**Naloga:**

Dana je tabela, ki vsebuje premi pregled dvojiškega drevesa.

* a) Sestavi algoritem, ki ugotovi, če je to drevo lahko iskalno. Npr. za tabele [8, 4, 2, 1, 6, 5, 10, 9, 30, 11], [1, 2, 3], [8, 10, 9, 30, 11], [3, 2, 1] in [3, 1, 2] vrne True, za tabele [8, 4, 10, 2], [8, 4, 2, 1, 6, 5, 10, 7, 30, 11] in [1, 2, 4, 1] pa False. Algoritem lahko zapišeš v programskem jeziku Python, s psevdokodo ali kot besedilni postopek. Predpostavi, da so v tabeli gotovo sama različna cela števila.
* b) Oceni časovno zahtevnost svojega algoritma. Odgovor utemelji.

**Algoritem:**

Če imamo dani premi pregled dvojiškega drevesa, vemo, da je prvi element seznama koren. Potem pa ne vemo, do kod sega levo poddrevo in do kod desno. Opazimo pa lahko, da morajo biti od korena do desnega poddrevesa vsi elementi manjši od korena, ker je to levo poddrevo. Torej, ko naletimo na prvo število, ki je večje od korena, morajo biti vsa naslednja prav tako večja od njega. V primeru, da so od večjega števila naprej tudi manjša, pomeni, da to ne more biti dvojiško iskalno drevo.

A computer screen shot of a code

Description automatically generated

**Časovna zahtevnost:**

Naj bo n število elementov danega seznama in naj bo primerjanje karakteristična operacija.

Vsakič naredimo 2\*(n-2) primerjav, kar nam da časovno zahtevnost **O(n)**. Podrobnejša razlaga od kod ta formula:

Vzemimo primer seznama [8,4,2,1,6,5]. Zapišimo katero število s kom primerjamo pri posameznem indeksu:

i = 1: 4 in 2 primerjamo s korenom

i = 2: 2 in 1 primerjamo s korenom

i = 3: 1 in 6 primerjamo s korenom

i = 4: 6 in 5 primerjamo s korenom

Sedaj lahko vidimo, da

* število 4 in 5 primerjamo enkrat,
* števila 2, 1 in 6 pa dvakrat.

Za ta primer imamo potem skupno število primerjav 1 + 2 + 2 + 2 + 1 = 2\*(6 – 2) = 8. Potem 6 zamenjamo z n, saj je 6 dolžina tega seznama iz primera.

Ko uporabimo funkcijo O, dobimo, da je **časovna zahtevnost tako enaka O(n).**