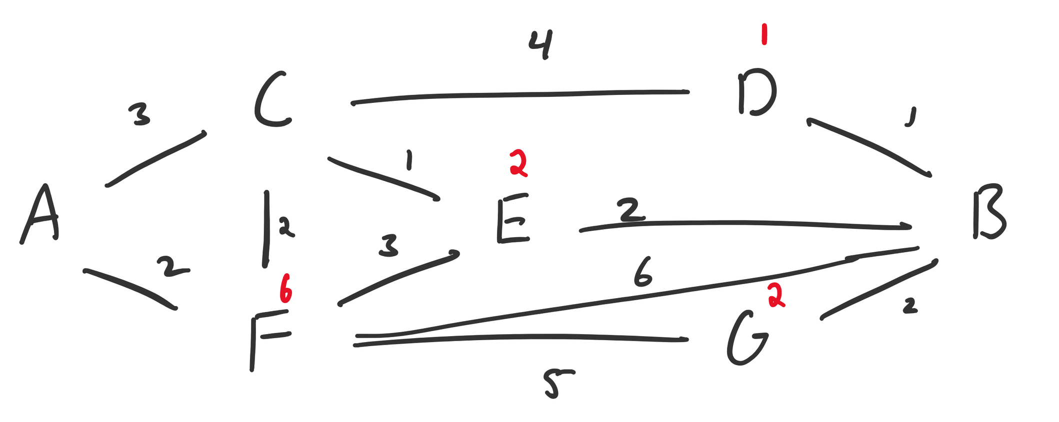
D*

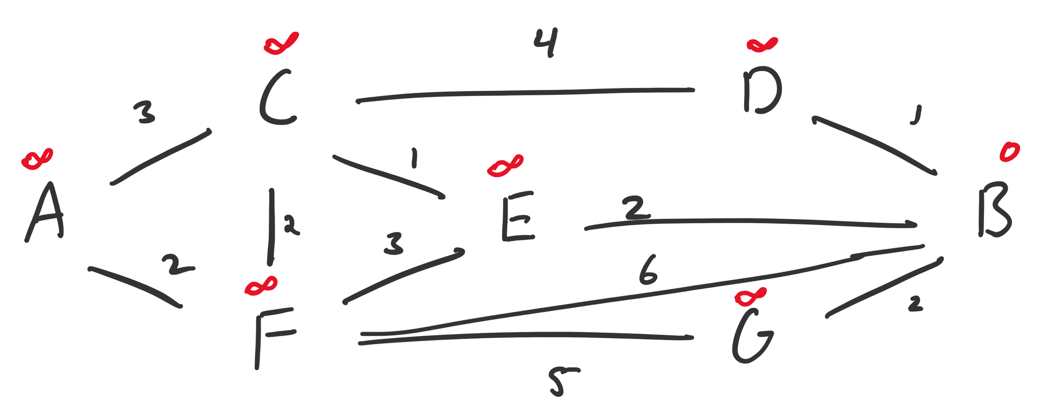
Friday, July 25, 2025 11:05 AM



Notes:

1. Works in reverse from Goal to Start
2. distance (u) - shortest distance from goal to vertex u
3. OPEN - priority queue of nodes to be processed
4. Predecessor - Previous node in the optimal path

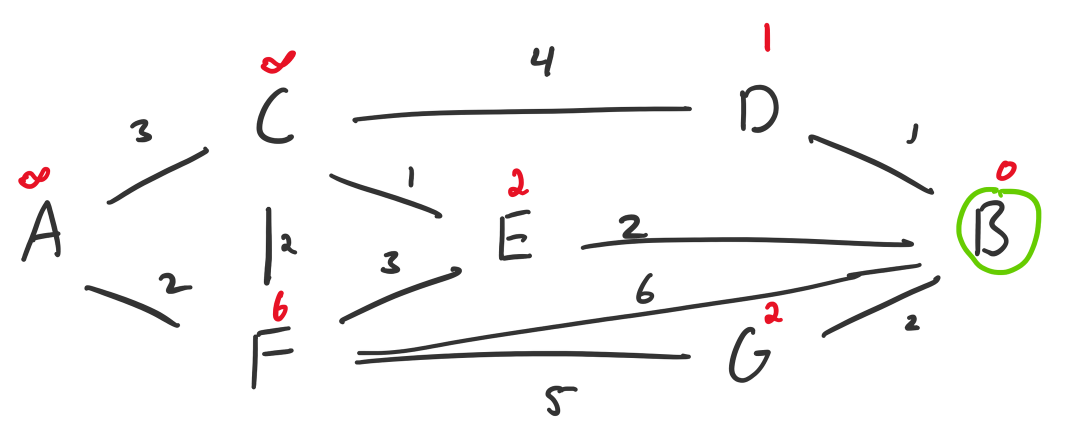
Step 1 - Initialize



- set dist(B) = 0, all else = ∞ ← is this true??
- OPEN = [B]

Step 2 - Process B

OPEN = [ ]



D:

dist(D) = dist(B) + cost (D, B)

= 0 + 1 = 1

E:

dist(E) = 0 + 2 = 2

F:

dist(F) = 6

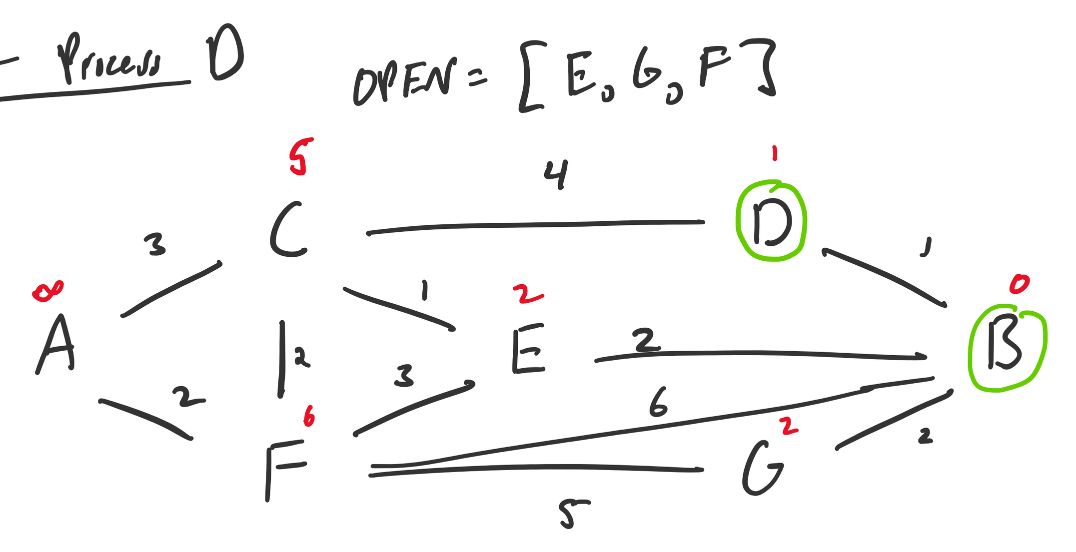
G:

dist(G) = 2

OPEN = [D [1], E [2], G [2], F [6]]

Step 3 - Process D

OPEN = [E, G, F]



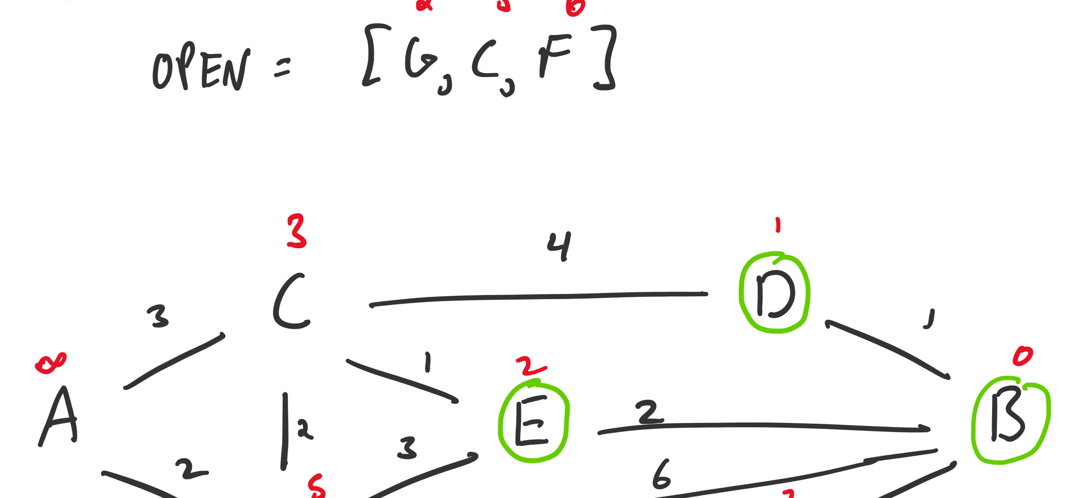
C:

dist(C) = 1+4 = 5

OPEN = [E [2], G [2], C [5], F [6]]

Step 4: Process E

OPEN = [G [2], C [5], F [6]]



C:

dist(C) = 2 + 1 = 3 < 5

so update ✓

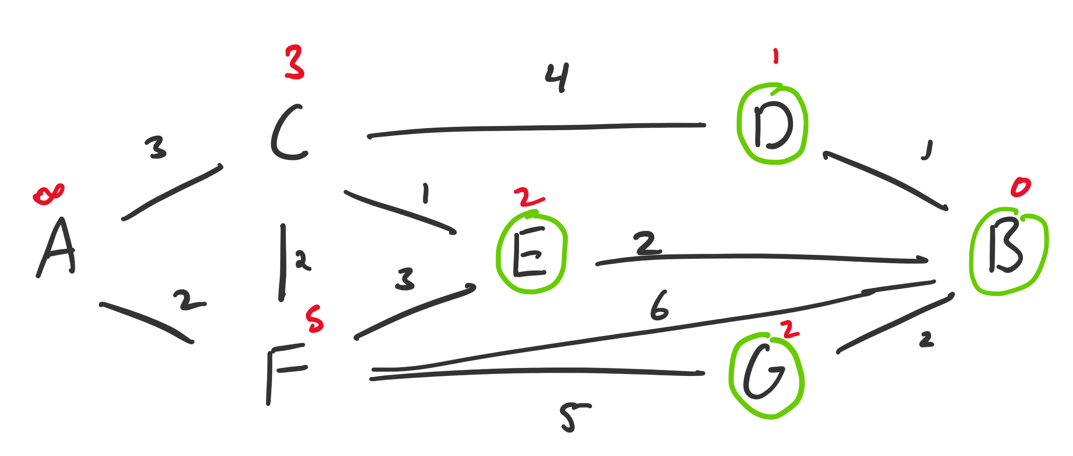
F:

dist(F) = 2 + 3 = 5 < 6

so update ✓

Step 5: Process G

OPEN: [C [3], F [5]]



F:

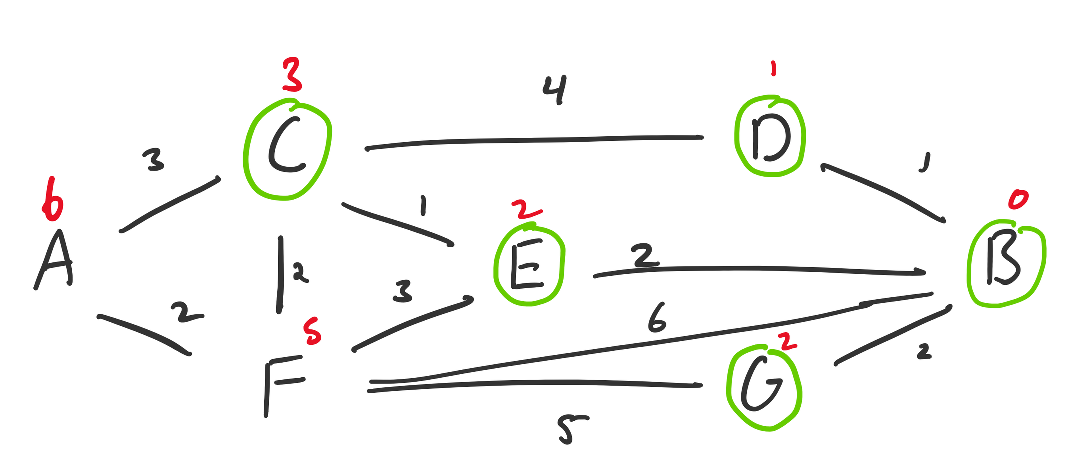
dist(F) = 2 + 5 = 7 > 5

no update ✗

STEP 6: Process C

- Add A to list & remove C

OPEN = [F [5], A [∞]]



A:

dist(A) = 3 + 3 = 6 < ∞

update ✓

D:

dist(D) = 3 + 1 = 4 > 1

no update ✗

E:

dist E = 3 + 1 = 4 > 2

no update ✗

F:

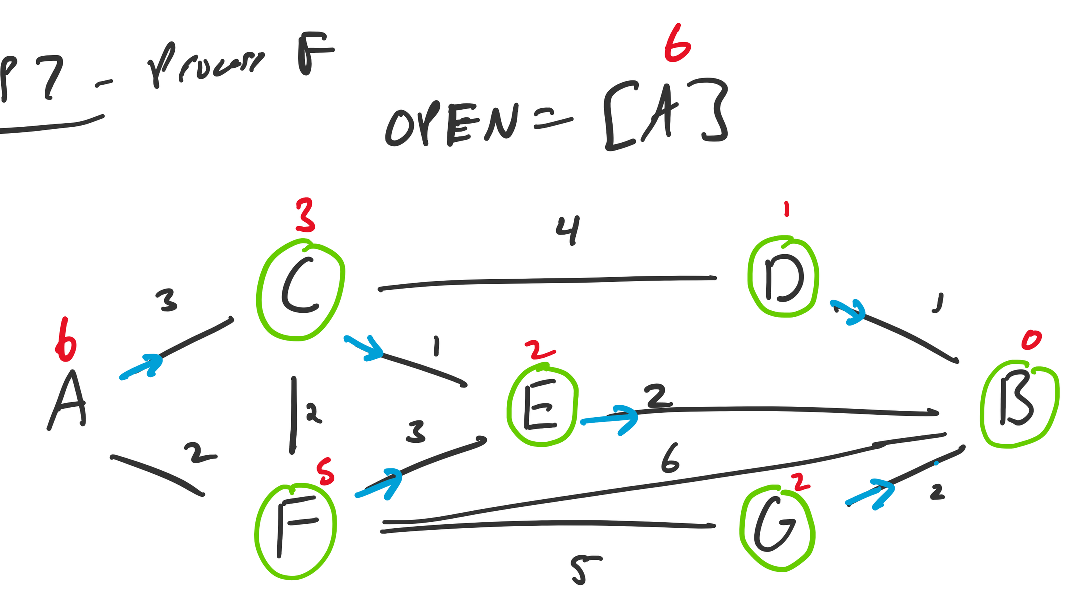
dist(F) = 3 + 2 = 5 = 5

no update ✗

OPEN = [F [5], A [6]]

STEP 7 - Process F

OPEN = [A [6]]



C:

dist (C) = 5 + 2 = 7 > 3

no update ✗

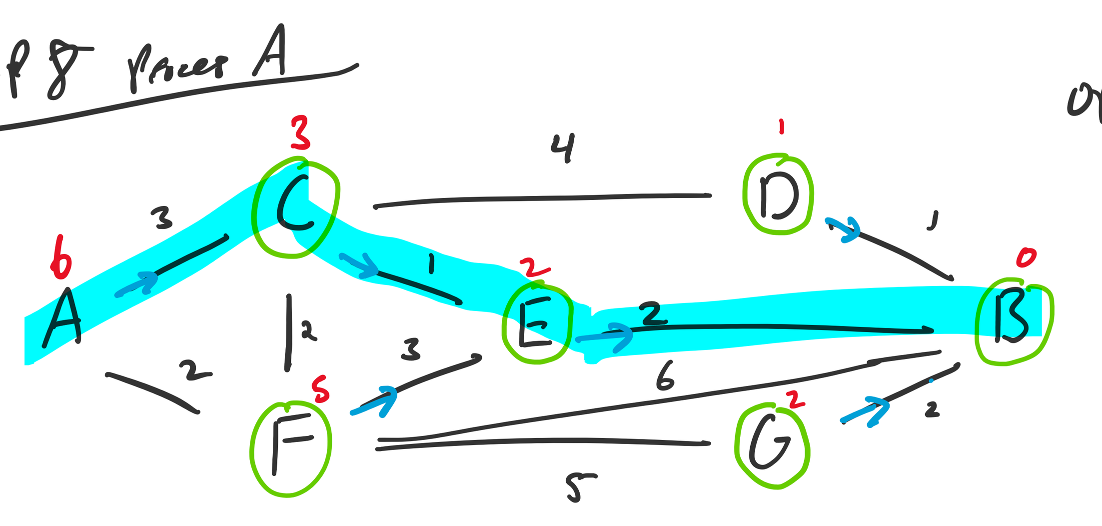
A:

dist(A) = 5 + 2 = 7 > 6

no update ✗

STEP 8 Process A

OPEN = [ ]



The goal !!

Map shortest path