is said to be **strongly connected** if for each pair of vertices u and v there exists a path from both u to v and v to u. A strongly connected digraph is also called a **strong** digraph. A strong digraph have a subset S called a **seperator** if $D - S$ is not strong, we also say that S **seperates** D. A seperator S is called **minimal seperator** of D if there exists no proper subset $X \subset S$ that seperates D. Now we can introduce a k**-strong** digraph D which is a strong digraph with $|V| \geq k$ and a minimal seperator S on $|S| = k$.

In a digraph $D(V, A)$ we mostly use the **dregree** as two different degrees namely **out degree**, $d^+(v)$, and **in degree**, $d^-(v)$, that is the arcs from v and to v respectively. In a digraph D we can talk about the over all **minimum out degree**, $\delta^+(D) = \min\{d^+(v)|v \in V\}$ and **minimum in degree**, $\delta^-(D) = \min\{d^-(v)|v \in V\}$ somtime we is going to need the minimum of these to $\delta(D) = \min\{\delta^+(D), \delta^-(D)\}$ and called the **minimum degree**. For every vertex v the vertices that is **adjacent** with v we denote $N^+(v)$ and $N^-(v)$ as the set of vertices that is dominated by (**out-nieghbours**)v and dominates (**in-nieghbours**)v, respectivly. This means that $d^+(v) = |N^+(v)|$ and $d^-(v) = |N^-(v)|$.
For simplicity when mentioning paths and cycles in digraphs it will be **directed** paths and cycles if not anything else is mensioned. By **directed** means that we go from tail to head on every arc on the path or cycle. When mentioning paths in a digraph it sometimes makes more sence specifing the head an tail of the path, so a path from s to t is denoted **(**s, t**)-path**. In some digraphs there is more than one path between the same two vertices these paths can use the same arcs or same vertices or be totally distinct from eachother, the maximum number of disjoint path between two vertices in a digraph is denoted $\lambda_D(s, t)$

1.2 Computational complexity

In this section we will go over how time is measured for an algorithm and what it means for a problem to be polynomially solveable or polynomially verifyable. Also what it means for a problem to be NP-hard and NP-complete and how we found out if a problem is either of them.

1.2.1 Measure time of algorithm (Polynomial, exponential)

The runing time of an algorithm is based on how many steps it is going thourgh which is somtimes based on the input that the algorithm takes we are going to denote an algorithms running time as a function $f(n)$ over the input n. This is how different functions can descirbe the running time of an algorithm, if an algorithm has the same number of steps no matter what the input is it has a constat running time where the constant is the number of steps the algorithm uses.

An algorithm can also take the form of an polynomial function or even exponentiel, if this is the case we uses some notation as big-O notation or θ. Big-oh is the most used one and is the notation we are going to use in this thises, if the algorithm takes $f(n) = 4n^3 + 2n^2 - n + 2$ time we denote it in big-oh notation as $O(n^3)$ as it is the biggest term of $f(n)$.

Since the shorter the runningtime is the better the algortihm is. Since the exponentiel runningtime algortihms take forever on large inputs, we would want to improve them, but sometimes you are left with problems where that is not a possibility.

So we are going to classify the problems in gruops of how long time it take to decide or verify the problems solution. a problem that is decided i polynomiel time is in the class called P. Which means for every given time of input in a problem from **P** we can find the solution for the problem in polynomiel time.

1.2.2 NP problems and classifications

As shortly described above there is something called a **polynomial verifier** for a problem. That means given a problem and then given a solution we can in polynomial time verify if it is a solution to the given problem. This is the class we call NP.

Definition 1.2.1. ***NP*** *is the class of languages that have polynomial time verifiers.*

Obiously if you can find a solution in polynomial time you can also verify whether a solution is correct in polynomial time. So $P \subseteq NP$. There is also a class called $NP - Hard$ but before we can explain that we need to explain what it means for a problem to be polynomial reduceable to another problem. For a specific problem A and another problem B then if there exists an algorithm that can take a solution from A and make it a solution for B in polynomial time. When such a algorithm exists it is called a polynomial verifier and we say that A is **polynomial reducable** to B or just that A is **reduced** to B. **NP-Hard** are the class of problems that every NP problem can be polynomiel reduced to. A problems in the class of NP-Hard problems does not nessesarily mean that it is NP it-self. If a problem is both **NP** and **NP-Hard** we call it **NP-Complete**. The problems we are mostly focusing on is in the class of **NP-Complete** problems.

1.3 Classes of Digraphs

We can classefy specific collection of graphs the reason for this is that digraphs of smaller collections of digraphs (like tournaments is a smaller collection of semicomplete digraphs) might be because of problems that is hard to solve on general digraph but is easy/polynomial solvable on specific types of digraphs.

A group of these problems is called NP-complete problems which sometimes sound easy solvable for graphs but only for some specific graphs we know how to solve it in polynomial time. Like finding paths in digraphs or cycles or more specific things, but in general des more we know about a digraph we can use to solve hard problems which in general would be time consuming like the problems that are NP-hard. By some quick fast algorithm you can checks wheter a digraph belongs to a certion **class** of digraphs. A class of digraph is a collection of digraph with certain properties incommen like **tournaments**.

1.3.1 introduction to some digraph classes

Tournaments is a digraph where the underlying graph is complete. So a complete graph of order 5 any orientation of the edges concludes in a tournament. Strong digraphs is also in it self a classification of digraphs. Classes of digraphs can be overlapping each other or