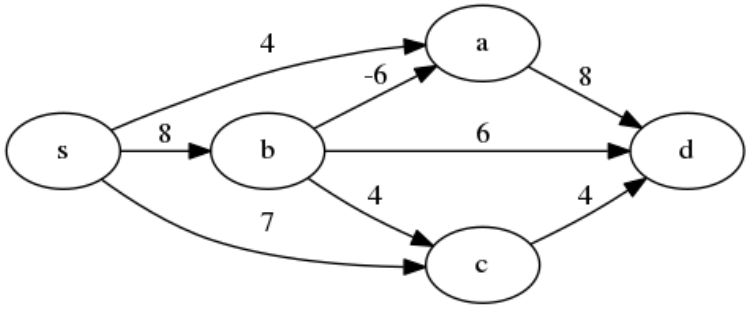
OME: Tutorial, Lecture 2

Question 1



1. Initialise:

$Q = (s, 0), (a, \infty), (b, \infty), (c, \infty), (d, \infty)$

v	s	a	b	c	d
$d[v]$	0	∞	∞	∞	∞
$parent[v]$	nil	nil	nil	nil	nil

2. Relax from s

$Q = (a, 4), (c, 7), (b, 8), (d, \infty)$

v	s	a	b	c	d
$d[v]$	0	4	8	7	∞
$parent[v]$	nil	s	s	s	nil

3. Relax from a

$Q = (c, 7), (b, 8), (d, 12)$

v	s	a	b	c	d
$d[v]$	0	4	8	7	12
$parent[v]$	nil	s	s	s	a

3. Relax from c

$Q = (b, 8), (d, 11)$

v	s	a	b	c	d
$d[v]$	0	4	8	7	11
$parent[v]$	nil	s	s	s	c

4. Relax from b

$Q = (d, 11)$

v	s	a	b	c	d
$d[v]$	0	2	8	7	11
$parent[v]$	nil	b	s	s	c

5. Relax from d

$Q = \varnothing$

v	s	a	b	c	d
$d[v]$	0	2	8	7	11
$parent[v]$	nil	b	s	s	c

The result of Dijkstra's algorithm (above) records the shortest path to d as 11 via $\langle s, c, d \rangle$, but it can be achieved in 10 via $\langle s, b, d \rangle$.

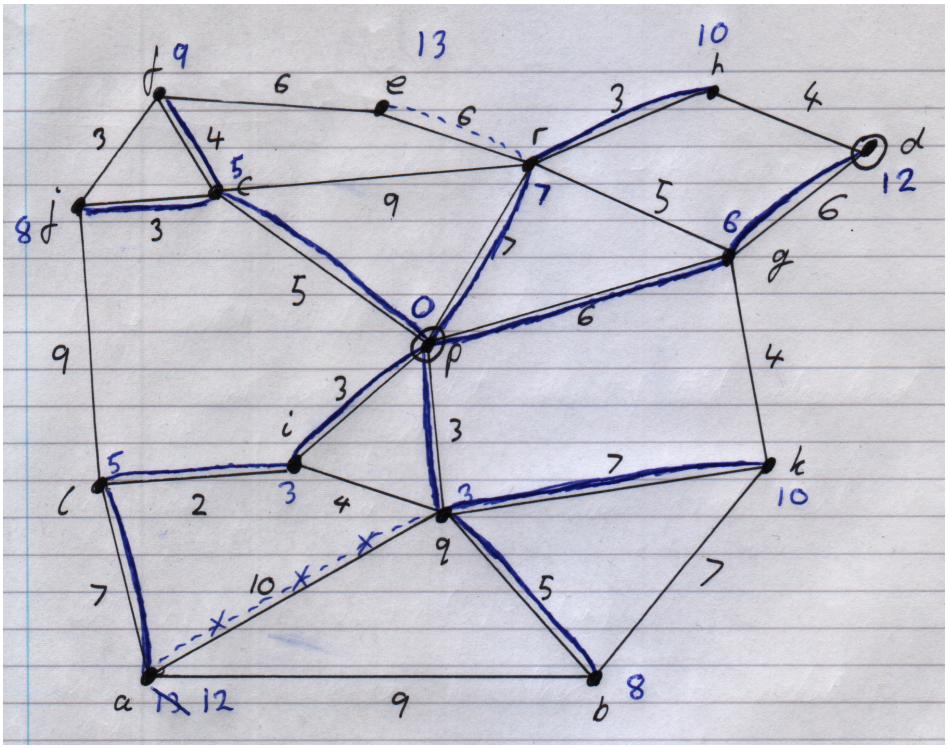
Question 2

```
output = empty list
unmarked = empty set

fun TOPOLOGICAL—SORT(G):
    add all vertices to unmarked

    while (unmarked is not empty):
        n = select vertex from unmarked
        VISIT(n)

fun VISIT(n):
    if (n has a temporary mark):
        return FALSE // not a DAG

    if (n is not marked):
        mark n temporarily
        for each m in adj[n], do:
            visit(m)
        mark n permanently
        remove n from unmarked
        add n to head of L
```

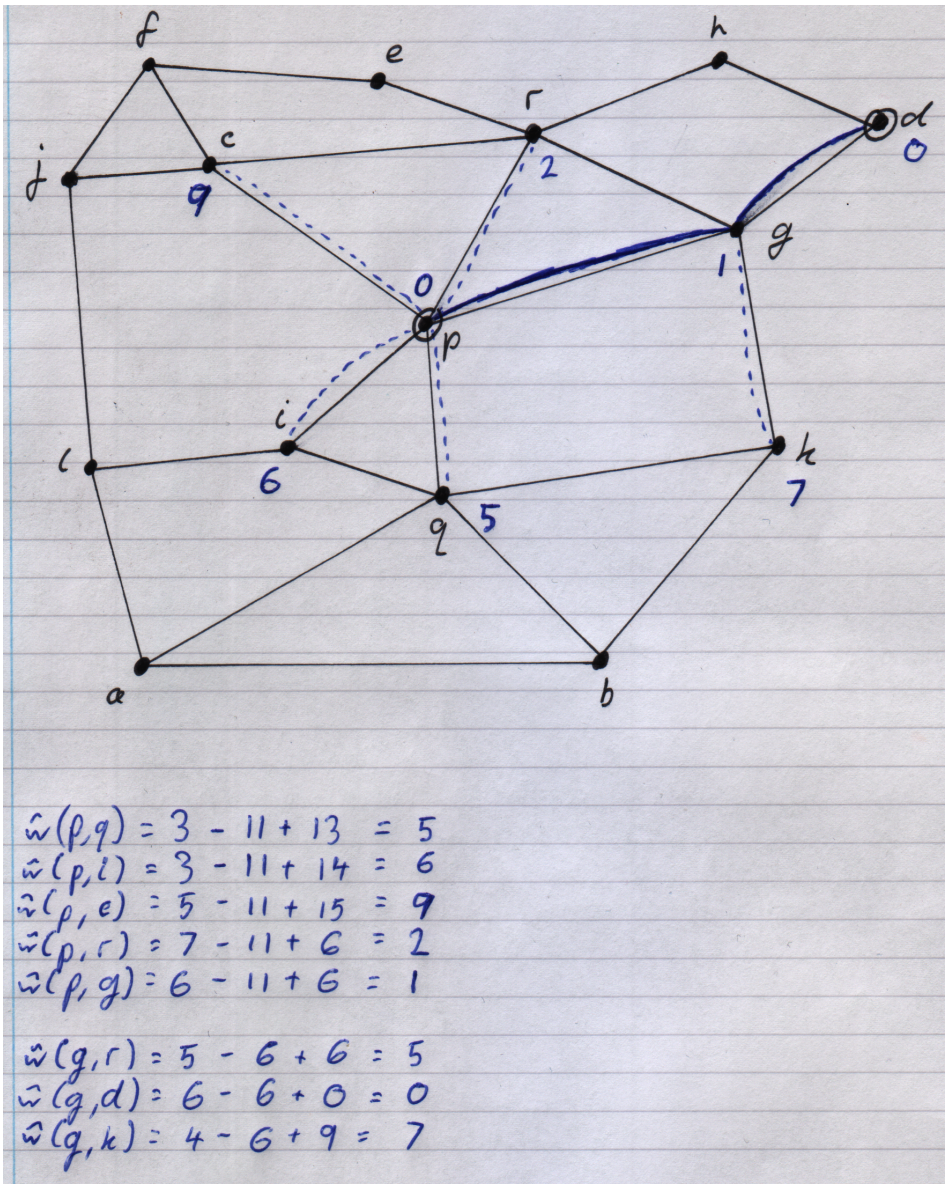
Run-time: $\Theta(n + m) = \Theta(|V| + |E|)$

Question 3

Dijkstra:



Geographic Re-weighted Dijkstra:



Question 4

```
fun BREADTH-FIRST-SEARCH(G, s):
    open = empty queue
    closed = empty set

    open.enqueue(s)

    while (open is not empty):
        n = open.dequeue()
        closed.add(n)

        VISIT(n)

        for each m in adj[n], do:
            if (closed does not contain m):
                open.enqueue(m)
```

Question 5

```
fun DEPTH-FIRST-SEARCH(G, s):
    open = empty stack
    closed = empty set

    open.push(s)

    while (open is not empty):
        n = open.pop()
        closed.add(n)

        VISIT(n)

        for each m in adj[n], do:
            if (closed does not contain m):
                open.enqueue(m)
```