CSCI 255: Lab #9 Graph Shortest Path

Darwin Jacob Groskleg

Tuesday, November 12th, 2019

Contents

Contents i

Console Output 1

main.cpp 2

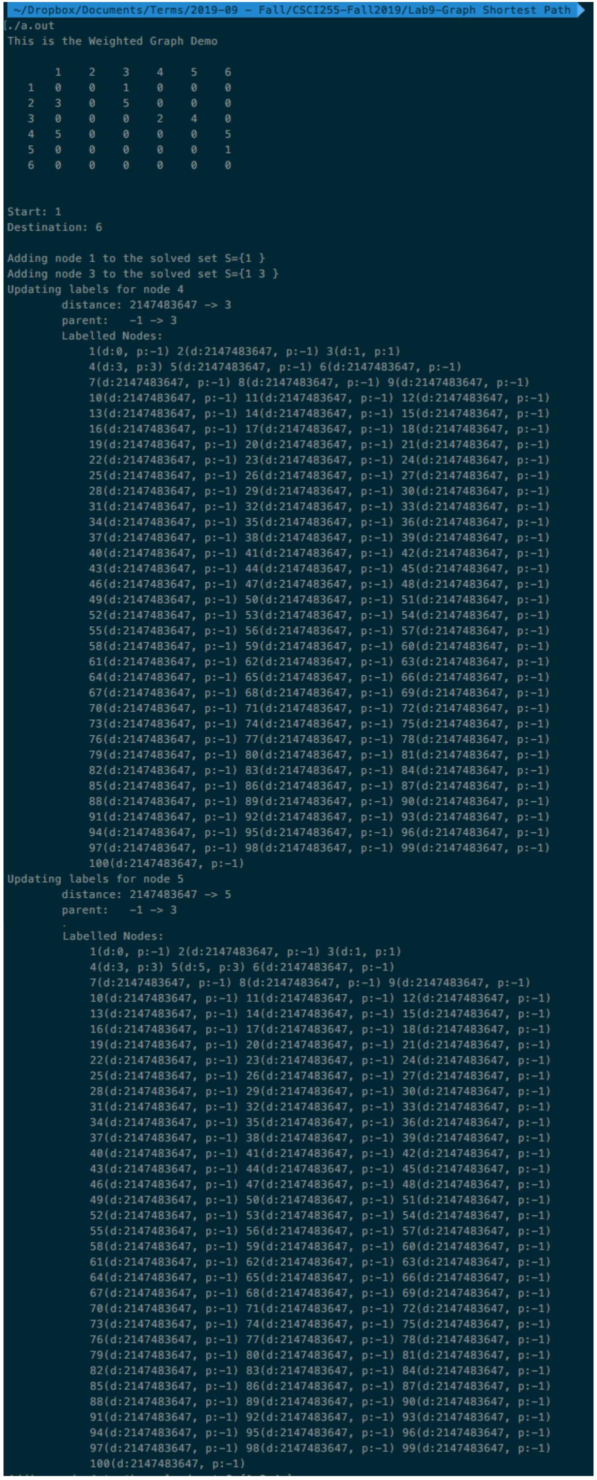
graph.hpp 6

graph.cpp 7

Console Output

Compiles and Runs with Input: 1 6

See main.cpp for easier to read output.

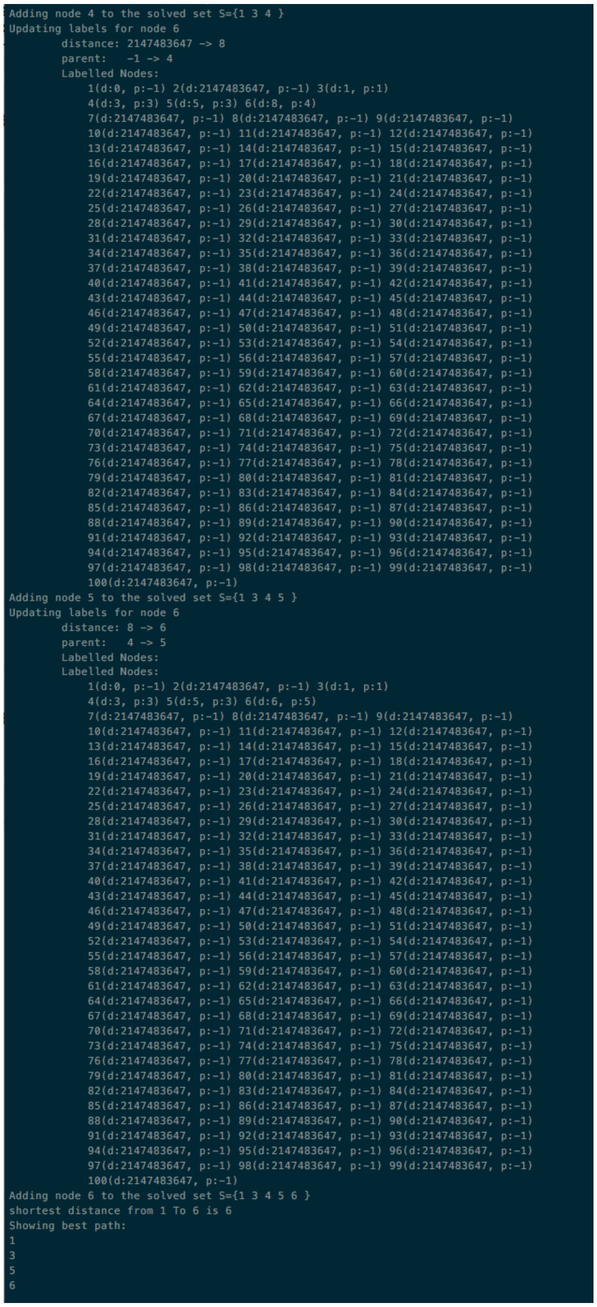


```
 ~/Dropbox/Documents/Terms/2019-09 - Fall/CSCI255-Fall2019/Lab9-Graph Shortest Path
[./a.out
This is the Weighted Graph Demo

       1    2    3    4    5    6
   1   0    0    1    0    0    0
   2   3    0    5    0    0    0
   3   0    0    0    2    4    0
   4   5    0    0    0    0    5
   5   0    0    0    0    0    1
   6   0    0    0    0    0    0


Start: 1
Destination: 6

Adding node 1 to the solved set S={1 }
Adding node 3 to the solved set S={1 3 }
Updating labels for node 4
        distance: 2147483647 -> 3
        parent:   -1 -> 3
        Labelled Nodes:
            1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
            4(d:3, p:3) 5(d:2147483647, p:-1) 6(d:2147483647, p:-1)
            7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
            10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
            13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
            16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
            19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
            22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
            25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
            28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
            31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
            34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
            37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
            40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
            43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
            46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
            49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
            52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
            55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
            58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
            61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
            64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
            67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
            70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
            73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
            76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
            79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
            82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
            85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
            88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
            91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
            94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
            97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
            100(d:2147483647, p:-1)
Updating labels for node 5
        distance: 2147483647 -> 5
        parent:   -1 -> 3
        .
        Labelled Nodes:
            1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
            4(d:3, p:3) 5(d:5, p:3) 6(d:2147483647, p:-1)
            7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
            10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
            13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
            16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
            19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
            22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
            25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
            28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
            31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
            34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
            37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
            40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
            43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
            46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
            49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
            52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
            55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
            58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
            61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
            64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
            67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
            70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
            73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
            76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
            79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
            82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
            85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
            88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
            91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
            94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
            97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
            100(d:2147483647, p:-1)
```



```
Adding node 4 to the solved set S={1 3 4 }
Updating labels for node 6
        distance: 2147483647 -> 8
        parent:   -1 -> 4
        Labelled Nodes:
            1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
            4(d:3, p:3) 5(d:5, p:3) 6(d:8, p:4)
            7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
            10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
            13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
            16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
            19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
            22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
            25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
            28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
            31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
            34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
            37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
            40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
            43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
            46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
            49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
            52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
            55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
            58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
            61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
            64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
            67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
            70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
            73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
            76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
            79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
            82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
            85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
            88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
            91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
            94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
            97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
            100(d:2147483647, p:-1)
Adding node 5 to the solved set S={1 3 4 5 }
Updating labels for node 6
        distance: 8 -> 6
        parent:   4 -> 5
        Labelled Nodes:
        Labelled Nodes:
            1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
            4(d:3, p:3) 5(d:5, p:3) 6(d:6, p:5)
            7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
            10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
            13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
            16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
            19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
            22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
            25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
            28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
            31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
            34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
            37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
            40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
            43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
            46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
            49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
            52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
            55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
            58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
            61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
            64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
            67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
            70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
            73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
            76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
            79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
            82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
            85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
            88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
            91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
            94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
            97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
            100(d:2147483647, p:-1)
Adding node 6 to the solved set S={1 3 4 5 6 }
shortest distance from 1 To 6 is 6
Showing best path:
1
3
5
6
```

main.cpp

```
/* main.cpp (TestGraph.cpp)
 * ------------------------
 * Authors: Darwin Jacob Groskleg, Man Lin
 *
 * Purpose: to rewrite the Best_Path method on Weighted_Graph to show the
 *          detailed steps of the shortest path algorithm.
 *
 * CONSOLE SAMPLE
--------------------------------------------------------------------------------
This is the Weighted Graph Demo

       1    2    3    4    5    6
   1   0    0    1    0    0    0
   2   3    0    5    0    0    0
   3   0    0    0    2    4    0
   4   5    0    0    0    0    5
   5   0    0    0    0    0    1
   6   0    0    0    0    0    0


Start: 1
Destination: 6
Adding node 1 to the solved set S={1 }
Adding node 3 to the solved set S={1 3 }
Updating labels for node 4
    distance: 2147483647 -> 3
    parent:   -1 -> 3
    Node labels:
        1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
        4(d:3, p:3) 5(d:2147483647, p:-1) 6(d:2147483647, p:-1)
        7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
        10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
        13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
        16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
        19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
        22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
        25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
        28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
        31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
        34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
        37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
        40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
        43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
        46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
        49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
        52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
        55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
        58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
        61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
        64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
        67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
        70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
        73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
        76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
        79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
        82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
        85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
        88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
        91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
        94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
        97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
        100(d:2147483647, p:-1)
Updating labels for node 5
    distance: 2147483647 -> 5
    parent:   -1 -> 3
    Node labels:
        1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
        4(d:3, p:3) 5(d:5, p:3) 6(d:2147483647, p:-1)
        7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
        10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
        13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
        16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
        19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
        22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
        25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
        28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
        31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
        34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
        37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
        40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
        43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
        46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
        49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
        52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
        55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
        58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
        61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
        64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
        67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
        70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
        73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
        76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
        79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
        82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
        85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
        88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
        91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
        94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
        97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
        100(d:2147483647, p:-1)
Adding node 4 to the solved set S={1 3 4 }
Updating labels for node 6
    distance: 2147483647 -> 8
    parent:   -1 -> 4
    Node labels:
        1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
        4(d:3, p:3) 5(d:5, p:3) 6(d:8, p:4)
        7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
        10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
        13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
        16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
        19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
        22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
        25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
        28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
        31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
        34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
        37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
        40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
        43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
        46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
        49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
        52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
        55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
        58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
        61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
        64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
        67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
        70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
        73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
        76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
        79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
        82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
        85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
        88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
        91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
        94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
        97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
        100(d:2147483647, p:-1)
Adding node 5 to the solved set S={1 3 4 5 }
Updating labels for node 6
    distance: 8 -> 6
    parent:   4 -> 5
    Node labels:
        1(d:0, p:-1) 2(d:2147483647, p:-1) 3(d:1, p:1)
        4(d:3, p:3) 5(d:5, p:3) 6(d:6, p:5)
        7(d:2147483647, p:-1) 8(d:2147483647, p:-1) 9(d:2147483647, p:-1)
        10(d:2147483647, p:-1) 11(d:2147483647, p:-1) 12(d:2147483647, p:-1)
        13(d:2147483647, p:-1) 14(d:2147483647, p:-1) 15(d:2147483647, p:-1)
        16(d:2147483647, p:-1) 17(d:2147483647, p:-1) 18(d:2147483647, p:-1)
        19(d:2147483647, p:-1) 20(d:2147483647, p:-1) 21(d:2147483647, p:-1)
        22(d:2147483647, p:-1) 23(d:2147483647, p:-1) 24(d:2147483647, p:-1)
        25(d:2147483647, p:-1) 26(d:2147483647, p:-1) 27(d:2147483647, p:-1)
        28(d:2147483647, p:-1) 29(d:2147483647, p:-1) 30(d:2147483647, p:-1)
        31(d:2147483647, p:-1) 32(d:2147483647, p:-1) 33(d:2147483647, p:-1)
        34(d:2147483647, p:-1) 35(d:2147483647, p:-1) 36(d:2147483647, p:-1)
        37(d:2147483647, p:-1) 38(d:2147483647, p:-1) 39(d:2147483647, p:-1)
        40(d:2147483647, p:-1) 41(d:2147483647, p:-1) 42(d:2147483647, p:-1)
        43(d:2147483647, p:-1) 44(d:2147483647, p:-1) 45(d:2147483647, p:-1)
        46(d:2147483647, p:-1) 47(d:2147483647, p:-1) 48(d:2147483647, p:-1)
        49(d:2147483647, p:-1) 50(d:2147483647, p:-1) 51(d:2147483647, p:-1)
        52(d:2147483647, p:-1) 53(d:2147483647, p:-1) 54(d:2147483647, p:-1)
        55(d:2147483647, p:-1) 56(d:2147483647, p:-1) 57(d:2147483647, p:-1)
        58(d:2147483647, p:-1) 59(d:2147483647, p:-1) 60(d:2147483647, p:-1)
        61(d:2147483647, p:-1) 62(d:2147483647, p:-1) 63(d:2147483647, p:-1)
        64(d:2147483647, p:-1) 65(d:2147483647, p:-1) 66(d:2147483647, p:-1)
```

```cpp
        67(d:2147483647, p:-1) 68(d:2147483647, p:-1) 69(d:2147483647, p:-1)
        70(d:2147483647, p:-1) 71(d:2147483647, p:-1) 72(d:2147483647, p:-1)
        73(d:2147483647, p:-1) 74(d:2147483647, p:-1) 75(d:2147483647, p:-1)
        76(d:2147483647, p:-1) 77(d:2147483647, p:-1) 78(d:2147483647, p:-1)
        79(d:2147483647, p:-1) 80(d:2147483647, p:-1) 81(d:2147483647, p:-1)
        82(d:2147483647, p:-1) 83(d:2147483647, p:-1) 84(d:2147483647, p:-1)
        85(d:2147483647, p:-1) 86(d:2147483647, p:-1) 87(d:2147483647, p:-1)
        88(d:2147483647, p:-1) 89(d:2147483647, p:-1) 90(d:2147483647, p:-1)
        91(d:2147483647, p:-1) 92(d:2147483647, p:-1) 93(d:2147483647, p:-1)
        94(d:2147483647, p:-1) 95(d:2147483647, p:-1) 96(d:2147483647, p:-1)
        97(d:2147483647, p:-1) 98(d:2147483647, p:-1) 99(d:2147483647, p:-1)
        100(d:2147483647, p:-1)
Adding node 6 to the solved set S={1 3 4 5 6 }
shortest distance from 1 To 6 is 6
Showing best path:
1
3
5
6

--------------------------------------------------------------------------------
 * END OF CONSOLE SAMPLE
 */
#include <iostream>
#include "graph.hpp"

using namespace std;

int main (void) {
    Weighted_Graph graph(6); // SAY NO TO GLOBALS!

    cerr << "This is the Weighted Graph Demo\n\n";

    graph.Add_Edge(2, 1, 3);
    graph.Add_Edge(2, 3, 5);
    graph.Add_Edge(3, 4, 2);
    graph.Add_Edge(4, 1, 5);
    graph.Add_Edge(1, 3, 1);
    graph.Add_Edge(3, 5, 4);
    graph.Add_Edge(5, 6, 1);
    graph.Add_Edge(4, 6, 5);

    graph.Display();

    int start;
    int dest;

    clog << "\n\n" << "Start: ";
    cin >> start;
    clog << "Destination: ";
    cin >> dest;
    cout << '\n';
    graph.Show_Best_Path(start, dest);

    return 0;
}
```

graph.hpp

```cpp
/* graph.hpp (Graph.h)
 * --------------------
 * Authors: Darwin Jacob Groskleg, Man Lin
 */
#ifndef GRAPH_HPP_INCLUDED
#define GRAPH_HPP_INCLUDED

#include <deque>

#ifndef MAX_NODES
#define MAX_NODES 100
#endif

//modify from Turner's Weighted_Graph_demo
class Weighted_Graph
{
  public:
    Weighted_Graph(int node_count);

    //add an edge to the graph
    void Add_Edge(const int Node_1, const int Node_2, int edge_weight);

    // Print the ajacency matrix to stdout
    void Display() const;

    //find the best path from start to dest
    auto Best_Path(const int Start, const int Dest, bool verbose=false) const
        -> std::deque<int>;

    // Prints the shortest path between two nodes to stderr/console
    void Show_Best_Path(int Start, int Dest) const;

  private:
    int number_of_nodes; //the number of vertexes in the graph
    // The Node ID 0 is not used.  The first real node has ID 1
    int weight[MAX_NODES+1][MAX_NODES+1];   //store the weight of the edges

    bool validVertex(int node) const;
};

#endif // GRAPH_HPP_INCLUDED
```

graph.cpp

```cpp
/* graph.cpp
 * ---------
 * Authors: Darwin Jacob Groskleg
 *
 * IMPLEMENTATION of Weighted_Graph
 */
#include ”graph.hpp”

#include <iostream>
#include <iomanip>
#include <algorithm>

#include <climits> // INT_MAX

using namespace std;

// parameterized constructor
    Weighted_Graph::
Weighted_Graph(int node_count) :
    number_of_nodes(node_count)
{
    if (node_count > MAX_NODES)
        throw ”Fail to Construct: node_count exceeds MAX_NODES!”;
}

//add an edge to the graph, update the vertex and edge structure
    void Weighted_Graph::
Add_Edge(const int Node_1, const int Node_2, int edge_weight)
{
    if (!validVertex(Node_1) || !validVertex(Node_2)) throw ”Invalid node!”;
    weight[Node_1][Node_2] = edge_weight;
}

//display the weight between the vertexes
    void Weighted_Graph::
Display() const
{
    cout << setw(4) << ” ”;
    for (int i = 1; i <= number_of_nodes; ++i) {
        cout << setw(4) << i << ” ”;
    }
    cout << '\n';

    for (int i = 1; i <= number_of_nodes; ++i) {
        cout << setw(4) << i;
        for (int j = 1; j <= number_of_nodes; ++j) {
            cout << setw(4) << weight[i][j] << ” ”;
        }
        cout << '\n';
    }
}

// Find Shortest path between Start and Dest
//  Approach: use Dijkstra's algorithm, greedy
```

```
    deque<int> Weighted_Graph::
Best_Path(const int Start_ID, const int Dest_ID, bool verbose) const
{
    deque<int> best_path; //a queue to store the best path
    if (!(validVertex(Start_ID) && validVertex(Dest_ID))) {
        cout << ”Invalid start or destination\n”;
        return best_path;  // empty deque
    }

    //an array storing the distance label for each vertex
    // distance from start node
    int distance[MAX_NODES+1];
    //an array storing the parent label for each vertex
    int parent[MAX_NODES+1];
    //a array indicating whether the vertex is already in the solved set
    bool is_solved[MAX_NODES+1];

    for (int i = 1; i <= MAX_NODES; ++i) {
        // if node i is connected to Start node
        if (weight[Start_ID][i] > 0) {
            distance[i] = weight[Start_ID][i];
            parent[i] = Start_ID;
        }
        else {
            distance[i] = INT_MAX;
            parent[i] = -1;
        }
        // want to initialize the array
        is_solved[i] = false;
    }

    // HELPER LAMBDAS FOR LOGGED OPERATIONS
    auto addToSolved = [&] (int node) {
        // Operation 1
        is_solved[node] = true;
        if (verbose) {
            clog << ”Adding node ” << node << ” to the solved set S={”;
            // Operation 2: show solved set S
            //  nodes are false if not solved
            for (int x=1; x<=MAX_NODES; x++)
                if (is_solved[x]) clog << x << ' ';
            clog << ”}\n”;
        }
    };
    auto updateDistanceLabel = [&] (int parent_node, int node) {
        // Operation 3: show nodes whose labels are updated and the
        //              corresponding updated label (distance and parent).
        auto new_distance = distance[parent_node] + weight[parent_node][node];
        if (verbose)
            clog << ”Updating labels for node ” << node
                << ”\n\tdistance: ” << distance[node] << ” -> ” << new_distance
                << ”\n\tparent:   ” << parent[node]   << ” -> ” << parent_node
                << '\n';
        distance[node] = new_distance;
        parent[node] = parent_node;
        // Operation 4: show labels of all the nodes
        if (verbose) {
            clog << ”\tLabelled Nodes: ”;
            for (int x=1; x<=MAX_NODES; x++) {
                if (x%3 == 1) clog << ”\n\t    ”;
                clog << x << ”(d:” << distance[x] << ”, p:” << parent[x]<< ”) ”;
            }
            clog << '\n';
        }
    };

    // THE ALGORITHM
    distance[Start_ID] = 0;
    addToSolved(Start_ID);

    while (!is_solved[Dest_ID]) {
        // Determine the node with least distance among
        // all nodes whose best distance is not yet known.
        int min_best_dist = INT_MAX;
        int best_node_id = -1;
        // find the node that is not in the solved set and has the minimal
        // distance
        for (int i = 1; i <= number_of_nodes; ++i) {
            // best distance so far?
            if (!is_solved[i] && distance[i] < min_best_dist) {
                min_best_dist = distance[i];
                best_node_id = i;
            }
        }

        if (best_node_id == -1) {
            // Destination is unreachable.
            cerr << Dest_ID << ” is unreachable from ” << Start_ID << '\n';
            return best_path;  // empty deque
        }

        // Best total distance so far for this node is the actual
        // best total distance.
        int v = best_node_id;
        addToSolved(v);

        // if applicable, update the label of the neighbours of the active node
        // (that is, the distance and parent)
        for (int i = 1; i <= number_of_nodes; ++i) {
            if (!is_solved[i]
                && weight[v][i] > 0
                && (distance[v] + weight[v][i]) < distance[i]
            ) {
                // It does.
                updateDistanceLabel(v, i);
            }
        }
    }

    // At this point we know parent of each node on the
    // best path from Start to Dest
    clog << ”shortest distance from ” << Start_ID << ” To ” << Dest_ID
```

```
        << ” is ” << distance[Dest_ID] << '\n';
    best_path.push_front(Dest_ID);
    int next_node_id = Dest_ID;
    while (next_node_id != Start_ID) {
        next_node_id = parent[next_node_id];
        best_path.push_front(next_node_id);
    }
    return best_path;
}


void Weighted_Graph::Show_Best_Path(int Start, int Dest) const
{
    deque<int> best_path = this->Best_Path(Start, Dest, true);

    if (best_path.size() == 0) {
        clog << ”No path found\n”;
    }
    else {
        clog << ”Showing best path:\n”;
        while (best_path.size() > 0) {
            int next = best_path.front();
            best_path.pop_front();
            cout << next << '\n';
        }
    }
    clog << '\n';
}



    bool Weighted_Graph::
validVertex(int node) const
{
    return node > 0 && node <= number_of_nodes;
}
```