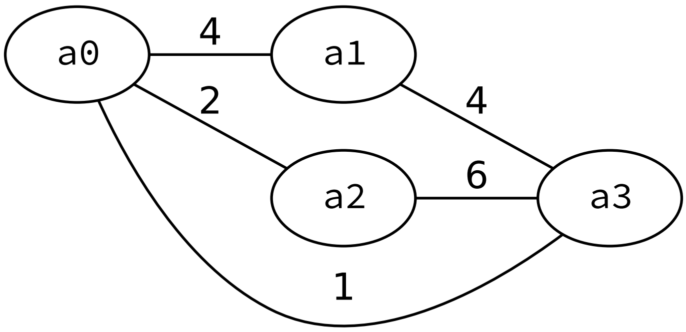
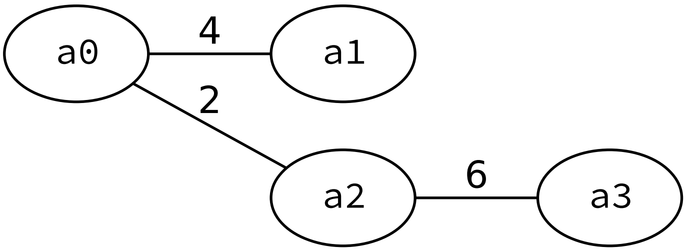
Weighted(Cost) Graph

>> Contains Cost or Weight while traversing from one node to another in the Graph.

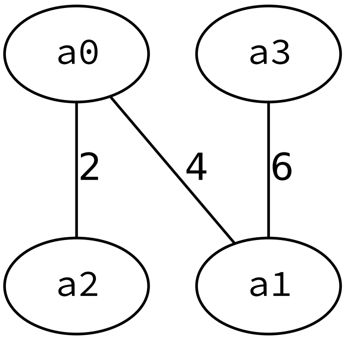


Spanning Tree

A tree representation, subset of a graph, that is constructed from minimum number of edges.



or



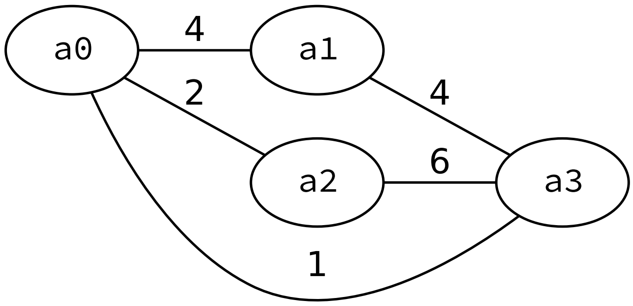
Total vertices = n

Total Edges = n-1

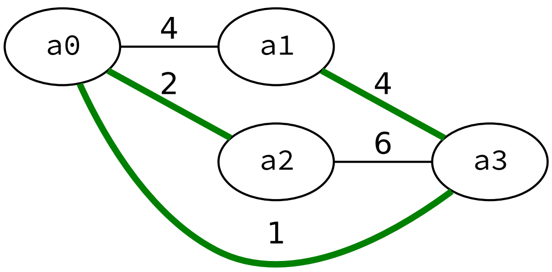
No. of Trees = n^{n-2}

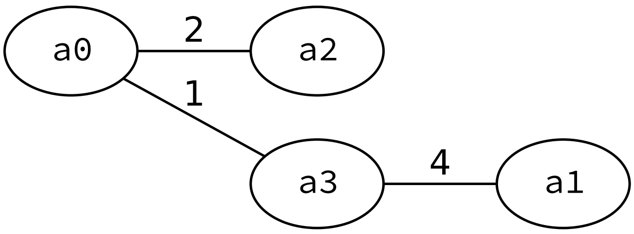
Minimum Spanning Tree

A spanning tree (does not have cycle), that is constructed from minimum cost edges.

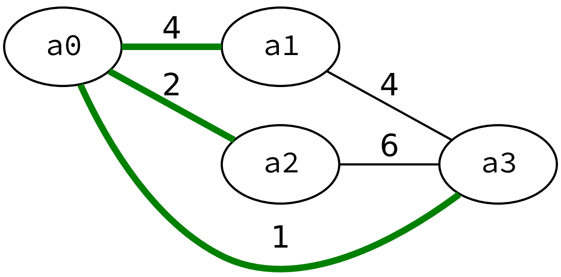


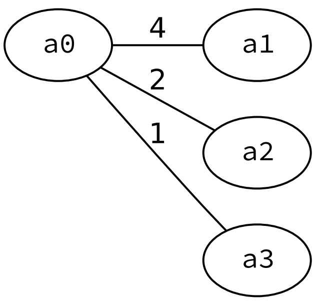
Solution - 1





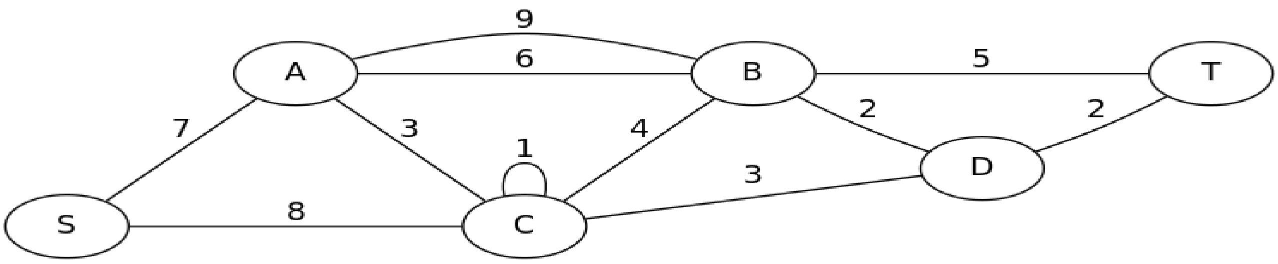
Solution - 2



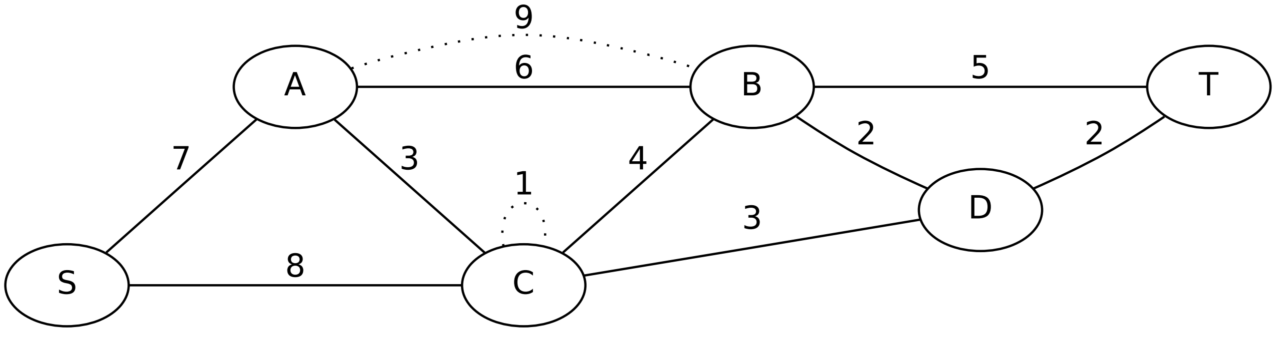


Kruskal’s Algorithm

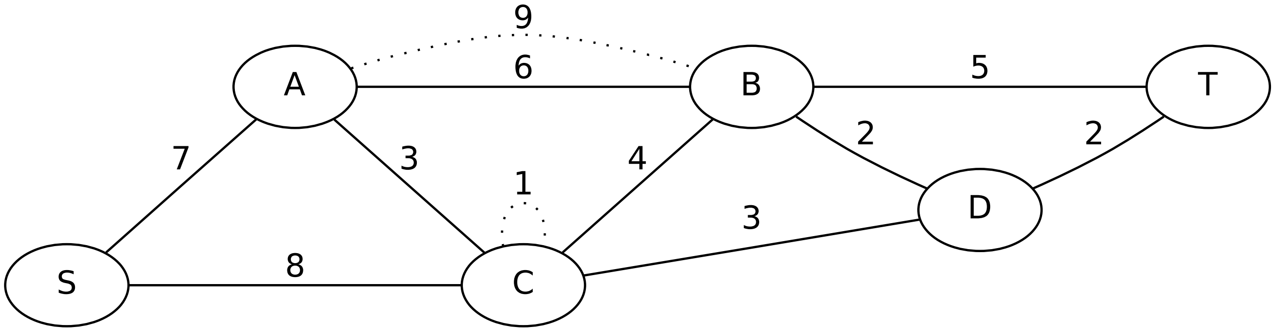
Example:



1. Remove self loop and parallel edges.

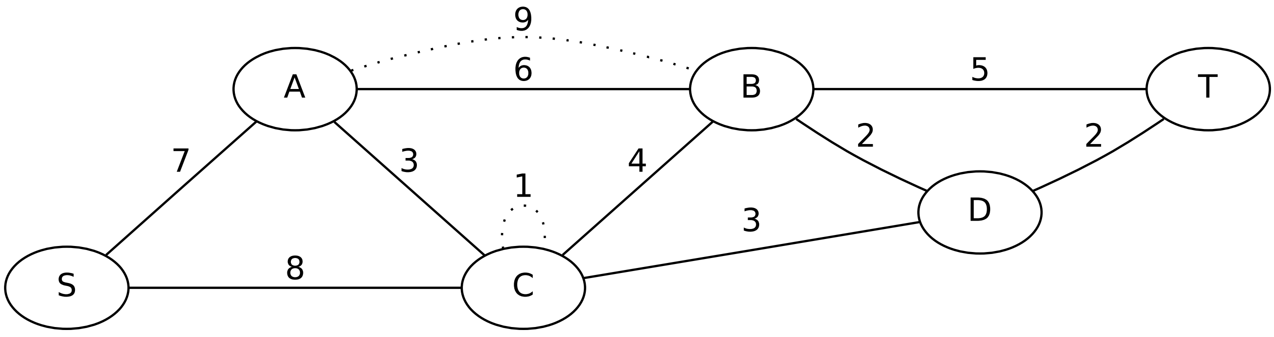


2. Arrange all edges with ascending order.

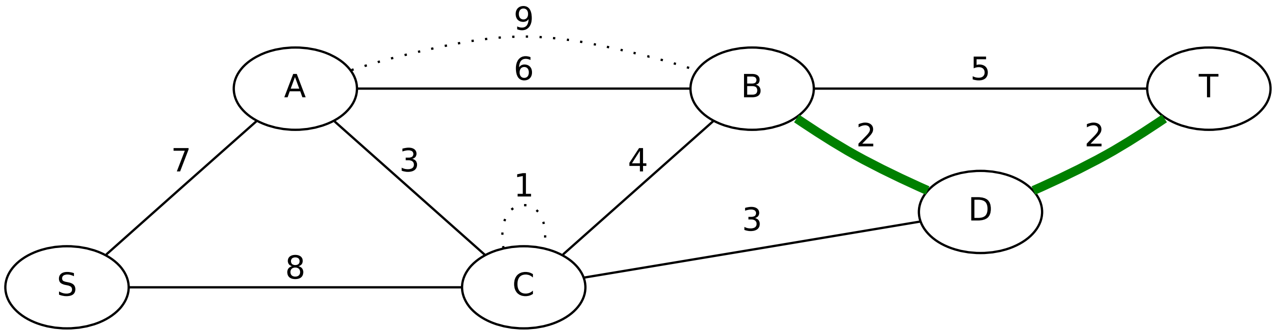


B→D	D→T	A→C	C→D	C→B	B→T	A→B	S→A	S→C
2	**2**	**3**	**3**	**4**	**5**	**6**	**7**	**8**

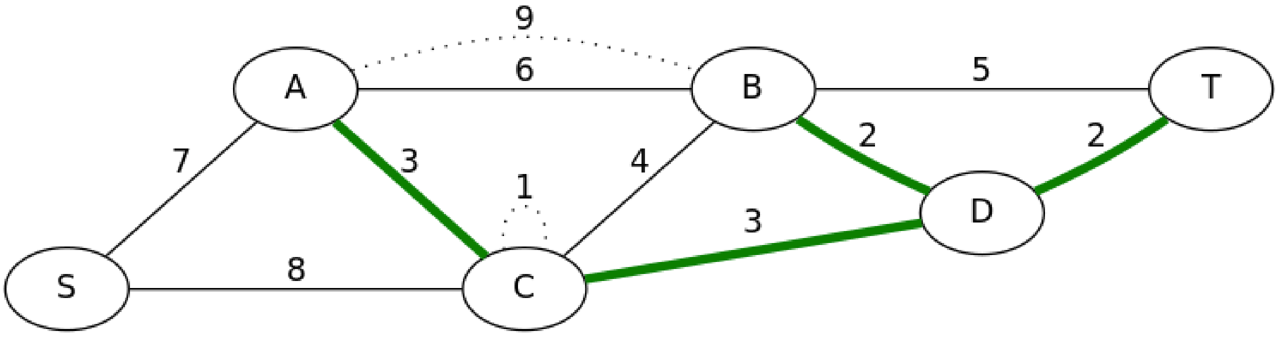
3. Add the edge with least weight.



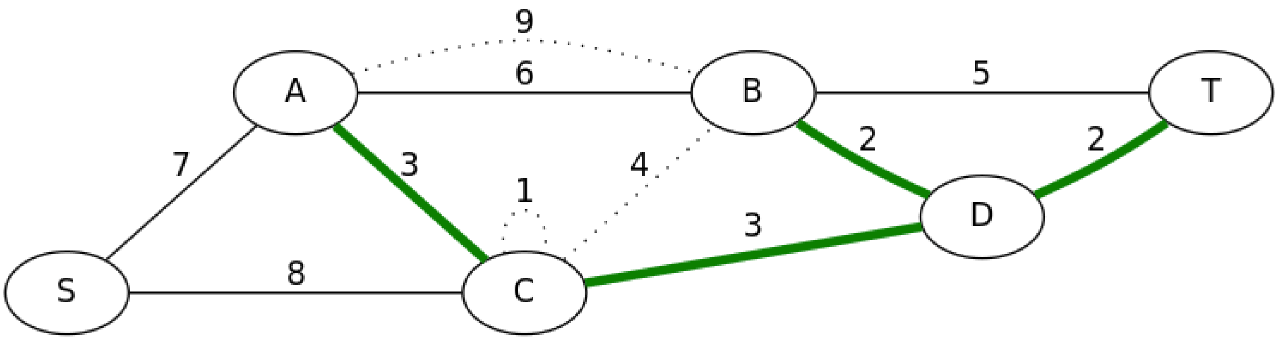
[least cost = 2]



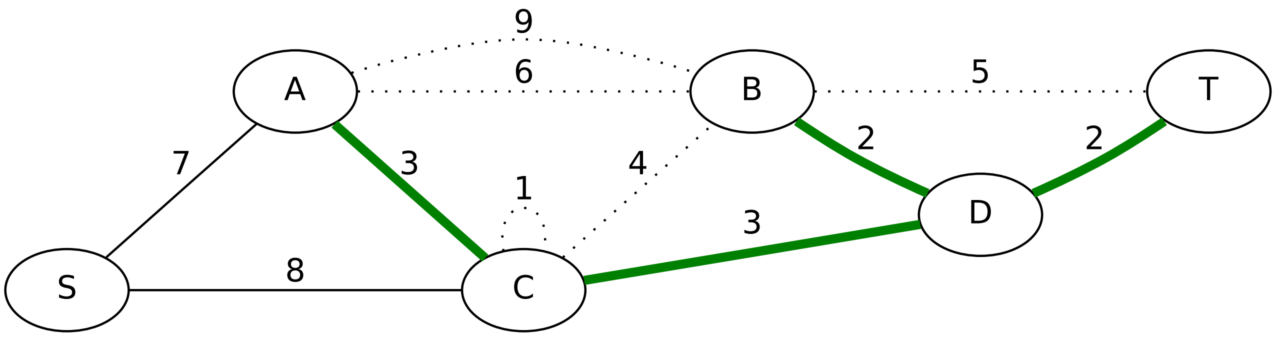
[least cost = 3]



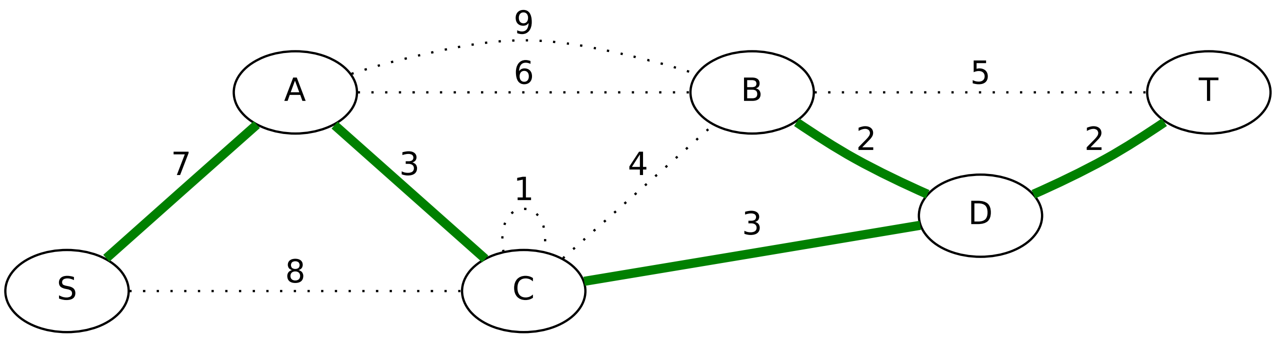
[least cost = 4, but makes cycle, so avoid it.]



[least cost = 5 & 6, but they also makes cycle, so avoid it.]



[Now, the least cost = 7 from A → S]



Hence, all the nodes are covered.

Total Nodes = 6.

Total Edges = 5[Green Colors]

Total Cost = 2 + 2 + 3 + 3 + 7

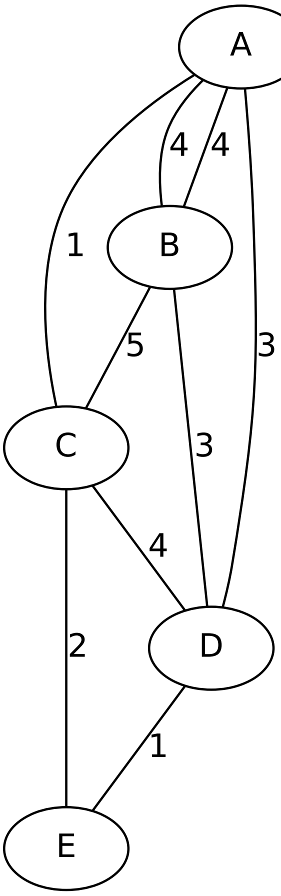
= **17**

Complexity = e log e, (e no. of edges.)

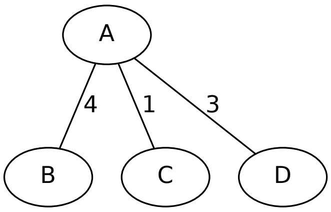
Round Robin Algorithm

(Tarjan And Cheriton's Algorithm For Finding MST. _pp.577 Of Book)

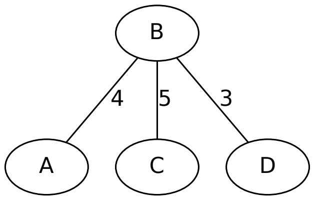
Generate all the priorities trees:



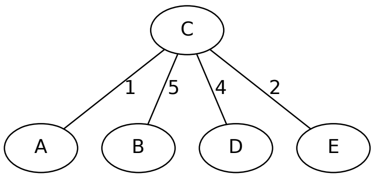
T1-Vertices(A) PQ: (A C 1), (A D 3), (A B 4)



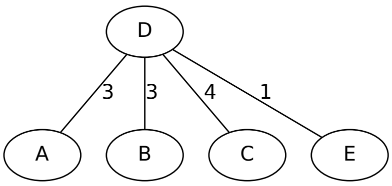
T2-Vertices(B) PQ: (B D 3), (B A 4), (B C 5)



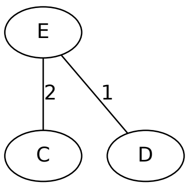
T3-Vertices(C) PQ:(C A 1),(C E 2),(C D 4),(C B 5)



T4-Vertices(D) PQ: (D E 1), (D A 3), (D B 3), (D C 4)



T5-Vertices(E) PQ: (E D 1), (E C 2)



Every partial tree has one vertex designated as its "root".

All Priorities Queues:

T1. Vertices: A PQ: **(A C 1)**, (A D 3), (A B 4)
T2. Vertices: B PQ: (B D 3), (B A 4), (B C 5)
T3. Vertices: C PQ: (C A 1), (C E 2), (C D 4), (C B 5)
T4. Vertices: D PQ: (D E 1), (D A 3), (D B 3), (D C 4)
T5. Vertices: E PQ: (E D 1), (E C 2)

Now,
T1. Vertices: A P1: (A C 1), (A D 3), (A B 4)
Remove (A C 1), say α, v1 and v2 are two vertices of α.

α: (A C 1) v1: A v2: C
(A C 1) is a component of the MST

T3. Vertices: C PQ: (C A 1), (C E 2), (C D 4), (C B 5)

Combine AC, (CA and AC edges are same with 1 weight.)
[**(A C 1) is used to combine the partial Tree.**]

T2. Vertices: B PQ: **(B D 3)**, (B A 4), (B C 5)
T4. Vertices: D PQ: (D E 1), (D A 3), (D B 3), (D C 4)
T5. Vertices: E PQ: (E D 1), (E C 2)
T13.Vertices: AC PQ: (C A 1), (C E 2), (A D 3), (A B 4), (C D 4), (C B 5)

Merging BD,
[**(B D 3) is used to combine the partial Tree.**]

T5. Vertices: E PQ: **(E D 1)**, (E C 2)
T13. Vertices: AC
PQ: (C A 1), (C E 2), (A D 3), (A B 4), (C D 4), (C B 5)
T24. Vertices: BD
PQ: (D E 1), (D A 3), (D B 3), (B A 4), (D C 4), (B C 5)

Merging EBD,
[**(E, D, 1) is used to combine the partial Tree.**]

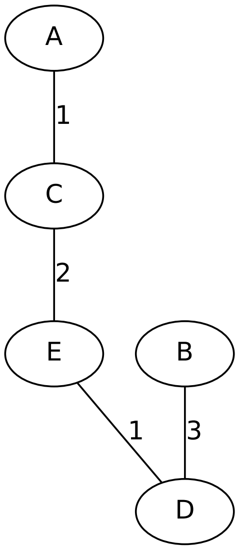
T13. Vertices: AC
PQ: (C A 1), **(C E 2)**, (A D 3), (A B 4), (C D 4), (C B 5)
T524. Vertices: EBD
PQ: (D E 1),(E C 2),(D A 3),(D B 3),(B A 4),(D C 4),(B C 5)

Repeating,
[**(C, E, 2) is used to combine the partial Tree.**]
T13524-Vertices: ACEBD
PQ: (D E 1), (E C 2), (A D 3), (D A 3), (D B 3),
(A B 4), (C D 4), (B A 4), (D C 4), (C B 5), (B C 5)

Result Tree i.e. MST:

Edges: {
(A C 1),
(B D 3),
(E D 1),
(C E 2)
}

And the MST is: O(e*loglogn)



Round Robin Algorithm

Algorithm 4.4.4: CHERITONTARJAN(G)

initialize *Tree* to contain no edges
for each $u \in V(G)$
do {
  initialize T_u to contain only u
  create PQ_u
  for each $v \longrightarrow u$
    do add uv to PQ_u
  comment: each edge will appear in two priority queue
}
comment: the forest currently has $|G|$ nodes and 0 edges

$t \leftarrow 0$
while $t < |G| - 1$
do {
  select a component T_u
  repeat
    select the minimum edge $xy \in PQ_u$ and determine
    which components x and y are in, say $x \in T_u$ and $y \in T_v$
  until $T_u \neq T_v$
  comment: xy connects two different components

  merge T_u and T_v
  merge PQ_u and PQ_v
  add xy to *Tree*
  $t \leftarrow t + 1$
}

Source: http://books.google.co.uk/books?id=zxSmHAoMiRUC&pg=PA78&lpg=PA78&dq=cheriton-tarjan%20algorithm&source=bl&ots=LQWYqzK7bq&sig=mmKjYZ7pevW3vkPD1Xmg7GC5BOI&hl=en&sa=X&ei=0AX4UsykO_Cp7AbWiICADg&ved=0CC0Q6AEwAA#v=onepage&q=cheriton-tarjan%20algorithm&f=false

Greedy Algorithm

- uses local optimum solution hoping to get global optimum solution.
- this happens by chance only, not favorable for all the problems.

Currency Problem Using Greedy Algo:

>> Accumulate Rs 18 from the available currencies [Rs1, Rs2, Rs5, Rs10.] with minimum numbers.

Solution: Rs[10 + 5 + 2 + 1], **Worked.**

>> Accumulate Rs 15 from the available currencies [Rs1, Rs7, Rs10].
Solution: Rs[10 + 1 + 1 + 1 + 1 + 1]
Not Worked.
Could be: Rs[7 + 7 + 1]