* Time & space complexity :-

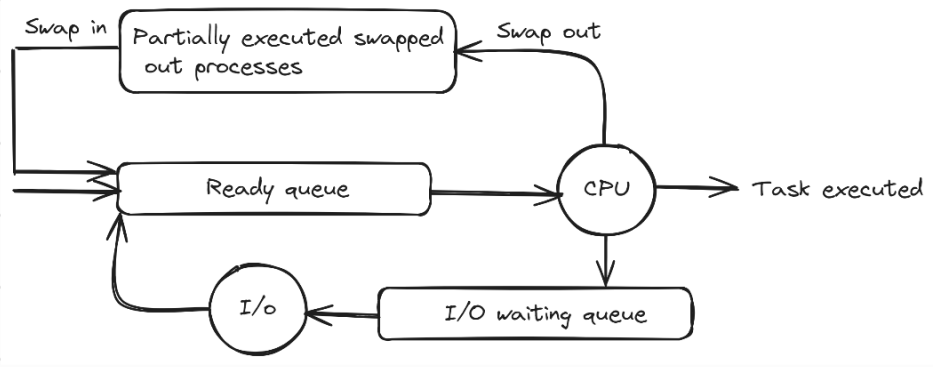
→ Efficiency of an algorithm is based on two things :-

i) Time taken

ii) Space / memory taken

→ The efficiency of an algorithm should be independent of the hardware that is being used.

* Multi-tasking in a single core CPU :-



→ In a single core CPU, practically there is no multi-tasking.

→ Consider a process P having 1000 instructions to be executed.

→ Initially, it will be waiting in the ready queue.

→ It will move to CPU for execution & CPU will only execute some instruction lets say 300 instructions out of all 1000 instructions.

→ It will swap out or remove process P from execution flow & move it to the area of partially executed swapped out processes.

→ Meanwhile there is another task X which is an I/O process.

→ Initially, it will wait in I/O waiting queue & then move to I/O.

→ When this process X is moved to ready queue again it only gets partially executed by CPU & is swapped out to partially executed swapped out processes area.

→ A process may get randomly picked from the waiting area & swap in back to ready queue.

→ CPU will again execute some set of instructions.

→ This loop will go on until the process is completely executed.

→ A single core CPU can work only in this way.

→ This is exactly like you eating, reading & playing a sport at the same instance of time.

→ Modern processors have multiple cores & threads for parallel execution of processes.

→ In 1s, approximately 10^8 instructions are executed.

* Experimental analysis :-

→ Here we determine how much time an algorithm takes based on the actual figure or numbers.

→ An algorithm can sometime get a priority to execute as the CPU is free & no other task are in queue.

→ In this scenario, it may execute in very less time, lets say 1 sec.

→ Same algorithm is executed when CPU is also executing other task & this time the algo took 3 secs.

→ This method of analysis is not efficient.

→ Measurement should be independent of hardware, factual values & background processes.

* Impact of input :-

→ The input type, size & the way input is provided can play an important role in measuring efficiency.

→ Since the computers can compute 10^8 instructions in 1 sec, for very small input, the instructions will be less, so it will take less time to compute.

→ Thus, we don't care about very small inputs as it will take less time for sure.

→ We consider input size to be very large for efficient measurement.

* Rate of growth :-

→ Rate at which the running time increases as a function of input is called Rate of growth.

→ If the time of an algo changes extremely fast with a small change in input then the rate of growth is high.

→ We judge the algorithms based on their rate of growth.

* Asymptotic analysis :-

→ It is based on two things :-

i) Rate of growth of algo w.r.t input size.

ii) Behavior of the rate at very large input value.

$$c < \log n < \sqrt{n} < n < n\log n < n\sqrt{n} < n^2 < n^3 < 2^n < n!$$

(Order of rate of growth)

↑ best — ↑ worst

→ Avoid the lower degree terms & constants in the time expression as they are comparatively negligible.

→ We only consider the term with the highest degree.