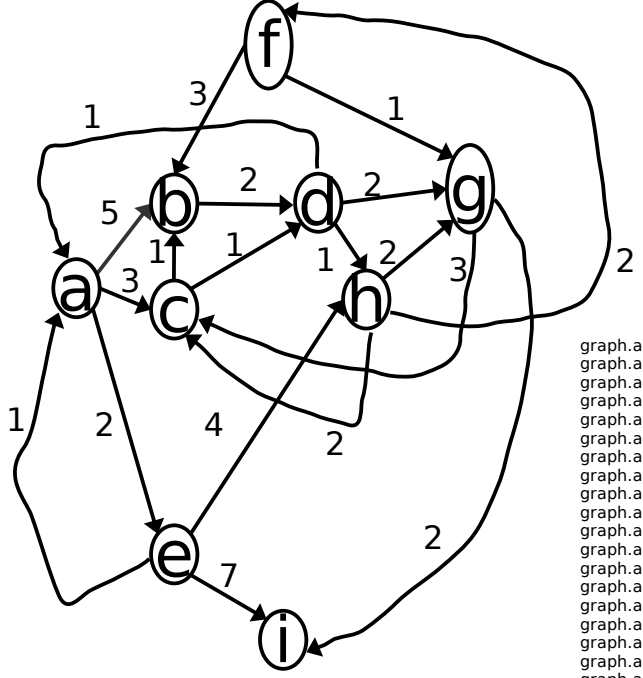


```
graph.add_edge('a', 'b', weight=5)
graph.add_edge('a', 'c', weight=3)
graph.add_edge('a', 'e', weight=2)
graph.add_edge('b', 'd', weight=2)
graph.add_edge('c', 'b', weight=1)
graph.add_edge('c', 'd', weight=1)
graph.add_edge('d', 'a', weight=1)
graph.add_edge('d', 'g', weight=2)
graph.add_edge('d', 'h', weight=1)
graph.add_edge('e', 'a', weight=1)
graph.add_edge('e', 'h', weight=4)
graph.add_edge('e', 'i', weight=7)
graph.add_edge('f', 'b', weight=3)
graph.add_edge('f', 'g', weight=1)
graph.add_edge('g', 'c', weight=3)
graph.add_edge('g', 'i', weight=2)
graph.add_edge('h', 'c', weight=2)
graph.add_edge('h', 'f', weight=2)
graph.add_edge('h', 'g', weight=2)


shortest_path = ShortestPath(graph)
result = shortest_path.find_shortest_path('a', 'I')    should be 8
self.assertEqual(result, ['a', 'c', 'd', 'g', 'i'])
```

unvisited																
a	b	c	d	e	f	g	h	i		a	b	c	d	e	f	g	h	i
										inf	inf	inf	inf	inf	inf	inf	inf	inf
									a									
									A=0	0	inf	inf	inf	inf	inf	inf	inf	inf
										0	5	3	inf	2	inf	inf	inf	inf
	b	c	d	e	f	g	h	i										
									E=2									
										0	5	3	inf	2	inf	inf	inf	inf
										0	5	3	inf	2	inf	inf	6	inf
										0	5	3	inf	2	inf	inf	6	9
	b	c	d		f	g	h	i										
									C=3	0	5	3	inf	2	inf	inf	6	9
										0	4	3	inf	2	inf	inf	6	9
										0	4	3	4	2	inf	inf	6	9
	b		d		f	g	h	i		0	4	3	4	2	inf	inf	6	9
									B=4	0	4	3	4	2	inf	inf	6	9
										0	4	3	4	2	inf	inf	6	9
			d		f	g	h	i		0	4	3	4	2	inf	inf	6	9
									D=4									
										0	4	3	4	2	inf	inf	6	9
										0	4	3	4	2	inf	6	6	9
										0	4	3	4	2	inf	6	5	9
				f	g	h	i			0	4	3	4	2	inf	6	5	9
									H=5									
										0	4	3	4	2	inf	6	5	9
										0	4	3	4	2	7	6	5	9
										0	4	3	4	2	7	6	5	9
				f	g		i			0	4	3	4	2	7	6	5	9
									G=6									
										0	4	3	4	2	7	6	5	9
										0	4	3	4	2	7	6	5	8
				f			i			0	4	3	4	2	7	6	5	8
									F=7									
										0	4	3	4	2	7	6	5	8
										0	4	3	4	2	7	6	5	8
								i		0	4	3	4	2	7	6	5	8
									I=8									
										0	4	3	4	2	7	6	5	8

aは指定される

aのadjacent nodes(b5/c3/e2。数値はaから)

weightが低いeからc、bへ(0+5=5, 0+3=3, 0+2=2) ← Cが最低値で更新される。fromはA

aはvisited

unvisitedの一番低いweightのノードを選択。eを選択

eのadjacent nodes(a1/h4/i7。数値はeから)

a=2+1=3で現在値0がそのまま

H=2+4=6

I=2+7=9

eはvisited

unvisitedの一番低いweightのノードを選択。cを選択

cのadjacent nodes(b1/d1。数値はcから)

B=3+1=4

D=3+1=4 ← Dが最低値で更新される。fromはC

cはvisited

unvisitedの一番低いweightのノードを選択。bを選択

bのadjacent nodes(d2。数値はbから)

D=4+2=6。現在値4なのでアップデートしない

bはvisited

unvisitedの一番低いweightのノードを選択。dを選択

dのadjacent nodes(a1/g2/h1。数値はdから)

A=4+1=5。アップデート無し

G=4+2=6 ← Gが最低値で更新される。fromはD

H=4+1=5

dはvisited

unvisitedの一番低いweightのノードを選択。hを選択

hのadjacent nodes(c2/f2/g2。数値はhから)

C=5+2=7

F=5+2=7

G=5+2=7

hはvisited

unvisitedの一番低いweightのノードを選択。gを選択

gのadjacent nodes(c3/i2。数値はgから)

C=6+3=9

I=6+2=8 ← iが最低値で更新される。fromはG

gはvisited

unvisitedの一番低いweightのノードを選択。fを選択

fのadjacent nodes(b3/g1。数値はfから)

B=7+3=10

G=7+1=8

fはvisited

unvisitedの一番低いweightのノードを選択。iを選択

iのadjacent nodes無し

iはvisited

unvisited無しで終わり