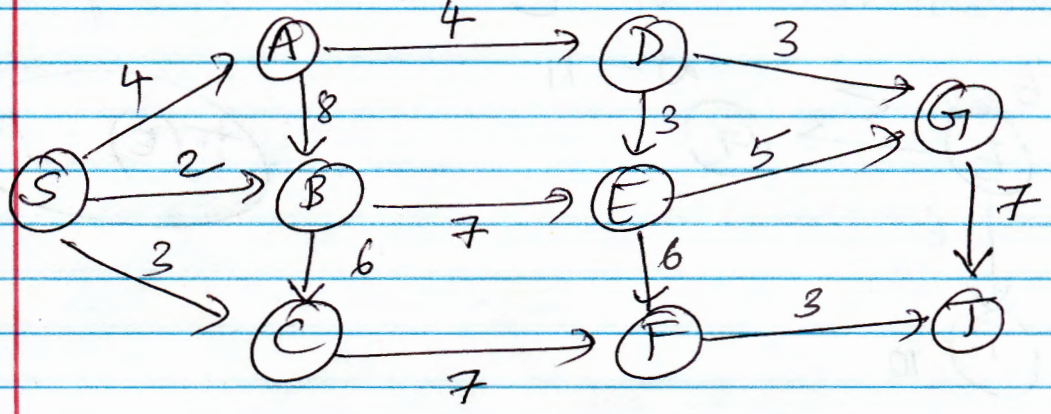
DIJKSTRAS



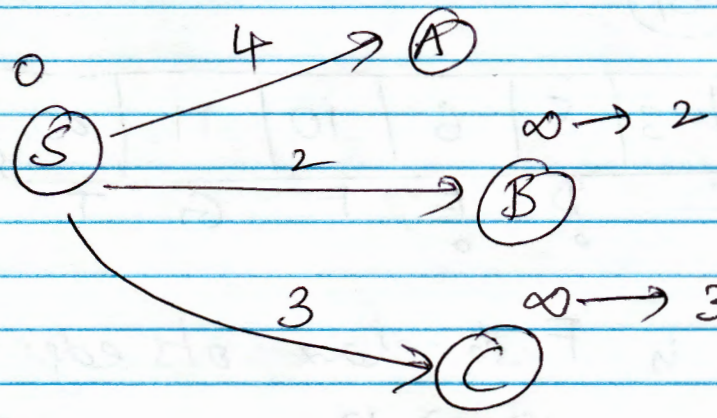
initiliase distance array

0	∞	∞	∞	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	T

~~Pick closest vertex to s~~

First pick S (source). Relax the edges SA, SB, SC

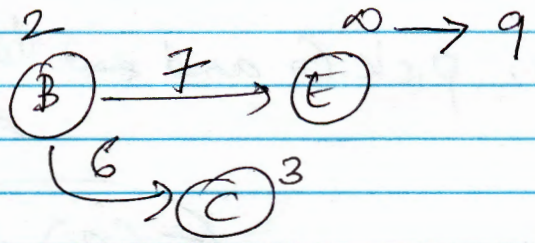
∞ → 4



0	4	2	3	∞	∞	∞	∞	∞
S	A	B	C	D	E	F	G	T

next closest vertex to S is B. So pick B and relax all edges of B (outgoing)

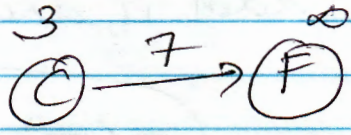
$\pi(B) = S$



0	4	2	3	∞	9	∞	∞	∞
S	A	B	C	D	E	F	G	T

next closest vertex to S is C. So pick C and relax all edges of C.

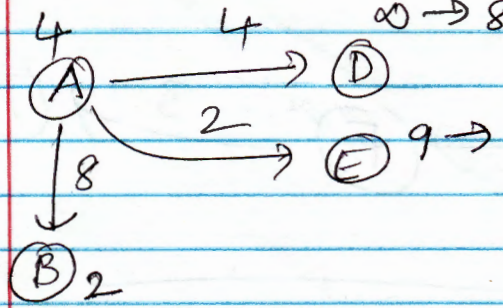
$\pi(C) = S$



0	4	2	3	∞	9	10	∞	∞
S	A	B	C	D	E	F	G	T

Next closest is A. Relax all outgoing edges of A

$\pi(A) = S$

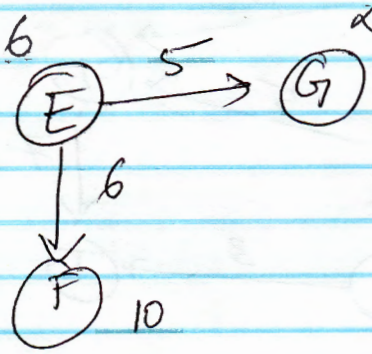


0	4	2	3	8	6	10	∞	∞
S	A	B	C	D	E	F	G	T

(3)

0	4	2	3	8	6	10	∞	∞	~~X~~
S	A	B	C	D	E	F	G	T	

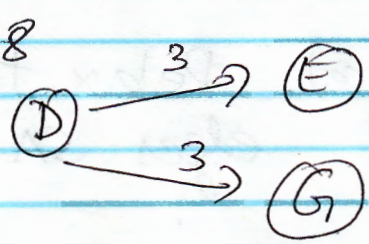
next closest vertex is E. so relax edges of E



$\pi(E) = A$

0	4	2	3	8	6	10	11	∞
S	A	B	C	D	E	F	G	T

next closest vertex is D so relax it's edges

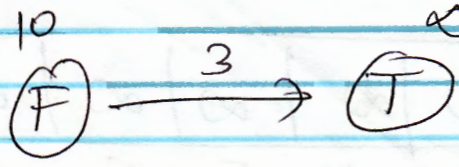


$\pi(D) = A$

no change

0	4	2	3	8	6	10	11	∞
S	A	B	C	D	E	F	G	T

next closest is F so relax its edges.

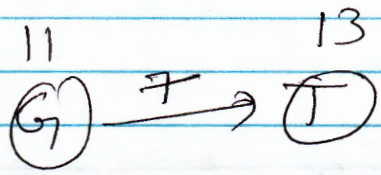


$\pi(F) = C$

(4)

0	4	2	3	8	6	10	11	13
S	A	B	C	D	E	F	G	T

next closest is G. pick G and ~~ex~~relax its edges.



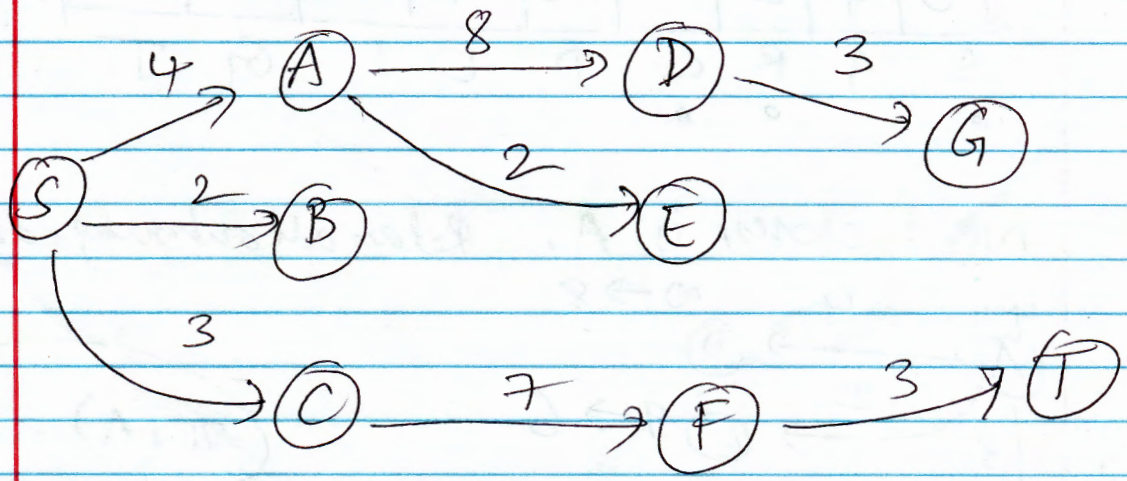
$\pi(G) = D$

0	4	2	3	8	6	10	11	13
S	A	B	C	D	E	F	G	T

T is the last vertex, add it and Done.

Shortest paths:

$\pi(T) = F$



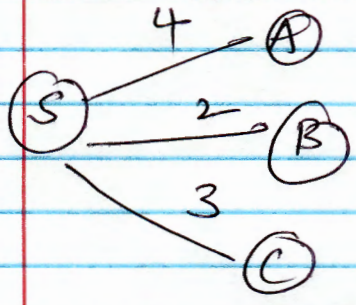
PRIMS

For prims, we maintain distance from FRONTIER

FRONTIER: set of vertices already included in the tree

0	∞	∞	∞	∞	∞	∞	∞	∞
S ✓	A	B	C	D	E	F	G	T

First, add S to Frontier. This allows us to explore all edges of S

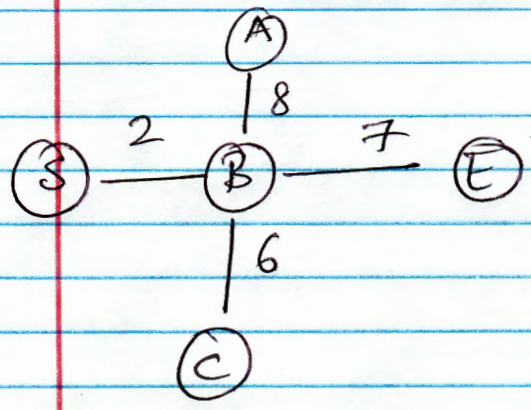


Frontier = {S}

0	4	2	3	∞	∞	∞	∞	∞
S ✓	A	B ✓✓	C	D	E	F	G	T

We add the closest vertex to the frontier i.e. B to the frontier. (edge: SB)

This allows us to explore edges of B

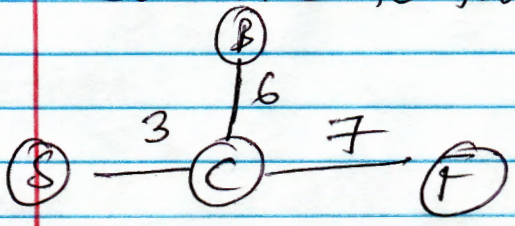


Frontier = {S, B}

0	4	2	3	∞	7	∞	∞	∞
S	A	B	C	D	E	F	G	T
✓		✓	✓✓					

→ edge (SC)

The closest vertex to the frontier is C.
So add C to frontier and explore C's edges:

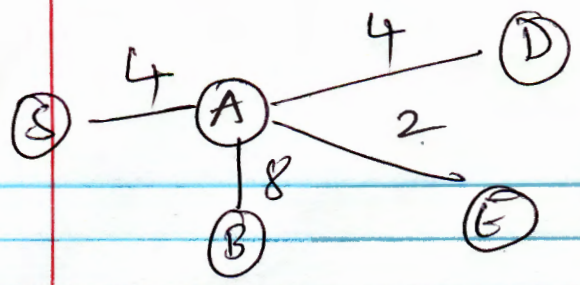


Frontier = {S, B, C}

0	4	2	3	∞	7	7	∞	∞
S	A	B	C	D	E	F	G	T
✓	✓✓	✓	✓					

→ edge (SA)

Next closest vertex to frontier is A.
add A and explore A's edges

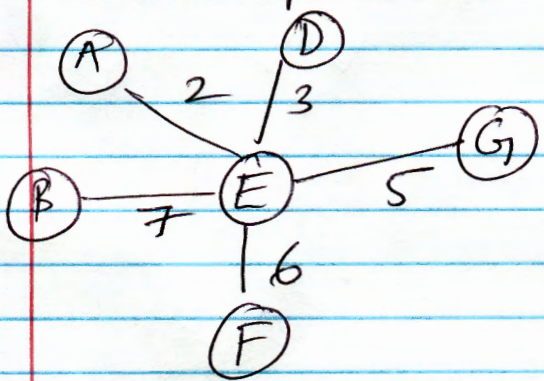


frontier = {S, B, C, A}

edge: (AE)

0	4	2	3	4	2	7	∞	∞
S	A	B	C	D	E	F	G	T
✓	✓	✓	✓	✓✓				

E is closest vertex to frontier now. So add E and explore E's edges:

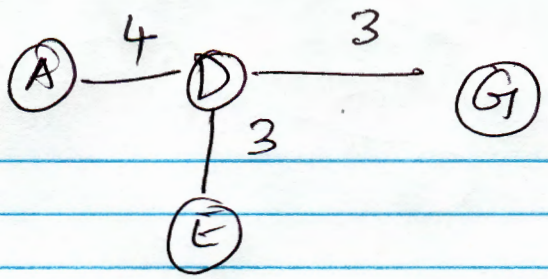


frontier = {S, B, C, A, E}

edge: (ED)

0	4	2	3	3	2	6	5	∞
S	A	B	C	D	E	F	G	T
✓	✓	✓	✓	✓✓	✓			

D is closest vertex to frontier. So add D and explore D's edges.

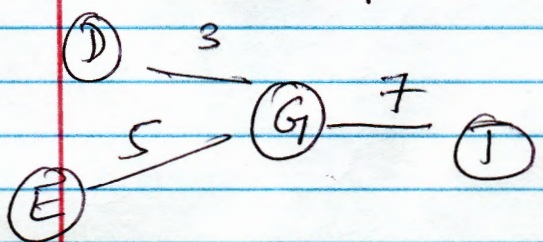


frontier
{S, B, C, A, E, D}

edge (DG)

0	4	2	3	3	2	6	3	∞
S	A	B	C	D	E	F	G	T
✓	✓	✓	✓	✓	✓		✓✓	

G is closest vertex to frontier. add G and explore G's edges

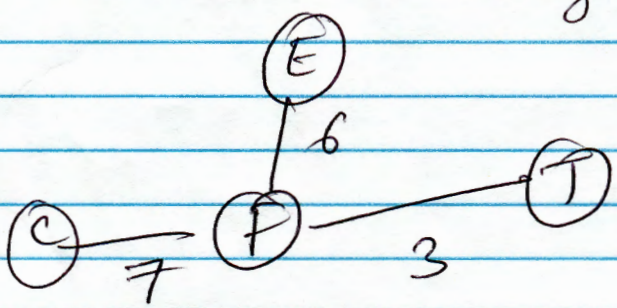


frontier
{S, B, C, A, E, D, G}

0	4	2	3	3	2	6	3	7
S	A	B	C	D	E	F	G	J
✓	✓	✓	✓	✓	✓	✓✓	✓	

edge (EF)

F is closest vertex to frontier. add F and explore F's edges.



frontier:
{S, B, C, A, E, D, G, F}

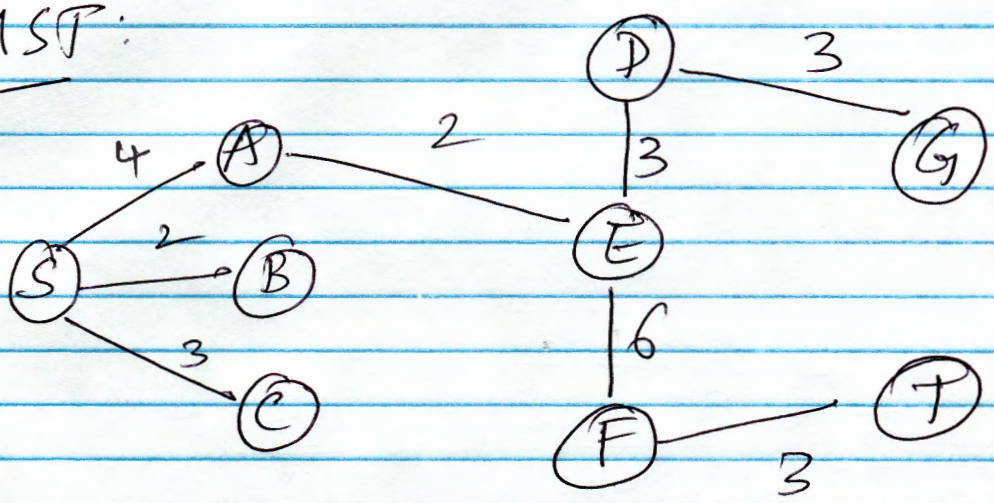
0	4	2	3	3	2	6	3	3	
S	A	B	C	D	E	F	G	T	
✓	✓	✓	✓	✓	✓	✓	✓	✓✓	

edge: (FT)

finally add T to frontier.

Since all vertices are in the frontier we are done.

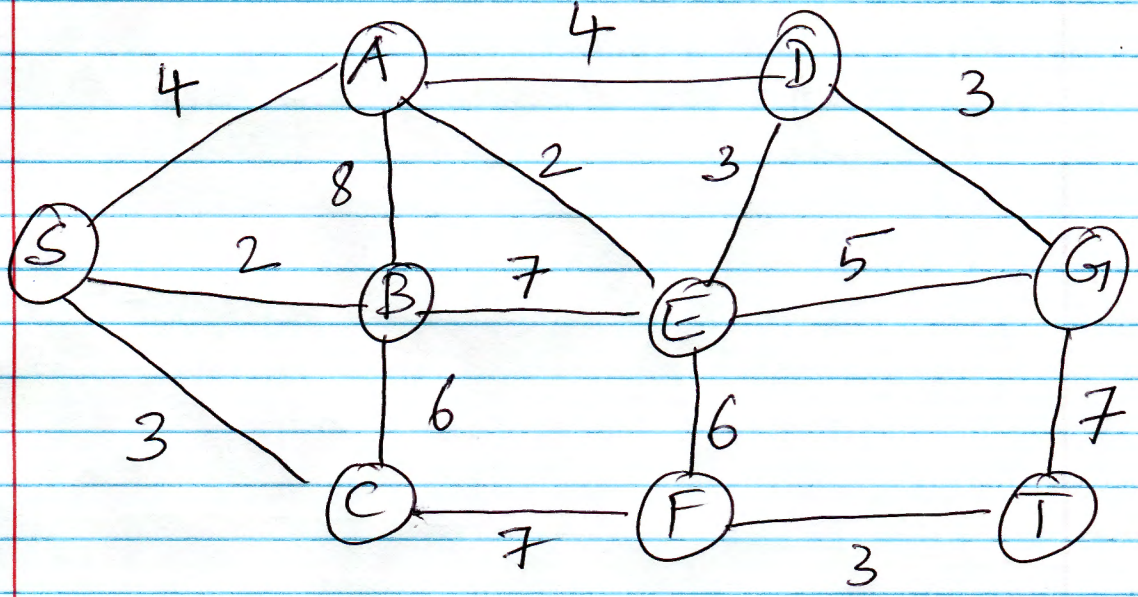
MST:



total cost of MST:

= 26

KRUSKAL'S



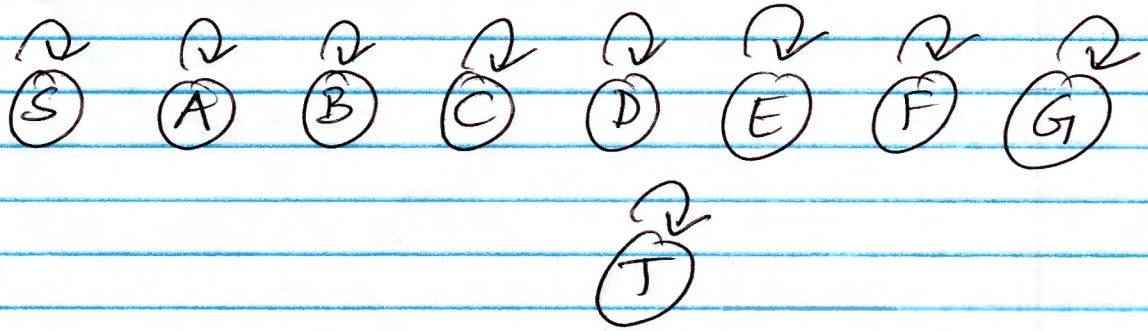
Edges increasing order

SB AE SC DE DG FT

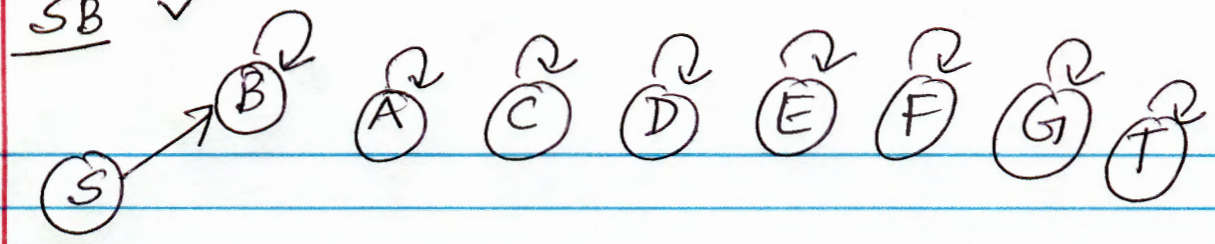
SA AD EG BC EF

BE CF GT AB

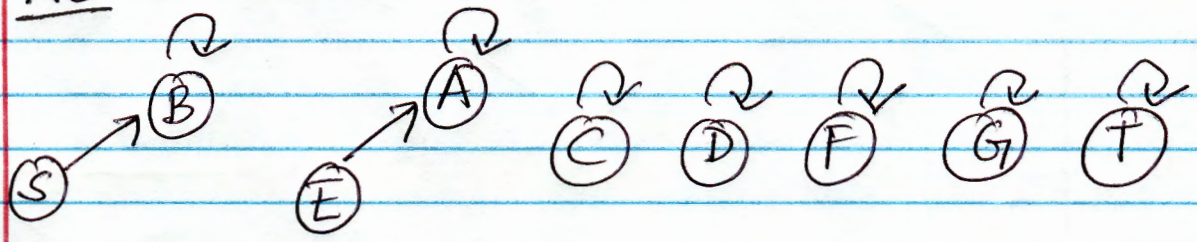
Before processing any edge



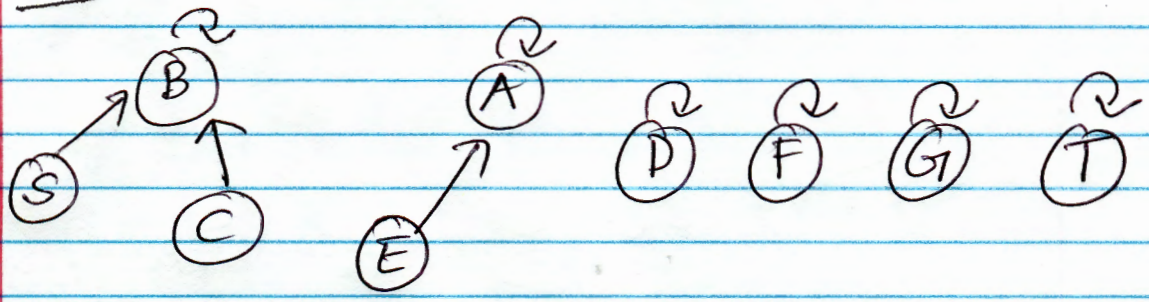
SB ✓



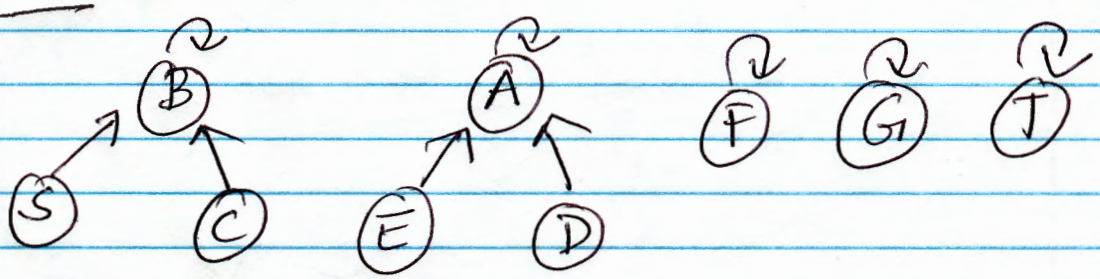
AE ✓



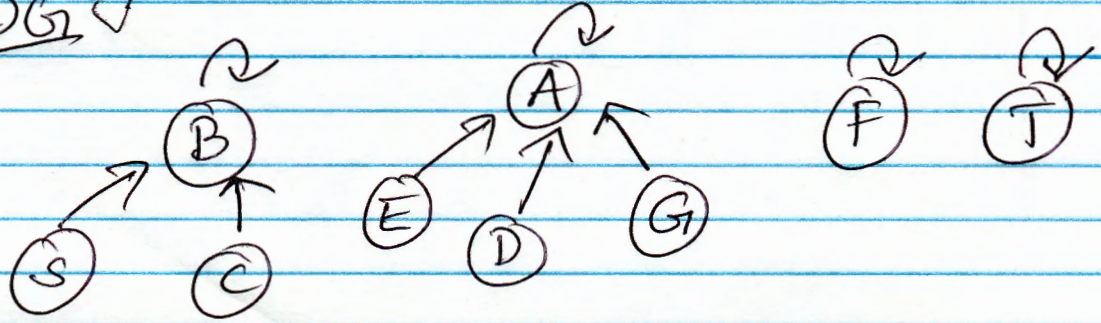
SC ✓



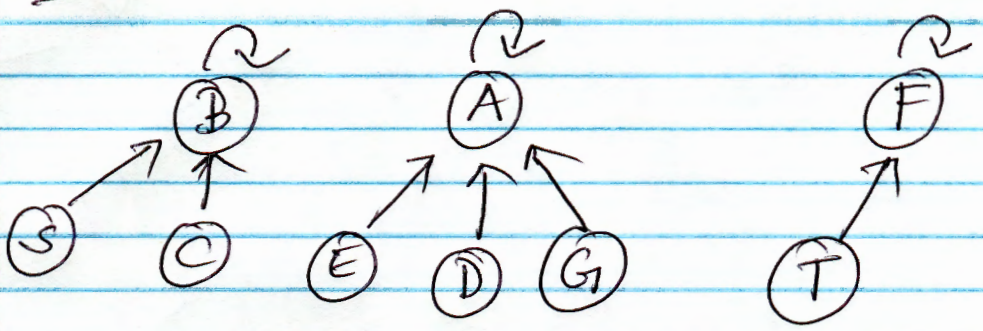
DE ✓



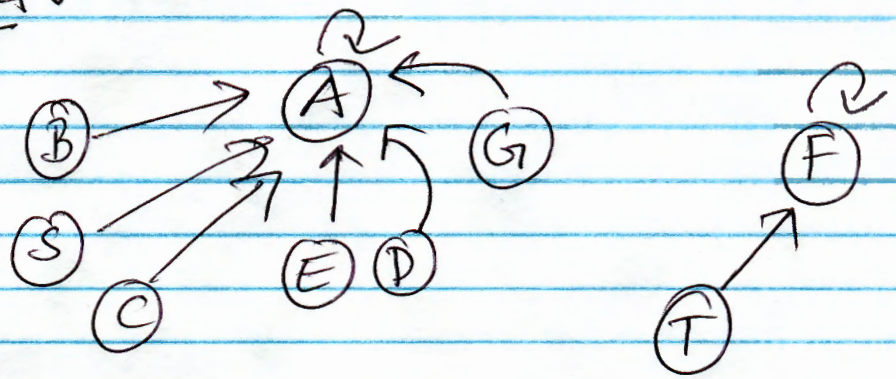
DG ✓



FT ✓



SA ✓

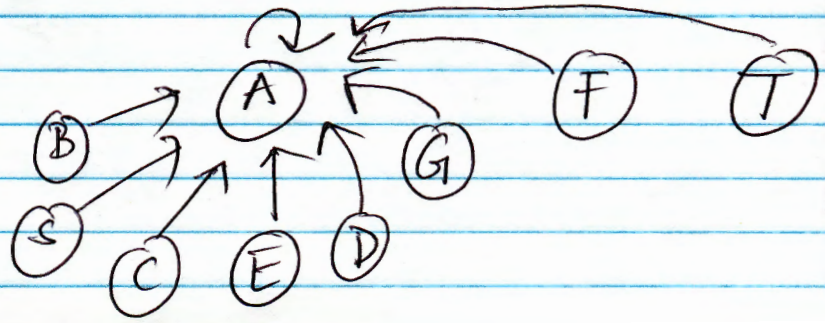


AD X (since both belong to ~~A~~ component A)

EG X (since both belong to component A)

BC X (")

EF ✓



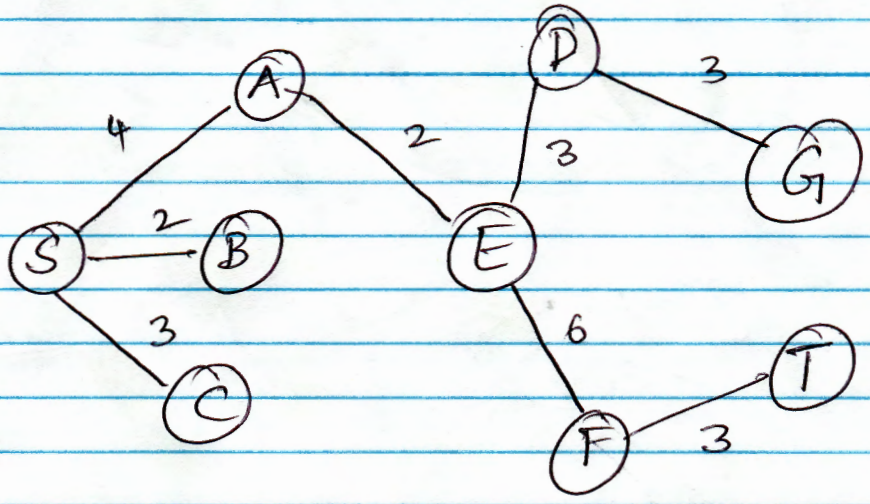
BE X (both ~~B E~~ belong to component A)

CF X "

GT X "

AB X "

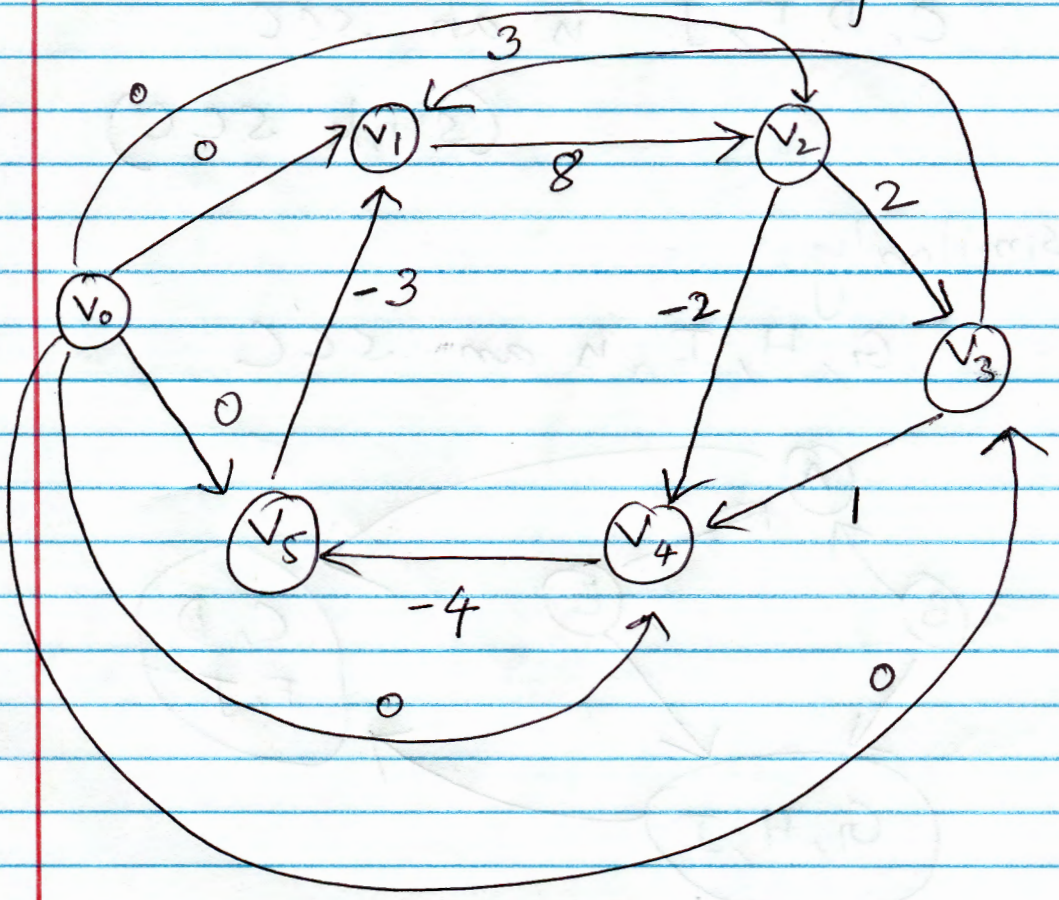
Final MST using kruskal's



total cost of MST

= 26

SOLVING INEQUALITIES using bellman ford



Bellman Ford

	it 0	it 1	it 2	it 3	it 4	it 5	it 6
V_0	0	0	0	0	0	0	
V_1	∞	0	-3	-7	-9	-9	
V_2	∞	0	0	0	0	-1	
V_3	∞	0	0	0	0	0	
V_4	∞	0	-2	-2	-2	-2	-3
V_5	∞	0	-4	-6	-6	-6	

Chain Matrix Multiplication

A_1 A_2 A_3 A_4

A_1 (50×10)

A_2 (10×30)

A_3 (30×20)

A_4 (20×100)

A_1A_2 A_2A_3 A_3A_4

$A_1A_2A_3$ $A_2A_3A_4$

$A_1A_2A_3A_4$

$c(i,j)$: most efficient way to compute $(A_i A_{i+1} \dots A_j)$

$c(1,2) = 50 \times 10 \times 30 = 15k$ $\quad c(2,3) = 10 \times 30 \times 20 = 6k$

$c(3,4) = 30 \times 20 \times 100 = 60k$

$A_1A_2A_3$: Two potential ways

$A_1 (A_2A_3)$ $\quad (A_1A_2) A_3$

$c(2,3) +$ $\quad c(1,2) +$

$m_0 \times m_1 \times m_3$ $\quad m_0 \times m_2 \times m_3$

$= 6k + 50 \times 10 \times 30$ $\quad = 15k + 50 \times 30 \times 20$

$= 21k$ $\quad = 45k$

← more efficient

$c(1,3) = 21k$

$C(2,4) = 26k$

$A_2 A_3 A_4$

$(A_2 A_3) A_4$ | $A_2 (A_3 A_4)$

$c(2,3) +$ | $c(3,4) +$

$m_1 \times m_3 \times m_4$ | $m_1 \times m_2 \times m_4$

$= 6k + 10 \times 20 \times 100$ | $= 60k + 10 \times 30 \times 100$

$= 26k$ ← more efficient | $= 90k$

$C(1,4) = 76k$

$A_1 A_2 A_3 A_4$

$A_1 (A_2 A_3 A_4)$ | $(A_1 A_2)(A_3 A_4)$ | $(A_1 A_2 A_3) A_4$

$c(2,4)$ | $c(1,2) + c(3,4)$ | $c(1,3) +$

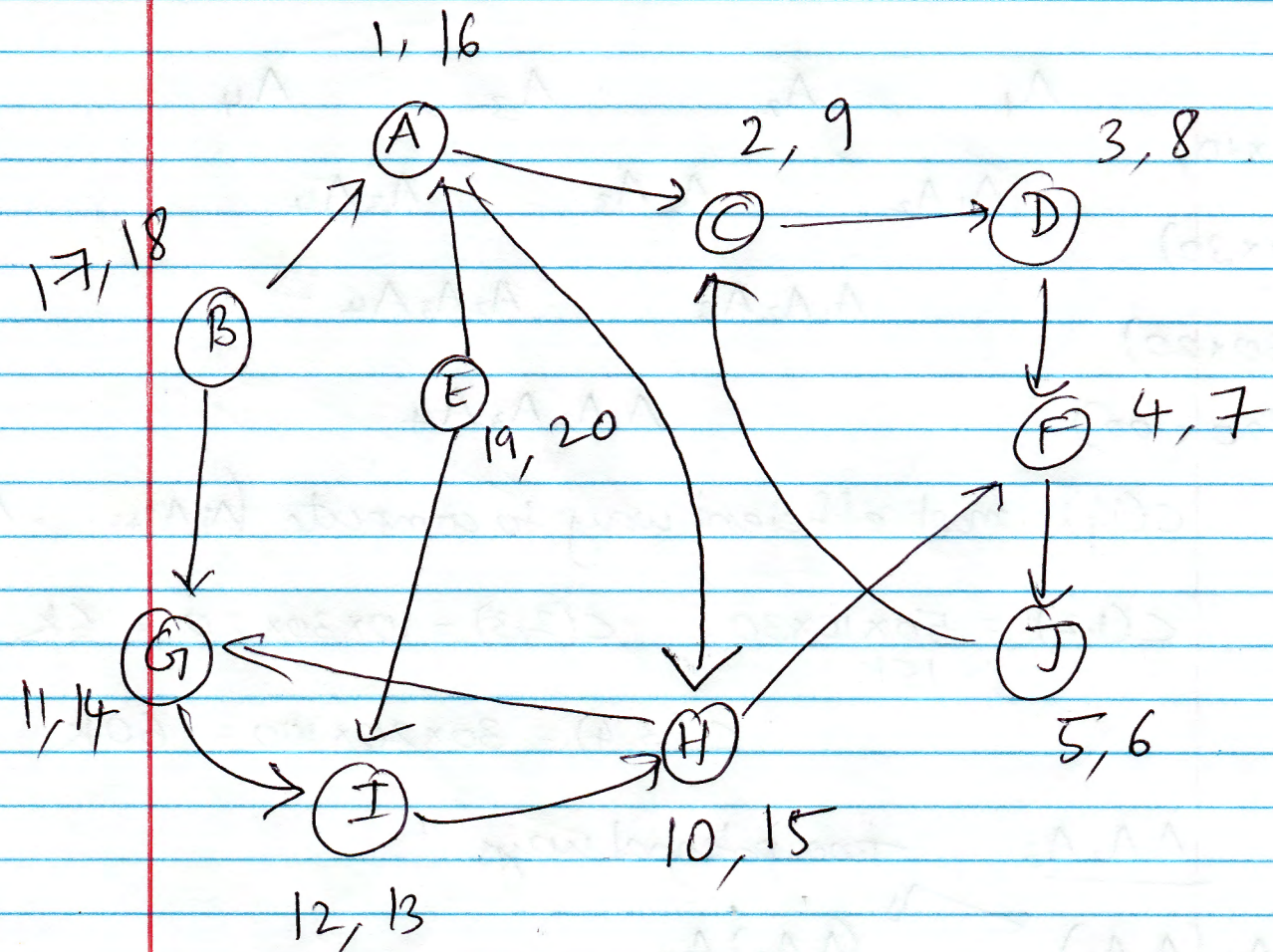
$+ m_0 \times m_1 \times m_4$ | $+ m_0 \times m_2 \times m_4$ | $m_0 \times m_3 \times m_4$

$= 26k +$ | $= 15k + 60k$ | $= 21k +$

$(50)(10)(100)$ | $+ (50)(30)(100)$ | $(50)(20)(100)$

$= 76k$ | $= 225k$ | $= 121k$

DFS



Tree edges

AC, CD, DF, FJ, AH, HG, GI

Back edges

JC, IH

Cross edges

HF, BA, BG, EA, EI

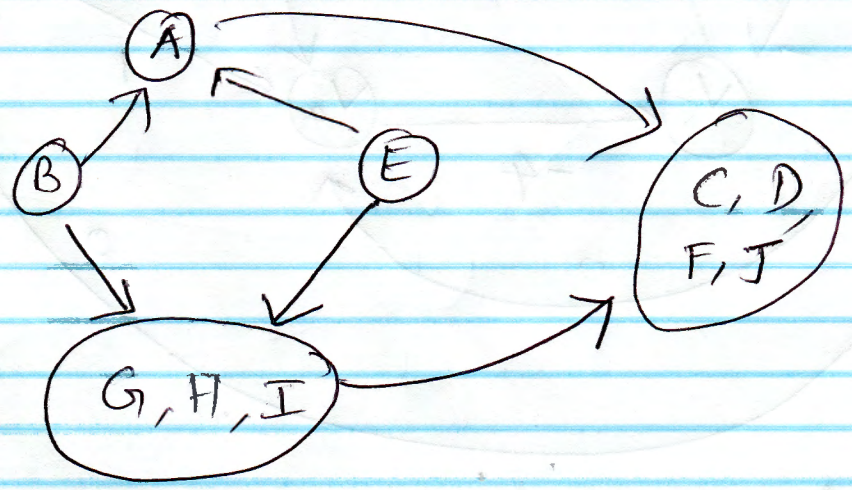
By inspection,

C, D, F, J is an SCC

(sink SCC)

similarly,

G, H, I is an SCC



clearly, A, B, E are all from different SCC's (since there is no way of going from A ⇝ B, A ⇝ E, B ⇝ E, E ⇝ B)

∴ 5 SCC's