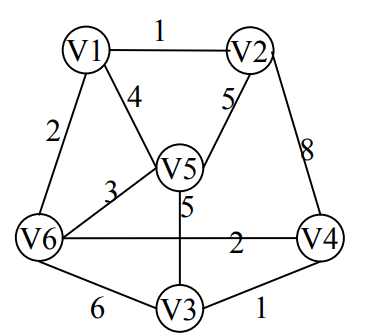
**Project #1: (programming) –** when we finish the dynamic programming discussion, please turn in your program (for project-1) within 1~2 weeks. Please finish grouping. Certainly it is fine to be by yourself. A group just needs to turn in one result/report. However, this one may be a little simpler, so welcome to exercise it by yourself even for a joint work.

**1. Dynamic Programming**:

Please use C++/C or python to write a dynamic programming program for travelling salesman problem (TSP) for Problem-6 in HW-1 (– case1)



For input format, we need to enter a graph. Let’s use the following simple description:

# comment (if line starts with “#”, then it is a comment)

# edgeIndex edgeWeight nameOfVertexU nameOfVertexV

e1 1 v1 v2

e2 4 v1 v5

e3 5 v2 v5

e4 2 v1 v6

e5 3 v6 v5

e6 2 v6 v4

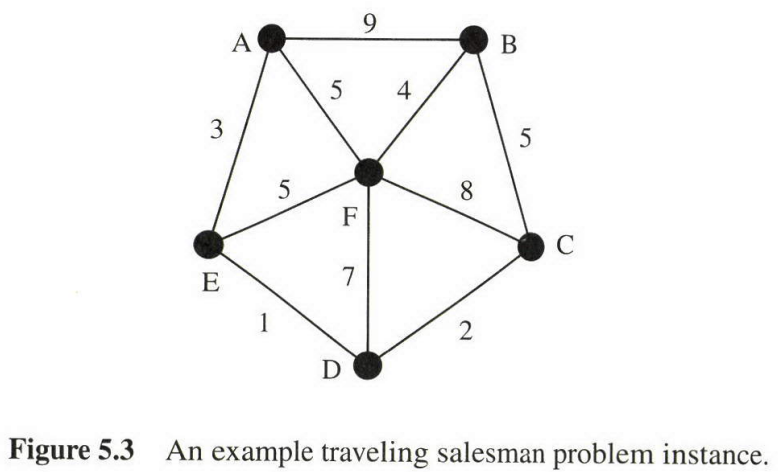
e7 6 v6 v3

e8 1 v3 v4

e9 5 v5 v3

e10 8 v4 v2

Welcome to run your program using the graph shown in ref. book page 58-64 (This dynamic programming approach for TSP will now be illustrated for the example graph of Figure 5.3 where the choice vs = A has been made) Please see page-64 of the book (pdf page-76).) And welcome to create your input file (– case2)



**About program submission**, please turn in

1. “Readme” file, describing how to build C/C++ and how to run your programs.
   1. For python, please tell us the version of Python you are using. (Our laptop, my CentOS-7 may not have the same version. Try to use version 3.x and above)
2. Source codes, makefile. (then, TA and I can try to compile them.)
   1. for python, we will run your codes. If some utility libraries are needed, please indicate them in your readme file.
3. Executable binaries (for C++/C) with naming, such as “studentID\_name\_proj1.”
4. Input data (please see the input format defined above) and output data,
   1. There are two graphs to be run.
5. Output data is the ordering of cities for TSP and total distance.
6. A brief report about your data structures 🡪 this will help you to understand more.

Thanks.

I found these public domain programs. Welcome to use and modify them.

<https://gist.github.com/jgcoded/d7ecba7aa3e210419471>

<https://github.com/kristiansandratama/travelling-salesman-problem-dynamic-programming>

<https://github.com/topics/travelling-salesman-problem?l=c%2B%2B>

<https://github.com/beckysag/traveling-salesman>

Another 3 test cases for dynamic programming 🡪 TA and I will run your program on these 3 cases; then, check the results.

Case-3:

# comment (if line starts with “#”, then it is a comment)

# edgeIndex edgeWeight nameOfVertexU nameOfVertexV

e1 1 v1 v2

e2 4 v1 v5

e3 5 v2 v5

e4 2 v1 v6

e5 3 v6 v5

e6 2 v6 v4

e7 6 v6 v3

e8 1 v3 v4

e9 5 v5 v3

e10 8 v4 v2

e11 3 v7 v8

e12 4 v6 v7

e13 2 v7 v9

e14 1 v7 v1

e15 2 v7 v5

e16 10 v8 v1

e17 4 v8 v3

e18 2 v8 v6

e20 3 v9 v1

e21 4 v8 v2

e22 2 v9 v3

e23 8 v9 v5

Case-4:

# comment (if line starts with “#”, then it is a comment)

# edgeIndex edgeWeight nameOfVertexU nameOfVertexV

e1 1 v1 v2

e2 4 v1 v5

e3 5 v2 v5

e4 2 v1 v6

e5 3 v6 v5

e6 2 v6 v4

e7 6 v6 v3

e8 1 v3 v4

e9 5 v5 v3

e10 8 v4 v2

e11 3 v7 v8

e12 4 v6 v7

e13 2 v7 v9

e14 1 v7 v1

e15 2 v7 v5

e16 10 v8 v1

e17 4 v8 v3

e18 2 v8 v6

e20 3 v9 v1

e21 4 v8 v2

e22 2 v9 v3

e23 8 v9 v5

e24 1 v10 v1

e25 2 v10 v3

e26 13 v10 v6

e27 3 v10 v5

e28 4 v10 v8

e29 3 v11 v2

e30 4 v11 v4

e31 5 v11 v6

e32 6 v11 v8

e33 1 v7 v2

Case-5:

# comment (if line starts with “#”, then it is a comment)

# edgeIndex edgeWeight nameOfVertexU nameOfVertexV

e1 1 v1 v2

e2 4 v1 v5

e3 5 v2 v5

e4 2 v1 v6

e5 3 v6 v5

e6 2 v6 v4

e7 6 v6 v3

e8 1 v3 v4

e9 5 v5 v3

e10 8 v4 v2

e11 3 v7 v8

e12 4 v6 v7

e13 2 v7 v9

e14 1 v7 v1

e15 2 v7 v5

e16 10 v8 v1

e17 4 v8 v3

e18 2 v8 v6

e20 3 v9 v1

e21 4 v8 v2

e22 2 v9 v3

e23 8 v9 v5

e24 1 v10 v1

e25 2 v10 v3

e26 13 v10 v6

e27 3 v10 v5

e28 4 v10 v8

e29 3 v11 v3

e30 4 v11 v4

e31 5 v11 v9

e32 6 v11 v8

e33 1 v7 v2

e34 4 v12 v1

e35 3 v12 v2

e36 10 v12 v11

e37 7 v12 v9

e38 3 v12 v8

e39 1 v12 v5

e40 2 v12 v7

e41 9 v7 v3