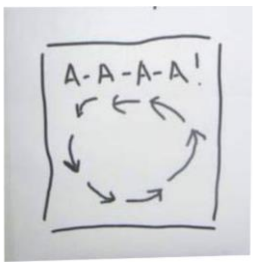
# Tutorial #15. Graph algorithms

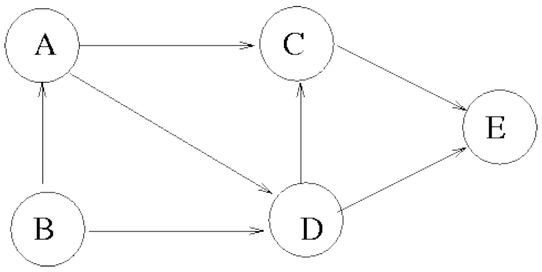
## Theoretical part

### Topological sort

Topological sort is important for scheduling work, usually of multiple agents, to achieve agreed and consistent timetable. **Result** of TS is a **list** where graph vertex order is preserved. The idea is that if you know that some action A is essential for another action B to execute (B depends on A directly or indirectly), than in a sorted list of actions action A will stand to the left from ("before") B. It is very easy to achieve this if your graph satisfies the only constraint: it is **acyclic (has no cycles)**.

Left graph cannot be sorted.





If the graph **acyclic**, than if you start going from any point by **any path**, you will once reach the "end of the world" (nowhere to go). That means, there's always **at least 1 vertex that has no successors**. Second idea about the acyclic graphs is if you remove one vertex from that graph, **it will remain acyclic**. These two ideas about acyclic graphs can converted to very easy algorithm for topological sort:

1) Find **one** lonely vertex (there can be few of them) at the end of the world. No one from the remaining graph depends on this vertex (so they all can seat to the left in the resulting sorted list). Attach this vertex to **the left of the list**.
2) Remove this vertex and all the adjacent edges from the graph.
3) Repeat 1-2 until graph is empty OR we find a cycle (no lonely edges).

### Shortest path: Dijkstra algorithm

Finds all shortest paths from starting vertex to all other in $O(v^2)$ time. Cannot work for negative weights. Algorithms at each iteration "visits" new vertex, calculates intermediate path length and refines path length for already visited. Stops when all vertices are visited. Starting with 0 for start node and $+\infty$ for other.

**Algorithm step**:

- Find **unvisited** vertex $V_i$ with minimal calculated path length.
- Consider all simple paths, where $V_i$ is the pre-last vertex – get all adjacent nodes to $V_j$ (let's call them $V_j$) and calculate new path length for each as $PL(V_j) = PL(V_i) + |(V_i, V_j)|$. If this values is smaller that already assigned to a node – replace with smaller. If you want to restore the path – replace also "origin value" for $V_i$.

## Practical part (for Students)

1) You should explain Euler's formula to your small brother
   $$e^{ix} = \cos x + i \sin x$$
   He knows nothing about the math that mean you should cover all these themes:
   - Natural numbers

- Integer numbers
- Rational numbers
- Real numbers
- Complex numbers
- Limits
- Exponentiation (bringing to a power)
- “e” constant
- Trigonometrical functions

Arrange these themes as a graph. Create a file with proposed graph. Create learning track for your brother using topological sort.

2) Read the graph from the file proposed at previous tutorial. Find the shortest path from «Вологда» to «Курск». Write distance and city sequence to a file.