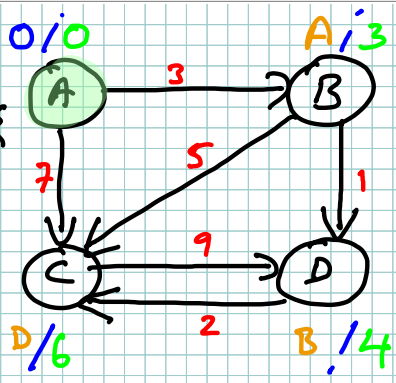
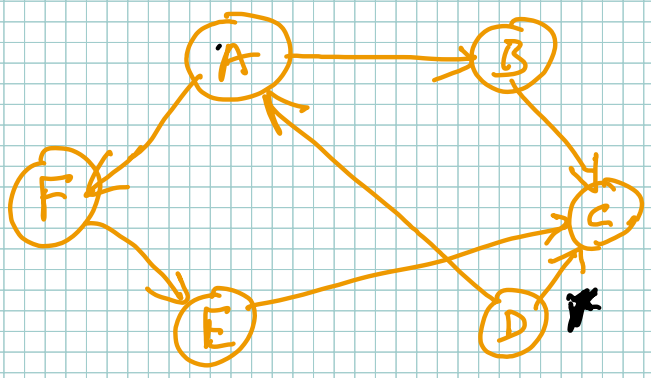
DYJKSTRA'S SHORTEST PATH

```
for each Vertex curV in GRAPH{
    curV.distance = ∞
    curV.pred = 0
    enqueue curV in UNVISITED V's
}
startV.distance = 0
while (unvisited V's not empty){
    dequeue curV with shortest distance
    for each adjV to curV {
        edgeW = weight of edge from curV to adjV
        altDist = curV.distance + edgeW
        if (altDist < adjV.distance)
            adjV.distance = altDistance
            adjV.pred = curV
    }
}
```




UNVISITED VERTICES



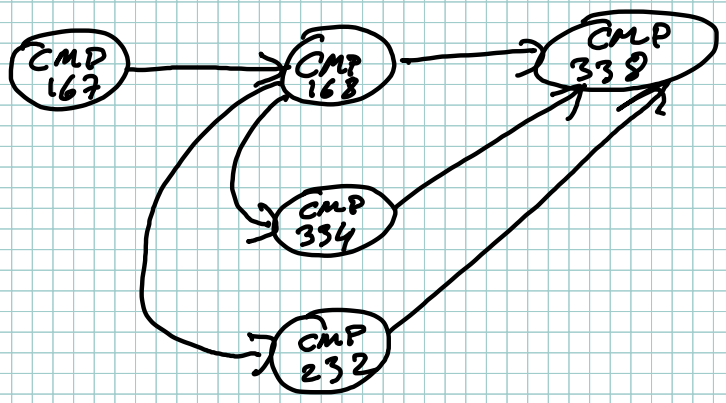

topological sort of an acyclic graph produces a list of the graph's vertices such that for every edge from vertex X to vertex Y, X comes before Y in list.

C, D, A, F, B, E  NO

D, A, F, E, C, B  NO

D, A, F, E, B, C  YES

D, A, F, B, E, C  YES




167, 168, 334, 232, 338

TOPOLOGICA SORT {

result list IS empty

NO INCOMING HAS VERTICES WITH NO INCOMMING

REMAINING EDGES

- While (NO INCOMMING NOT EMPTY) {

  CurV = remove a Vertex from list

  add curV to result list

  Remove outgoing edges of CurV from remaining edges

  for (each edge in CurE) {

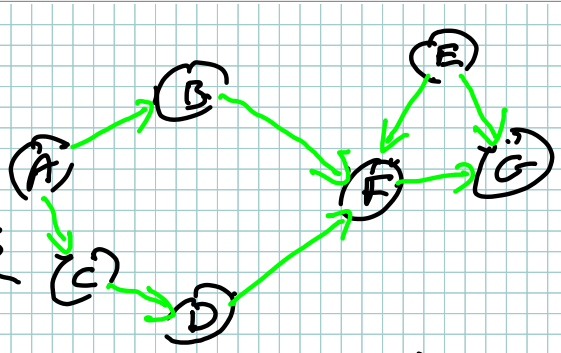
    if removing CurE results in destV have no incoming {

      add destV to no Incoming

    }

  }

}




RESULT LIST

A E B C D F G

NO INCOMMING

REMAINING EDGES

CurV = A

CurE

:

:

TOPOLOGICA SORT {

result list IS empty

NO INCOMING HAS VERTICES WITH NO INCOMMING

REMAINING EDGES

While (NO INCOMMING NOT EMPTY) {

CurV = remove a Vertex from list

add curV to result list

Remove outgoing edges of CurV from remaining edges

for (each edge in CurE) {

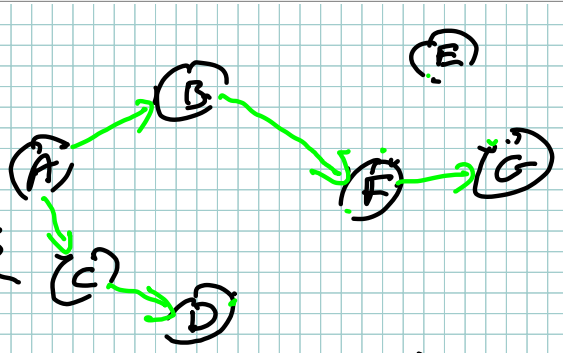
if removing CurE results in destV have no incoming {

add destV to no Incoming

}

}

}




RESULT LIST

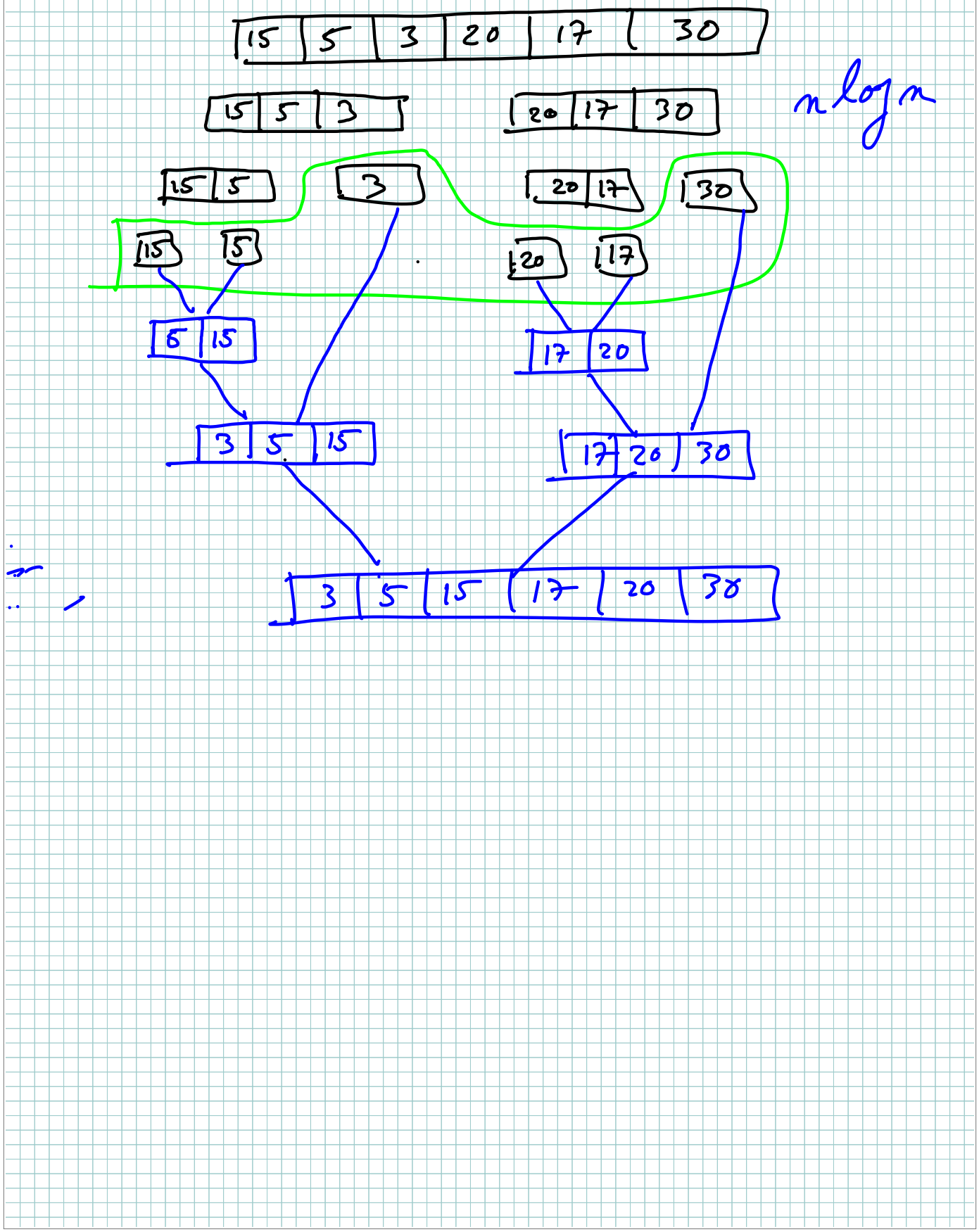
A E B C D F G

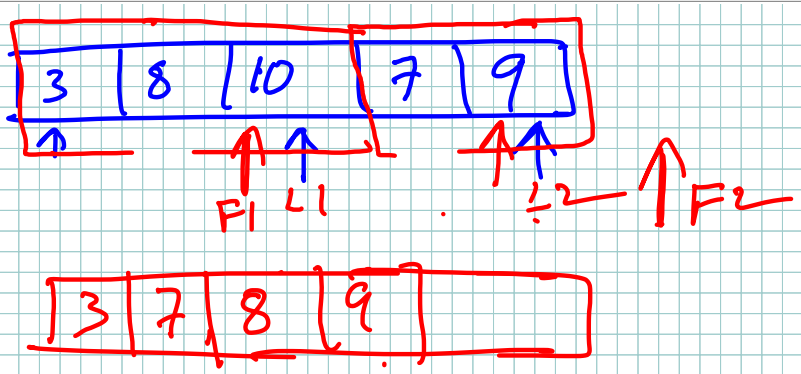
NO INCOMMING

REMAINING EDGES

CurV = A

CurE


15 5 3 20 17 30
15 5 3
20 17 30
15 5
3
20 17
30
15
5
20
17
5 15
17 20
3 5 15
17 20 30
3 5 15 17 20 38
n log n


3
8
10
7
9
F1
L1
L2
F2


3
7
8
9