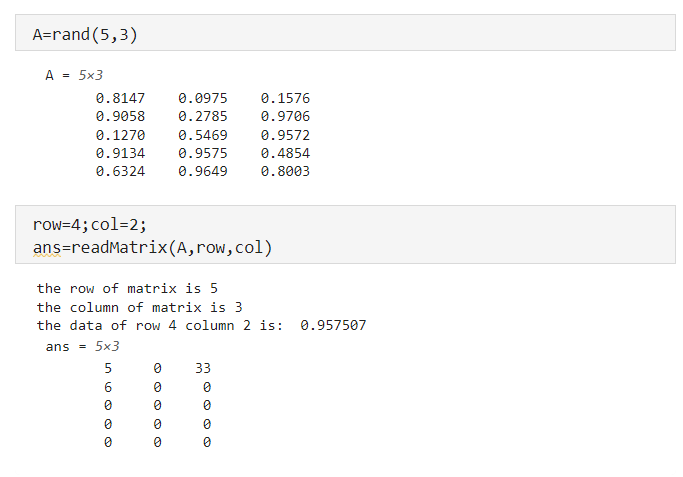


图 4 运行结果

### C++代码读取 struct 类型数据

利用 MATLAB 产生 struct 类型的数据。以下 stu 是一个向量（一维数组），每个元素是一个结构，该结构包括 3 个字段，分别是 name、list、id，其类型分别是 string、matrix、double，代码如下：

1. #include "mex.h"
2. #include <cstdio> *// 包含标准输入输出库，用于printf函数*
3. *// 定义mexFunction函数，这是MEX文件中必须定义的函数，用于实现MATLAB和C/C++之间的接口*
4. void mexFunction(int nlhs, mxArray\* plhs[], int nrhs, mxArray\* prhs[]) {
5. size\_t m, n;
6. *// 检查第一个输入参数是否为结构体数组*
7. if (mxIsStruct(prhs[0])) {
8. m = mxGetM(prhs[0]); *// 获取结构体数组的行数，即结构体的数量*
9. n = mxGetN(prhs[0]); *// 获取结构体数组的列数，这里应该是1*
10. size\_t numStruct = m \* n; *// 计算结构体的总数*
11. printf("the total number of struct is %d\n", numStruct);
13. *// 遍历每个结构体*
14. for (size\_t i = 0; i < numStruct; i++) {
15. printf("===the info of the %d student====\n", i + 1);
17. *// 读取结构体中的标量字段"id"*
18. mxArray\* ID = mxGetField(prhs[0], i, "id"); *// 获取字段"id"的mxArray指针*
19. if (ID) {
20. double id = mxGetScalar(ID); *// 获取标量值*
21. printf("id=%f\n", id);
22. }
24. *// 读取结构体中的矩阵字段"list"*
25. printf("matrix=\n");
26. mxArray\* tmpMatrix = mxGetField(prhs[0], i, "list"); *// 获取字段"list"的mxArray指针*
27. if (tmpMatrix) {
28. double\* tmpmatrix = mxGetPr(tmpMatrix); *// 获取矩阵的指针*
29. m = mxGetM(tmpMatrix); *// 获取矩阵的行数*
30. n = mxGetN(tmpMatrix); *// 获取矩阵的列数*
31. *// 遍历矩阵并打印每个元素*
32. for (size\_t i = 0; i < m; i++) {
33. for (size\_t j = 0; j < n; j++) {
34. printf("%f ", \*(tmpmatrix + j \* m + i));
35. }
36. printf("\n");
37. }
38. }
40. *// 读取结构体中的字符串字段"name"*
41. mxArray\* tmpString = mxGetField(prhs[0], i, "name"); *// 获取字段"name"的mxArray指针*
42. if (tmpString) {
43. char\* tmp = mxArrayToString(tmpString); *// 将mxArray转换为C字符串*
44. if (tmp)
45. printf("name=%s\n", tmp);
46. *// 释放由mxArrayToString分配的内存*
47. mxFree(tmp);
48. }
49. }
50. }
51. }

先在 matlab 控制台运行 mex readStruct.cpp：

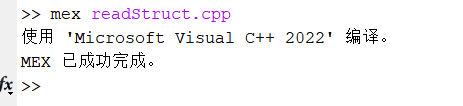


图 5 编译程序

在 Matlab 中的操作，观察、理解程序运行：

1. stu(1).name='aa';
2. stu(2).name='bb';
3. stu(3).name='cc';
4. for i=1:3
5. stu(i).list=rand(5,4);
6. stu(i).id=randi(100);
7. end
8. readStruct(stu);

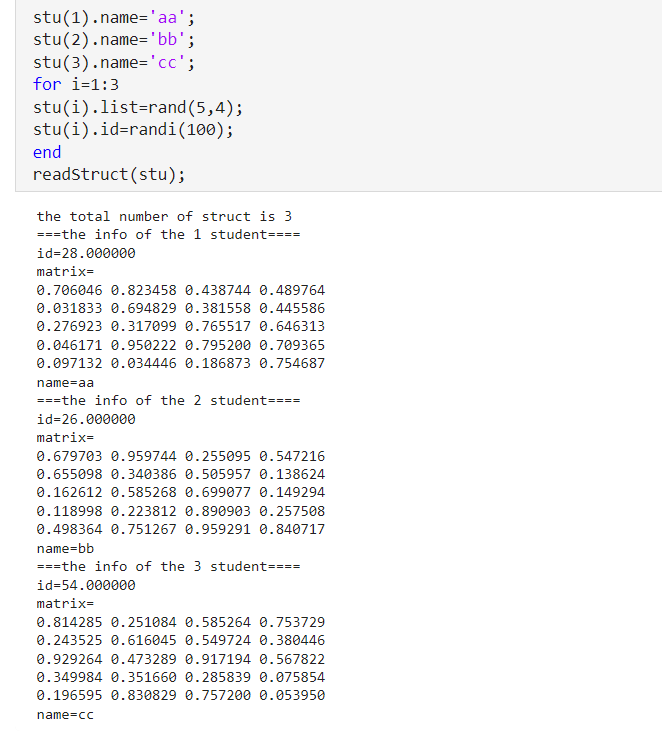


图 6 运行结果

### 模仿 3.1 的例程，完成 Matlab 环境下调用 C++代码（实现Dijkstra 迪杰斯特拉算法），计算网络的单源最短路径

算法思想：

设 G=(V,E)是一个带权有向图，把图中顶点集合 V 分为两组，第一组为已求出最短路径的顶点集合（用 S 表示，初始时 S 中只有一个源点，以后每求得一条最短路径 , 就将加入到集合 S 中，直到全部顶点都加入到 S 中，算法就结束了）,第二组为其余未确定最短路径的顶点集合（用 U 表示）,按最短路径的的递增次序依次把第二组中的顶点加入 S 中。在加入的过程中，总保持从源点 v 到 S 中各个顶点的最短路径长度不大于从源点 v 到 U 中任何路径的长度。此外，每个顶点对应一个距离，S 中的顶点的距离就是从 v 到此顶点的最短路径长度，U中的顶点的距离，是从 v 到此顶点只包括 S 中的顶点为中间顶点的当前路径的最短长度。

算法步骤：

a.初始时，只包括源点，即 S = {v}，v 的距离为 0。U 包含除 v 以外的其他顶点，即：U ={其余顶点}，若 v 与 U 中顶点 u 有边，则(u,v)为正常权值，若 u 不是 v 的出边邻接点,则(u,v)权值 ∞;

b.从 U 中选取一个距离 v 最小的顶点 k，把 k，加入 S 中（该选定的距离就是 v 到 k 的最短路径长度）。

c.以 k 为新考虑的中间点，修改 U 中各顶点的距离；若从源点 v 到顶点 u 的距离（经过顶点 k）比原来距离（不经过顶点 k）短，则修改顶点 u 的距离值，修改后的距离值的顶点 k的距离加上边上的权。

d.重复步骤 b 和 c 直到所有顶点都包含在 S 中。

代码如下：

1. #include "mex.h"
2. #include <vector>
3. #include <limits>
4. #include <cmath>
5. const int INF = 32767;
6. const int MAXN = 10;
7. *// Dijkstra算法的实现*
8. void Dijkstra(int v, const double A[MAXN][MAXN], double dis[MAXN], int path[MAXN]) {
9. bool S[MAXN] = { false };
10. int n = MAXN;
11. for (int i = 0; i < n; ++i) {
12. dis[i] = A[v][i];
13. if (dis[i] == INF) dis[i] = NAN; *// 如果不可达，设置为NaN*
14. path[i] = -1;
15. }
16. dis[v] = 0;
17. S[v] = true;
18. for (int i = 0; i < n; ++i) {
19. int u = -1;
20. double minDist = INF;
21. *// 寻找不在S中且距离最小的顶点u*
22. for (int j = 0; j < n; ++j) {
23. if (!S[j] && (u == -1 || dis[j] < minDist)) {
24. u = j;
25. minDist = dis[j];
26. }
27. }
28. if (u == -1) break; *// 剩余顶点都不可达*
29. S[u] = true;
30. *// 更新顶点u相邻的顶点的距离*
31. for (int j = 0; j < n; ++j) {
32. if (!S[j] && A[u][j] != INF && dis[u] + A[u][j] < dis[j]) {
33. dis[j] = dis[u] + A[u][j];
34. path[j] = u;
35. }
36. }
37. }
38. }
39. *// MEX文件的入口点*
40. void mexFunction(int nlhs, mxArray\* plhs[], int nrhs, const mxArray\* prhs[]) {
41. *// 获取输入参数*
42. int src = static\_cast<int>(mxGetScalar(prhs[0]));
43. double\* graphPtr = mxGetPr(prhs[1]);
44. *// 创建输出参数*
45. plhs[0] = mxCreateDoubleMatrix(MAXN, 1, mxREAL);
46. plhs[1] = mxCreateDoubleMatrix(MAXN, 1, mxREAL);
47. *// 将输入图转换为C++数组*
48. double A[MAXN][MAXN];
49. for (int i = 0; i < MAXN; ++i) {
50. for (int j = 0; j < MAXN; ++j) {
51. A[i][j] = graphPtr[i + j \* MAXN];
52. }
53. }
54. *// 调用Dijkstra算法*
55. double\* disPtr = mxGetPr(plhs[0]);
56. int\* pathPtr = reinterpret\_cast<int\*>(mxGetPr(plhs[1])); *// 使用reinterpret\_cast进行类型转换*
57. double dis[MAXN];
58. int path[MAXN];
59. Dijkstra(src, A, dis, path);
60. *// 将结果复制回Matlab矩阵*
61. for (int i = 0; i < MAXN; ++i) {
62. disPtr[i] = dis[i];
63. if (path[i] == -1) {
64. pathPtr[i] = 0; *// 如果没有路径，则设置为0*
65. }
66. else {
67. pathPtr[i] = path[i] + 1; *// MATLAB索引从1开始*
68. }
69. }
70. }

在 matlab 控制台运行 mex dijkstra.cpp：

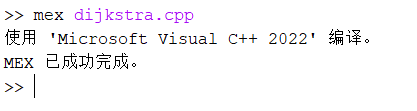


图 7 编译程序

在 Matlab 中的操作，观察、理解程序运行：

1. % 创建一个邻接矩阵，其中INF表示不相连
2. A = [0, 4, inf, inf, inf, inf, inf, 8, inf;
3. inf, 0, 8, inf, inf, inf, inf, 11, inf;
4. inf, 8, 0, 7, inf, 4, inf, inf, 2;
5. inf, inf, 7, 0, 9, 14, inf, inf, inf;
6. inf, inf, inf, 9, 0, 10, inf, inf, inf;
7. inf, inf, 4, 14, 10, 0, 2, inf, inf;
8. inf, inf, inf, inf, inf, 2, 0, 1, 6;
9. 8, 11, inf, inf, inf, inf, 1, 0, 7;
10. inf, inf, 2, inf, inf, inf, 6, 7, 0];
11. % 调用MEX函数，传入邻接矩阵和源点
12. [distances, paths] = Dijkstra(1, A);
13. % 显示结果
14. disp(distances);

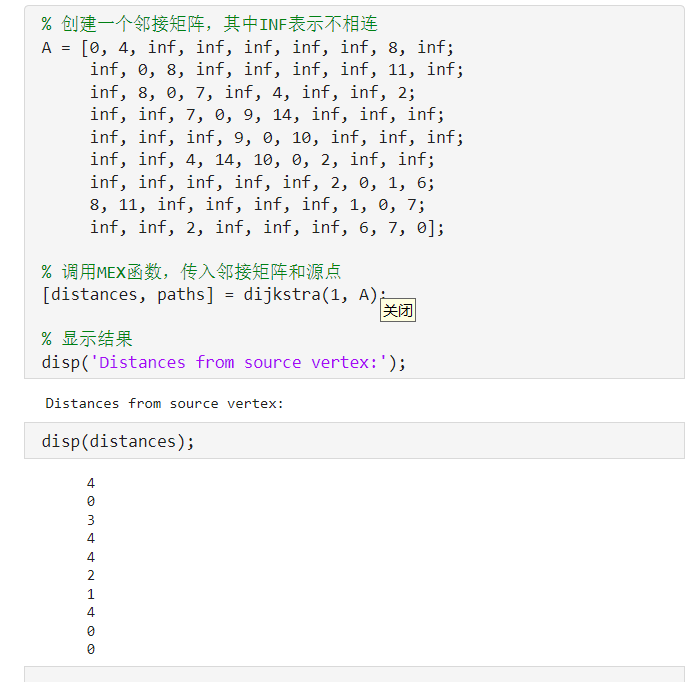


图 8 运行结果

1. **上机总结及感想**

在本次网络科学基础第三次上机实验中，我深入学习了Matlab与C++代码的交互，包括如何在C++中处理Matlab矩阵和结构体数据，并将结果反馈给Matlab。通过编写和调用MEX函数，我不仅提升了编程技能，还加深了对数据结构和算法的理解。特别是在实现Dijkstra迪杰斯特拉算法计算网络单源最短路径的过程中，我对图论和网络分析有了更深刻的认识。这次实验不仅增强了我的问题解决能力，也激发了我对网络科学和编程深入学习的热情，我期待将这些知识应用到未来的研究和项目中。