HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

Algorithms and Data Structures

Laboratory work #8

Michael Kosyakov
Associate Professor

Denis Tarakanov
Assistant

Aglaya Iliina
Associate Professor

hduitmo.ads@yandex.ru

Classes plan

1. Previous homework problem #1450 “Russian Pipelines”
2. Problem #1160 “Network”
3. Task for homework
4. Explanation of test 4

Problem #1450 "Russian Pipelines"

HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

- Link to the problem's description https://acm.timus.ru/problem.aspx?space=1&num=1450&locale=en

- Russian pipeline system consists of N transfer stations. For each of M pipelines the numbers of stations A and B, which are connected by this pipeline, and its profitability C are known. Profitability of a pipeline is an amount of dollars, which will be yielded in taxes by transferring the gas through this pipeline.
- Pipelines are unidirectional. If it is possible to transfer the gas from station X to Y (perhaps, through some intermediate stations), then the reverse transfer from Y to X is impossible.
- You need to find a route to transfer gas from the starting to the final station. A profitability of this route should be maximal.

Problem #1450 "Russian Pipelines"

- Target – find path with maximum cost (length)
- Main idea – adapt shortest path algorithm

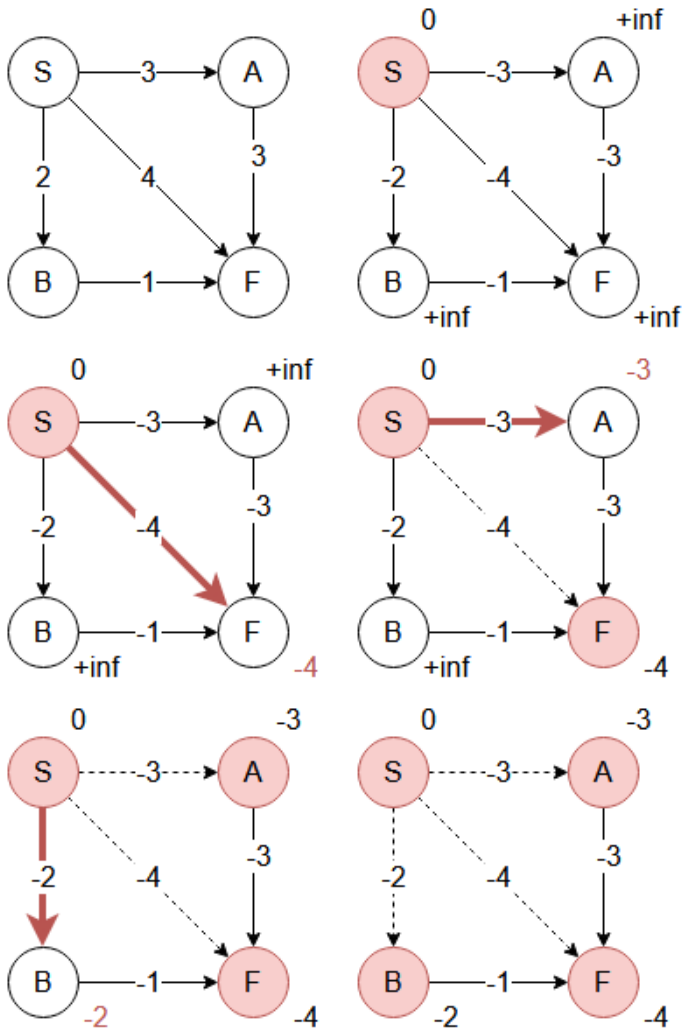
- Approach 1. Adapt Dijkstra's algorithm

- Profitability is positive integer
- What if we can change sign of all costs?
- What if we can change min to max?

Problem #1450 "Russian Pipelines"

HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

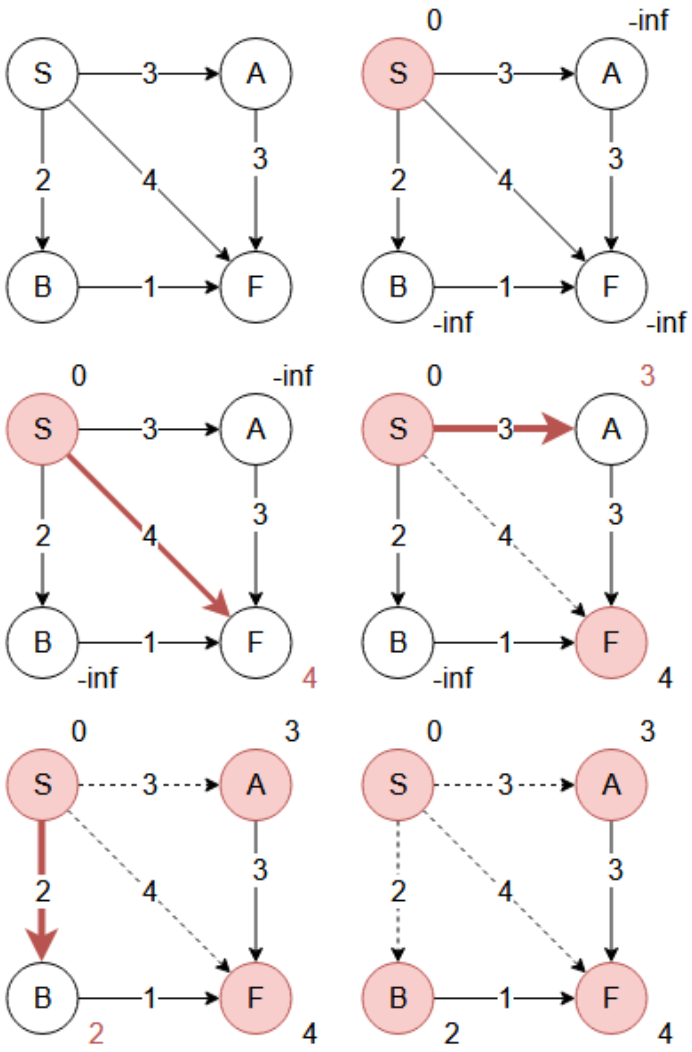
- Approach 1. Adapt Dijkstra's algorithm
  - Change sign for all edges
  - Find edge to unused vertex, which gives minimum DS
  - After all vertices were added algorithm stops
  - Path is not maximum!
  - Adding of new edge can reduce DC – algorithm doesn't handle it
  - Algorithm doesn't work



Problem #1450 "Russian Pipelines"

HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

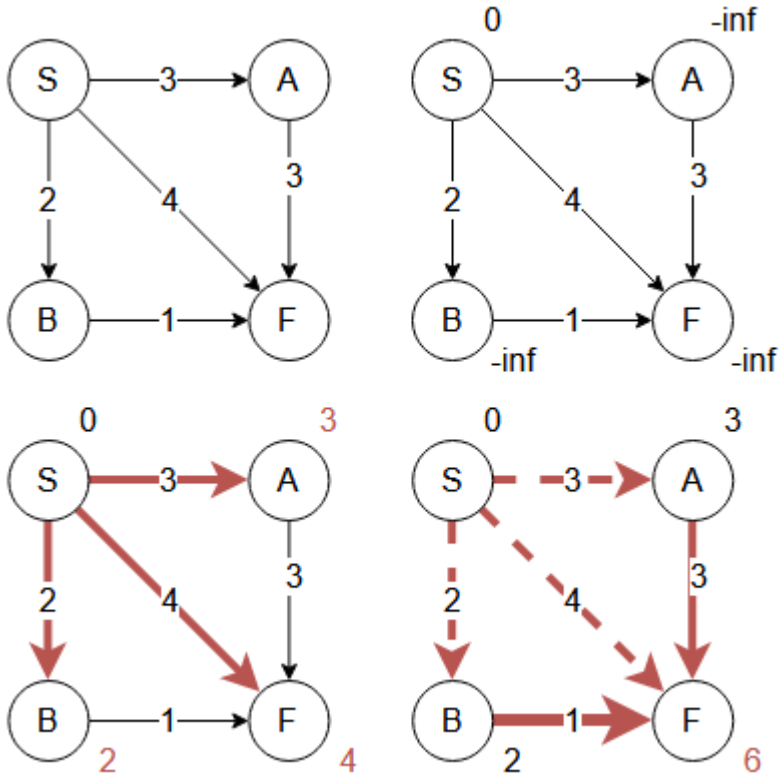
- Approach 1. Adapt Dijkstra's algorithm
  - Change min function to max
  - Apply -inf as Dijkstra score for unused vertices
  - Find edge to unused vertex, which gives maximum DS
  - After all vertices were added algorithm stops
  - Path is not maximum!
  - Algorithm doesn't work



Problem #1450 “Russian Pipelines”

HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

- Approach 2. Adapt Bellman-Ford algorithm
  - Supports negative weights
  - No negative cycles
  - Both approaches can work with Bellman-Ford algorithm
  - Complexity is $O(n * m)$

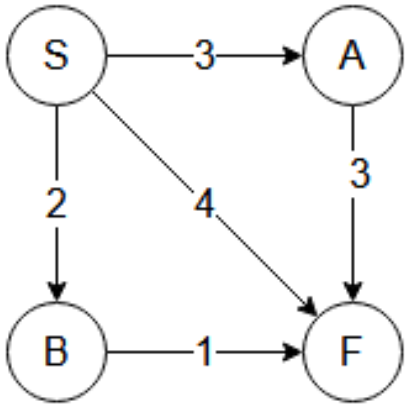


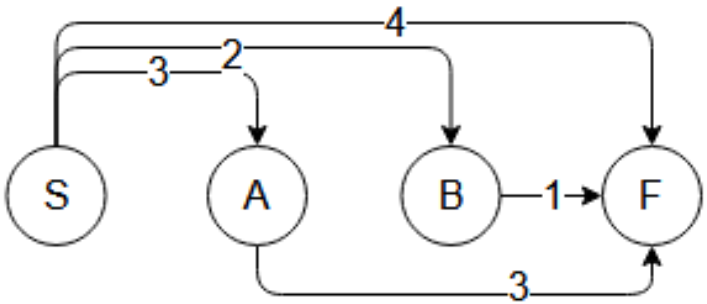
Problem #1450 "Russian Pipelines"

- Approach 3. Topological sort + dynamic
  - "If it is possible to transfer the gas from station X to Y (perhaps, through some intermediate stations), then the reverse transfer from Y to X is impossible"
  - Directed acyclic graph (DAG)
  - Topological sort can be applied
  - Calculate cost of paths in order of sort
  - For each vertex – paths to its predecessors are already calculated, no unused edges (compared to Dijkstra)!
  - Change min to max

Problem #1450 "Russian Pipelines"

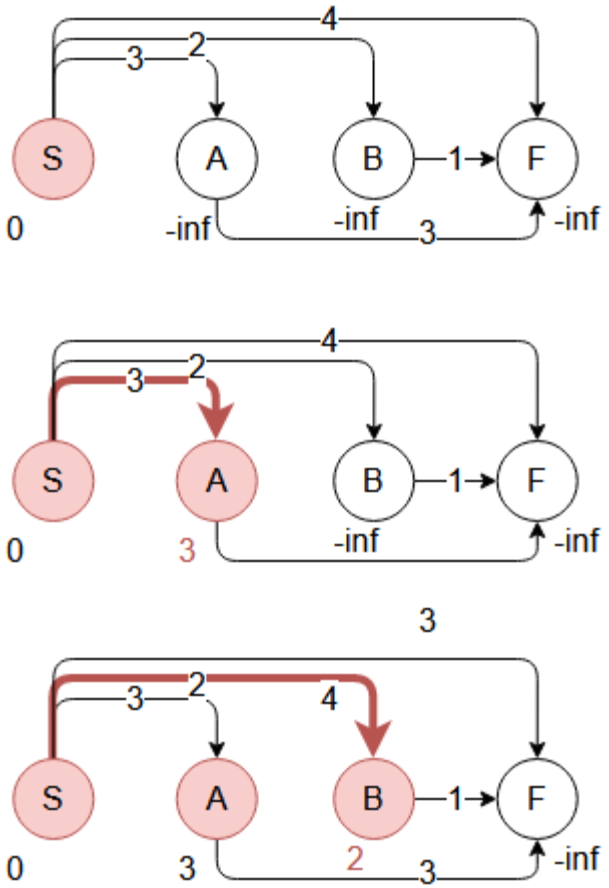
- Approach 3. Topological sort + dynamic
  - Sort the graph
  - Complexity is $O(n)$





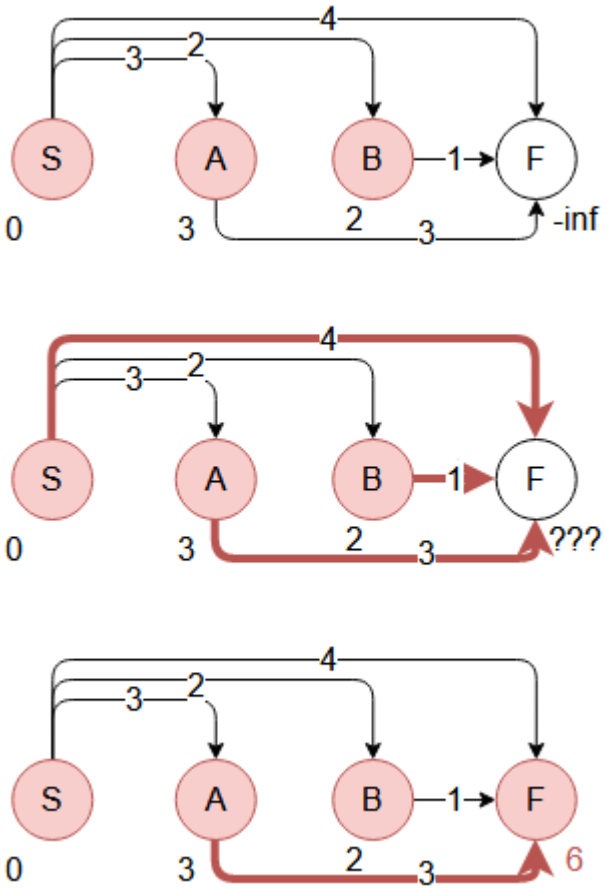
Problem #1450 "Russian Pipelines"

- Approach 3. Topological sort + dynamic
  - Set maximum cost of path to S as 0
  - In order of topological sort, calculate cost for each vertex $Cost = \max(ParentCost + EdgeFromParentWeight)$



Problem #1450 "Russian Pipelines"

- Approach 3. Topological sort + dynamic
  - Because of sorted order, on each step it is guaranteed that cost can't be greater
  - Complexity is $O(m)$
  - Total complexity is $O(n+m)$



- If there is no path from S to F – result is "No solution"

Problem #1160 “Network”

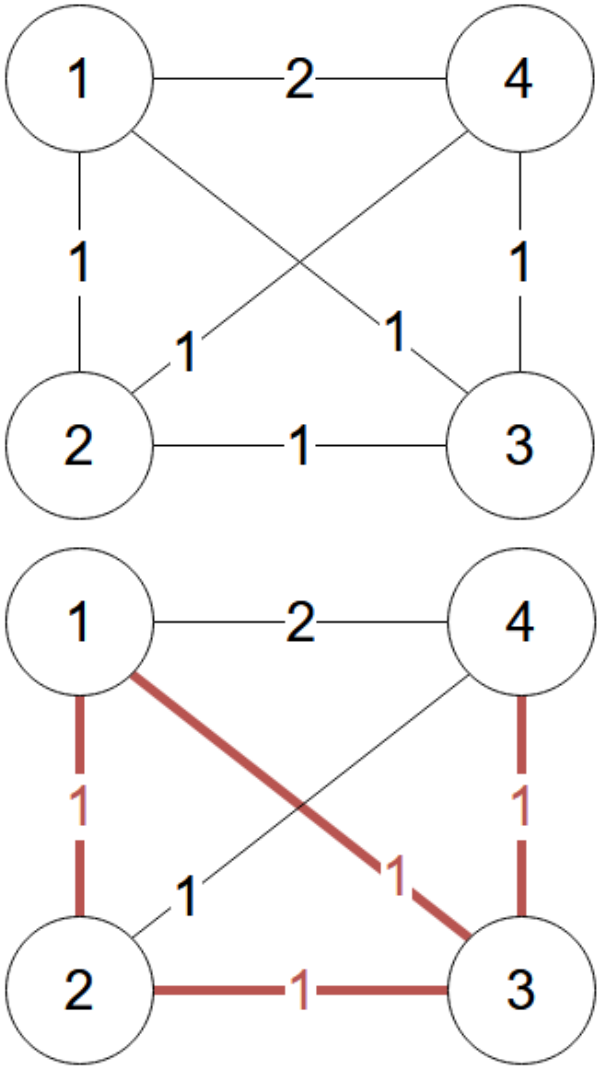
HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

- Link to the problem's description https://acm.timus.ru/problem.aspx?space=1&num=1160&locale=en

- There will be N hubs in the company, they can be connected to each other using cables. Since each worker of the company must have access to the whole network, each hub must be accessible by cables from any other hub.
- Since cables of different types are available and shorter ones are cheaper, it is necessary to make such a plan of hub connection, that the maximum length of a single cable is minimal. There is another problem – not each hub can be connected to any other one because of compatibility problems and building geometry limitations.

Problem #1160 "Network"

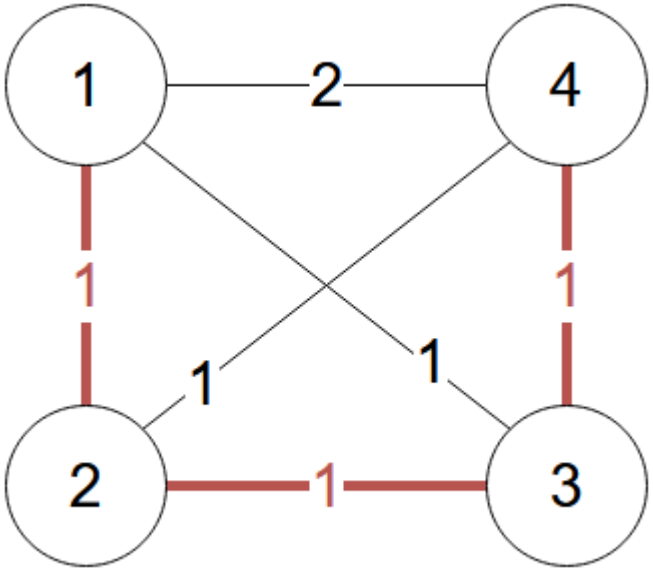
- Graph is connected and weighted
- Need to get such subgraph that maximum length of edges is minimized
- Are all edges of example required?



Problem #1160 "Network"

- Spanning tree is enough
- All vertices are connected

- Maximum length of edges is minimized
- Minimum bottleneck spanning tree (MBST)



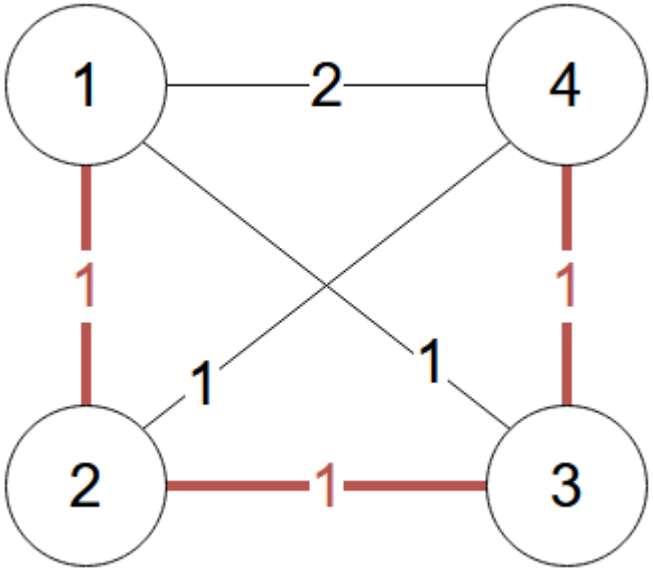
$$\sum weigth(edge_i) = 3$$

$$bottleneck = 1$$

Problem #1160 "Network"

HDU-ITMO Joint Institute
杭州电子科技大学 圣光机联合学院

- Minimum bottleneck spanning tree (MBST)
- How to find it in a graph?
- Minimum spanning tree (MST)

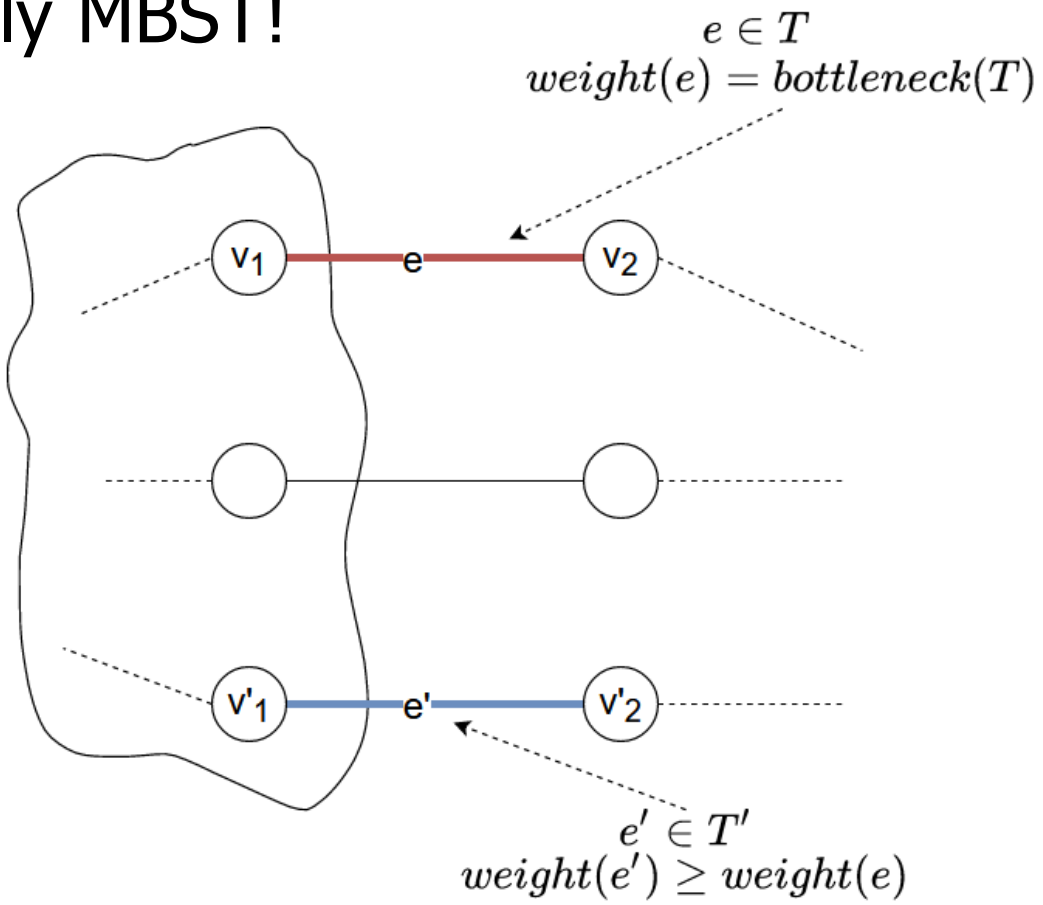


Problem #1160 "Network"

- MST is necessarily MBST!
- Let's prove it

- G – graph
- T – MST(G)
- T' – MBST(G)

- Cut property



Problem #1160 "Network"

- Two possibilities:

1. $weight(e') > weight(e)$

$$bottleneck(T') \geq weight(e') > weight(e) = bottleneck(T)$$

$$bottleneck(T') > bottleneck(T)$$

$$T' is\ MBST \Rightarrow contradiction$$

2. $weight(e') = weight(e)$

$$bottleneck(T') \geq weight(e') = weight(e) = bottleneck(T)$$

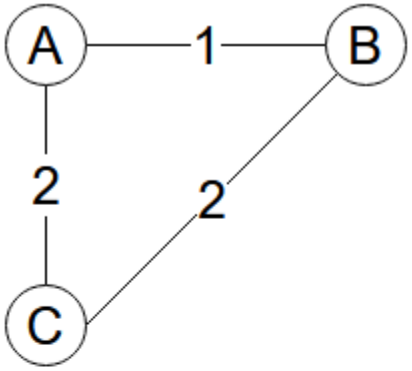
$$bottleneck(T') \geq bottleneck(T)$$

$$T' is\ MBST \Rightarrow bottleneck(T') = bottleneck(T)$$

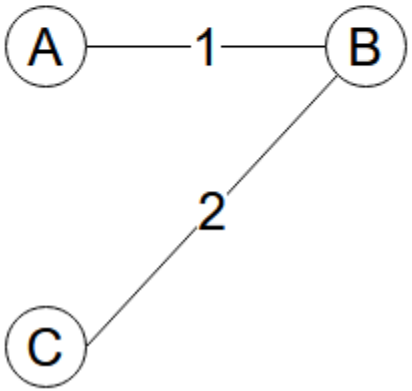
$$T\ is\ also\ MBST\ \blacksquare$$

Problem #1160 "Network"

- MBST is not necessarily MST!
- Check simple graph



Spanning tree 1:

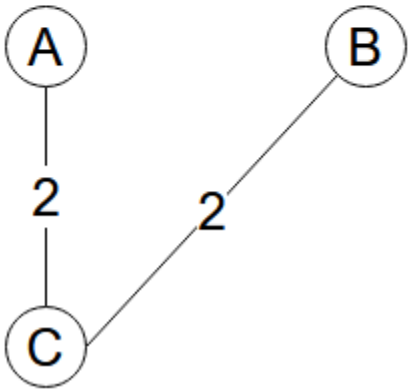


$bottleneck(S_1) = 2$

$\sum weight(e) = 3, e \in S_1$

MST and MBST

Spanning tree 2:



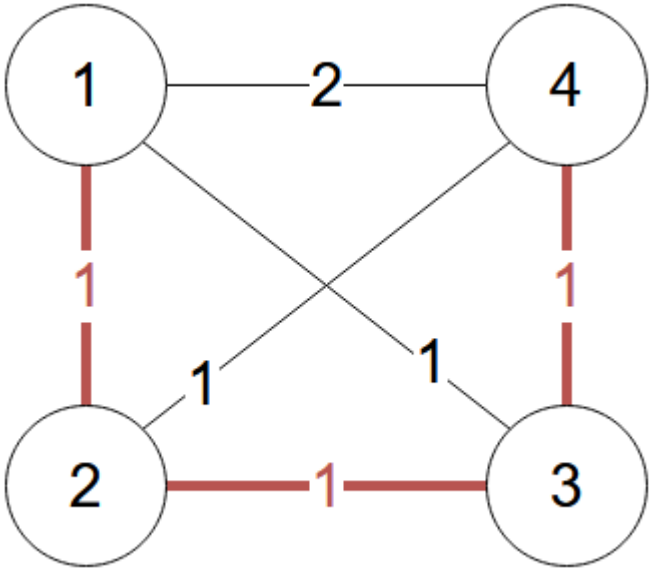
$bottleneck(S_2) = 2$

$\sum weight(e) = 4, e \in S_2$

MBST, but not MST!

Problem #1160 "Network"

- How to find MST?
- Prim's algorithm



Mandatory task

1. Prepare source code to solve problem #1160 “Network” https://acm.timus.ru/problem.aspx?space=1&num=1160&locale=en
2. Pass tests on Timus system for this problem https://acm.timus.ru/submit.aspx?space=1&num=1160
3. Prepare a report with algorithm complexity and explanation
   Use template.docx to prepare report and send it to hduitmo.ads@yandex.ru with correct subject

Task for homework

You can solve following problem to get extra 2 points:

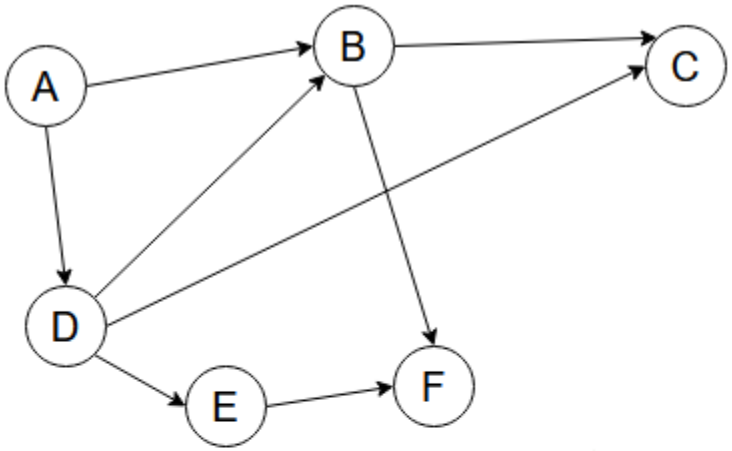
1. Problem #1162 “Currency Exchange”
   https://acm.timus.ru/problem.aspx?space=1&num=1162&locale=en
   N.B. Report for this problem should contain explanation, which algorithm was chosen

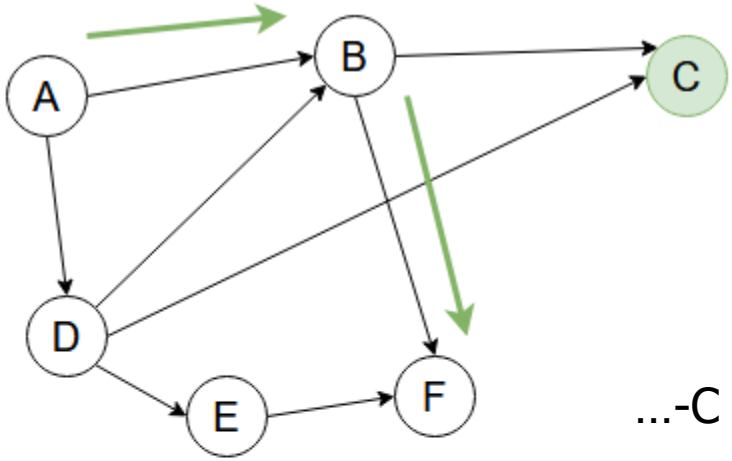
Explanation of test 4

- Task 1
- Topological sort algorithm
- Main rules and steps
  - Topological sort can be applied for DAG only
  - Call DFS for this graph
  - When DFS returns from vertex – all its children are sorted, insert it onto front of linked list
  - DFS always iterates in alphabetical order by description

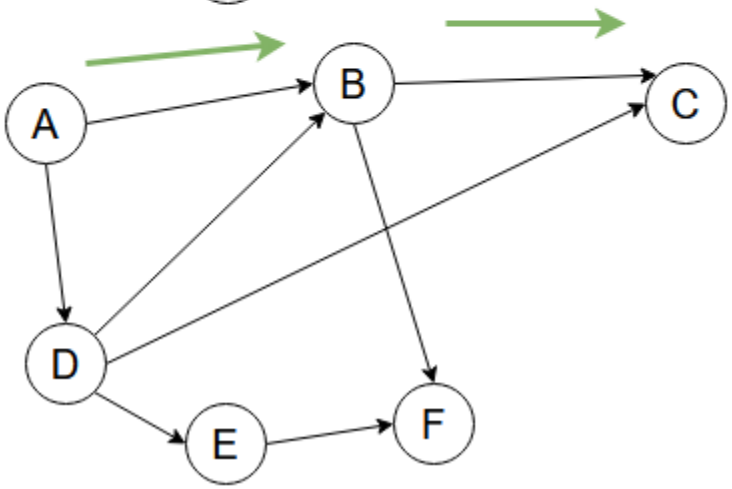
Explanation of test 4

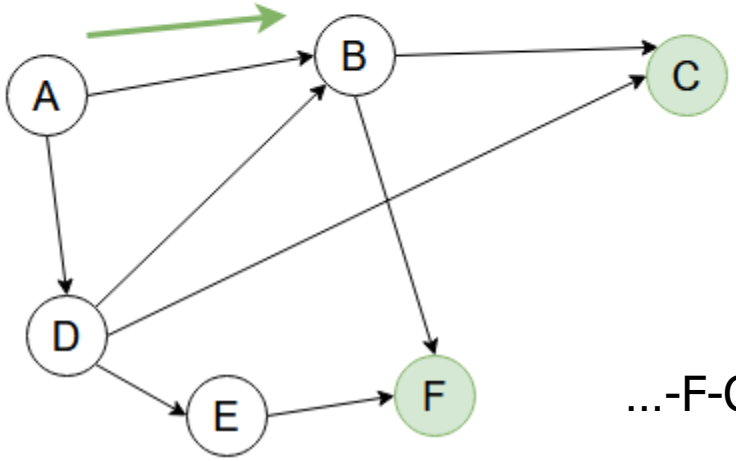
- Task 1. Example 1





...-C

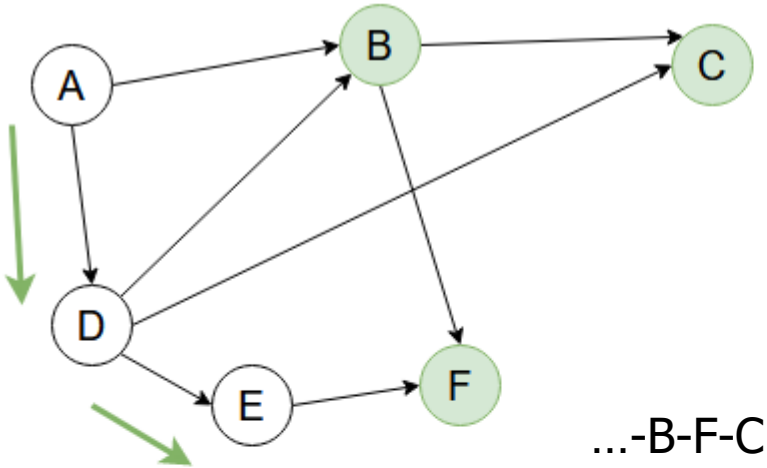




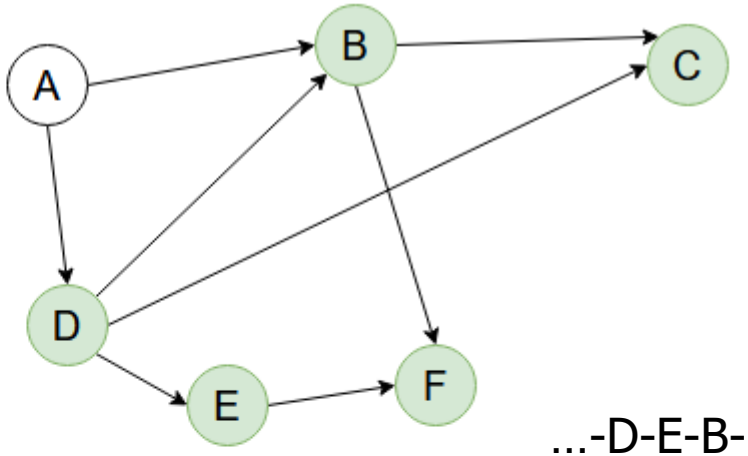
...-F-C

Explanation of test 4

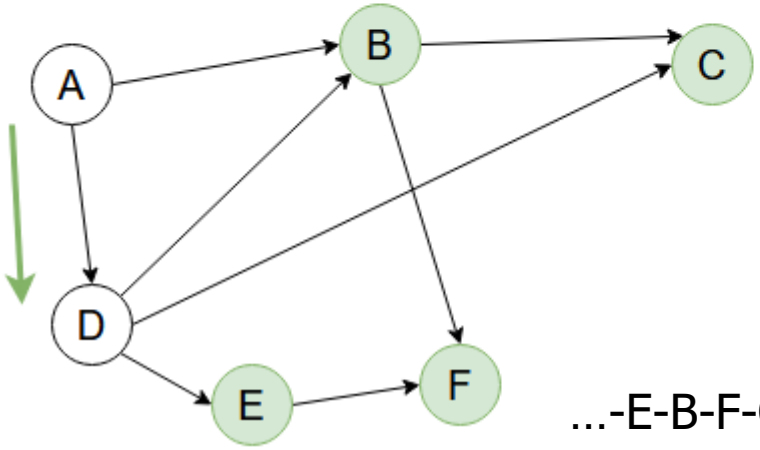
- Task 1. Example 1



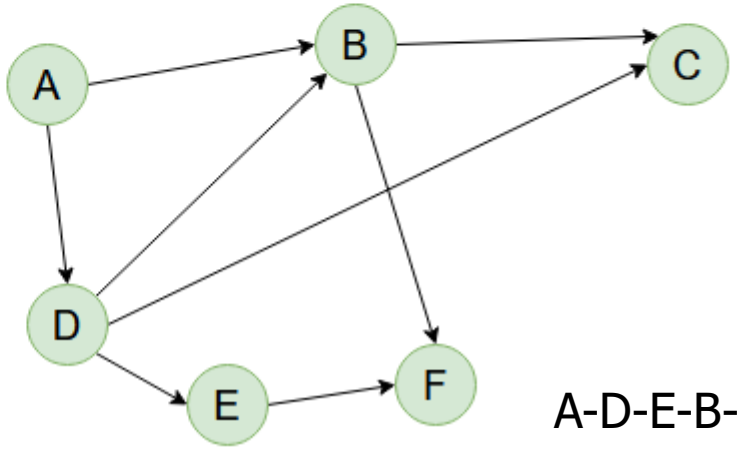
...-B-F-C



...-D-E-B-F-C



...-E-B-F-C

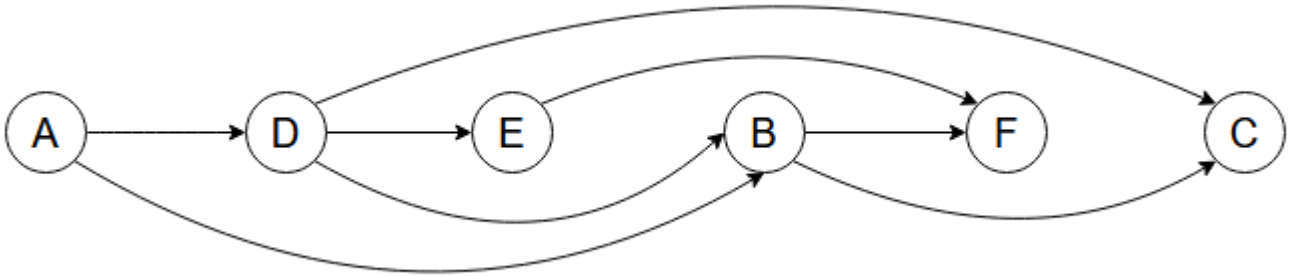


A-D-E-B-F-C

Explanation of test 4

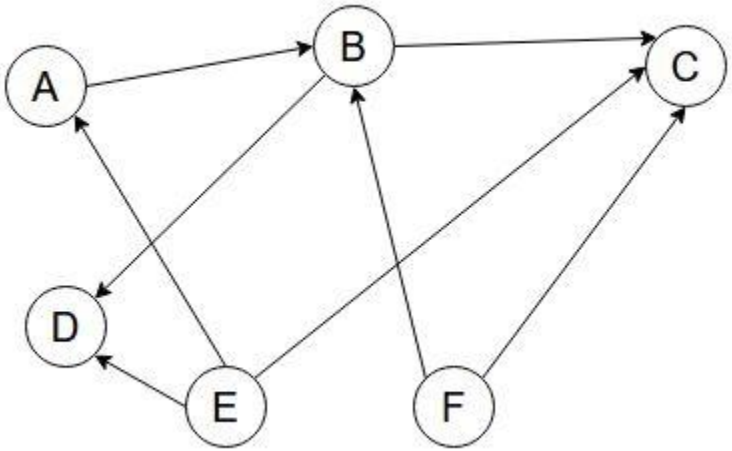
- Task 1. Example 1

- "Sorted" graph



Explanation of test 4

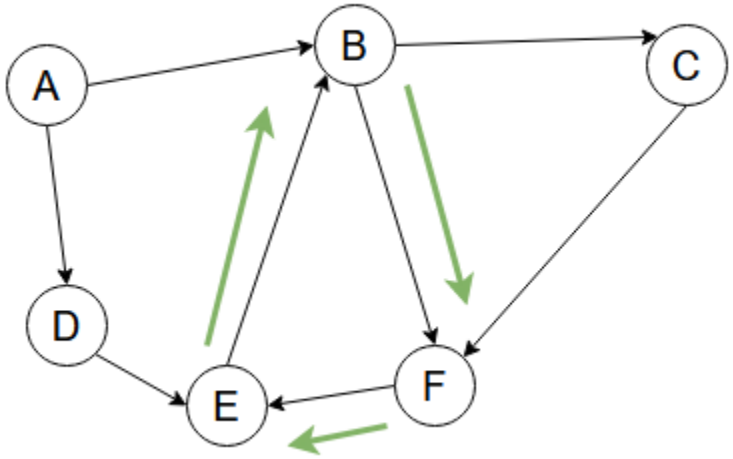
- Task 1. Example 2



- Correct answers:
  - F-E-A-B-D-C

Explanation of test 4

- Task 1. Example 3

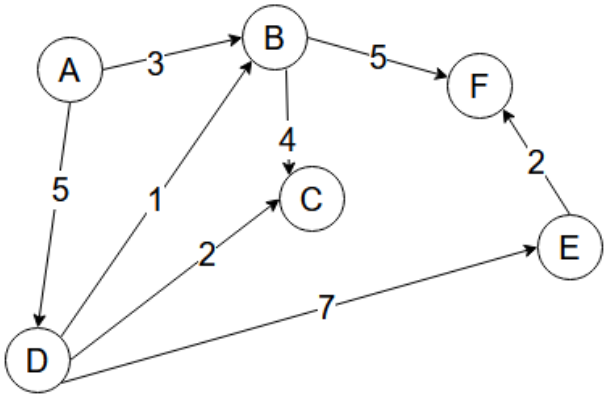


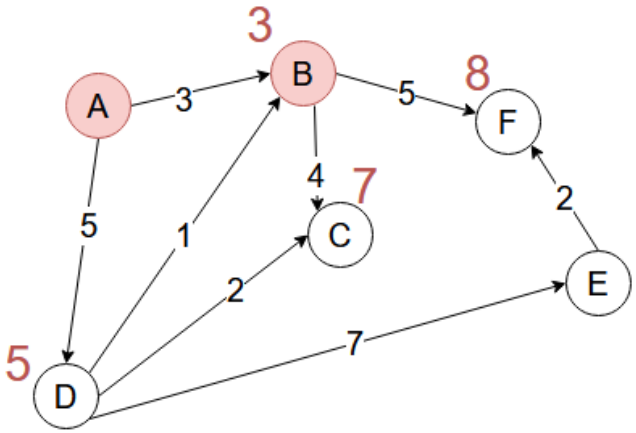
- Correct answers:
  - B-F-E is a cycle!
  - This vertices can’t be sorted

Explanation of test 4

- Task 2. Example 1
- Dijkstra's algorithm

A-0





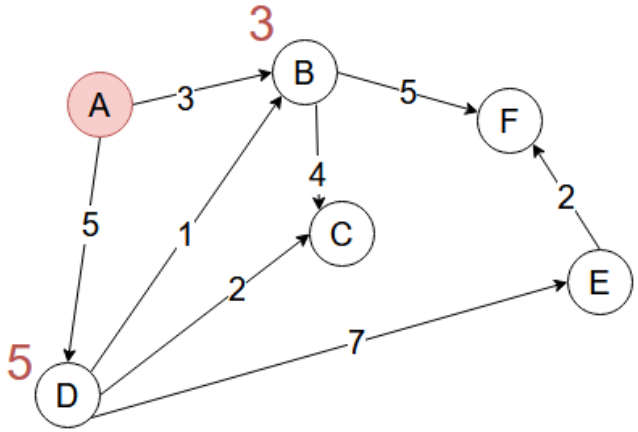
D-5

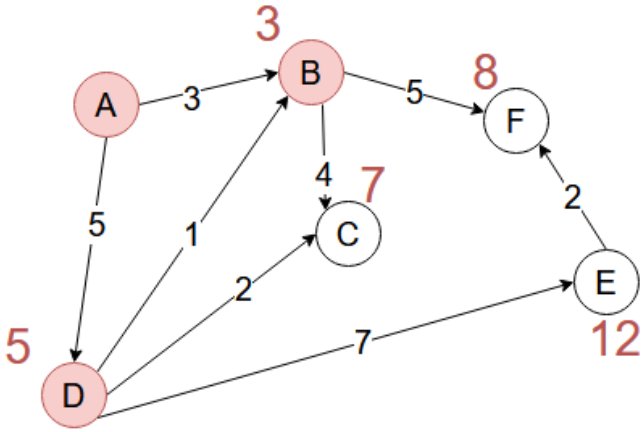
C-7

F-8

B-3

D-5





C-7

F-8

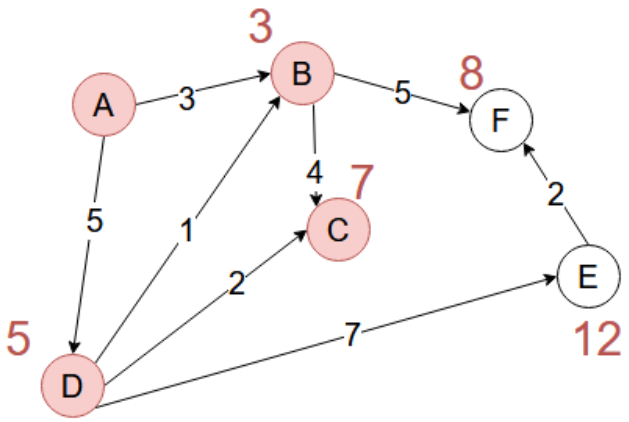
E-12

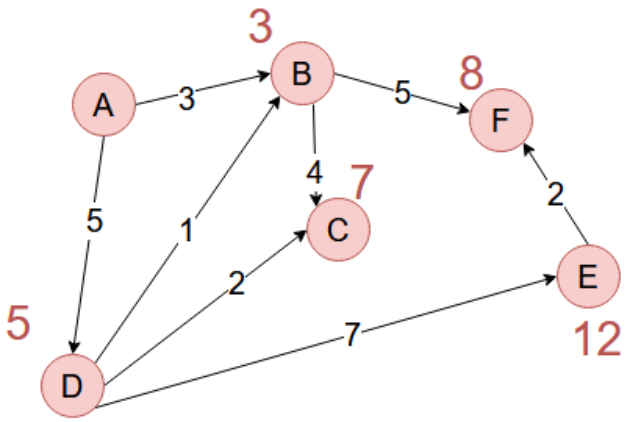
Explanation of test 4

- Task 2. Example 1
- Dijkstra's algorithm

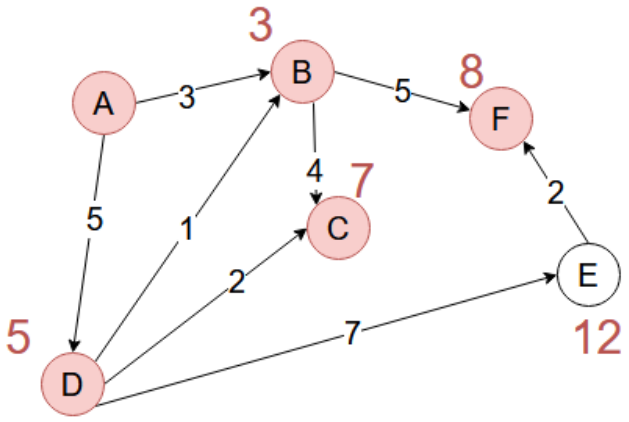
F-8

E-12





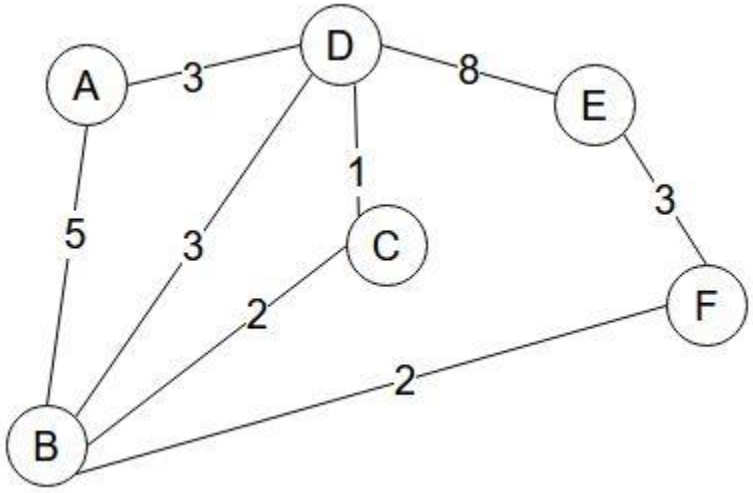
E-12



Order: A-B-D-C-F-E

Explanation of test 4

- Task 2. Example 2 and 3
- Dijkstra's algorithm

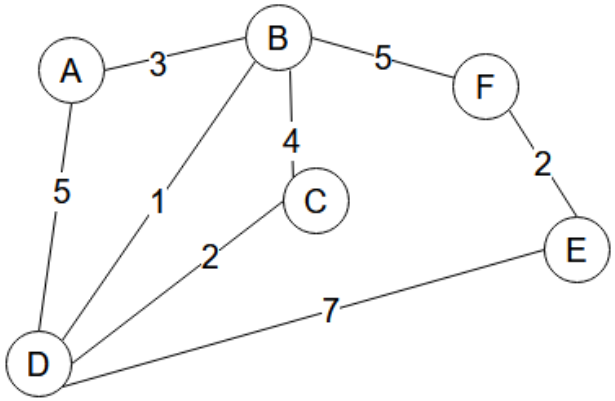


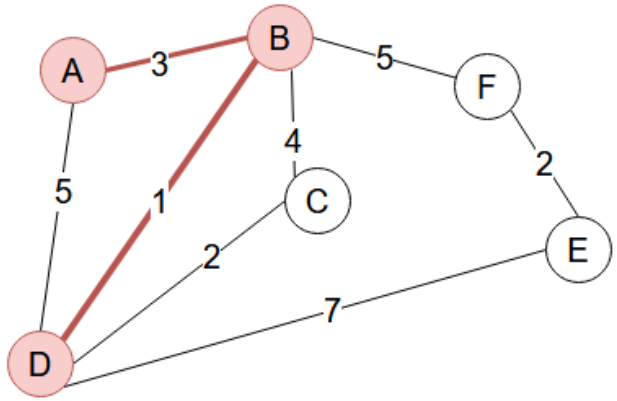
Start from A
Order: A-D-C-B-F-E

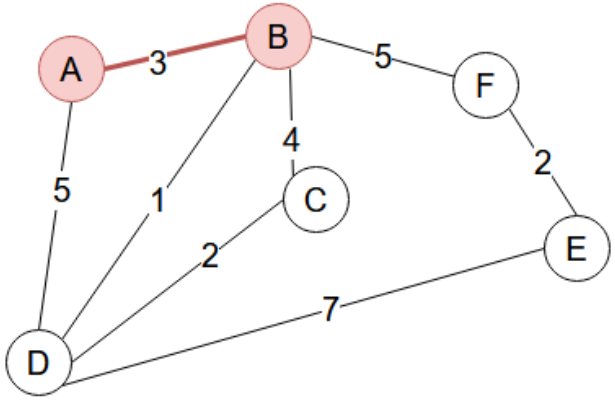
Start from F
Order: F-B-E-C-D-A

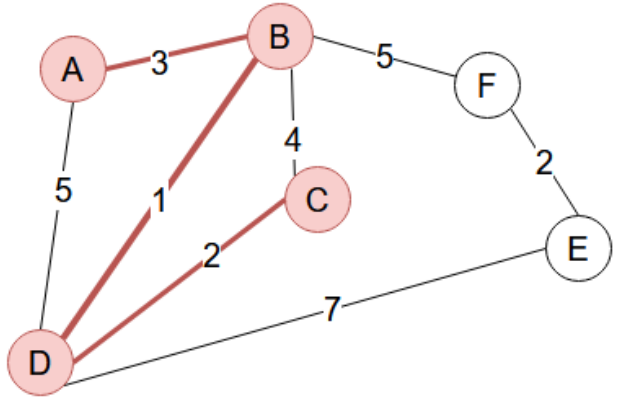
Explanation of test 4

- Task 3. Example 1
- Minimum Spanning Tree



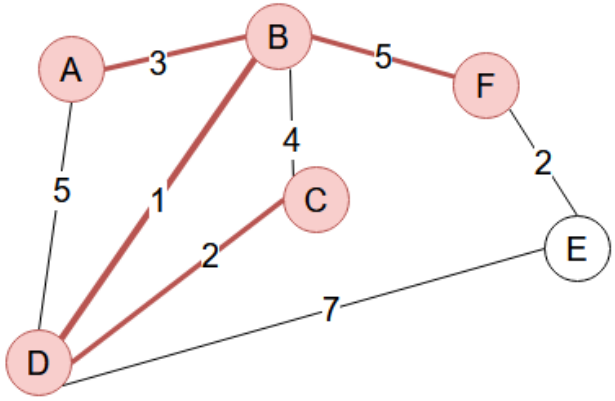




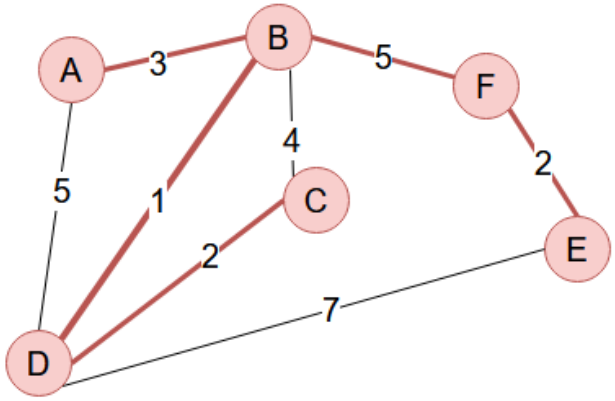


Explanation of test 4

- Task 3. Example 1
- Minimum Spanning Tree



Edges of MST:
AB; BD; CD; BF; EF

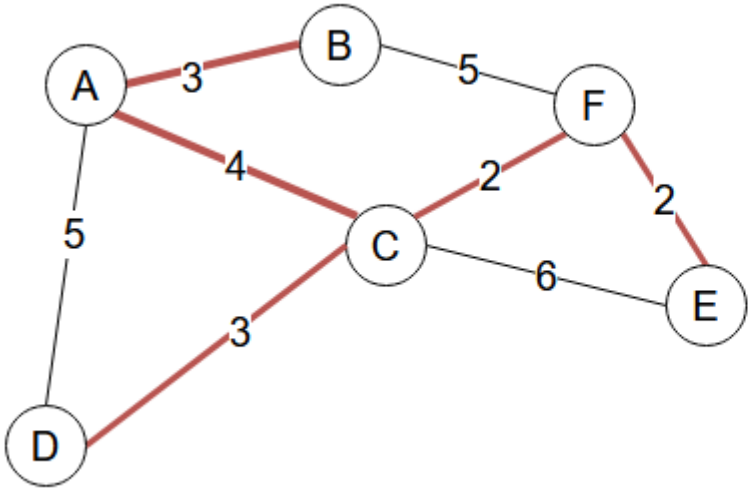


Don't forget, that MST – is a tree!

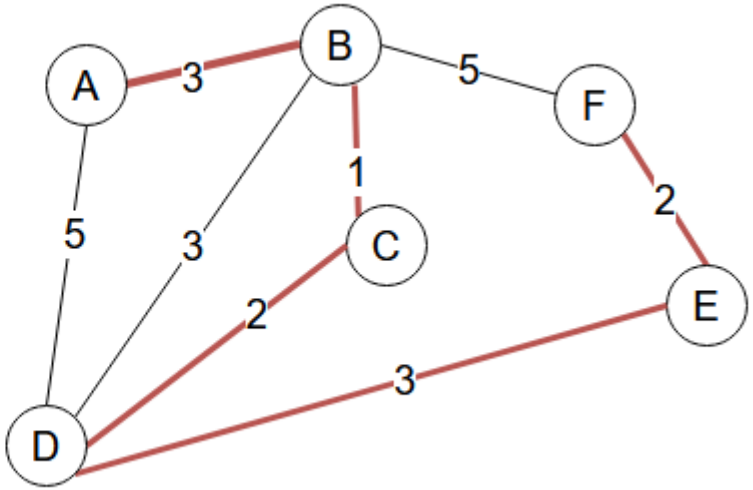
- MST covers all vertices of graph
- Sum of weights of edges is minimum
- No cycles

Explanation of test 4

- Task 3. Example 2 and 3
- Minimum Spanning Tree



Edges of MST:
AB; AC; CD; CF; EF



Edges of MST:
AB; BC; CD; DE; EF

Explanation of test 4

- Task 4
- Dijkstra vs Bellman-Ford
  - Different complexity: $O(m \log n)$ vs $O(mn)$
  - Bellman-Ford can work with negative weights
- Strongly connected vs connected
  - Strongly connected – for all pairs of vertices A and B in directed graph there is exist directed path from A to B and from B to A
  - Connected – all pairs of vertices in undirected graph are connected
- Topological sort
  - Linear ordering of all vertices of graph such that if graph has edge a->b, so a appears before b
  - Shortest path problem for DAG
  - Computing optimal order of execution in dependency graph

Thank you!