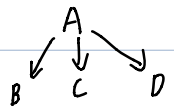
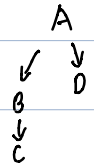
D. 1. same order



DFS A B C D

BFS A B C D

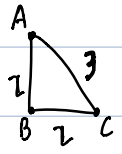
different order

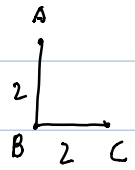


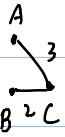
DFS A B C D

BFS A B D C

2. counter example:

graph 

minimum spanning tree: 

shortest path tree rooted at C. 

3. (a) assume x is the capital.

$M(x) = c(x)$

for other vertex v: $M(v) = \min_{(u,v) \text{ in } E} (M(u) + r(u,v) + c(v))$

algorithm:

① topologically sort the vertices of V

② initialize $M(x) = c(x)$ other vertex v $M(v) = \infty$

③ for each vertex u in topologically sorted order

for each vertex v in G.adj[u]

if $M(v) < M(u) + r(u,v) + c(v)$

$M(v) = M(u) + r(u,v) + c(v)$

(b). time complexity: ① topologically sorted $\Theta(V+E)$

② initialize $\Theta(V)$

③ for each vertex u in tologically sorted orde
    for each vertex v in $G.adj[u]$ } $\Theta(V+E)$
        $\Theta(1)$

total time $\Theta(V+E)$ time complexity $\Theta(V+E)$

correctness:

because the roads would not form any cycle

thus this graph is a DAG.

So the graph has a topologically sorted order.

prove by induction:

Inductive hy hypothesis: if all vertex before v in a topological sort order have been updated. then $M(v) = \delta(x,v)$

x is the source vertex.

Base case $M(x) = c(x) = \delta(x,x)$

other vertexs $M(v) = \infty = \delta(x,v)$

Inductive step:

- Consider a vertex v after s
- $M(v) = \min_{(u,v) \text{ in } E} (M(u) + c(v) + r(u,v))$
- By inductive hypothesis,

$$M(u) + c(v) + r(u,v) = \delta(x,u) + c(u) + r(u,v)$$

Since some (u,v) must be on the shortest path. by optimal substructure $M(v) = \delta(s,v)$

thus: this algorithm can compute the function M.

E.(1) use Dijkstra's algorithm

```
Dijkstra (G, w, s=0)
    for v in G.V
        v.d = ∞
        v.π = NIL                    } Θ(V)
    s.d = 0
    S = empty
    Q = G.V
    while Q ≠ empty:
        u = extract-min(Q)  (u.d is the smallest)   Θ(VlogV)
        S = S ∪ {u}
        for v in G.adj[u]
Realx:      if v.d > u.d + w(u,v)                   Θ(V+E)
                v.d = u.d + w(u,v)
                v.π = u
```

```
Get Minimum (G, w, s=0):
    Dijkstra (G, w, 0);
    for v in G.V
        return 2 X v.d
    (for factory v, the minimum amount).
```

(2). algorithm:

We take factory 0 as the source vertex.

0 using Dijkstra's algorithm to compute the shortest path and distance from factory 0 to other factory.

the for the return path, we trun each path to a reverse edge. such as. $3 \to 5$ to $5 \to 3$. then same as 0. find shorest path from 0 to i. this distance is equal to the shorest distance from i to 0.

the sum. of shortiest distance from 0 to i and shorest distance from i to 0 is the lowest mail fee for a round trip to and from factory i.

correct: we just need to find the shortest distance form 0 to i and shortest distance from i to 0. thus using twice Dijkstra's alogrithm is ok!

time complexity:

We use heap to store each u as u.d.

so the time complexity extract-min $= \Theta(\log V)$

other parts' time complexity is displayed above (1).

total time $(\Theta(V) + \Theta(V \log V) + \Theta(V+E)) \times 2$.

$= \Theta(V \log V + E) = O(E \log E)$

(3) using Bellman-Ford algorithm

```
Bellman-ford (G, w, s=0):
    for v in G.V:
        v.d = ∞
        v.π = NIL
    s.d = 0
```

```
for i = 1 to |G.V| - 1:
    for (u, v) in G.E:
Relax:  if v.d > u.d + w(u, v):
            v.d = u.d + w(u, v)
            v.π = u
for (u, v) in G.E        // find negative cycle
    if v.d > u.d + w(u, v)
        output ("I am rich")
```

like (2). we using Bellman-ford algorithm to find the shortest distance from 0 to i and shortest distance from i to 0. the sum is the answer.

(4). in (1) Dijkstra's algorithm:

advantage: time complexity $O(E \log E)$ is better than $O(VE)$

disadvantage: can't compute the shortest path for a graph that has the negative edge.

in (3) Bellman-ford's algorithm:

advantage: can compute the shortest pant for all cases

disadvatage: time complexity $O(VE)$

(5). reweighting:

for each edge : new weight = old weight · K − reliability of the front vertex

use Bell-ford algorithm to detect if exist a negative cycle.

correctness :

$$\sum_{\text{acycle}} w_i = \sum_{\text{acycle}} (w_i - \text{cold}) \cdot k - \sum \text{reliability of } v_i$$

$$= k \cdot \text{total mailing fees} - \text{total reliability}$$

if $\dfrac{\text{total reliability}}{\text{total mailing fees}} > k$

then $k \cdot \text{total mailing fees} - \text{total reliability} < 0$

thus $\sum_{\text{acycle}} w_i < 0$

thus detecting the negative cycle is to find a "trustful cycle".

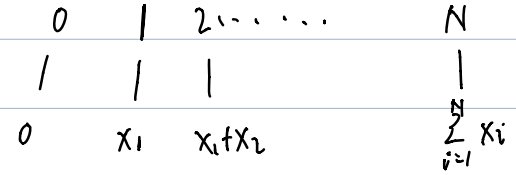
time complex: reweighting run time $O(N)$

Bellman-ford time complex $O(NE)$.

thus total time complexity is $O(N*E)$

F: We use N+1 vertexs, which stand for the prefix sum of X.

example:



each equation will be connected by these vertexs.

thus. this problem will be reduced to find the minimum spanning tree for this graph.

thus use Prim's algorithm is ok.

correctness:

when we compute the prefix sum of x from 0 to N

we can solve each X_i.

so, we just find a graph include all vertexs above. with minimum cost

thus. this problem can be reduced to find a minimum spanning tree.

time complexity:

all edges = all equations = $\frac{N(N+1)}{2}$

construct graph: $\Theta(N + \frac{N(N+1)}{2}) = \Theta(N^2)$

Prim's algorithm:

using Fibonacci heaps:

$O(E + V\log V) = O(N^2 + N\log N) = O(N^2)$

thus: total time complexity is $O(N^2)$