DESIGN & ANALYSIS OF ALGORITHM

PCC-CS501

DESIGN & ANALYSIS OF ALGORITHM SCHEDULE ----TOPIC WISE

	Topic	Sub Topic
1	INTRODUCTION	DESIGN OF ALGORITHM ,ANALYSIS OF ALGORITHM, ALGORITHM PROPERTIES
2	FRAMEWORK FOR ALGORITHM ANALYSIS	HOW TO COUNT EXECUTION TIME OF ALGORITHM,INPUT INSTANCES
3	ASYMPTOTIC NOTATION	BEST CASE,AVERAGE CASE, WORST CASE
4	SOLVING RECURRENCE RELATION	SUBSTITUTION METHOD, MASTER THEOREM
5	ALGORITHM DESIGN TECHNIQUES	DIVIDE & CONQUER, GREEDY,DYNAMIC PROGRAMMING, BACKTRACKING,
6	DISJOINT SET MANIPULATION	UNION FIND
7	NETWORK FLOW PROBLEM	FORD FULKERSON ALGORITHM
8	NP COMPLETENESS	NP,NP HARD………ALGORITHM
9	APPROXIMATION ALGORITHM	COMPLEXITY ANALYSIS OF NP COMPETE PROBLEM

NEXT CLASS DISJOINT SET MANIPULATION

- **Kruskal's Algorithm**

- INITIALIZE AN EMPTY EDGE SET T.
- SORT ALL GRAPH EDGES BY THE ASCENDING ORDER OF THEIR WEIGHT VALUES.
- FOR EACH EDGE IN THE SORTED EDGE LIST **CHECK WHETHER IT WILL CREATE A CYCLE** WITH THE EDGES INSIDE T.
- IF THE EDGE DOESN'T INTRODUCE ANY **CYCLES**, ADD IT INTO T.
- IF T HAS (V-1) EDGES, EXIT THE LOOP.
- RETURN T

DISJOINT SET

- DISJOINT SET & OPERATIONS
- DETECTING A CYCLE
- GRAPHICAL REPESENTATION
- ARRAY REPRESENTATION
- WEIGHTED UNION & COLLAPSING FIND

DISJOINT SET & OPERATIONS

- SET with **Distinct elements.**

  Example- A :{2,4,8,16} B:{3,9,18}
- **FIND** – Find a vertex & returns the disjoint set.
- **UNION** – If Find operation generates two different set for an edge then union combines them to a single disjoint set.
- **Application-** kruskal's Algorithm

DISJOINT SET & OPERATIONS

- A :{2,4,8,16} B:{3,9,18}
- Find(2) output A
- Find(3) output B
- Union(2,3)→ C:{2,4,8,16, 3,9,18}

DISJOINT SET & OPERATIONS

- Find(Edge)-----Edge(x,y)-----Find(x) & Find(y)
- If Find(x) & Find(y) outputs are different then using Union(x ,y) can merge the two set into a single set otherwise cycle is confirmed.

NEED OF FIND & UNION OPERATIONS

- TO DETECT CYCLE.

DETECT CYCLE USING IN KRUSKAL'S

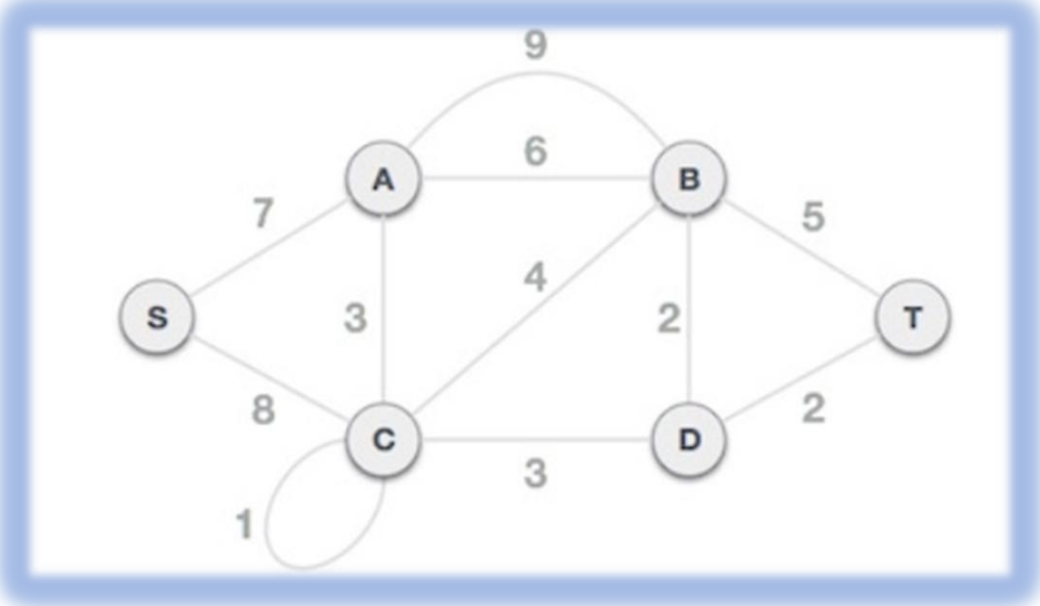
UNIVERSAL SET(U) {S,A,C,B,D,T}

Edge BD find(B)- B find(D)-D union (B,D) E1-{B,D}
Edge DT find(D)- E1 find(T)-U union(D,T) E2- B,D,T
Edge AC find(A)- A find(C)-C union(A,C) E3- A,C
Edge CD find(C)- E3 find(D)-E2 union(C,D) E4- A,B,C,D,T

Edge BC find(B)- E4 find(C)- E4 **CYCLE DETECTION**

Edge BT find(B)- E4 find(T)- E4 **CYCLE DETECTION**
Edge AB find(A)- E4 find(B)- E4 **CYCLE DETECTION**

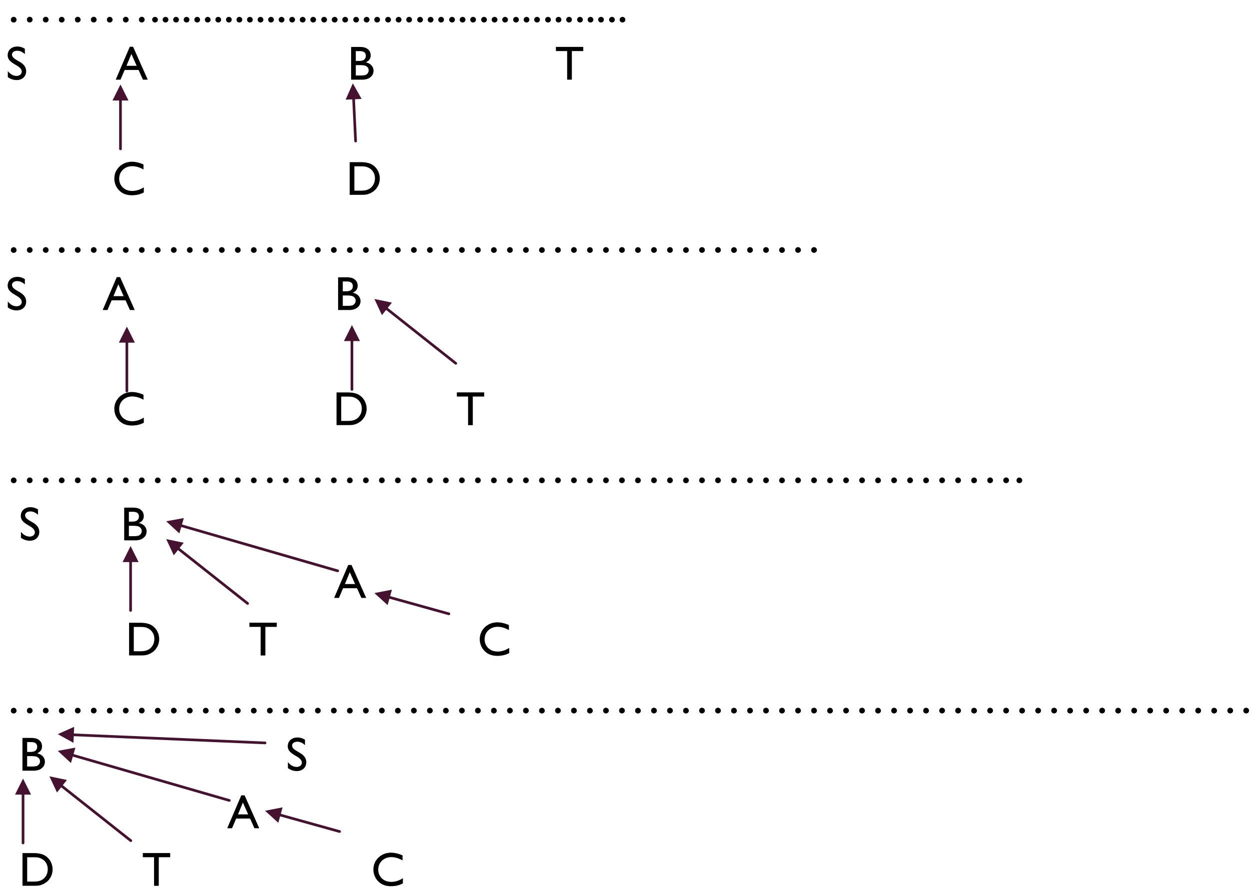
Edge AS find(A)- E4 find(S)- S union(A,S) E5-A, B, C, D,T, S

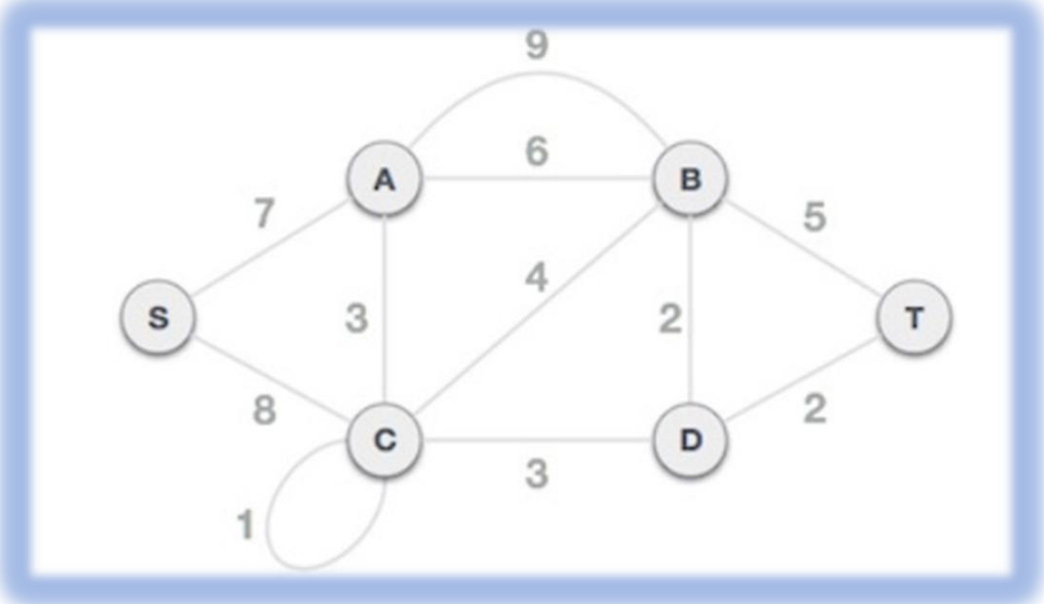


B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

GRAPHICAL REPRESENTATION

UNIVERSAL SET(U) {S,A,C,B,D,T}



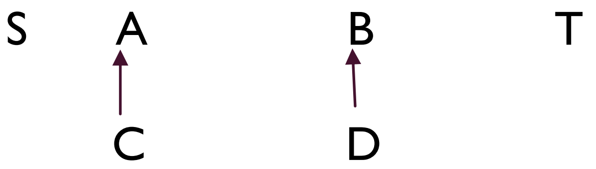


B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

GRAPHICAL & ARRAY REPRESENTATION

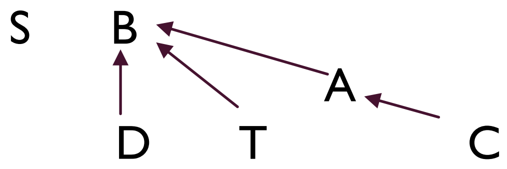
UNIVERSAL SET(U) {S,A,C,B,D,T}

-1	-1	-1	-1	-1	-1
S	A	B	C	D	T

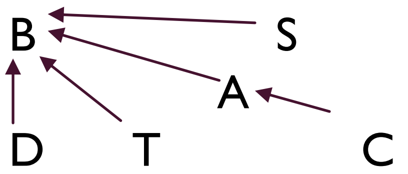


-1	-2	-2	2	3	-1
1	2	3	4	5	6

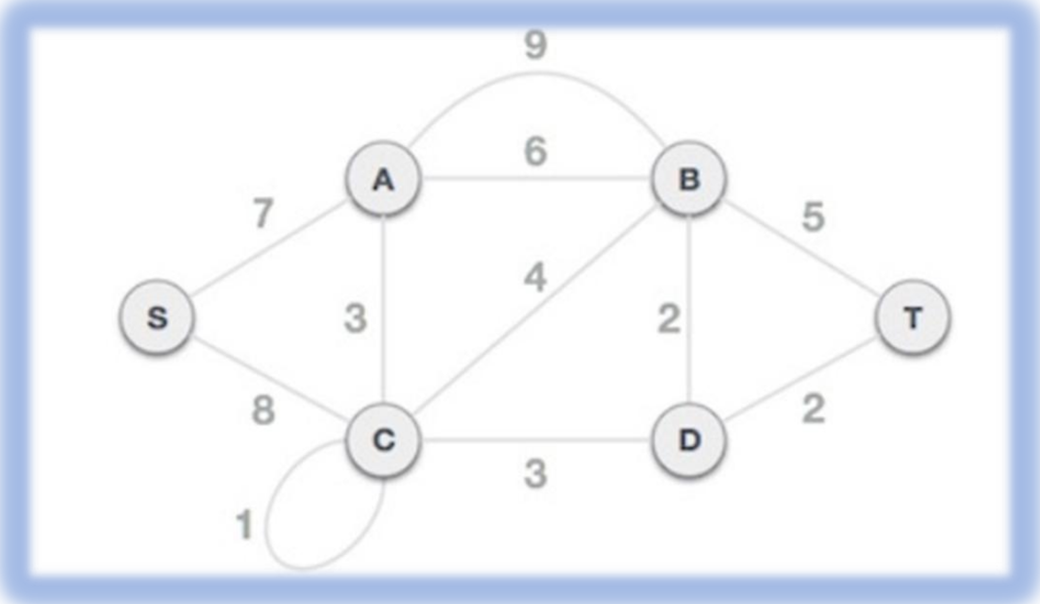
-1	-2	-3	2	3	3



S	A	B	C	D	T
-1	3	-5	2	3	3



3	3	-6	2	3	3

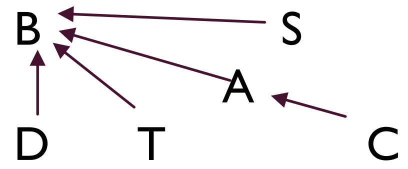


B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

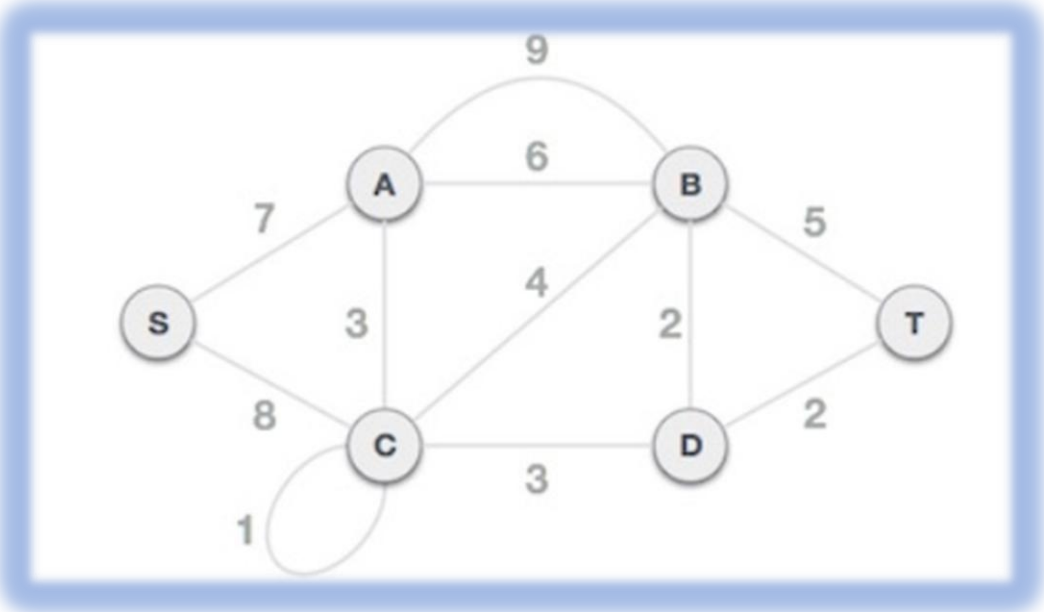
WEIGHTED UNION & COLLAPSING FIND

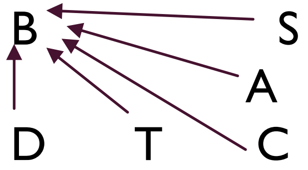
UNIVERSAL SET(U) {S,A,C,B,D,T}

-1	-1	-1	-1	-1	-1
S	A	B	C	D	T



3	3	-6	2	3	3
1	2	3	4	5	6





3	3	-6	3	3	3

Find() takes constant time

B, D	D, T	A, C	C, D	C, B	B, T	A, B	S, A	S, C
2	2	3	3	4	5	6	7	8

NEXT CLASS

- Single Source Shortest Path Problem / Dijkstra’s Algorithm