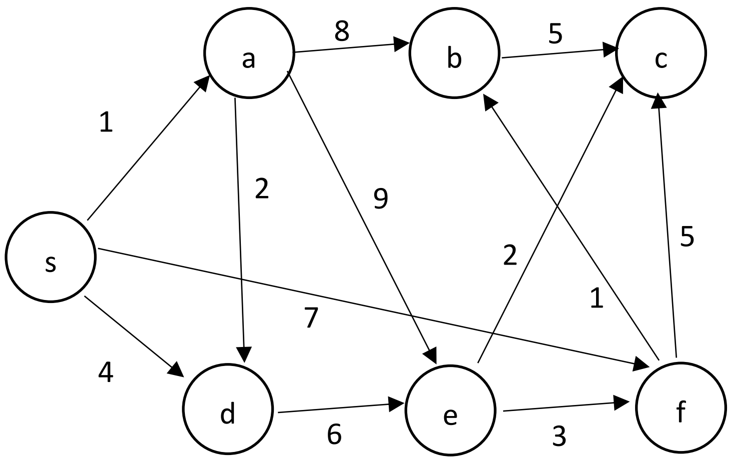
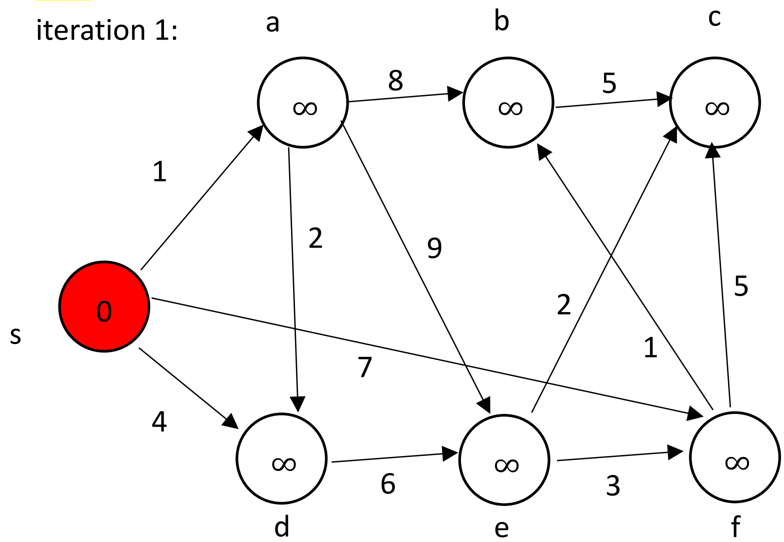
Algorithm 2021 Spring HW4 Solution

(15pts)1. Use Dijkstra's algorithm to find the shortest paths from vertex s to other vertices. (You need to show your process.)



Ans:

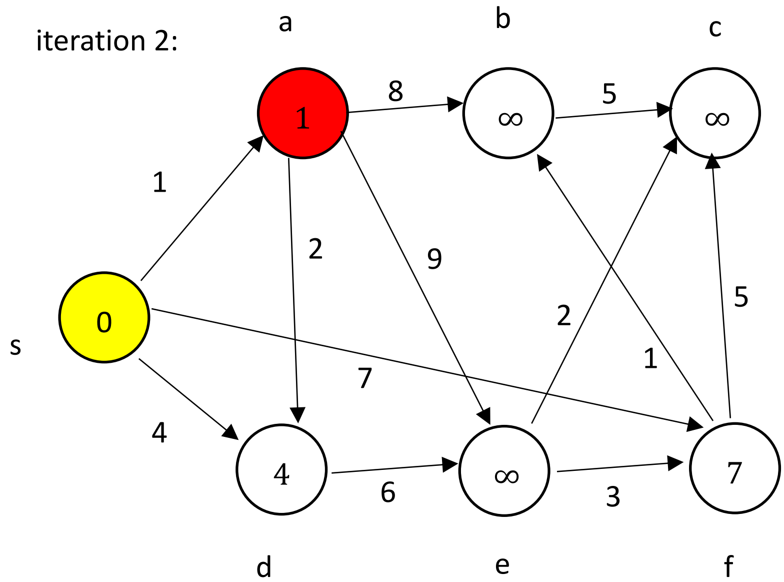
iteration 1:



Q: s, a, b, c, d, e, f

S:

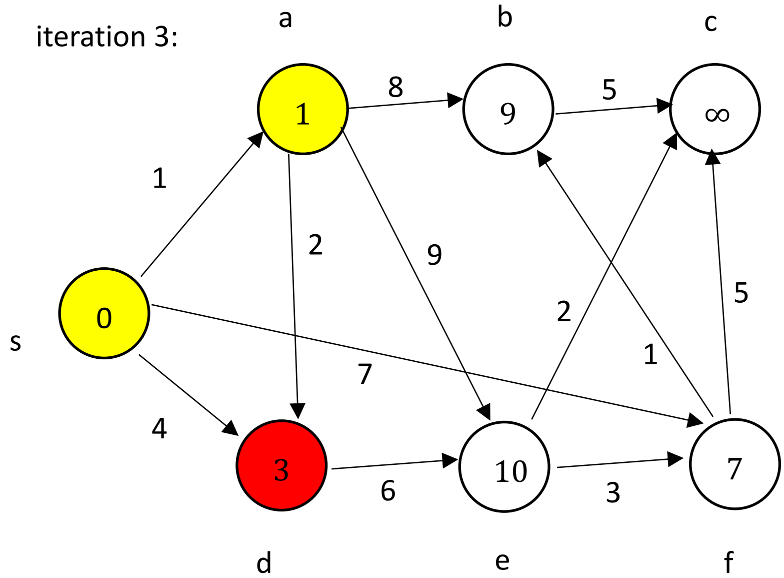
iteration 2:



Q: a, b, c, d, e, f

S: s

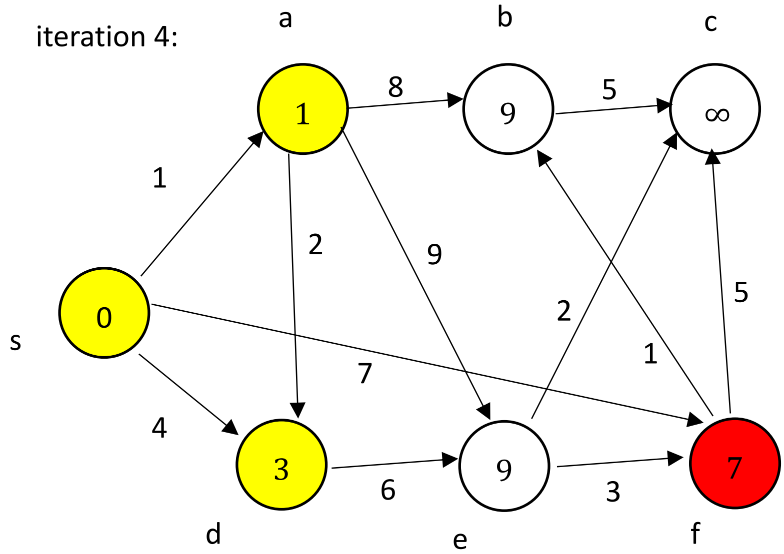
iteration 3:



Q: b, c, d, e, f

S: s, a

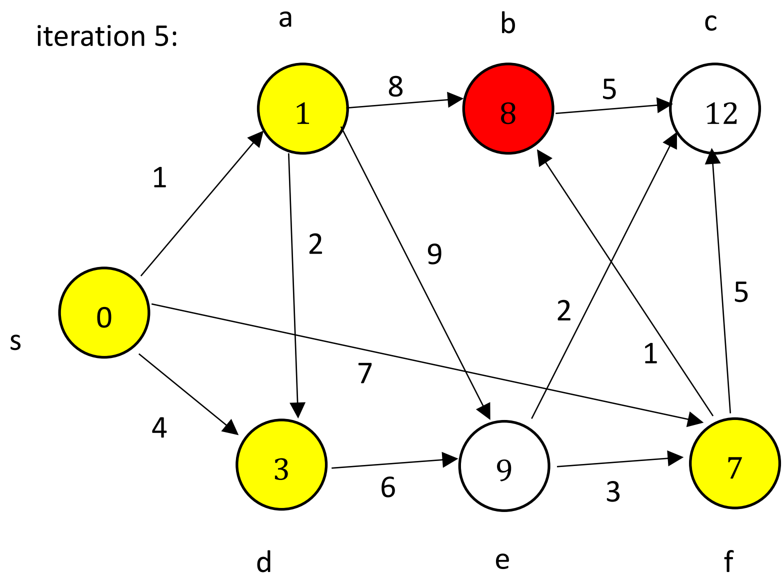
iteration 4:



Q: b, c, e, f

S: s, a, d

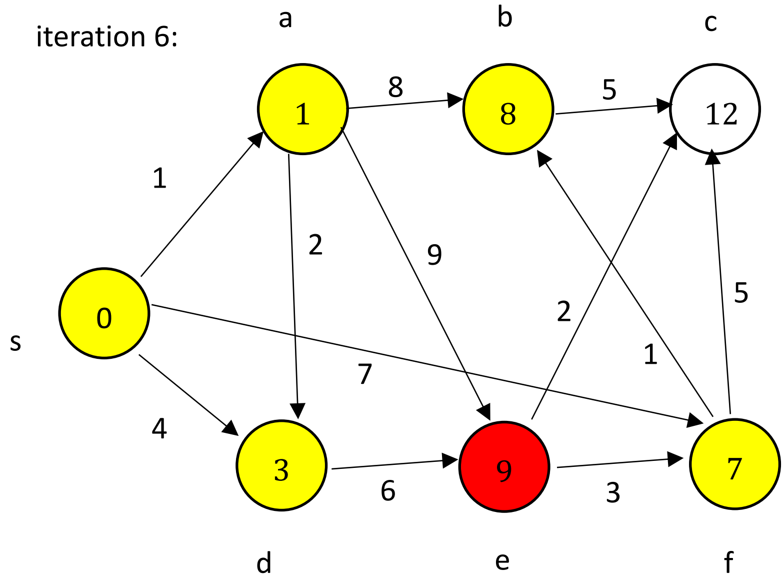
iteration 5:



Q: b, c, e

S: s, a, d, f

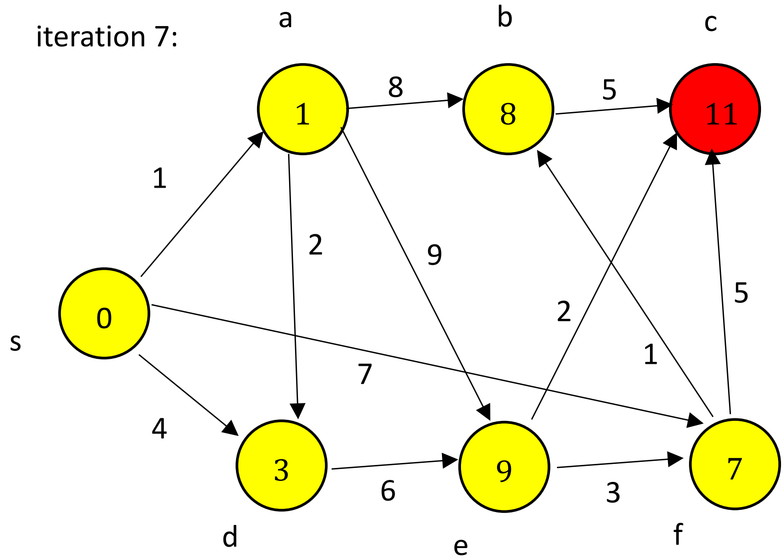
iteration 6:



Q: c, e

S: s, a, d, f, b

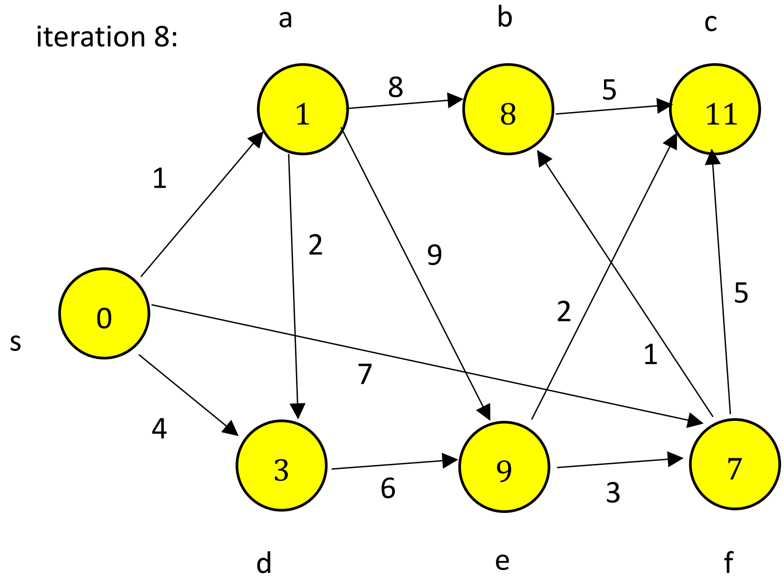
iteration 7:



Q: c

S: s, a, d, f, b, e

iteration 8:



Q:

S: s, a, d, f, b, e, c

(10pts)2. Given the DFS code below, please fill in the blanks.

DFS(G):

```
for each vertex u ∈ G.V
        u.color = WHITE
        u.π = NIL
time = 0
for each vertex u ∈ G.V
        if u.color == WHITE
                DFS-VISIT(G, u)
```

DFS-VISIT (G, u):

```
time = time + 1
u.d = time
u.color = GRAY
for each vertex v ∈ G.Adj[u]
        if ______(1)______
                print "(u, v) is a tree edge."
                v.π = u
                DFS-VISIT (G, v)
        else if ______(2)______
                print "(u, v) is a back edge."
        else if ______(3)______
                print "(u, v) is a forward edge."
        else
                print "(u, v) is a cross edge."
u.color = BLACK
time = time + 1
u.f = time
```

Ans:

(1) v.color == WHITE

(2) v.color == GRAY

(3) v.d > u.d

(25pts)3. Given a set of requests $\{1,2,\dots,n\}$, i^{th} request corresponds an interval with start time $s(i)$ and finish time $f(i)$ ie. Interval i: $[s(i), f(i))$.

(a) Please give a greedy algorithm to partition these requests into a minimum number of compatible subsets, each corresponds to one resource. (hint: A subset of intervals is compatible if no two intervals overlap)

(b) Please prove that your greedy algorithm is correct and optimal.

Ans:

(a)

Sort the requests by their start times, breaking ties arbitrarily.

Let $I_1, I_2, \dots, I_n$ denote the requests in this order.

For $j = 1, 2, 3 \dots, n$

  For each request I_i that precedes I_j in sorted order and overlaps it

    Exclude the label of I_i from consideration for I_j

  End

  If there is any label from $\{1,2,\dots,d\}$ that has not been excluded then

    Assign a non-excluded label to I_j

  Else

    Leave I_j unlabeled

  Endif

Endfor

(b)

we define the depth of a set of intervals to be the maximum number that pass over any single point on the time-line.

Theorem 1

In any instance of Interval Partitioning, the number of resources needed is at least the depth of the set of intervals.

Proof:

Suppose a set of intervals has depth *d*, and let $I_1, \dots, I_d$ all pass over a common point on the time-line. Then each of these intervals must be scheduled on a different resource, so the whole instance needs at least *d* resources.

Claim 1

If we use the greedy algorithm above, every interval will be assigned a label, and no two overlapping intervals will receive the same label.

Proof:

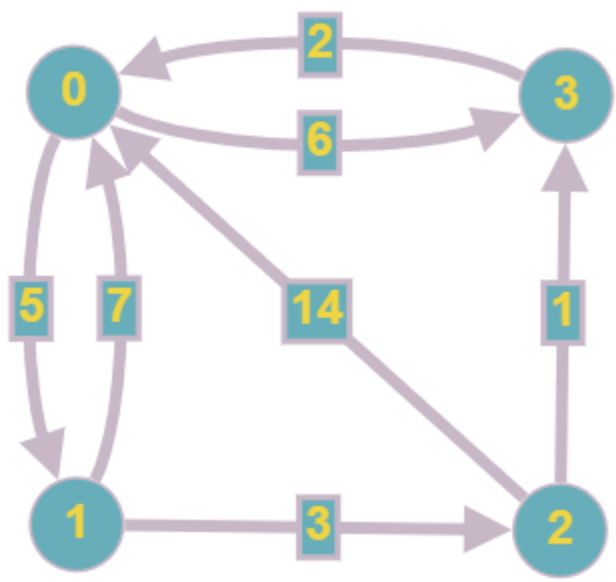
First let's argue that no interval ends up unlabeled. Consider one of the intervals I_j, and suppose there are *t* intervals earlier in the sorted order that overlap it. These t intervals, together with I_j, form a set of $t + 1$ intervals that all pass over a common point on the time-line (namely, the start time of I_j), and so $t + 1 \leq d$. Thus $t \leq d - 1$. It follows that at least one of the d labels is not excluded by this set of t intervals, and so there is a label that can be assigned to I_j.

Next we claim that no two overlapping intervals are assigned the same label. Indeed, consider any two intervals I and I' that overlap, and suppose I precedes I' in the sorted order. Then when I' is considered by the algorithm, I is in the set of intervals whose labels are excluded from consideration; consequently, the algorithm will not assign to I' the label that it used for I.

Essentially, if you have d labels at your disposal, then as you sweep through the intervals from left to right, assigning an available label to each interval you encounter, you can never reach a point where all the labels are currently in use. Since our algorithm is using d labels, we can use Theorem 1 to conclude that it is, in fact, always using the minimum possible number of labels. We sum this up as follows.

The greedy algorithm above schedules every interval on a resource, using a number of resources equal to the depth of the set of intervals. This is the optimal number of resources needed.

(10pts)4. Find all pairs shortest path using Floyd-Warshall.



Ans:

solution		0	1	2	3
	0	0	5	8	6
	1	6	0	3	4
	2	3	8	0	1
	3	2	7	10	0

(15pts)5. There are five cities in a network. The cost of building a road directly between i and j is the entry $\boldsymbol{a_{i,j}}$ in the matrix below. An infinite entry indicates that there is a mountain in the way and the road cannot be built. Determine the least cost of making all the cities reachable from each other.

	0	1	2	3	4
0	0	5	6	11	15
1	5	0	X	2	13
2	6	X	0	7	1
3	11	2	7	0	8
4	15	13	1	8	0

Ans:

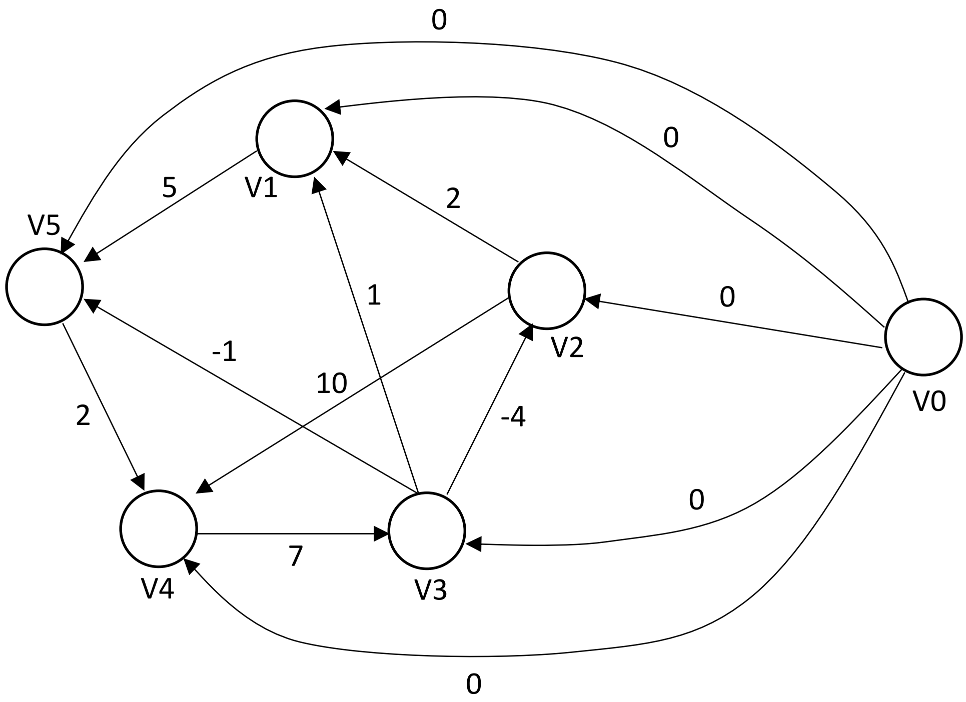
1+2+5+6 = 14

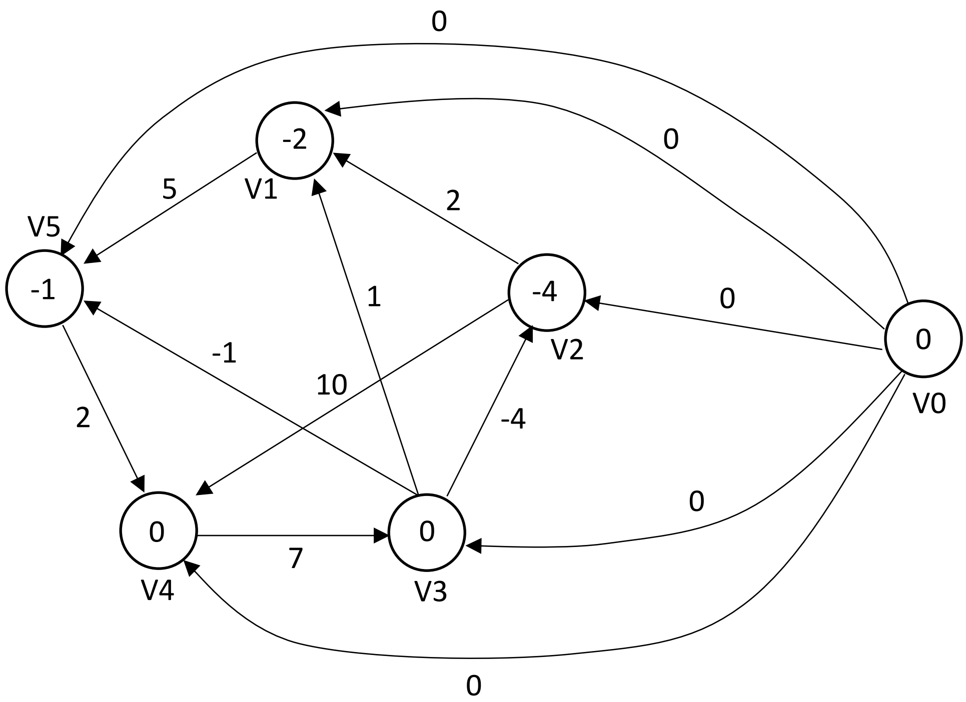
(15pts)6. Find a feasible solution or determine that no feasible solution exists for the following system of difference constraints:

$x_1 - x_3 \leq 1$
$x_2 - x_3 \leq -4$
$x_4 - x_5 \leq 2$
$x_3 - x_4 \leq 7$
$x_5 - x_1 \leq 5$
$x_4 - x_2 \leq 10$
$x_1 - x_2 \leq 2$
$x_5 - x_3 \leq -1$

Ans:

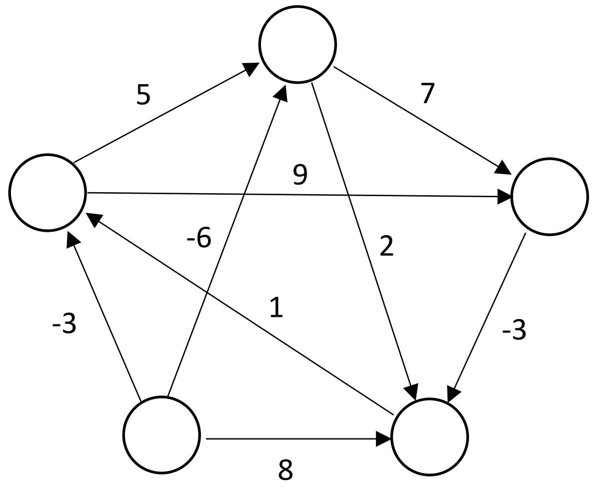
Add the new source v0, then perform Bellman-Ford's.





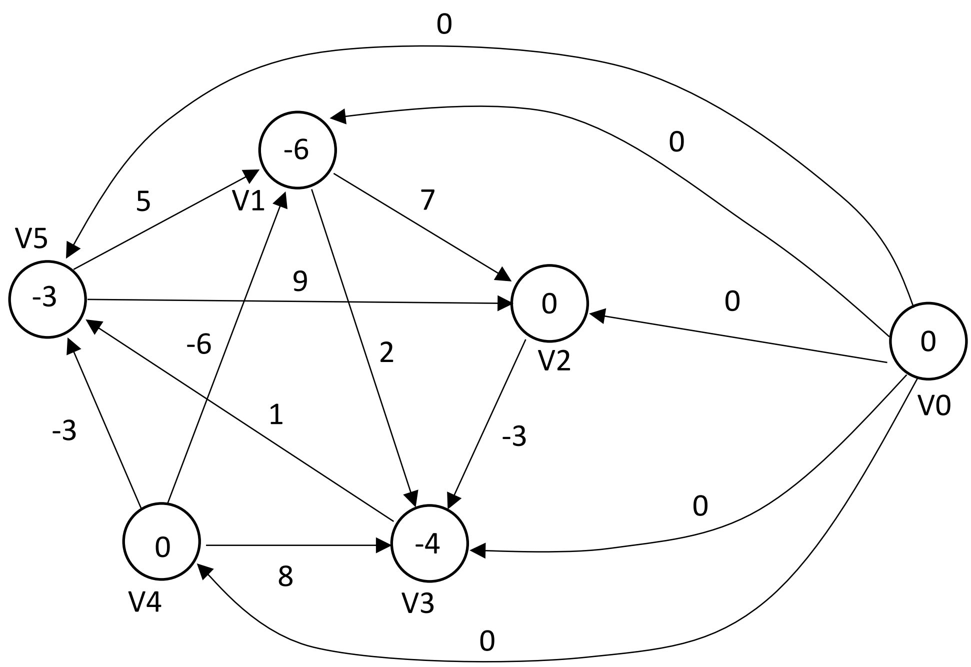
x = (-2, -4, 0, 0, -1) is a feasible solution, so is x+d for any constant d.

(10pts)7. Reweight the edges using the method used in Johnson's algorithm.



Ans:

Add the new source v0, then perform Bellman-Ford's.



For each edge (u, v) do

    w'(u, v) = w(u, v) + h(u) – h(v)

w'(v1, v2) = w(v1, v2) + h(v1) – h(v2)=7-6+0=1

w'(v1, v3) = w(v1, v3) + h(v1) – h(v3)=2-6+4=0

w'(v2, v3) = w(v2, v3) + h(v2) – h(v3)=-3+0+4=1

w'(v3, v5) = w(v3, v5) + h(v3) – h(v5)=1-4+3=0

w'(v4, v1) = w(v4, v1) + h(v4) – h(v1)=-6+0+6=0

$w'(v4, v3) = w(v4, v3) + h(v4) - h(v3) = 8 + 0 + 4 = 12$

$w'(v4, v5) = w(v4, v5) + h(v4) - h(v5) = -3 + 0 + 3 = 0$

$w'(v5, v1) = w(v5, v1) + h(v5) - h(v1) = 5 - 3 + 6 = 8$

$w'(v5, v2) = w(v5, v2) + h(v5) - h(v2) = 9 - 3 + 0 = 6$