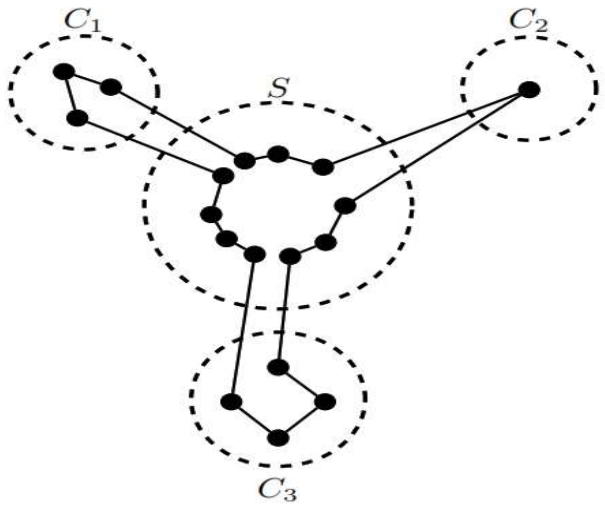
LECTURE # 10, 11 & 12

Proposition 2.11 If G is a graph with a hamiltonian cycle, then $G - S$ has at most $|S|$ components for every nonempty set $S \subseteq V$.

Proof: Assume G has a hamiltonian cycle C and S is a nonempty subset of vertices. Then as we traverse C we will leave and return to S from distinct vertices in S since no vertex can be used more than once in a cycle. Since C must span $V(G)$, we know S must have at least as many vertices as the number of components of $G - S$.

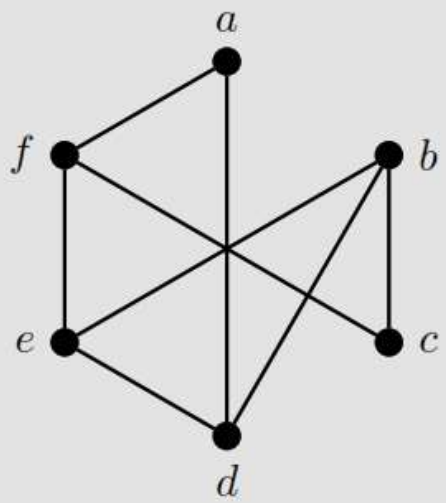


Theorem 2.12 (Dirac's Theorem) Let G be a graph with $n \geq 3$ vertices. If every vertex of G satisfies $\deg(v) \geq \frac{n}{2}$, then G has a hamiltonian cycle.

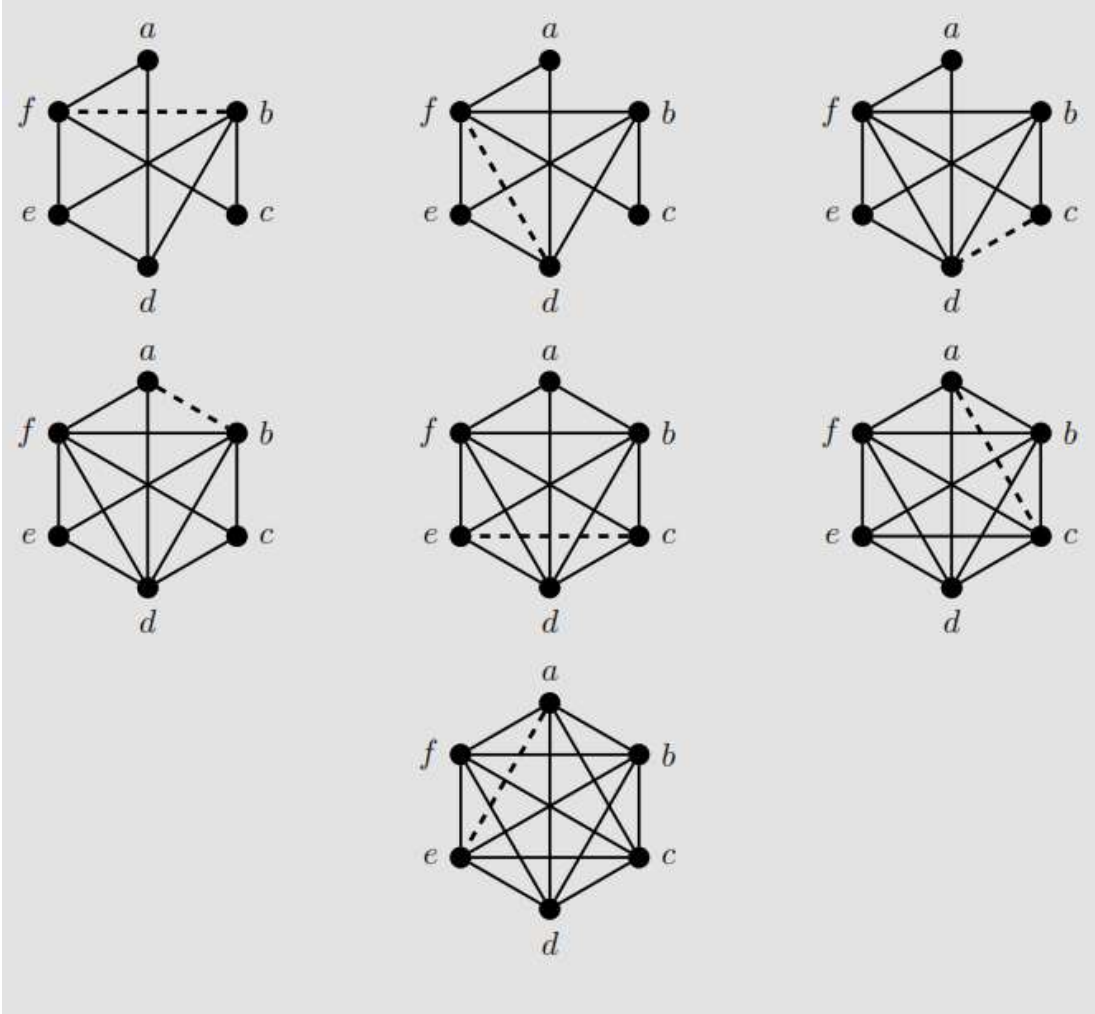
Theorem 2.13 (Ore's Theorem) Let G be a graph with $n \geq 3$ vertices. If $\deg(x) + \deg(y) \geq n$ for all distinct nonadjacent vertices, then G has a hamiltonian cycle.

Definition 2.14 The ***hamiltonian closure*** of a graph G, denoted $cl(G)$, is the graph obtained from G by iteratively adding edges between pairs of nonadjacent vertices whose degree sum is at least n, that is we add an edge between x and y if $\deg(x) + \deg(y) \geq n$, until no such pair remains.

Example 2.10 Find a hamiltonian closure for the graph below.



Solution: First note that the degrees of the vertices are $2, 3, 2, 3, 3, 3$ (in alphabetic order). Since the closure of G is formed by putting an edge between nonadjacent vertices whose degree sum is at least 6, we must begin with edge bf or df. We chose to start with bf. In the sequence of graphs shown on the next page, the edge added at each stage is a thick dashed line.



Complete graphs K_n, $n \geq 3$, are Hamiltonian.

Theorem 2.16 A graph G is hamiltonian if and only if its closure $cl(G)$ is hamiltonian.

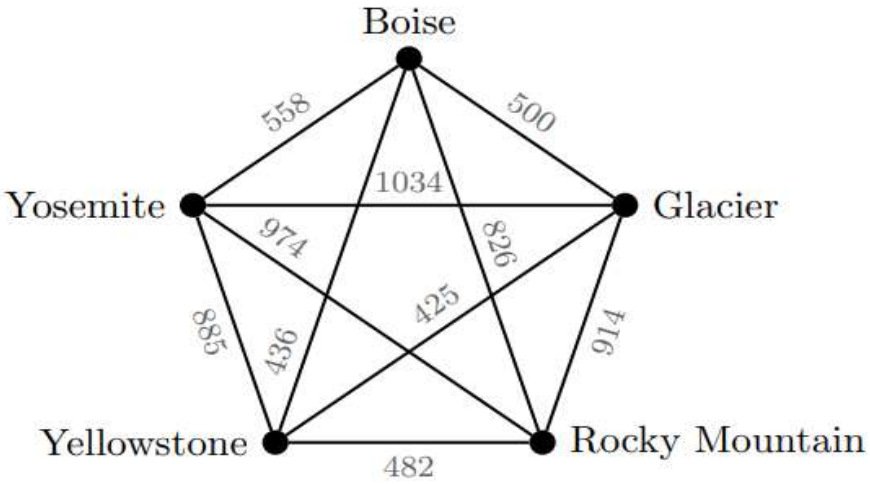
Lemma 2.17 If G is a graph with at least 3 vertices such that its closure $cl(G)$ is complete, then G is hamiltonian.

Theorem 2.18 Let G be a simple graph where the vertices have degree $d_1, d_2, \ldots, d_n$ such that $n \geq 3$ and the degrees are listed in increasing order. If for any $i < \frac{n}{2}$ either $d_i > i$ or $d_{n-i} \geq n-i$, then G is hamiltonian.

2.2.1 The Traveling Salesman Problem

The discussion above should make clear the difficulty in determining if a graph is hamiltonian. But what if a graph is know to have a hamiltonian cycle? For example, every complete graph K_n (for $n \geq 3$) must contain a hamiltonian cycle since it satisfies the criteria of Dirac's Theorem. In this scenario, finding a hamiltonian cycle is quite elementary, and so, as mathematicians do, we generalize the problem to one in which the edges are no longer equivalent and have a weight associated to them. Then instead of asking whether a graph simply *has* a hamiltonian cycle, we can now ask how do we find the *best* hamiltonian cycle.

Historically, the extensive study of hamiltonian circuits arose in part from a simple question: A traveling salesman has customers in numerous cities; he must visit each of them and return home, but wishes to do this with the least total cost; determine the cheapest route possible for the salesman. In fact,

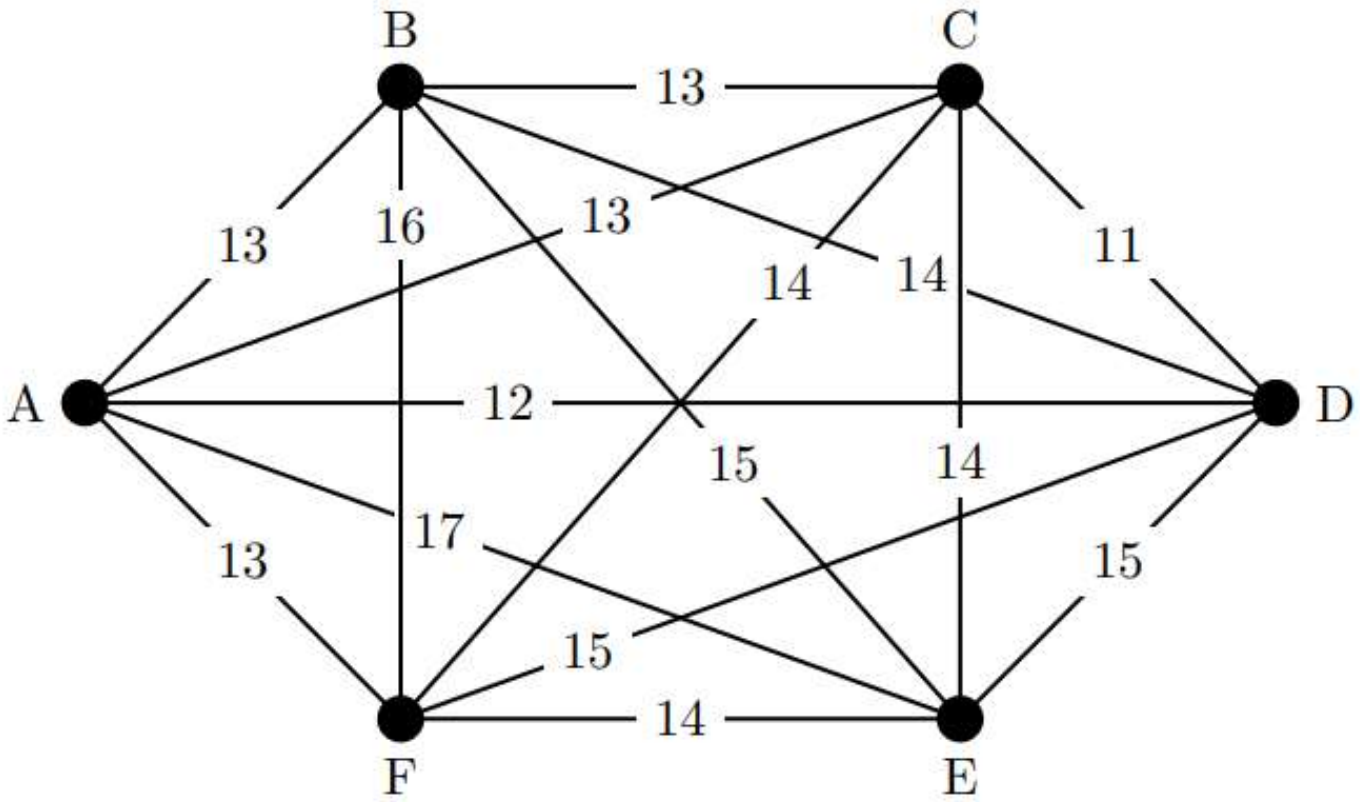


The graph that models the general Traveling Salesman Problem (TSP) is a *weighted complete graph*, such as the one shown above from Example 1.7.

Example 5.4.1. Mr. Lee is a salesman living in town A. Every day, he needs to travel from his house to all the other 5 towns (B to F) to promote his products before returning home in the evening. The 6 towns are well connected by public transport and there are direct buses between each pair of towns. The time (in minutes) it takes for Mr. Lee to travel between each pair of towns is shown in the table below.

Between	A	B	C	D	E	F
A	–	13	13	12	17	13
B	–	–	13	14	15	16
C	–	–	–	11	14	14
D	–	–	–	–	15	15
E	–	–	–	–	–	14
F	–	–	–	–	–	–

How should Mr. Lee plan his route from his home to the other 5 towns so as to minimize his total traveling time?


B
C
13
13
16
13
14
14
11
A
12
D
15
14
17
13
15
15
14
F
E

Clearly, G has many spanning cycles, each representing a route that Mr. Lee can take. The total traveling time would be the sum of all the weights of edges in a particular spanning cycle. The question now reduces to finding a spanning cycle with the **minimum weight**.

The Traveling Salesman Problem. (TSP) Let G be a weighted complete graph of order n. Find a spanning cycle C in G such that

$$w(C) = \sum_{e \in E(C)} w(e)$$

is the minimum among all spanning cycles of G.

Remark 5.4.2.

(1) Unlike the shortest path problem, minimum spanning tree problem, chinese postman problem, there are currently no efficient algorithms for solving the traveling salesman problem.

(2) Research questions relating to the traveling salesman problem are still being worked on by many mathematicians and computer scientists. Interested readers may visit http://www.tsp.gatech.edu/.

(3) Some instances of the traveling salesman problem have the weighted complete graphs G satisfying the triangle inequality

$$w(xy) \leq w(xz) + w(zy)$$

for any vertices x, y, z in G.

Algorithms for finding Hamiltonian Cycle with least possible cost

Brute Force Algorithm

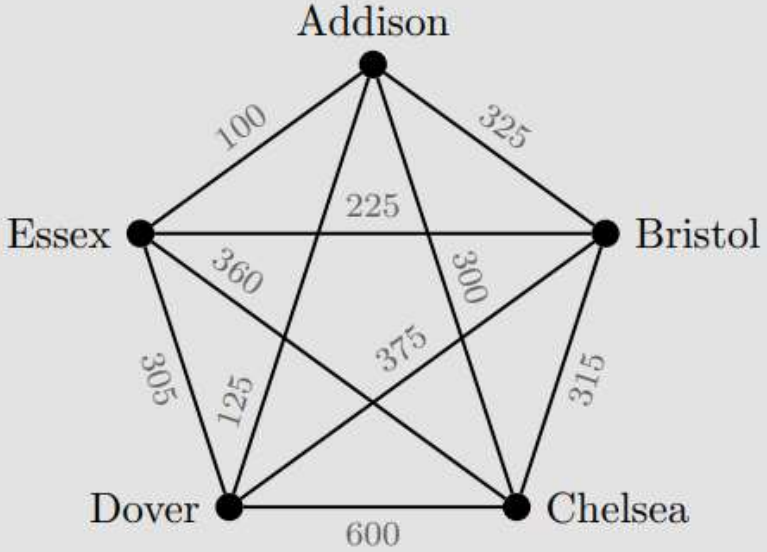
Input: Weighted complete graph K_n.

Steps:

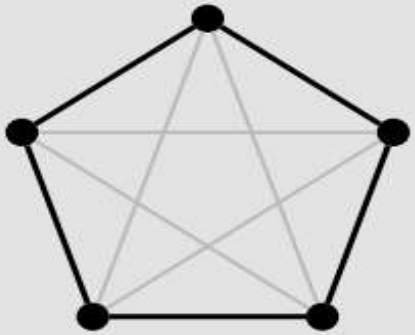
1. Choose a starting vertex, call it v.
2. Find all hamiltonian cycles starting at v. Calculate the total weight of each cycle.
3. Compare all $(n-1)!$ cycles. Pick one with the least total weight. (Note: there should be at least two options).

Output: Minimum hamiltonian cycle.

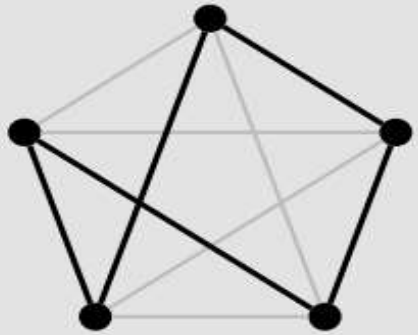
Example 2.11 Sam is planning his next business trip from his hometown of Addison and has determined the cost for travel between any of the five cities he must visit. This information is modeled in the weighted complete graph on the next page, where the weight is given in terms of dollars. Use Brute Force to find all possible routes for his trip.



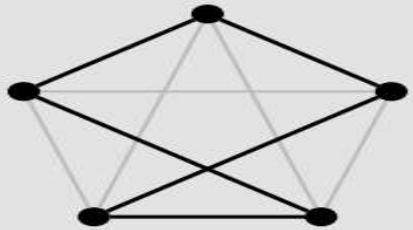
Solution: One method for finding all hamiltonian cycles, and ensuring you indeed have all 24, is to use alphabetical or lexicographic ordering of the cycles. Note that all cycles must start and end at Addison and we will abbreviate all cities with their first letter. For example, the first cycle is $a\,b\,c\,d\,e\,a$ and appears first in the list below. Below each cycle is its reversal and total weight.



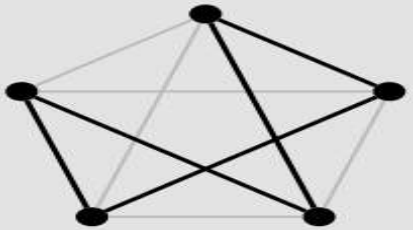
$a\,b\,c\,d\,e\,a$
$a\,e\,d\,c\,b\,a$
1645



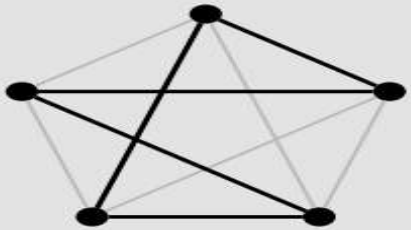
$a\,b\,c\,e\,d\,a$
$a\,d\,e\,c\,b\,a$
1430



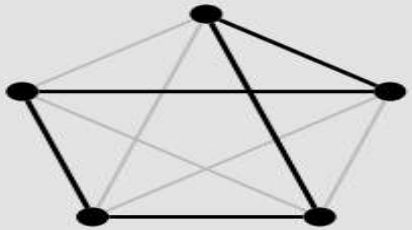
$abdcea$
$aecdba$
1760



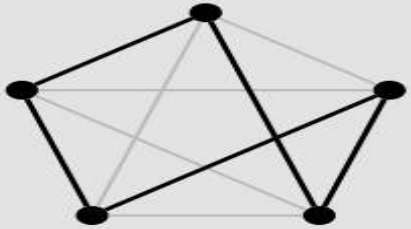
$abdeca$
$acedba$
1665



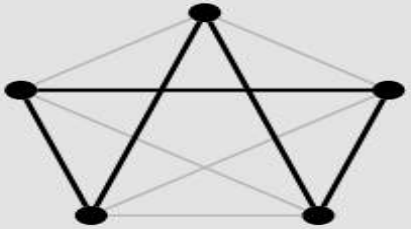
$abecda$
$adceba$
1635



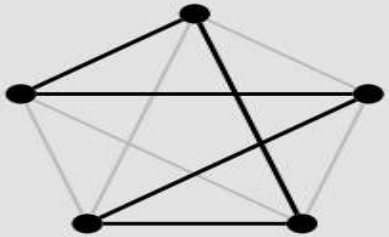
$abedca$
$acdeba$
1755



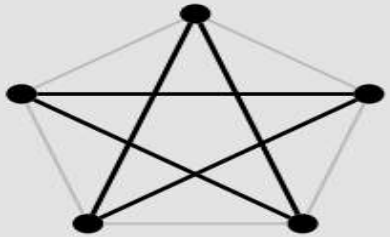
$acbdea$
$aedbca$
1395



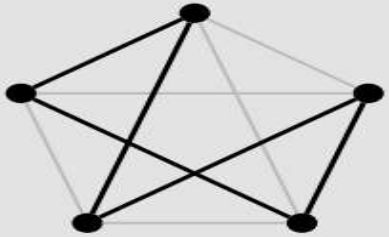
$acbeda$
$adebca$
1270



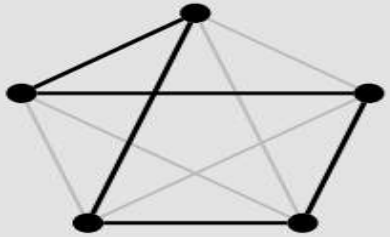
$a\,c\,d\,b\,e\,a$
$a\,e\,b\,d\,c\,a$
1600



$a\,c\,e\,b\,d\,a$
$a\,d\,b\,e\,c\,a$
1385



$a\,d\,b\,c\,e\,a$
$a\,e\,c\,b\,d\,a$
1275



$a\,d\,c\,b\,e\,a$
$a\,e\,b\,c\,d\,a$
1365

From the data above we can identify the optimal cycle as $a\,c\,b\,e\,d\,a$, which provides Sam with the optimal route of Addison to Chelsea to Bristol to Essex to Dover and back to Addison, for a total at a cost of \$1270.

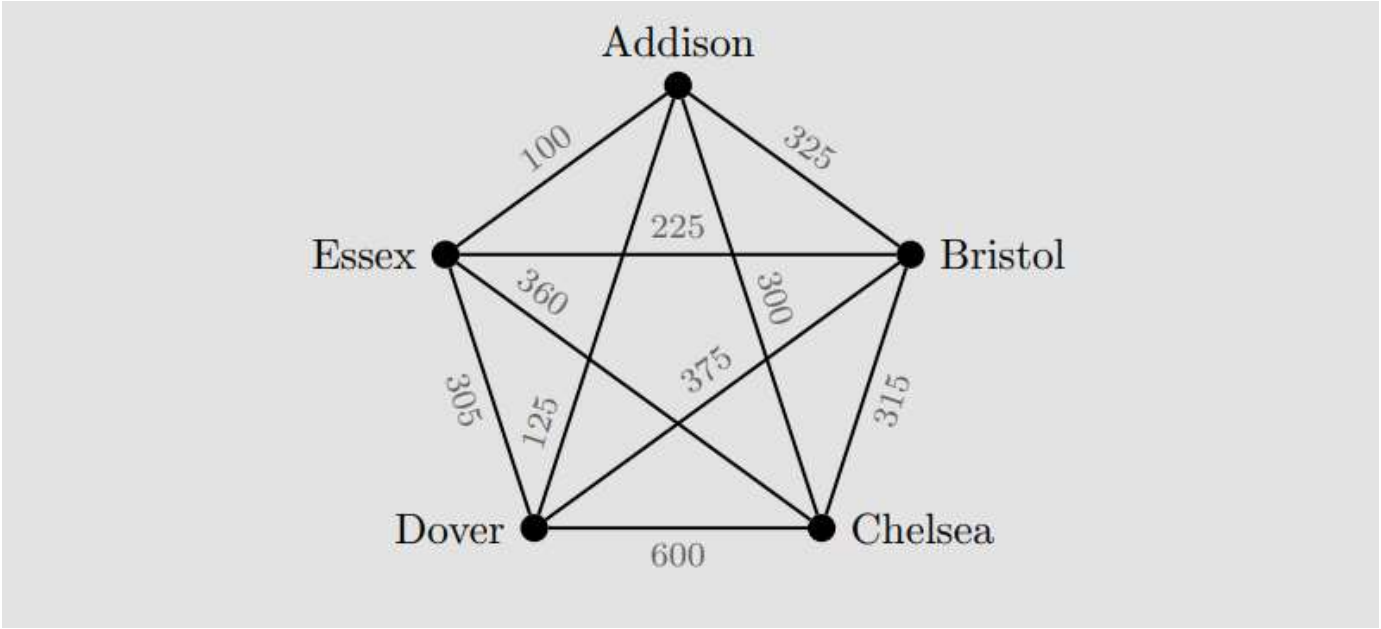
Nearest Neighbor Algorithm

Input: Weighted complete graph K_n.

Steps:

1. Choose a starting vertex, call it v. Highlight v.
2. Among all edges incident to v, pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one.
3. Highlight the edge and move to its other endpoint u. Highlight u.
4. Repeat Steps (2) and (3), where only edges to unhighlighted vertices are considered.
5. Close the cycle by adding the edge to v from the last vertex highlighted. Calculate the total weight.

Output: hamiltonian cycle.

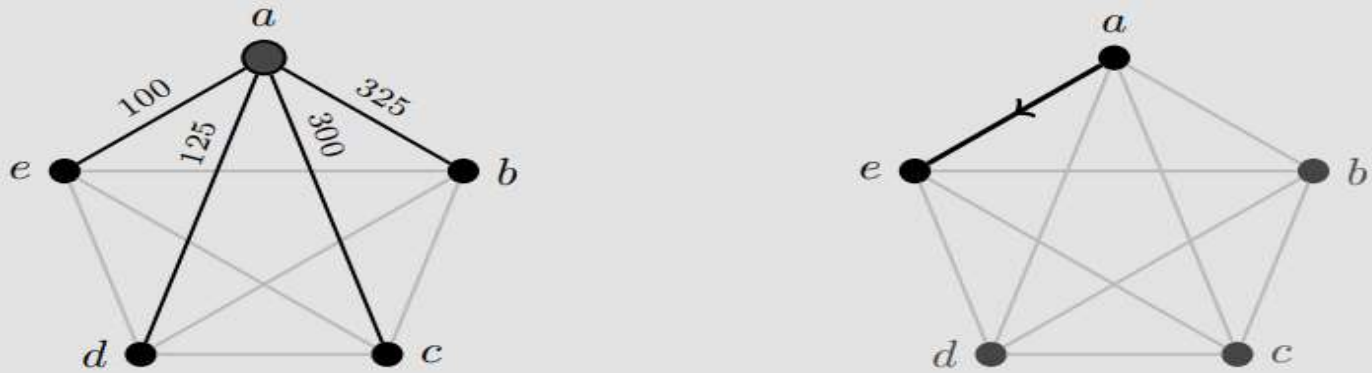

Addison
Essex
Bristol
Dover
Chelsea
100
325
225
360
300
375
305
125
315
600

Example 2.12 Apply the Nearest Neighbor Algorithm to the graph from Example 2.11.

Solution: At each step we will show two copies of the graph. One will indicate the edges under consideration and the other traces the route under construction.

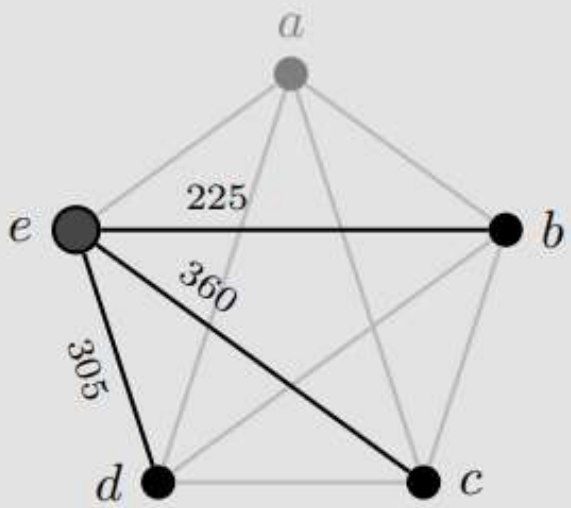
Step 1: The starting vertex is a.

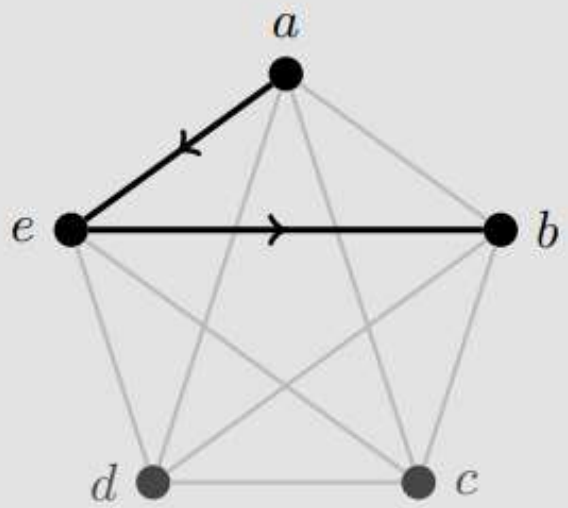
Step 2: The edge of smallest weight incident to a is ae with weight 100.



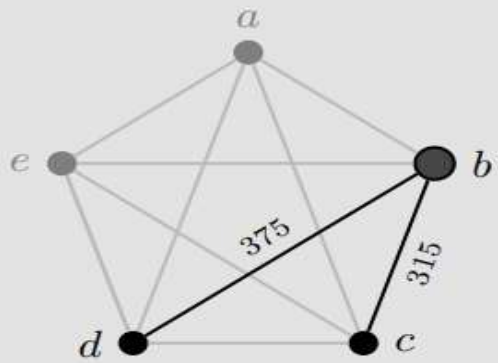
Step 3: From e we only consider edges to b, c, or d. Choose edge eb with weight 225.

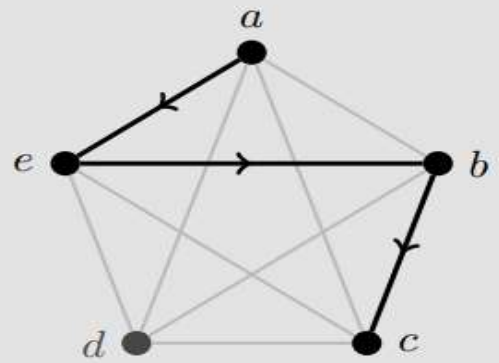
Step 3: From e we only consider edges to b, c, or d. Choose edge eb with weight 225.



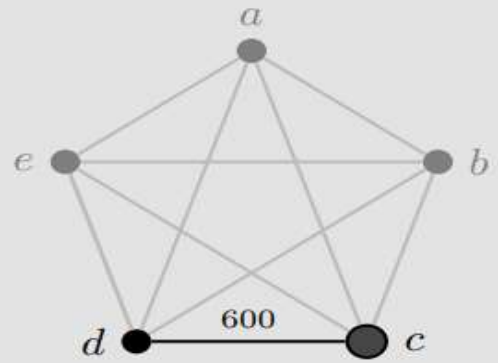


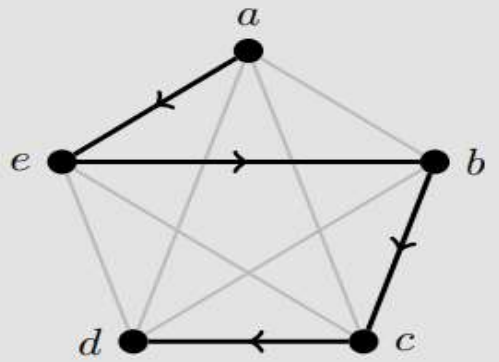
Step 4: From b we consider the edges to c or d. The edge of smallest weight is bc with weight 315.



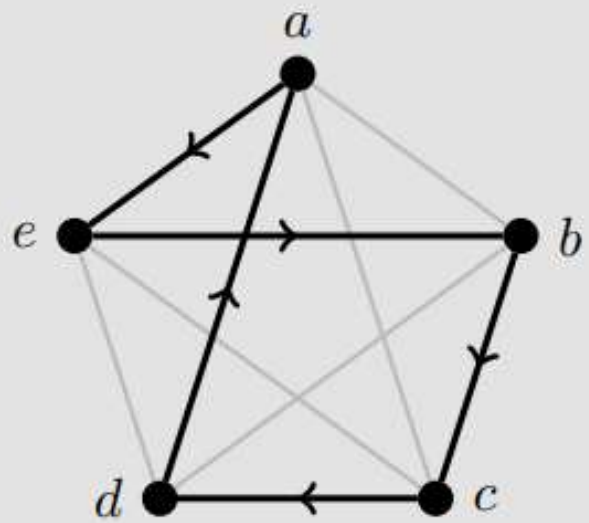


Step 5: Even though cd does not have the smallest weight among all edges incident to c, it is the only choice available.





Step 6: Close the circuit by adding da.



Output: The circuit is $a\,e\,b\,c\,d\,a$ with total weight 1365.

In the previous example, the final circuit was Addison to Essex to Bristol to Chelsea to Dover and back to Addison. Although Nearest Neighbor did not find the optimal circuit of total cost \$1270, it did produce a fairly good circuit with total cost \$1365. Perhaps most important was the speed with which Nearest Neighbor found this circuit.

Repetitive Nearest Neighbor Algorithm

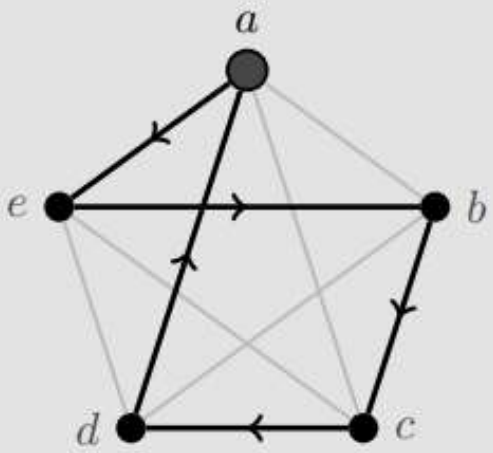
Input: Weighted complete graph K_n.

Steps:

1. Choose a starting vertex, call it v.
2. Apply the Nearest Neighbor Algorithm.
3. Repeat Steps (1) and (2) so each vertex of K_n serves as the starting vertex.
4. Choose the cycle of least total weight. Rewrite it with the desired reference point.

Output: hamiltonian cycle.

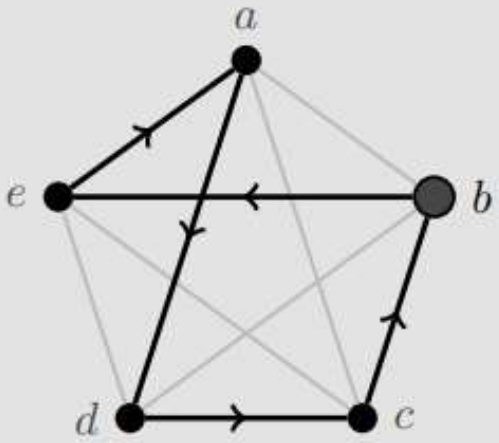
Solution: The five cycles are shown below, with the original name, the rewritten form with a as the reference point, and the total weight of the cycle. You should notice that the cycle starting at d is the same as the one starting at a, and the cycle starting at b is their reversal.



$a\,e\,b\,c\,d\,a$

$a\,e\,b\,c\,d\,a$

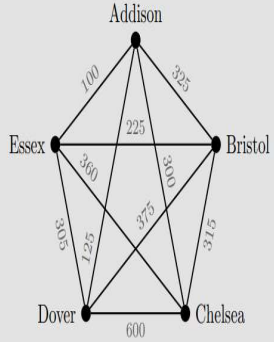
1365

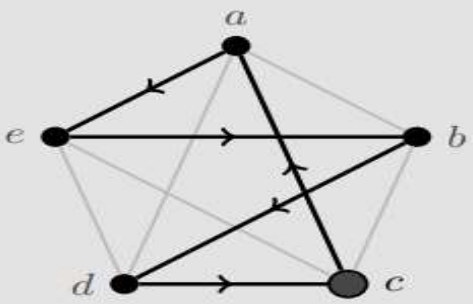


$b\,e\,a\,d\,c\,b$

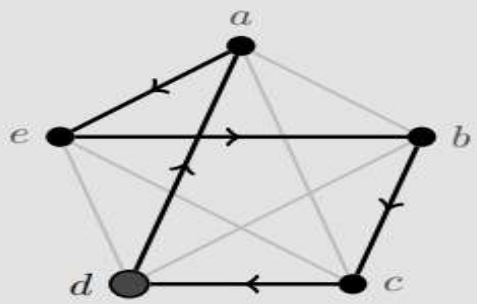
$a\,d\,c\,b\,e\,a$

1365

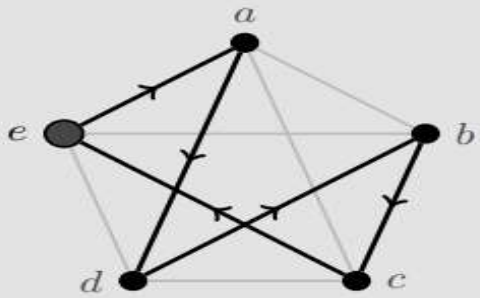




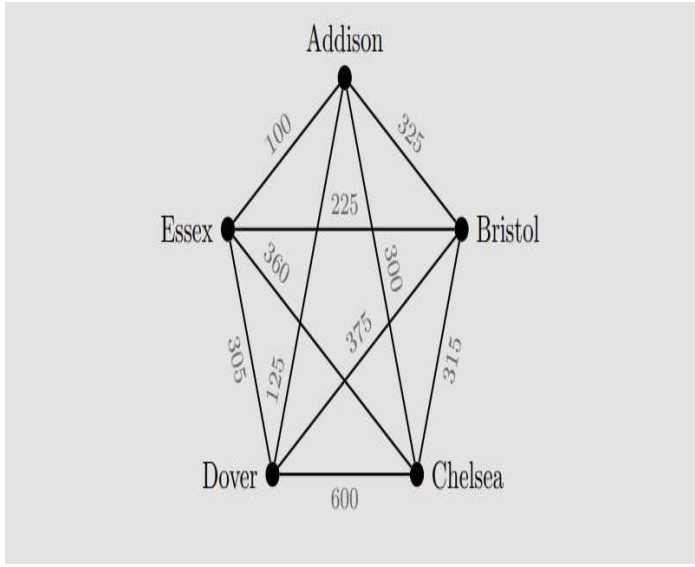
$caebdc$
$aebdca$
1600



$daebcd$
$aebcda$
1365



$eadbce$
$adbcea$
1275



It should come as no surprise that Repetitive Nearest Neighbor performs better than Nearest Neighbor; however, there is no guarantee that this improvement will produce the optimal cycle

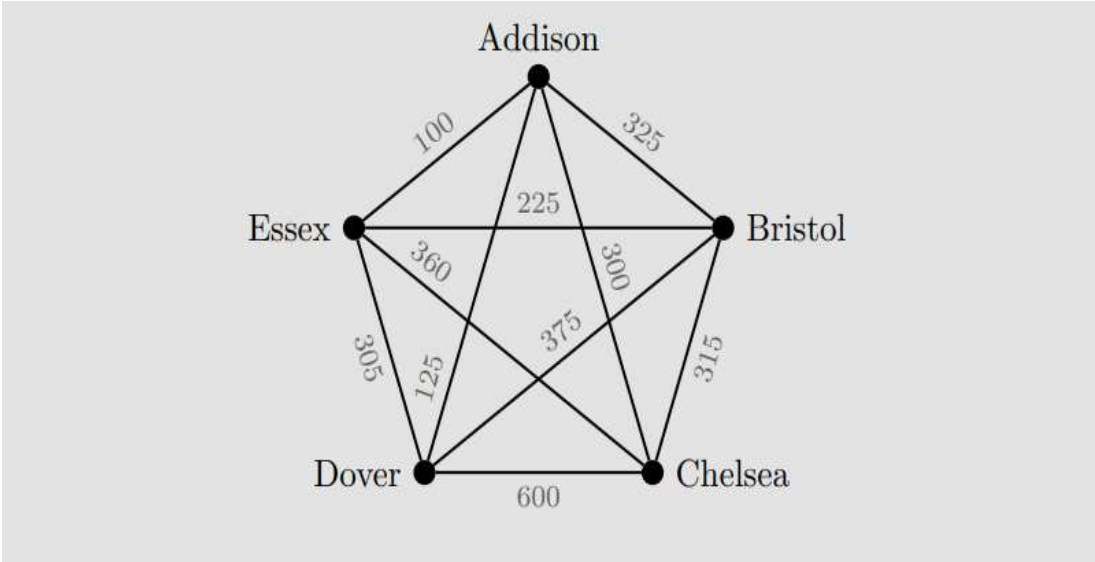
Cheapest Link Algorithm

Input: Weighted complete graph K_n.

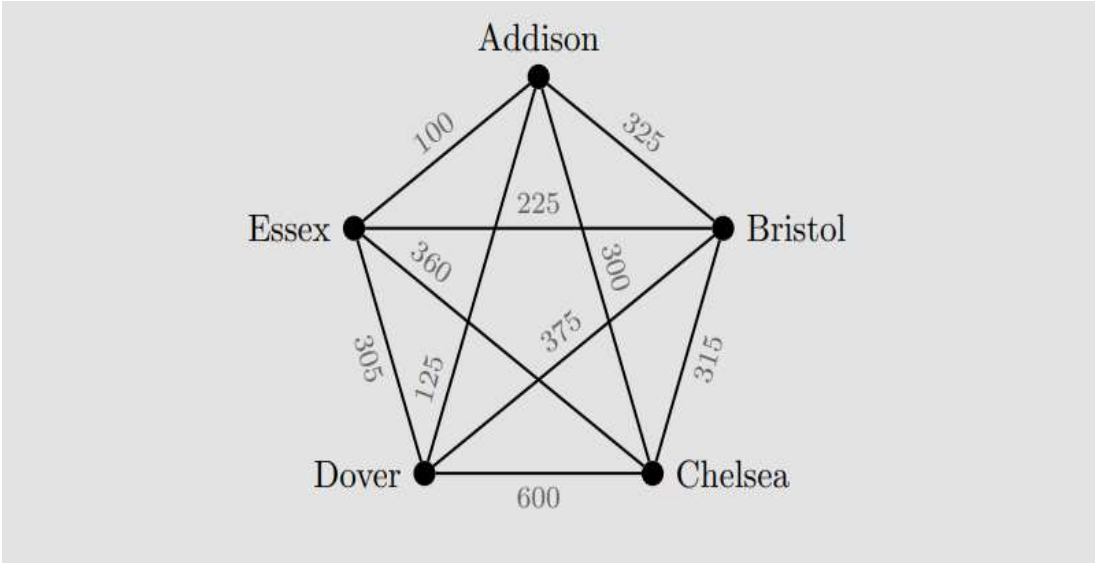
Steps:

1. Among all edges in the graph pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one. Highlight the edge.
2. Repeat Step (1) with the added conditions:
   (a) no vertex has three highlighted edges incident to it; and
   (b) no edge is chosen so that a cycle closes before hitting all the vertices.
3. Calculate the total weight.

Output: hamiltonian cycle.



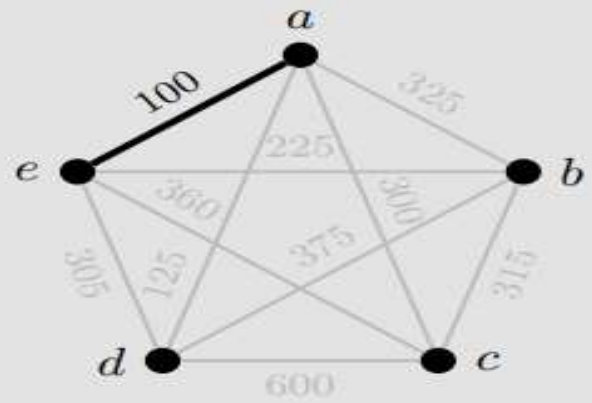
Example 2.14 Apply the Cheapest Link Algorithm to the graph from Example 2.11.



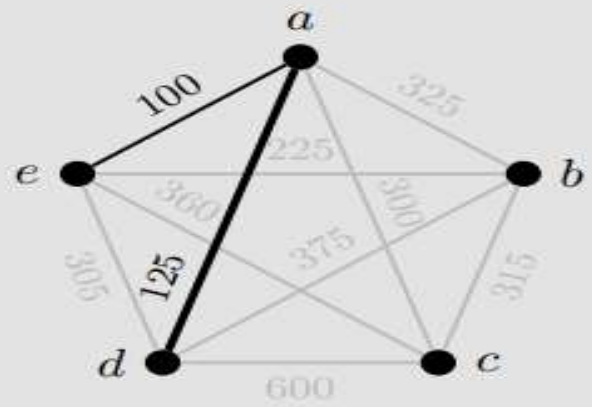
Example 2.14 Apply the Cheapest Link Algorithm to the graph from Example 2.11.

Solution: In each step shown, unchosen edges are shown in gray, previously chosen edges are in black, and the newly chosen edge in bold. An edge that is skipped will be marked with an X.

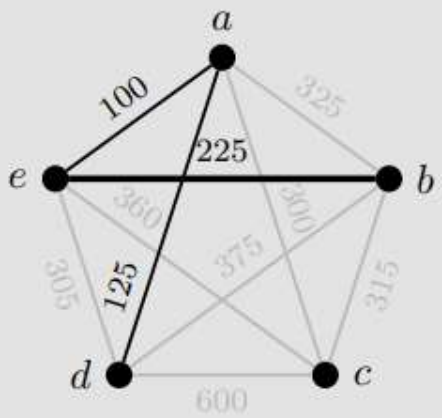
Step 1: The smallest weight is 100 for edge ae.



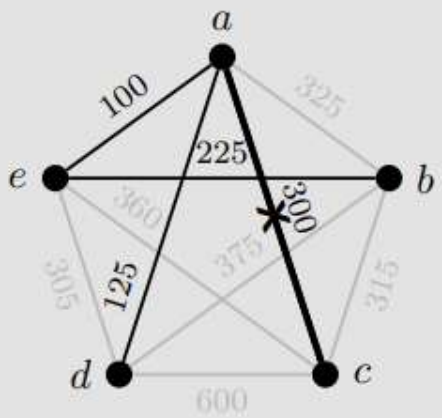
Step 2: The next smallest weight is 125 with edge ad.



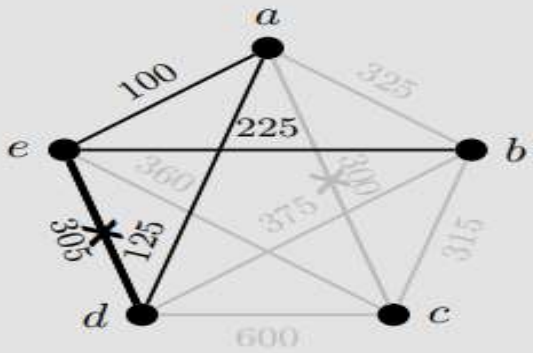
Step 3: The next smallest weight is 225 for edge be.



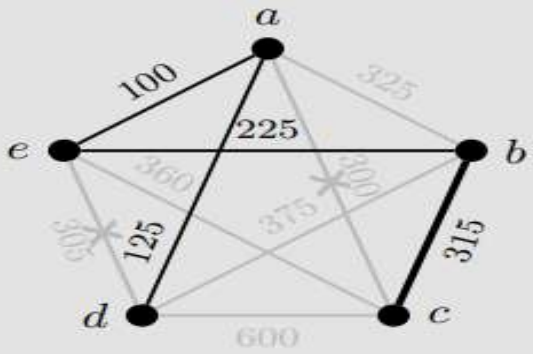
Step 4: Even though ac has weight 300, we must bypass it as it would force a to have three incident edges that are highlighted.



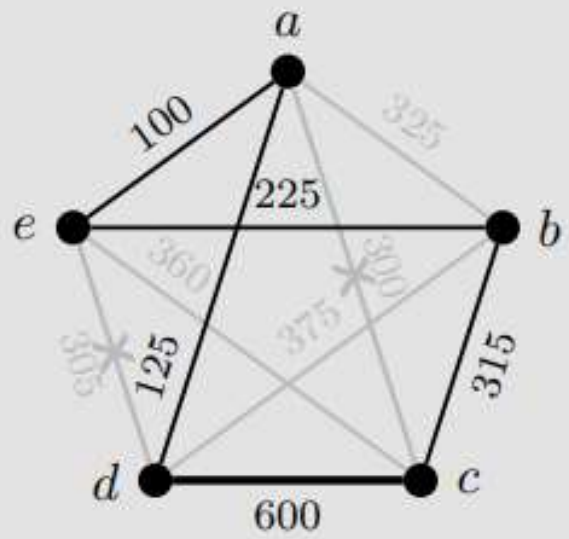
Step 5: The next smallest weight is 305 for edge ed, but again we must bypass it as it would close a cycle too early as well as force e to have three incident edges that are highlighted.



Step 6: The next available is bc with weight 315.



Step 7: At this point, we must close the cycle with the only one choice of cd.



Output: The resulting cycle is $a\,e\,b\,c\,d\,a$ with total weight 1365.

Nearest Insertion Algorithm

Input: Weighted complete graph K_n.

Steps:

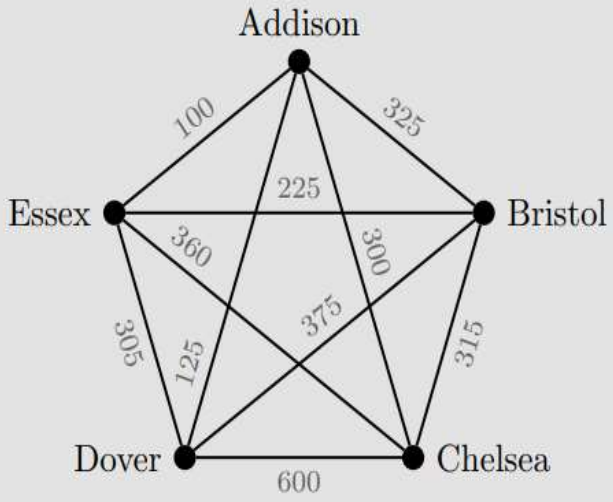
1. Among all edges in the graph, pick the one with the smallest weight. If two possible choices have the same weight, you may randomly pick one. Highlight the edge and its endpoints.
2. Pick a vertex that is closest to one of the two already chosen vertices. Highlight the new vertex and its edges to both of the previously chosen vertices.
3. Pick a vertex that is closest to one of the three already chosen vertices. Calculate the increase in weight obtained by adding two new edges and deleting a previously chosen edge. Choose the scenario with the smallest total. For example, if the cycle obtained from (2) was $a-b-c-a$ and d is the new vertex that is closest to c, we calculate:

   $$w(dc) + w(db) - w(cb) \text{ and } w(dc) + w(da) - w(ca)$$

   and choose the option that produces the smaller total.
4. Repeat Step (3) until all vertices have been included in the cycle.
5. Calculate the total weight.

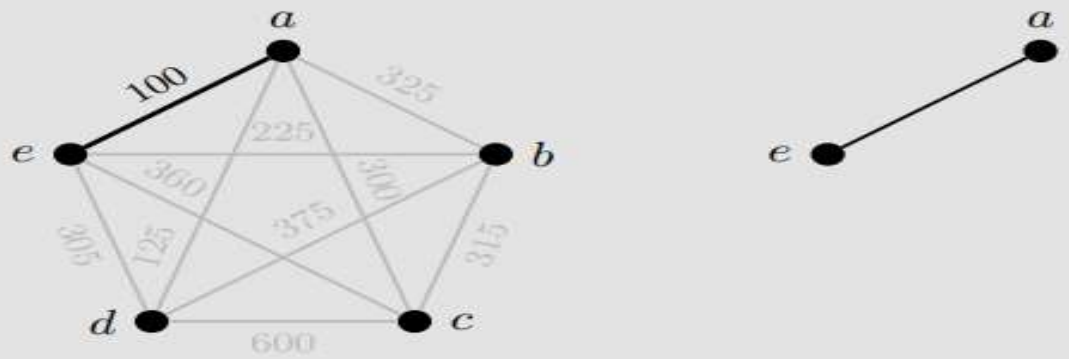
Output: hamiltonian cycle.

Example 2.15 Apply the Nearest Insertion Algorithm to the graph from Example 2.11.

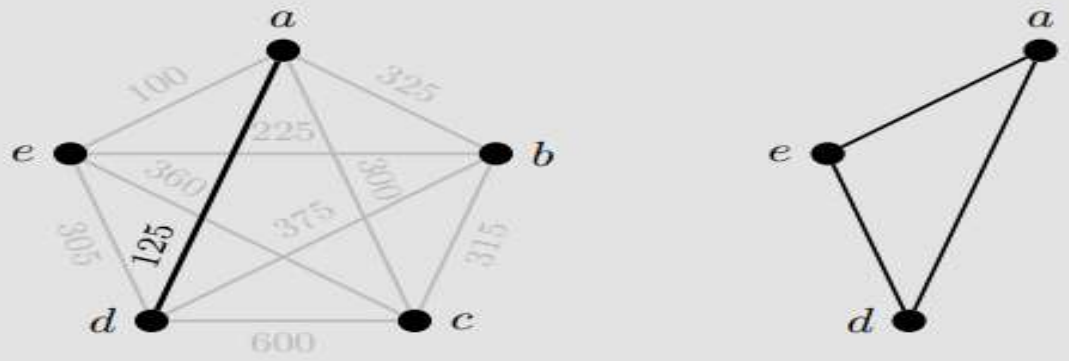


Solution: At each step shown, the graph on the left highlights the edge being added and the graph on the right shows how the cycle is built.

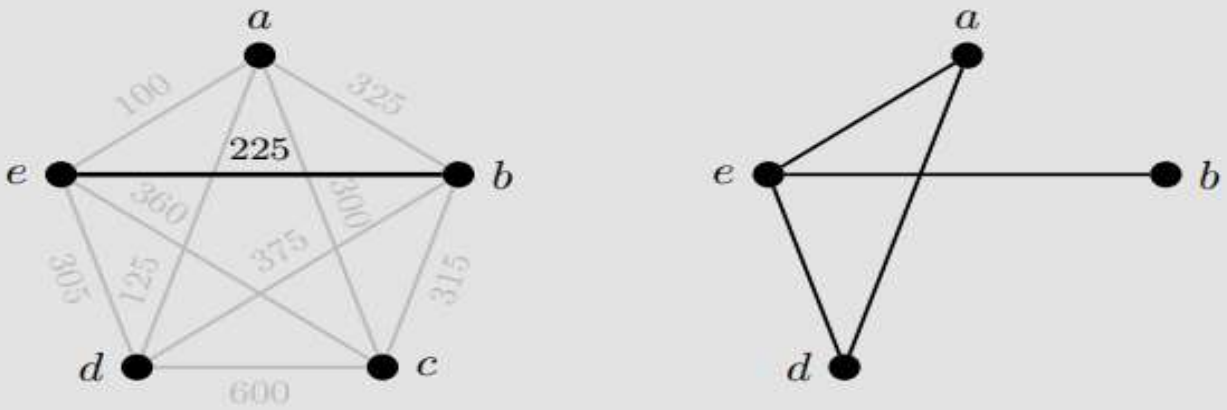
Step 1: The smallest weight edge is ae at 100.



Step 2: The closest vertex to either a or e is d through the edge ad of weight 125. Form a cycle by adding ad and de.



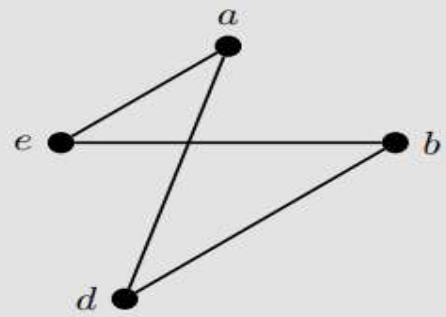
Step 3: The closest vertex to any of a, d, or e is b through the edge be with weight 225.



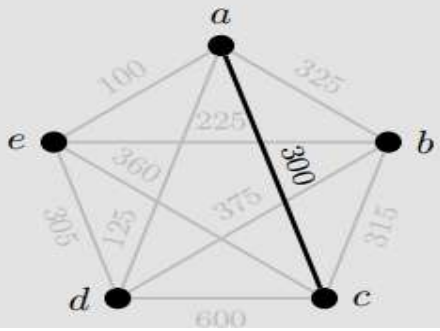
In adding edge be, either ae or de must be removed so that only two edges are incident to e. To determine which is the better choice, compute the following expressions:

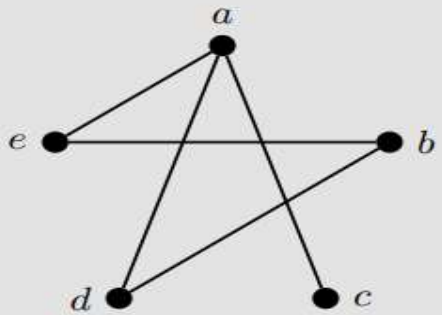
$$be + ba - ea = 225 + 325 - 100 = 450$$
$$be + bd - ed = 225 + 375 - 305 = 295$$

Since the second total is smaller, we create a larger cycle by adding edge bd and removing ed.



Step 4: The only vertex remaining is c, and the minimum edge to the other vertices is ac with weight 300.

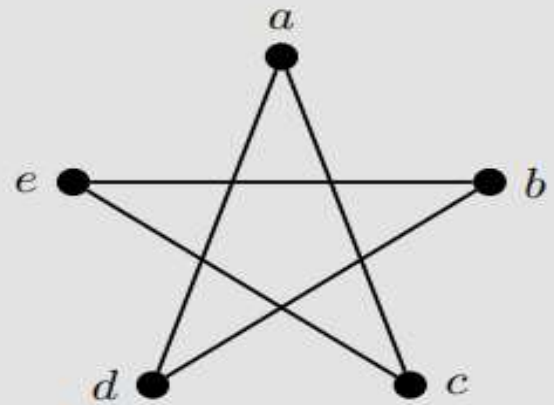




Either ae or ad must be removed. As in the previous step, we compute the following expressions:

$$ca + cd - ad = 300 + 600 - 125 = 775$$
$$ca + ce - ae = 300 + 360 - 100 = 560$$

The second total is again smaller, so we add ce and remove ae.



Output: The cycle is $a\,c\,e\,b\,d\,a$ with total weight 1385.

In the example above, Nearest Insertion performed slightly worse than Cheapest Link and Repetitive Nearest Neighbor. Among all three algorithms, Repetitive Nearest Neighbor found the cycle closest to optimal. In general, when comparing algorithm performance we focus less on absolute error ($1275 - 1270 = 5$) but rather on relative error.

Definition 2.19 The ***relative error*** for a solution is given by

$$\epsilon_r = \frac{Solution - Optimal}{Optimal}$$

Example 2.16 Find the relative error for each of the algorithms performed on the graph from Example 2.11.

Solution:

- Repetitive Nearest Neighbor: $\epsilon_r = \dfrac{1275 - 1270}{1270} = 0.003937 \approx 0.39\%$
- Cheapest Link: $\epsilon_r = \dfrac{1365 - 1270}{1270} = 0.074803 \approx 7.48\%$
- Nearest Insertion: $\epsilon_r = \dfrac{1385 - 1270}{1270} = 0.090551 \approx 9.05\%$

National University of computer and emerging sciences

Shortest Path

Foundation for Advancement of Science and Technology

Dijkstra's Algorithm

Input: Weighted connected simple graph $G = (V, E, w)$ and designated *Start* vertex.

Steps:

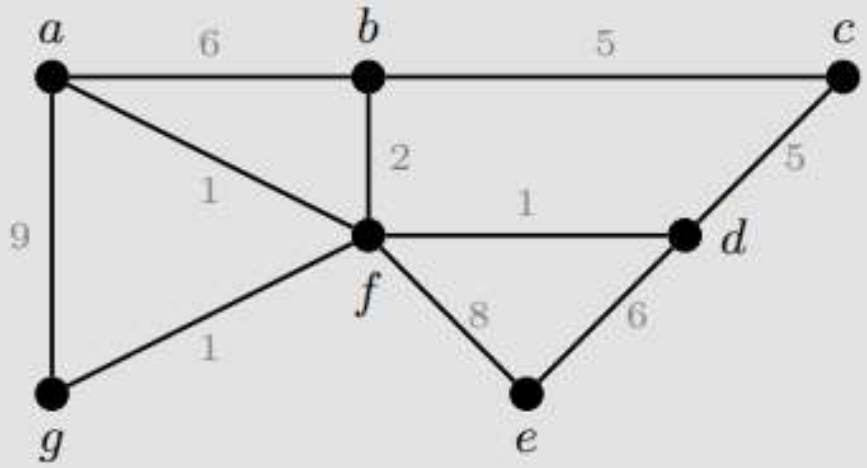
1. For each vertex x of G, assign a label $L(x)$ so that $L(x) = (-, 0)$ if $x = Start$ and $L(x) = (-, \infty)$ otherwise. Highlight *Start*.
2. Let $u = Start$ and define F to be the neighbors of u. Update the labels for each vertex v in F as follows:

   if $w(u) + w(uv) < w(v)$, then redefine $L(v) = (u, w(u) + w(uv))$

   otherwise do not change $L(v)$
3. Highlight the vertex with lowest weight as well as the edge uv used to update the label. Redefine $u = v$.
4. Repeat Steps (2) and (3) until each vertex has been reached. In all future iterations, F consists of the un-highlighted neighbors of all previously highlighted vertices and the labels are updated only for those vertices that are adjacent to the last vertex that was highlighted.
5. The shortest path from *Start* to any other vertex is found by tracing back using the first component of the labels. The total weight of the path is the weight given in the second component of the ending vertex.

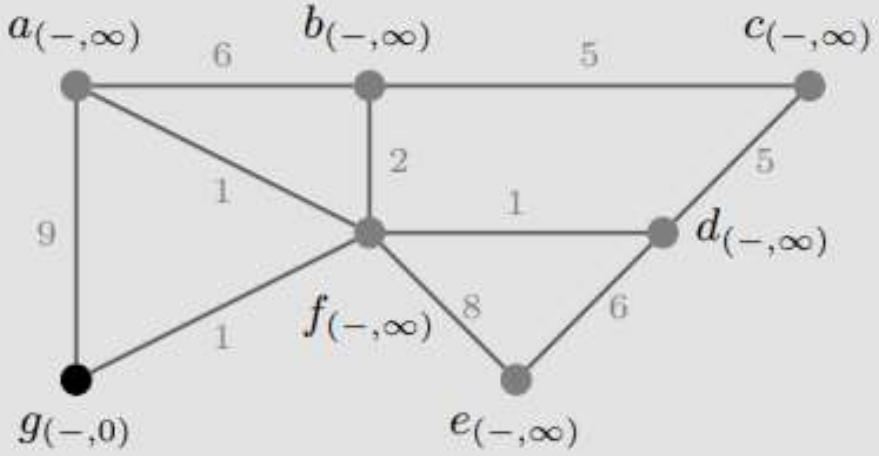
Output: Highlighted path from *Start* to any vertex x of weight $w(x)$.

Example 2.17 Apply Dijkstra's Algorithm to the graph below where $Start = g$.



Solution: In each step, the label of a vertex will be shown in the table on the right.

Step 1: Highlight g. Define $L(g) = (-, 0)$ and $L(x) = (-, \infty)$ for all $x = a, \cdots, f$.



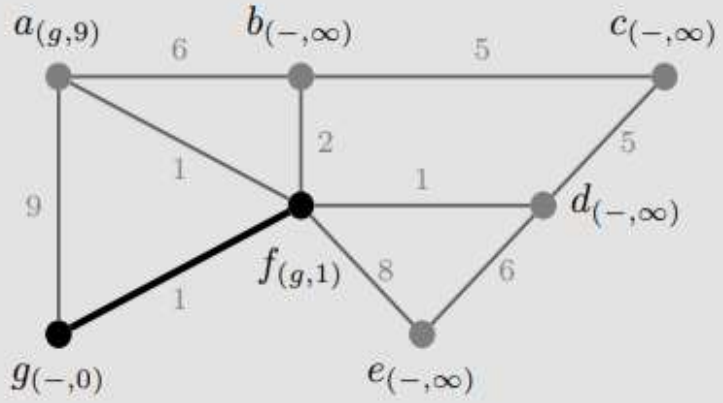
$F = \{\}$

a	$(-, \infty)$
b	$(-, \infty)$
c	$(-, \infty)$
d	$(-, \infty)$
e	$(-, \infty)$
f	$(-, \infty)$
g	$(-, 0)$

Step 2: Let $u = g$. Then the neighbors of g comprise $F = \{a, f\}$. We compute

$$w(g) + w(ga) = 0 + 9 = 9 < \infty = w(a)$$
$$w(g) + w(gf) = 0 + 1 = 1 < \infty = w(f)$$

Update $L(a) = (g, 9)$ and $L(f) = (g, 1)$. Since the minimum weight for all vertices in F is that of f, we highlight the edge gf and the vertex f.



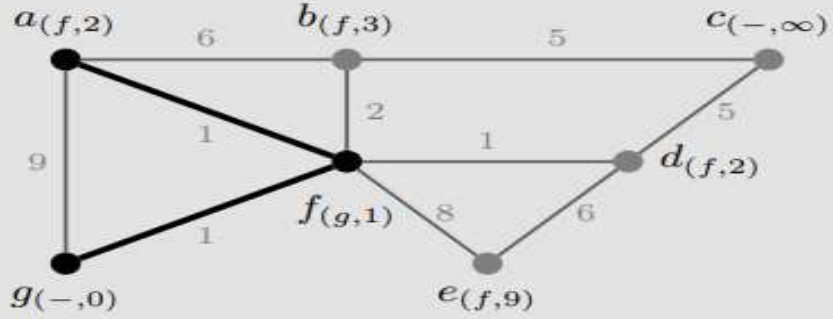
$F = \{a, f\}$

a	$(-,\infty) \to (g,9)$
b	$(-,\infty)$
c	$(-,\infty)$
d	$(-,\infty)$
e	$(-,\infty)$
f	$(-,\infty) \to (g,1)$
g	$(-,0)$

Step 3: Let $u = f$. Then the neighbors of all highlighted vertices are $F = \{a, b, d, e\}$. We compute

$$w(f) + w(fa) = 1 + 1 = 2 < 9 = w(a)$$
$$w(f) + w(fb) = 1 + 2 = 3 < \infty = w(b)$$
$$w(f) + w(fd) = 1 + 1 = 2 < \infty = w(d)$$
$$w(f) + w(fe) = 1 + 8 = 9 < \infty = w(e)$$

Update $L(a) = (f, 2)$, $L(b) = (f, 3)$, $L(d) = (f, 2)$ and $L(e) = (f, 9)$. Since the minimum weight for all vertices in F is that of a or d, we choose to highlight the edge fa and the vertex a.



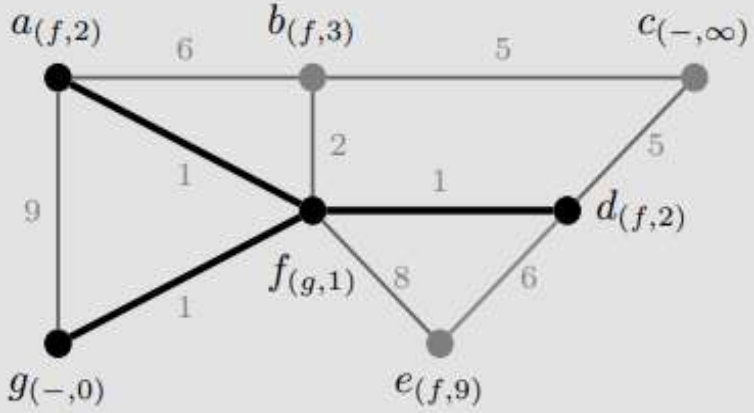
$F = \{a, b, d, e\}$

a	$(g, 9) \to (f, 2)$
b	$(-, \infty) \to (f, 3)$
c	$(-, \infty)$
d	$(-, \infty) \to (f, 2)$
e	$(-, \infty) \to (f, 9)$
f	$(g, 1)$
g	$(-, 0)$

Step 4: Let $u = a$. Then the neighbors of all highlighted vertices are $F = \{b, d, e\}$. Note, we only consider updating the label for b since this is the only vertex adjacent to a, the vertex highlighted in the previous step.

$$w(a) + w(ba) = 2 + 6 = 8 \not< 2 = w(b)$$

We do not update the label for b since the computation above is not less than the current weight of b. The minimum weight for all vertices in F is that of d, and so we highlight the edge fd and the vertex d.



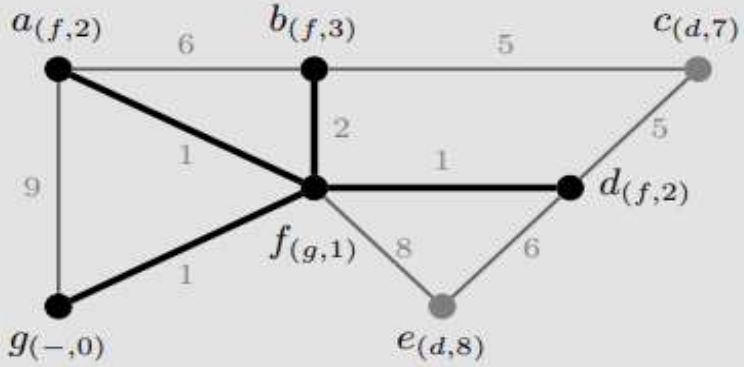
$F = \{b, d, e\}$

a	$(f, 2)$
b	$(f, 3)$
c	$(-, \infty)$
d	$(f, 2)$
e	$(f, 9)$
f	$(g, 1)$
g	$(-, 0)$

Step 5: Let $u = d$. Then the neighbors of all highlighted vertices are $F = \{b, c, e\}$. We compute

$$w(d) + w(dc) = 2 + 5 = 7 < \infty = w(c)$$
$$w(d) + w(de) = 2 + 6 = 8 < 9 = w(e)$$

Update $L(c) = (d, 7)$ and $L(e) = (d, 8)$. Since the minimum weight for all vertices in F is that of b, we highlight the edge bf and the vertex b.



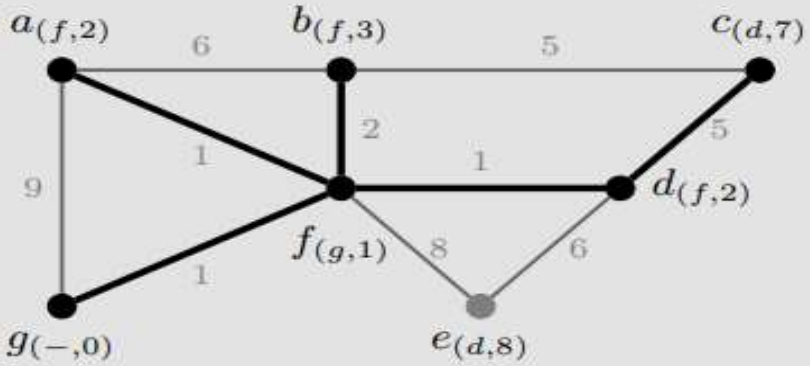
$F = \{b, c, e\}$

a	$(f, 2)$
b	$(f, 3)$
c	$(-, \infty) \to (d, 7)$
d	$(f, 2)$
e	$(f, 9) \to (d, 8)$
f	$(g, 1)$
g	$(-, 0)$

Step 6: Let $u = b$. Then the neighbors of all highlighted vertices are $F = \{c, e\}$. However, we only consider updating the label of c since e is not adjacent to b. Since

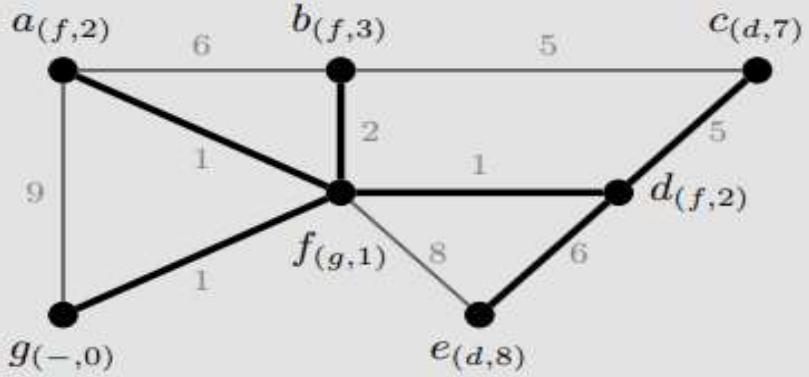
$$w(b) + w(bc) = 3 + 5 = 8 \not< 7 = w(c)$$

we do not update the labels of any vertices. Since the minimum weight for all vertices in F is that of c we highlight the edge dc and the vertex c. This terminates the iterations of the algorithm since our ending vertex has been reached.



$F = \{c, e\}$	
a	$(f, 2)$
b	$(f, 3)$
c	$(d, 7)$
d	$(f, 2)$
e	$(d, 8)$
f	$(g, 1)$
g	$(-, 0)$

Step 7: Let $u = c$. Then the neighbors of all highlighted vertices are $F = \{e\}$. However, we do not need to update any labels since c and e are not adjacent. Thus we highlight the edge de and the vertex e. This terminates the iterations of the algorithm since all vertices are now highlighted.



$F = \{e\}$	
a	$(f, 2)$
b	$(f, 3)$
c	$(d, 7)$
d	$(f, 2)$
e	$(d, 8)$
f	$(g, 1)$
g	$(-, 0)$

Output: The shortest paths from g to all other vertices can be found highlighted above. For example the shortest path from g to c is $g\,f\,d\,c$ and has a total weight 7, as shown by the label of c.