Computer programming

Just another programming weblog
2007-11-29

One Source Shortest Path: The Bellman-Ford Algorithm

Posted by scvalex under Algorithms, Graphs | Tags: algorithm, bellman-ford, C, explanation, graph, Graphs, programming, shortest path, sourcecode, tutorial |
[112] Comments

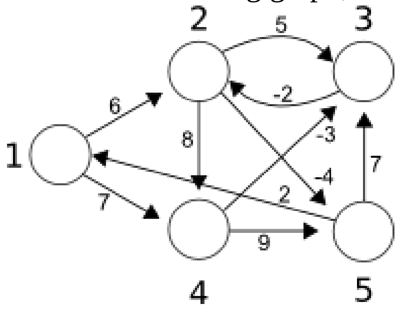
In this article, I describe the Bellman-Ford algorithm for finding the one-source shortest paths in a graph, give an informal proof and provide the source code in C for a simple implementation.

To understand this you should know what a graph is, and how to store one in memory. If in doubt check this and this.

Another solution to this problem is Dijkstra's algorithm.

The Problem

Given the following graph, calculate the length of the shortest path from **node 1** to **node 2**.



It's obvious that there's a direct route of length *6*, but take a look at path: *1 -> 4 -> 3 -> 2*. The length of the path is *7 – 3 – 2 = 2*, which is less than *6*. BTW, you don't need negative edge weights to get such a situation, but they do clarify the problem.

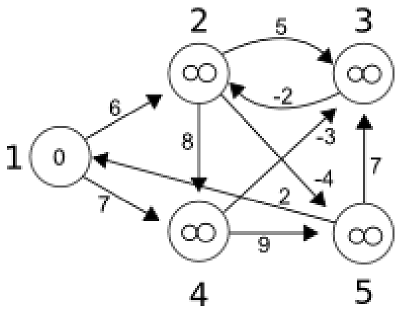
This also suggests a property of shortest path algorithms: to find the shortest path form *x* to *y*, you need to know, beforehand, the shortest paths to *y*'s neighbours. For this, you need to know the paths to *y*'s neighbours' neighbours… In the end, you must calculate the shortest path to the connected component of the graph in which *x* and *y* are found.

That said, you usually calculate **the shortest path to all nodes** and then pick the ones you're intrested in.

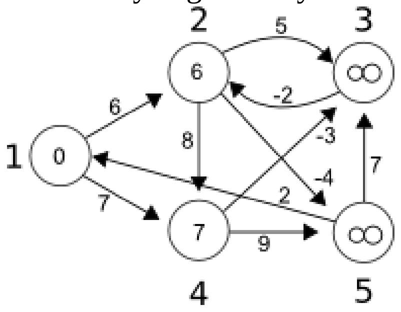
The Algorithm

The Bellman-Ford algorithm is one of the classic solutions to this problem. It calculates the shortest path to all nodes in the graph from a single source.

The basic idea is simple:
Start by considering that the shortest path to all nodes, less the source, is infinity. Mark the length of the path to the source as *0*:

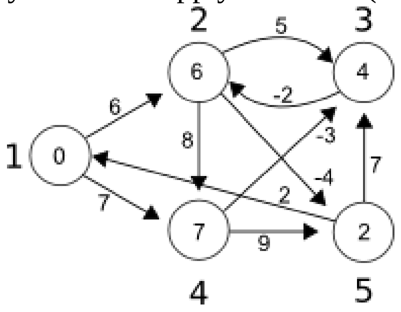


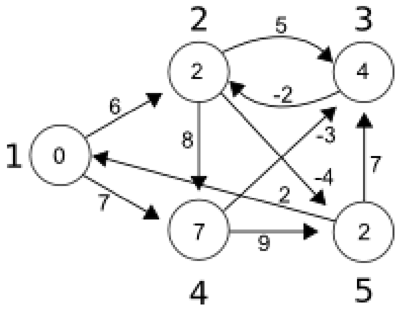
Take every edge and try to *relax* it:

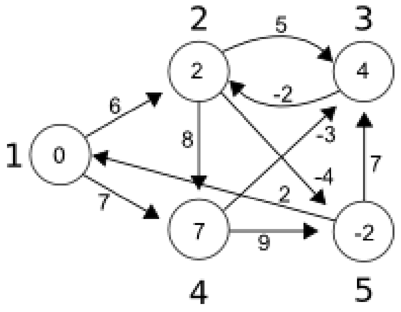


Relaxing an edge means checking to see if the path to the node the edge is pointing to can't be shortened, and if so, doing it. In the above graph, by checking the **edge 1 -> 2** of length *6*, you find that the length of the shortest path to **node 1** plus the length of the **edge 1 -> 2** is less then infinity. So, you replace infinity in **node 2** with *6*. The same can be said for edge *1 -> 4* of length *7*. It's also worth noting that, practically, you can't relax the edges whose start has the shortest path of length infinity to it.

Now, you apply the previous step *n – 1* times, where n is the number of nodes in the graph. In this example, you have to apply it *4* times (that's *3* more times).







That’s it, here’s the algorithm in a condensed form:

```
void bellman_ford(int s) {
        int i, j;

        for (i = 0; i < n; ++i)
                d[i] = INFINITY;

        d[s] = 0;

        for (i = 0; i < n - 1; ++i)
                for (j = 0; j < e; ++j)
                        if (d[edges[j].u] + edges[j].w < d[edges[j].v])
                                d[edges[j].v] = d[edges[j].u] + edges[j].w;
}
```

Here, **d[i]** is the shortest path to node **i**, **e** is the number of edges and **edges[i]** is the **i**-th edge.

It may not be obvious why this works, but take a look at what is certain after each step. After the first step, any path made up of at most *2* nodes will be optimal. After the step *2*, any path made up of at most *3* nodes will be optimal… After the *(n – 1)*-th step, any path made up of at most *n* nodes will be optimal.

The Programme

The following programme just puts the **bellman_ford** function into context. It runs in **O(VE)** time, so for the example graph it will do something on the lines of **5 * 9 = 45** relaxations. Keep in mind that this algorithm works quite well on graphs with few edges, but is very slow for dense graphs (graphs with almost n^2 edges). For graphs with lots of edges, you’re better off with Dijkstra’s algorithm.

Here’s the source code in C (bellmanford.c):

```
#include <stdio.h>

typedef struct {
        int u, v, w;
} Edge;

int n; /* the number of nodes */
int e; /* the number of edges */
Edge edges[1024]; /* large enough for n <= 2^5=32 */
int d[32]; /* d[i] is the minimum distance from node s to node i */

#define INFINITY 10000
```

```
void printDist() {
        int i;

        printf("Distances:\n");

        for (i = 0; i < n; ++i)
                printf("to %d\t", i + 1);
        printf("\n");

        for (i = 0; i < n; ++i)
                printf("%d\t", d[i]);

        printf("\n\n");
}

void bellman_ford(int s) {
        int i, j;

        for (i = 0; i < n; ++i)
                d[i] = INFINITY;

        d[s] = 0;

        for (i = 0; i < n - 1; ++i)
                for (j = 0; j < e; ++j)
                        if (d[edges[j].u] + edges[j].w < d[edges[j].v])
                                d[edges[j].v] = d[edges[j].u] + edges[j].w;
}

int main(int argc, char *argv[]) {
        int i, j;
        int w;

        FILE *fin = fopen("dist.txt", "r");
        fscanf(fin, "%d", &n);
        e = 0;

        for (i = 0; i < n; ++i)
                for (j = 0; j < n; ++j) {
                        fscanf(fin, "%d", &w);
                        if (w != 0) {
                                edges[e].u = i;
                                edges[e].v = j;
                                edges[e].w = w;
                                ++e;
                        }
                }
        fclose(fin);

        /* printDist(); */

        bellman_ford(0);
        printDist();

        return 0;
}
```

And here’s the input file used in the example (dist.txt):

```
5
0 6 0 7 0
0 0 5 8 -4
0 -2 0 0 0
0 0 -3 9 0
2 0 7 0 0
```

That’s an adjacency matrix.

That’s it. Have fun. Always open to comments.

About these ads

112 Responses to “One Source Shortest Path: The Bellman-Ford Algorithm”

1. *Raghavendra* Says:

   2008-02-08 at 12:58
   Thanks a lot for explanation.I have a doubt it detects negative edge cycles.What about the shortest will that be found.Will that include that negative edge also.Will that negative edge be skipped while calculating the shortest path.

   Reply

   1. *DavivMcD* Says:

      2009-12-08 at 1:23
      You can use this algorithm to detect negative edge cycles by going 1 step further (for a total of n loops). If a change occurs between the (n-1)th and nth loop, there is a negative edge cycle.

      Unfortunately, leaving this vertex out in any shortest path decisions is a much more difficult thing to do.

      To put it simply: The Bellman-Ford algorithm can detect negative loops, but it can’t work around them.

      Reply

      1. *Sumit* Says:

         2011-02-22 at 10:53

To be specific, any negative weigthed cycle reachable from source only not each!!

2. *papoo* Says:

   2008-02-29 at 15:27
   thank you :)

   Reply

3. *Rara* Says:

   2008-03-12 at 6:01
   what's the difference between Dijkstra and Bellman-Ford? i think it's similar. would u like to differ them by their each steps?

   Reply

   1. *DavivMcD* Says:

      2009-12-08 at 1:31
      Basically the Dijkstra's method marks each node as 'visited' once it relaxes all edges from that node, and will not try to relax any edges leading back to that node after that point. It then chooses the node with the lowest value from the initial node, as the next 'current' node.

      Since at each step i out of n nodes, you are relaxing at most n-i edges, (instead of n-1 like Bellman-Ford's), this algorithm completes much quicker than Bellman-Ford's, and scales better with a larger number of nodes. The main drawback is that it will not work with negative edge weights.

      Reply

   2. *sangamesh* Says:

      2012-02-12 at 7:33
      dijkstra algorithm can be applied to only positive weights, but bellman ford algorithm is applied for both positive and negative weights….

      Reply

4. *suvo* Says:

   2008-03-18 at 18:17
   I think there is a simple difference btween Dijkstra and Bellman-Ford….like – Bellman-Ford work with negetive weight cycles and relax many time….but Dijkstra not work with negetive cycle and relax only one time…

   Reply

   1. *LuVar* Says:

2010-03-18 at 16:38
Dijkstra cannot work with negative weights at all. If you try it, result will not be shortest path. In JUNG implementation of dijkstra it throws exception if some edge has negative weight.

Reply

2. *DEEPAK UNIYAL* Says:

   2011-07-06 at 20:07
   No algorithm works with negative weight cycle.
   Every algorithm fails in case of negative weight cycle.
   Bellman ford works with negative weight edges whereas Dijkstra doesn't.

   Reply

5. *chyrel* Says:

   2008-05-07 at 21:28
   Thanks for the explanation. By the way, can you show me how to find the shortest path in sudoku problem?

   Reply

6. *george* Says:

   2008-05-27 at 22:17
   can you tell us how to print also the path of the shortest distance.thanks a lot.

   Reply

7. *Jolmash* Says:

   2008-09-09 at 5:53
   Yeees, how to print also the path of the shortest distance? thanks in advance.

   Reply

8. *Max* Says:

   2008-10-21 at 16:12
   Very good explanations. Much better than my textbook (and my professor). Thanks

   Reply

9. *Miro* Says:

   2008-11-24 at 22:21
   thanx a lot for explaining, you explained it better than my teacher :)

   Reply

10. *Abid* Says:

    2009-02-28 at 22:56
    This tutorial is excellent.It has helped me a lot.More tutorial should be published.

    Reply
11. *Varuna* Says:

    2009-03-01 at 12:21
    thanx… lot of help to the assignment, the dijkstras and bellman ford does arises some complexities but they do differ from negative weights…. thanx again………… keep it up

    Reply
12. *pavlos* Says:

    2009-03-23 at 21:57
    Bellman-Ford's algorithm returns TRUE if the graph contains no negative weight cycles that are reachable from the source.
    So the above source needs some additions to check the above requirement.
    For pseudo code you may check this link
    http://www.cs.rpi.edu/~musser/gp/algorithm-concepts/bellman-ford-screen.pdf

    Reply
13. *m@q* Says:

    2009-03-25 at 15:55
    Thanks a lot!! :)

    Reply
14. *harro* Says:

    2009-03-26 at 17:06
    is there any way to print out the actual path? i.e. node 1 to 3 to 5

    Reply
    1. *Sumit* Says:

       2011-02-22 at 11:02
       yes but in reverse. reminding u've set parent to the node.. use that..

       or u can use recursion to print in sequence

       Reply

15. *andre* Says:

    2009-03-27 at 13:30
    Ford the best car :)

    Reply

16. *Hela* Says:

    2009-03-29 at 4:10
    Hi … Does anyone have an example of a real situation where negative weights are used?? Thanks !

    Reply

17. *sarang s sane* Says:

    2009-03-31 at 17:09
    If i have to laydown the telephone network around the 50 houses around my apartment …Which shortest path method should i use to get it work for me …please explane..

    Reply

18. ������������ Says:

    2009-04-02 at 17:17
    Ford is a good car :)

    Reply

19. *JP* Says:

    2009-04-19 at 16:07
    Thanks a lot.

    Reply

20. *Stephen* Says:

    2009-05-06 at 0:37
    Can the source ever become negative?

    Reply

    1. *ramen* Says:

       2011-02-03 at 14:26
       If the source were to ever become negative, then it would be due to a negative weight cycle.

Reply

21. *muslima* Says:

    2009-05-18 at 19:11
    Thanks alot..
    This is simple and helpful :)

    Reply

22. *rohit* Says:

    2009-07-01 at 15:47
    Great article! However, I was interested in knowing how we could modify your algorithm to find the Kth shortest path from a source to a node,ie, for K=2,how to find the second shortest path..?

    Reply

    1. *Durjay* Says:

       2009-09-07 at 17:08
       how i find shortest distance when source node 2,3,4,5.

       Reply

23. *Alyse* Says:

    2009-07-25 at 12:38
    thanks alot…

    Reply

24. *coolguy* Says:

    2009-10-21 at 4:12
    Very lucid explanation! Thanks for your efforts!

    Reply

25. *ishika jain* Says:

    2009-11-11 at 10:05
    thanks a lot…

    Reply

26. *Anil Gupta* Says:

    2009-11-16 at 1:20

awesome tuto…it really helps in understanding this algo.Thanks for your efforts!

Reply

27. *xxas2axx* Says:

    2009-11-17 at 23:59
    **** you nerds

    Reply

28. *Osmar* Says:

    2009-11-22 at 18:26
    Excelente este manual…..
    Me sirvio para entender mi clase

    Gracias

    Reply

29. *akther* Says:

    2009-12-05 at 7:53
    thanks a lot…but is it detect negative cycle??

    Reply

30. *himanshu* Says:

    2009-12-08 at 23:12
    good explation

    Reply

31. *Shw05* Says:

    2009-12-12 at 7:33
    can you please tell me how to print Path in Bellman algorithm ..like the way its done in Dijsktra.
    Please provide the code sample if you have..Thanks in advance.

    Reply

32. *Determinant* Says:

    2009-12-21 at 22:30
    Thank you very much, excellent explanation (Y) really thank you.

    I have only one question kindly answer it: what u,v,w represent in this code? it seems to be a silly question.

    Thanks in advance

Reply

33. *Nelson* Says:

    2010-01-02 at 15:21
    Hi,

    thank you very much.
    I think you adjecency matrix has a little mistake: the nine ought to be one spot to the right?
    To detect negative circles:
    Just let the first for loop go one further, then make an if statement in the existing if statement like if (n == n-1) … an then stop the function and print "negative circle"

    Reply

34. *yuri* Says:

    2010-01-10 at 16:18
    thank you so much , but the result of this program , is only a matrice , how can'i found the shortest path from onr node to a destination , for example : from 1 to 5.
    thank you

    Reply

35. *Murat Özen* Says:

    2010-03-27 at 15:48
    Thanks for your clear explanation. I have a question that how can I improve this code to find K-th shortest paths.

    thank you

    Reply

36. *Ramkumar* Says:

    2010-04-01 at 9:31
    Cool Dude !!!

    Reply

37. *Xerxes* Says:

    2010-04-22 at 7:08
    wow… who are you buddy ??? This article is incredibly good, clear and simple. Thank you very much for your article. I'll must recommend my pals to read your article.

    Reply

38. *Bayan* Says:

    2010-05-06 at 19:03
    thaks very much

    Reply
39. *ketut nadha* Says:

    2010-06-05 at 5:22
    bagus!!

    Reply
40. *Martin* Says:

    2010-07-23 at 3:53
    Thanks! The best explanation of Bellman-Ford I have encountered.

    Reply
41. *saad* Says:

    2010-08-12 at 19:52
    Thanks a lot…nice tutorial..

    Reply
42. *UVa : 558 (Wormholes) « Solved Programming Problems : Python's Blog* Says:

    2010-09-01 at 17:11
    […] algorithm (https://compprog.wordpress.com/2007/11/29/one-source-shortest-path-the-bellman-ford-algorithm/ […]

    Reply
43. *Nikita Vadher* Says:

    2010-10-19 at 16:02
    Thankx a lot……..

    Reply
44. *Liviu* Says:

    2010-12-03 at 15:21
    So bellman-ford detects if there's a negative cycle and tops there?

    Doesn't it find the shortest route by skipping the negative cycle also?

Reply

45. *Liviu* Says:

    2010-12-03 at 15:22
    So bellman-ford detects if there's a negative cycle and tops there?

    Doesn't it find the shortest route by skipping the negative cycle also?

    liviu.c.doro@gmail.com

    Reply

46. *Fahad* Says:

    2010-12-18 at 5:27
    has anyone executed this algorithm by taking the source instead of 0?

    Reply

47. *Patrick* Says:

    2011-02-03 at 15:24
    Can someone please tell me how the algorithm publishes the shortest path?

    Reply

48. *elif* Says:

    2011-03-13 at 18:27
    hey i have a question for u all? Can i use Dijsktra algoritm instead of Bellmans for megative weighted graphs, if I add a number for every arc in graph that makes every arc non-negative?

    Reply

49. *revendera* Says:

    2011-04-17 at 7:27
    thanks

    Reply

50. *Bellman-Ford Algorithm « Me, when programming* Says:

    2011-05-15 at 0:35
    […] For example Click here […]

    Reply

51. *ájem* Says:

2011-05-28 at 20:56
In belmann_ford()

for (j = 0; j < e; ++j)
if (d[edges[j].u] + edges[j].w<d[edges[j].v])
printf("Negativeee Circle\n\n");

hm?

otherwise(){thanks:)}

Reply

52. *Pravin* Says:

2011-08-05 at 8:37
can u any1 tell me what's the meaning of negative significance(negative weight cycle) in bellman algoritm

Reply

53. *Reena* Says:

2011-08-22 at 1:19
It was helpful. Thank u so much:-)

Reply

54. *John Oke* Says:

2011-10-02 at 9:19
So what is thew shortst path and how do you find/work it out
Thanking you in advance
Indiana

Reply

55. *sheetal (@sheetal0807)* Says:

2011-10-08 at 18:41
Can You plz provide the matlab code for Bellman ford algorithm.

Sonal

Reply

56. *Pankaj Nayak* Says:

2011-10-11 at 13:19
lalala pila

Reply

57. *abnerself* Says:

2011-10-15 at 20:57
hola quiero descargar(aquí está el archivo de entrada utilizado en el ejemplo (DIST.TXT):) pero no lo se como soy nuevo usuario, porfavor si pueden me lo embian mi cuenta es abnerself…, o abner.systems@hotmail.com

Reply

58. *taman negara pahang* Says:

2011-11-05 at 1:56
taman negara pahang…

[…]One Source Shortest Path: The Bellman-Ford Algorithm « Computer programming[…]…

Reply

59. *Andy Carloff* Says:

2011-11-13 at 0:23
Very well done. The description of "relaxing an edge" appears in Wikipedia, but they don't give any explanation of what that means. Thanks.

Reply

60. *Shahal Tharique* Says:

2011-11-21 at 19:30
I still don't get this. In the first loop iteration of the outer loop, let's say by ur example, u first modify edge 1->2 and edge 1->4, what's the problem in modifying the edge 2->3, 2->5, 4->3, 4->5, since we have d[2] and d[4] now.

Reply

61. *medichismes* Says:

2011-11-25 at 8:12
Thanks!!! =D

Reply

62. *Dhritisundar Maity* Says:

2011-11-26 at 16:03
Given a weighted, directed graph G = (V, E) with no negative-weight cycles, let m be the
maximum over all pairs of vertices u, v V of the minimum number of edges in a shortest
path from u to v. (Here, the shortest path is by weight, not the number of edges.) Suggest a
simple change to the Bellman-Ford algorithm that allows it to terminate in m + 1 passes.plz solve this…

Reply

63. *wsnets* Says:

    2011-12-04 at 6:13
    Is there any Matlab code for the Bellman-ford algorithm. I will appreciate if someone can point me to one please. Thanks.

    Reply
64. *wsnets* Says:

    2011-12-04 at 6:14
    Is there any Matlab code for the Bellman-ford algorithm. Thanks.

    Reply
65. *Jhonatam* Says:

    2011-12-15 at 4:17
    Hello I wonder if it is possible to print the nodes where he found the best way would be and how to do this?

    Reply
66. *sara* Says:

    2012-01-01 at 15:29
    thamks alot This really helped me….

    Reply
67. *maternity fashion* Says:

    2012-01-09 at 21:05
    maternity fashion…

    […]One Source Shortest Path: The Bellman-Ford Algorithm « Computer programming[…]…

    Reply
68. *Samir* Says:

    2012-01-11 at 2:02
    Samir…

    […]One Source Shortest Path: The Bellman-Ford Algorithm « Computer programming[…]…

    Reply
69. *venu]* Says:

    2012-03-03 at 16:41
    its not clear

Reply

70. *abhi* Says:

    2012-03-06 at 6:25
    can u explain the problem in stepwise..

    Reply

71. *akhileshazad* Says:

    2012-04-22 at 13:41
    i am not be able to open dist.txt file in Bellman–Ford algorithm example plz help me

    thanks ;

    Reply

72. *ameenos2000* Says:

    2012-04-22 at 18:50
    Reblogged this on ameenos2000.

    Reply

73. *n1ght3* Says:

    2012-04-26 at 8:04
    Simple. Concise. I liked it.

    Reply

74. *Gos* Says:

    2012-04-26 at 16:18
    Thanks ! For an awesome explanation! was struggling for it….
    but I needed a java program for this….thanks anyways!

    Reply

75. *ricky...* Says:

    2012-05-14 at 1:01
    nice

    Reply

76. *Hussnain* Says:

2012-05-16 at 18:05
thanks a lotttttttttttttttttt

Reply

77. *nimisha* Says:

2012-06-18 at 15:47
when there is positive weights than dijikstras will be applied and for negative weights bellmanford is applied

Reply

78. *foufidoom* Says:

2012-06-25 at 15:23
Hi there! I am trying to do some kind of programming exercise here on Bellman-Ford algorithm, I'm good with algorithms but I don't have much experience with program languages, not even c. I understood a lot from the way you built your program but unfortunately I can't open the text you were using to import your data, it takes me to a page where it says that should be logged in (I am) and a member of your blog (following is not enough? :( ). I'd really appreciate it if you could help me see that text. Thank you very much.

Reply

79. *nilesh* Says:

2012-07-31 at 17:40
Can anyone tell me the application of Bellman ford algo and where these negative edge are used?

Reply

80. *sam* Says:

2012-08-12 at 20:35
how can i print the shortest path?

Reply

81. *namit* Says:

2012-09-25 at 14:25
it's gud…………….

Reply

82. *indri* Says:

2012-10-07 at 17:21

hi, i want to ask, how about the implementation on java?
is it has big difference with C?
thank you :)

Reply

83. *karan* Says:

    2012-10-31 at 11:30
    no comments…………….

    Reply

84. *ramlal* Says:

    2012-11-11 at 20:12
    panditAJI

    Reply

85. *Morpheus* Says:

    2012-11-19 at 20:21
    Thanks a million!

    Reply

86. *rizkisatyautami* Says:

    2012-12-17 at 3:31
    Thank you so much :D

    Reply

87. *hema* Says:

    2012-12-18 at 9:30
    Am getting segmentation fault when I execute this program.

    Reply

88. *ntz* Says:

    2013-02-16 at 10:53
    ultimate….
    thank u..

    Reply

89. *make money online* Says:

 2013-04-03 at 1:11
 Hey There. I found your blog using msn. This is a very neatly written article. I will be sure to bookmark it and come back to read extra of your helpful info. Thank you for the post. I will definitely return.

 Reply

90. *mouliyerram* Says:

 2013-04-04 at 14:54
 can you explain about the input file dist.txt

 Reply

91. *utkarsh* Says:

 2013-05-28 at 13:50
 where are the value in the vertices stored???

 Reply

92. *vanita* Says:

 2013-06-26 at 12:40
 Thanks !!!! I searched various sites for this explanation.. This one was the best.. It make me understand the algorithm properly !!

 Reply

93. *Ramona* Says:

 2013-07-17 at 18:18
 It seems to me finding the shortest path with Floyd Warshal and Bellman-Ford is very similar, the algorithm differs, but the pattern seems the same to me. What's the difference really I can't seem to find it, when going through the graph?

 Reply

94. *aichaboulidame* Says:

 2014-01-27 at 20:25
 thank's . i can't open the file dist.txt plz help me

 Reply

95. *how rtd\u0027s work* Says:

2014-03-12 at 20:06
Good post. I learn something new and challenging on blogs
I stumbleupon on a daily basis. It will always be interesting to read through articles
from other writers and use a little something from other web sites.

Reply

96. *Hungry Shark Evolution Hack Triche Gratuit* Says:

    2014-09-24 at 3:05
    Hungry Shark Evolution Hack Triche Gratuit

    One Source Shortest Path: The Bellman-Ford Algorithm | Computer programming

    Reply

97. *Castle Clash Triche* Says:

    2014-09-24 at 4:53
    Castle Clash Triche

    One Source Shortest Path: The Bellman-Ford Algorithm | Computer programming

    Reply

98. *Https://Umich.Academia.Edu* Says:

    2014-09-30 at 15:31
    I'm not that much of a internet reader to be honest but your sites really nice, keep it up!
    I'll go ahead and bookmark your website to come back later.
    Many thanks

    Reply

99. *Injustice Hack* Says:

    2014-10-01 at 13:56
    Injustice Hack

    One Source Shortest Path: The Bellman-Ford Algorithm | Computer programming

    Reply

100. *War Commander Hack* Says:

    2014-10-07 at 11:14
    It's great that you are getting thoughts from this article as
    well as from our argument made at this place.

    Reply

101. *Camille* Says:

    2014-10-07 at 12:15

It’s awesome to pay a visit this web page and reading the views of all colleagues regarding this post, while I am also eager of getting know-how.

Reply

102. *Data Structures and Algorithms Tutorials | TheShayna.Com* Says:

2014-11-11 at 2:13
[…] with Implementation 2 […]

Reply

103. *aravinds187* Says:

2014-11-20 at 9:09
Inputs: A sequence A = (a1, . . . , an) of n real numbers, and two real numbers p and q.
Output: The number of elements of A that are more than or equal to p and less than or equal to q.
Give a recursive algorithm, NumberInRange(A, p, q, i) that solves this problem. e algorithm, NumberInRange(A, p, q, i) that solves this problem. Note that the parameter i is needed to record which part of the sequence A is being considered for a given recursive call. So NumberInRange(A, p, q, i) returns the number of values within the range that are in the subsequence (a1, . . . , ai)

Reply