## Topological Sort

*Description:*

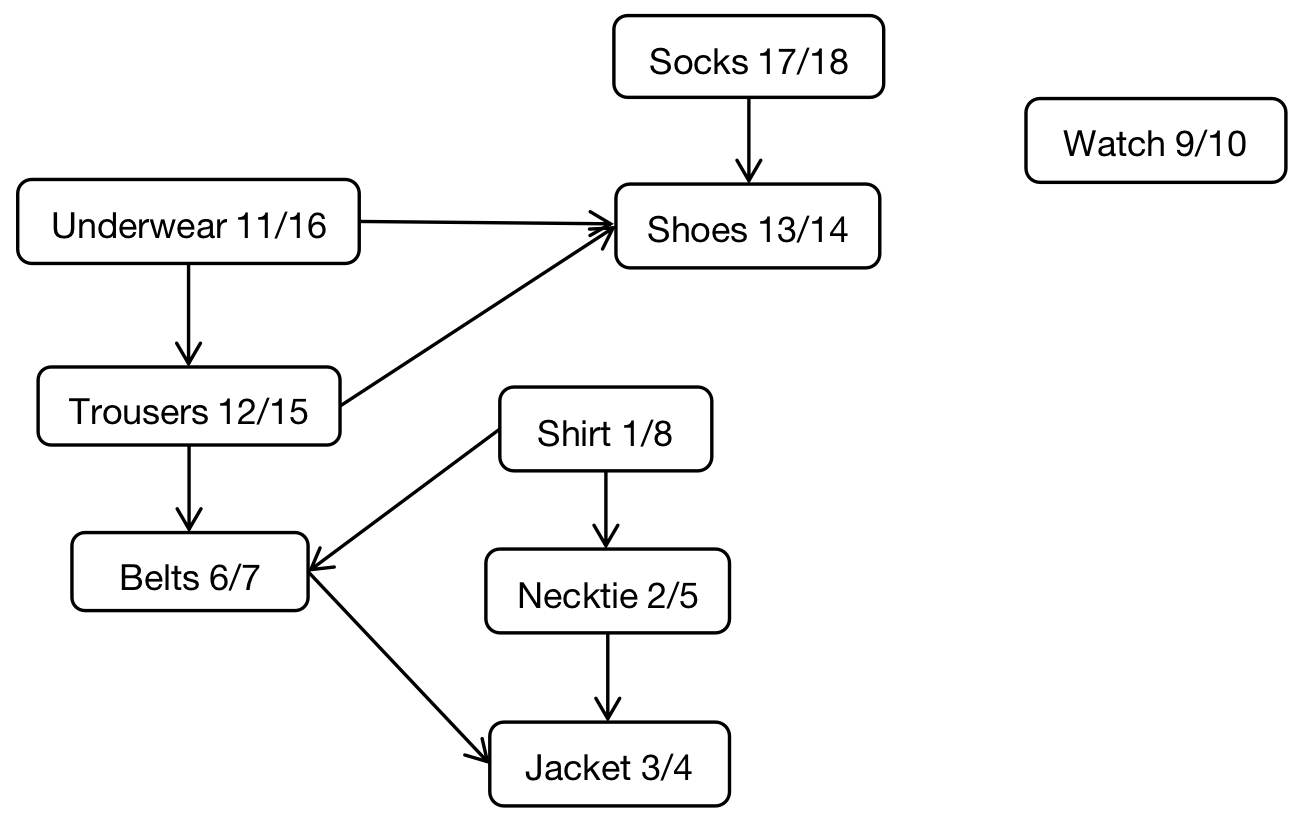
In this chapter, the topological sort algorithm is used to topological sort for the Directed Acyclic Graph. For one Directed Acyclic Graph G = (V, E), the Topological Sort Algorithm is used to sequentially sort of all nodes in Graph G, and this sequence satisfies the below condition: if there exists edge ( u, v ) in the Graph G, then the node u is before the node v. *( Attention that if there exist cyclic path in the Graph, then it is impossible to get a linear sequence. )*

However, the topological sort of graph is used to sort all nodes along one Horizontal Line, and all directed edges are all from left to right.

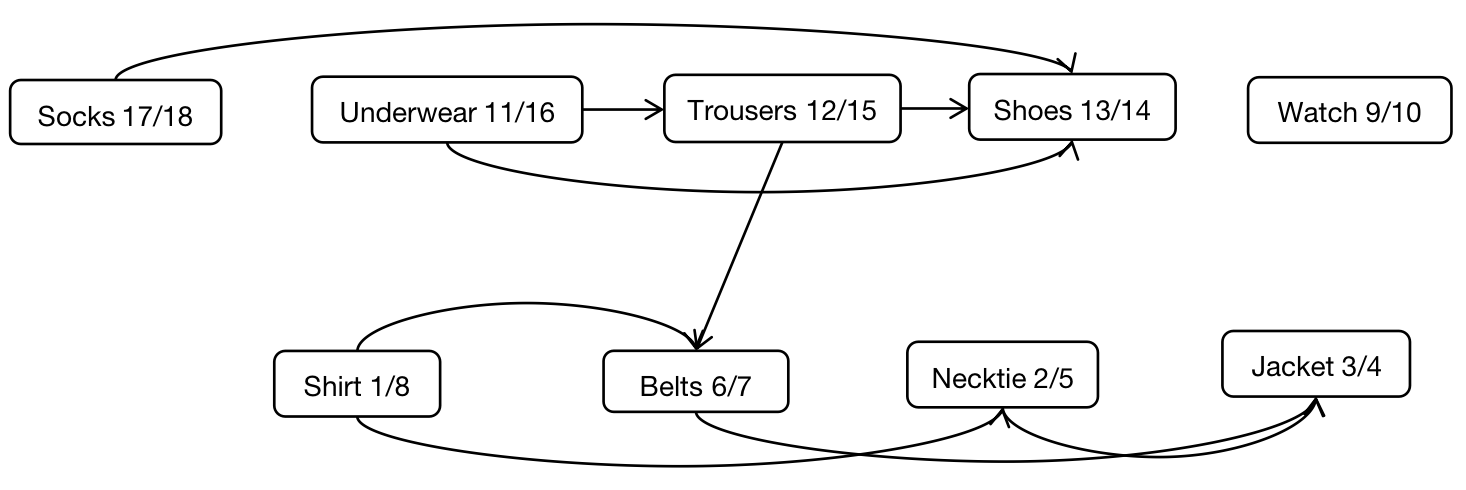
*Example:*

The image below describes all evidence events each morning the professor Bumstead needs to do.

The professor needs to wear some clothes first, and then wear some other clothes. Some clothes can be wore according to some sequence, for example, the socks needs to be wore before shoes. While some clothes can be wore at the random sequence, for example, the socks and trousers can be wore at the random sequence.



*In Directed Acyclic Graph, the Directed Edge (u, v) states that the clothes u must be wore before the clothes v. After the topological sorting of Directed Acyclic Graph, we can get the reasonable sequence of cloth wearing.*



*The Directed Acyclic Graph after proceeding the topological sort, can be displayed in the Horizontal Line. Of course, in the horizontal line, all Directed Edges are pointed from left to right.*

*Pseudo Code:*

Topological\_Sort ( Graph \* G ) {

*/\**

*\*\* The DFS Graph Algorithm is used to get all nodes final time when professor*

*\*\* finishes wearing all clothes.*

*\*/*

DFS ( G );

*/\**

*\*\* The vector vec is used to store all nodes from Graph.*

*\*/*

vector <Node \*> vec;

*/\**

*\*\* Push all nodes from G->V collection into vector vec, and sort all nodes according*

*\*\* to the second time when nodes are being visited, which means that all neighbor*

*\*\* nodes of the current node have been visited thoroughly, by descending order of*

*\*\* all end time.*

*\*/*

Push all nodes from G->V array into vec;

sort ( vec.begin(), vec.end(), func);

return vec;

}