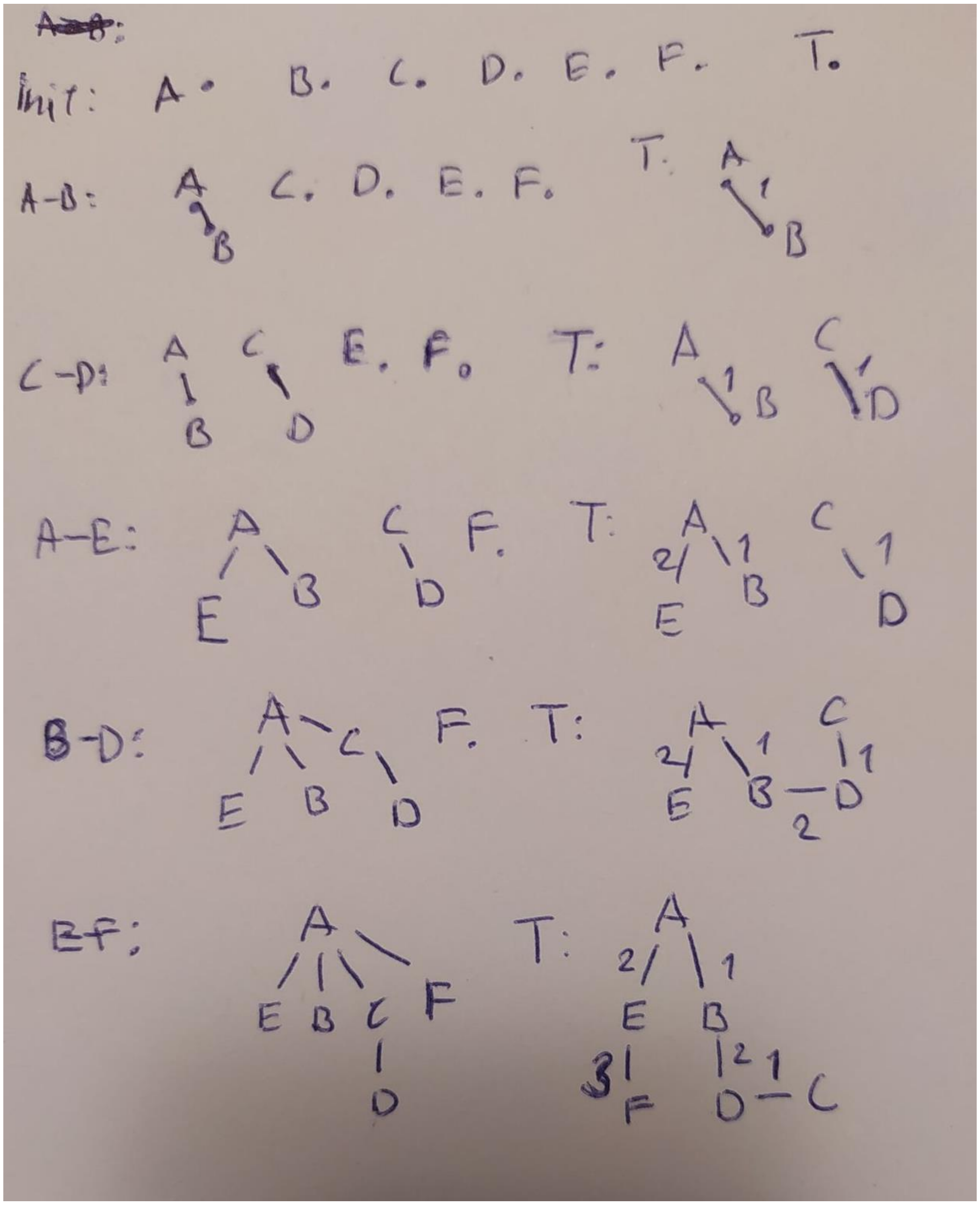
Ochirgarid Chinzorig, Assignment-11B

1. Sorted Edges: E={(A-B,1), (C-D,1),(A-E,2),(B-D,2),(E-F,3),(A-F,4),(F-D,5),(B-C,6),(A-D,7)}
   Disjoint Sets: S={{A},{B},{C},{D},{E},{F}}
   Tree: T={}



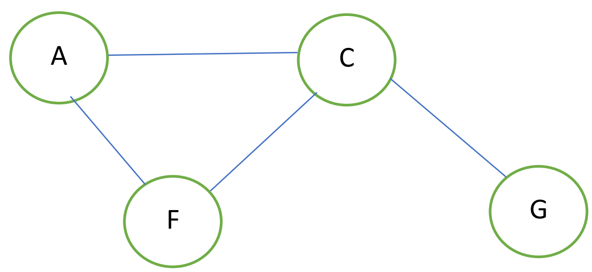
2. G=(V,E), the base for G is U
   a. True. For every edge in E, V has both ends of the edge which means V is a base for G.
   b. There is. A graph without any edge could have an empty base.
   c. V={A,B,C,D,E,F,G,H}, E={(A-B), (A-C), (A-D), (A-E), (A-F), (A-G), (A-H)} has a base U={A}
   d. Every base of the G=(V={V_1,V_2,..,V_{2n}}, E={(V_1,V_{n+1}), (V_2,V_{n+2}),…, (V_n,V_{2n})}) has size of at least n.

e. The running time of the algorithm is $O(M*2^N)$.

```
Algorithm SmallestBase(G):
Input: graph G
Output: smallest base of the G
      U = []
      for v in powerset of V do:
            if v (is base for G) do:
                  if U.size() > v.size()do:
                        U = V
      return U
```

3. G=(V,E), U is the spanning cycle of G

a. Any connected graph that does not contain a cycle is an example. Also, There is some that contains a cycle but does not have any spanning cycle.



b. We can check if there is a cycle exists in every possible permutation of the vertices. The running time of the algorithm will be O(N * N!)

```
Algorithm SpanningCycleDetect(G):
Input: graph G
Output: True if there is a spanning cycle, false otherwise
      for U in (all possible permutation of the V):
            if U is cycle do:
                  return True
      return False
```