1. Algoritem:

Naredimo števec, ki ga na začetku postavimo na minus neskončno, zato da bo manjši od vseh elementov in ga lahko povečujemo. V vrsto postavimo stražarja, da se ne za ciklamo. Naredimo še eno spremenljivko veriga, ki naj bo na začetku None.

Dokler ne pridemo do stražarja:

Vsak element vrste posebej preberemo, ga shranimo v spremenljivko veriga in ga odstranimo. Poračunamo vsoto, če je vsota večja od števca potem, števcu priredimo vrednost te vsote in verigo shranjeno v veriga prestavimo v vrsto. To naredimo zato, da res ne bomo imeli padajočih vsot in da se naše ne padajoče zaporedje nahaja za stražarjem. Če pa je vsota manjša od števca samo nadaljujemo, saj je člen manjši po vsoti od prejšnjega in ga ne potrebujemo.

Ko pridemo do stražarja, stražarja odstranimo in končamo.

1. Časovna zahtevnost:

Če predpostavimo, da je ena operacija izračun vsote verige. Torej za vsak element vrste naredimo en izračun sledi, da je časovna zahtevnost O(n), kjer je n število elementov v vrsti.

1. Vsota verige:

Algoritem:

Naredimo spremenljivko vsota, ki bo na začetku 0. Naredimo kazalec trenutni, ki na začetku kaže na verigo(začetek verige). Sedaj se sprehajamo po verigi, dokler kazalec na pokaže na None.

V vsakem koraku preberemo vrednost vozla in jo prištejemo vsoti(vsota+=trenutni.vrednost). Trenutni pa spremenimo, da kaže na naslednji vozel(trenutni=trenutni.naslednji). Vrnemo vsoto.

Časovna zahtevnost: Za vsak element imamo eno branje vrednosti, povečanje vsote in spreminjanje kazalca. Če ima naša veriga m členov ja časovna zahtevnost 3m oziroma O(m).

1. Skupna časovna zahtevnost torej če bi za prvotni algoritem vzeli operacije od vsota verige, bi bila O(k), kjer je k število vozlišč v vseh verigah v vrsti.