Lihtsate algoritmide keerukuste näited

Keerukus	**Näide**	**Kinnitamine aja mõõtmisega**

O(1) – konstantne keerukus Lihtsad operatsioonid, nagu +, -, *, / ja nende jadad omavad konstantset keerukust.	`phi = (1 + math.sqrt(5)) / 2`
	`f = phi**n - (1-phi)**n) / 5**0.5`
	`fibo = int(round((f))`
	`print(fibo)`
		Algoritmi tööaeg jääb samaks, sõltumata sisendi suurusest *n*.

O(n) – lineaarne keerukus. Tegevus, mis rakendub *n* elemendile. Tsükkel pikkusega *n*.	`for i in range(n):` `  print(i)`	Iga kord kui sisendi suurus kasvab *k* korda pikeneb algoritmi tööaeg *k* korda.

O(n^2) – ruutkeerukus. Tsükkel, mille sees on tsükkel ja mõlema tsükli sammude arv on *n*. Tegevus, mis rakendub loendi pikkusega *n* kõigile võimalikele paaridele.	`for i in range(n):`
	`  for j in range(n):`
	`  print(i, j)`
		Iga kord kui sisendi suurus kasvab *k* korda pikeneb algoritmi tööaeg k^2 korda.

O(n^3) – kuupkeerukus. Tsükkel, mille sees on tsükkel, mille sees on tsükkel, kõigi sammude arv on *n*. Tegevus, mis rakendub loendi pikkusega *n* kõigile võimalikele kolmikutele.	`for i in range(n):`
	`  for j in range(n):`
	`  for k in range(n):`
	`  print(i, j, k)`
		Iga kord kui sisendi suurus kasvab *k* korda pikeneb algoritmi tööaeg k^3 korda.

O(n^k) – polünomiaalne keerukus.	`...`	...

O(log n) – logaritmiline keerukus. Igal tsükli või rekursiooni sammul väheneb läbivaadatavate andmete suurus kaks, kolm või k korda.	`while n > 1:`
	`  n = n / 2`
	`if n == 1:`
	`  print("n oli kahe aste")`
		Iga kord kui sisendi suurus kasvab *k* korda pikeneb algoritmi tööaeg *m* võrra.

O(e^n) – eksponentsiaalne keerukus. Näiteks *n* elemendilise hulga kõikvõimalike alamhulkade läbivaatamine on keerukusega 2^n (alamhulkade arv). Rekursioonipuu.	`def alamhulgad(loend, alamhulk=[]):`
	`  if len(loend)==0:`
	`  print(alamhulk)`
	`  else:`
	`  alamhulgad(loend[1:],`
	`  alamhulk)`
	`  lisa = alamhulk + [loend[0]]`
	`  alamhulgad(loend[1:], lisa)`
		Iga kord kui sisendi suurus kasvab *k* võrra pikeneb algoritmi tööaeg *m* korda.

O(n!) - Faktoriaalne keerukus. Näiteks *n* elemendilise hulga kõikvõimalike permu-tatsioonide läbivaatamine	`def stringi_perm(s):`
	`  if len(s) <= 1:`
	`  return [s]`
	`  tulem = []`
	`  for i in range(len(s)):`
	`  s2 = s[:i] + s[i+1:]`
	`  for perm in stringi_perm(s2):`
	`  tulem.append(s[i]+perm)`
	`  return tulem`	