You have washed a full washing machine of black socks. You assume that no piece is lost and every sock has a pair. Now you should find a pair for each sock, since each pair is a little bit different. How would you describe the process of find a pair for each sock "algorithmically"?  How fast does the pairing go in the best case (and what is the best case like?)

for n in range(socks)

Pick a sock

If it is the pair of one of the previous unpaired socks,

match them together

remove them from matching table

else

put it in a new place on the matching table

stop

It is important to know how long an algorithm take to run and what would be the algorithm’s growth rate with changes in the input size. Does the algorithm become excitingly slow when we increase the size of the input or it remains as quick as before? In my opinion, the main goal of asymptotic notation is to answer to this kind of questions. Asymptotic notation determines the behavior of an algorithm in the sense of running time as a function of input size of the algorithm. It is a mathematical tool to measure the efficiency of an algorithm and specially the time complexity of an algorithm. In order to measure the growth rate of an algorithm using asymptotic notation, we discard the less important parts and also the constants while remaining the most significant terms. A good example has shown in the reference link when we have the running time of the algorithm as a function of input size (n). But what is difficult to understand for me is that how we can achieve this function and then drop the less significant part of it? Where does this time complexity function come from?