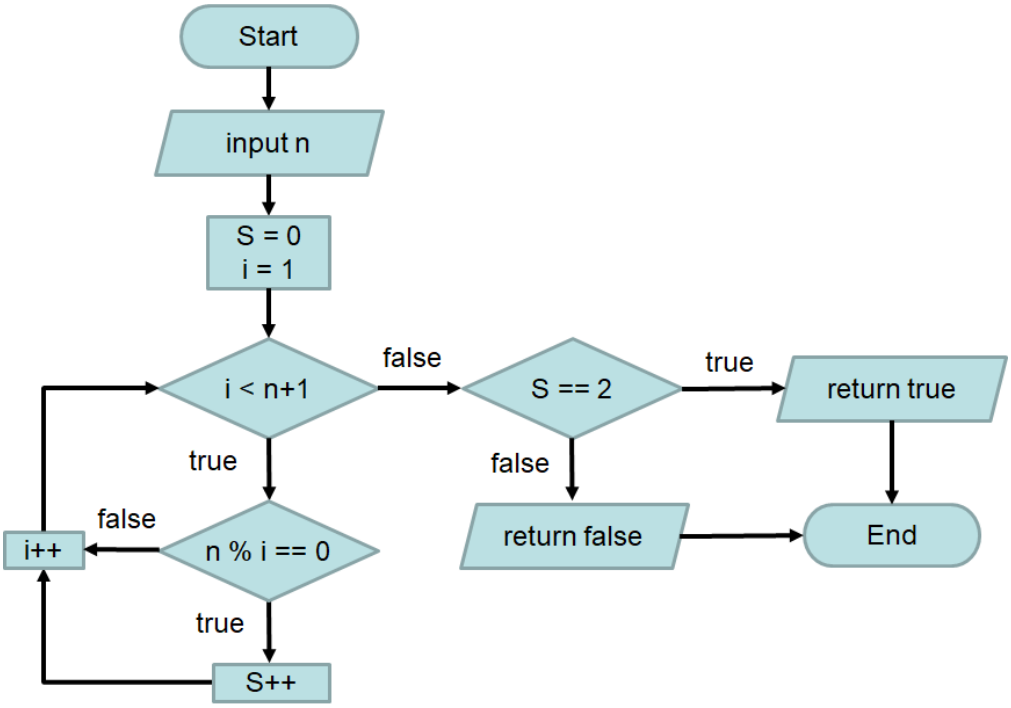
Dalyko „Algoritmai ir duomenų struktūros“ egzamino užduočių sprendimai

1. Paprasčiausio algoritmo, patikrinančio, ar natūralusis skaičius n yra pirminis, pseudokodas:

```
1. isprime(n)
2.     S = 0
3.     for i = 1 to n do
4.         if n % i == 0 then
5.             S++
6.     if S == 2 then
7.         return(true)
8.     else
9.         return(false)
```

   To paties algoritmo blokinė schema:



2. Pagal sąlygą parinkus mažiausią galimą šešioliktainį skaičių teisinga lygybė

$$(A900)_{16} = (\mathbf{43}264)_{10},$$

   parinkus didžiausią galimą šešioliktainį skaičių teisinga tokia lygybė

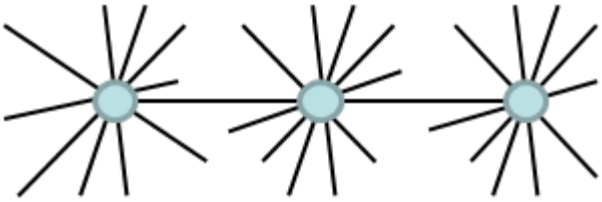
$$(A9FF)_{16} = (\mathbf{43}519)_{10}.$$

   Išvada – pirmieji du dešimtainio skaičiaus skaitmenys yra 4 ir 3. Belieka iš kito galo apskaičiuoti nežinomus šešioliktainio skaičiaus skaitmenis:

$$(\mathbf{43}452)_{10} = (A9\mathbf{BC})_{16}.$$

3. Užtenka pastebėti, kad Priuferio kode esantys skaičiai yra vidinių medžio viršūnių numeriai (prisiminkime, kad sudarydami Priuferio kodą niekada į jį neįterpsime medžio lapo numerio). Išvada – vidinės medžio viršūnės tik trys, kurias jungia 2 briaunų takas:

Prie šių vidinių viršūnių pridėję trūkstamus medžio lapus, niekada tokiame medyje nerasime ilgesnio nei **4 briaunų** tako:



4. a) Faktą, kad *Selection Sort* algoritmas nėra stabilus užtenka pagrįsti pavyzdžiu, išrikiuojant tokią skaičių seką:

[3 (id #1), 3 (id #2), 1 (id #1)].

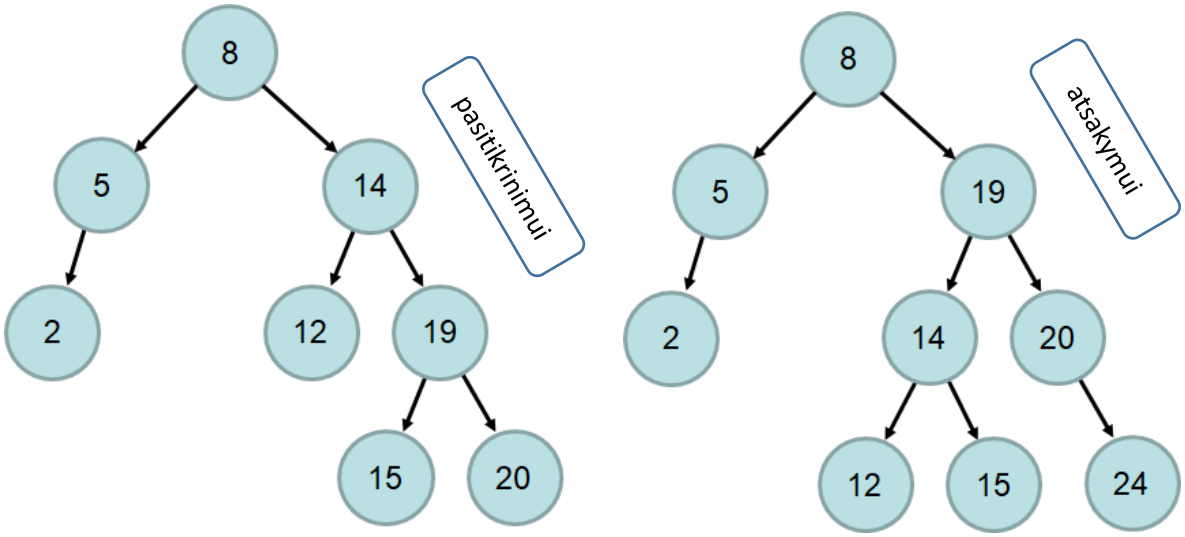
Po rikiavimo išrinkimu bus sukeisti 1 ir 3 elementai vietomis:

[1 (id #1), 3 (id #2), 3 (id #1)].

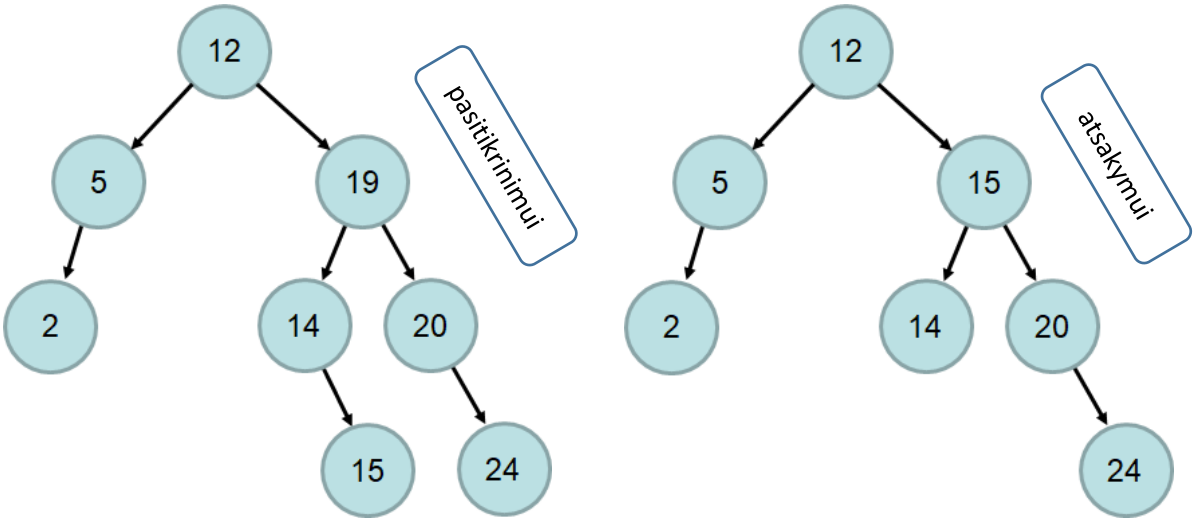
Šiuo pavyzdžiu parodėme, kad realizavus *Selection Sort* algoritmą vienodi elementai buvo sukeisti vietomis tarpusavio atžvilgiu, todėl šis rikiavimo algoritmas nėra stabilus.

b) *Bubble Sort* algoritmo eigoje vietomis galima sukeisti tik gretimus rikiuojamo sąrašo elementus, kurie nėra tarpusavyje lygūs. Jei neišrikiuotame sąraše bus keli vienodi elementai, burbuliuko rikiavimo algoritmo eigoje jokie 2 vienodi elementai nebus sukeisti vietomis. Išvada – po rikiavimo bus išlaikomas bet kurių 2 vienodų elementų tarpusavio eiliškumas, vadinasi, bus išlaikomas ir visų vienodų elementų tarpusavio eiliškumas.

5. AVL medis po reikšmių 2, 15, 20 įterpimo ir po reikšmės 24 įterpimo:



AVL medis po reikšmės 8 pašalinimo ir po reikšmės 19 pašalinimo:



6. Užtenka pastebėti, kad 100 × 100 gretimumo matricos galimos reikšmės 0 ir 1 išsidėsčiusios „šachmatų lentos“ principu:

0 1 0 1 0 1 ... 0 1
1 0 1 0 1 0 ... 1 0
0 1 0 1 0 1 ... 0 1
1 0 1 0 1 0 ... 1 0
.....
0 1 0 1 0 1 ... 0 1
1 0 1 0 1 0 ... 1 0

Kiekvienoje eilutėje yra lyginis elementų skaičius, o gretimi elementai yra skirtingi, todėl kiekvienos eilutės elementų suma lygi 50. Eilučių yra iš viso 100, todėl visos matricos elementų suma lygi 5000. Kadangi šios simetrinės matricos pagrindinėje įstrižainėje – tik nuliai, todėl grafas yra bekilpis ir jo briaunų skaičius lygus pusei matricos elementų sumos, t. y. 2500.

7. Apdoroti *scan3D.xyz* failą pašalinant pasikartojančius taškus galima daugeliu algoritmų, kurių sudėtingumas geresnis nei kvadratinis.

I būdas.

Algoritmo procedūra	**Sudėtingumas**
Nuskaitant *scan3D.xyz* failą kiekviena jo eilutė paverčiama *string* kintamuoju ir įterpiama į masyvą: ["-4.5 2.54 0.01", "-3.64 2.62 -0.01", ...].	O(n)
Masyvo elementai išrikiuojami naudojant skaitmeninio rikiavimo algoritmą. (Užduoties pavyzdyje kiekvienos eilutės ilgis neviršija 16 simbolių, kurių iš viso yra 13 rūšių: "0123456789.- ").	$O(n \cdot k) = O(n \cdot 13 \cdot 16) = O(208n)$
Realizavus skaitmeninį rikiavimą pasikartojantys masyvo elementai bus gretimi. Nuosekliai lyginant kiekvieną išrikiuoto masyvo elementą su ankstesniu, atgal į *scan3D.xyz* failą skirtingose eilutėse reikia įrašyti tik tokias *string* reikšmes, kurios nesikartoja.	O(n)

Šio algoritmo sudėtingumas lygus $O(n) + O(208n) + O(n) = O(n)$ ir yra daug geresnis negu kvadratinis $O(n^2)$.

II būdas.

Algoritmo procedūra	**Sudėtingumas**
Nuskaitant *scan3D.xyz* failą kiekvienos eilutės 3 reikšmės paverčiamos *float* kintamuoju ir įterpiamos į masyvą (kiekvienas masyvo elementas susideda iš x, y ir z koordinačių): [[-4.5, 2.54, 0.01], [-3.64, 2.62, -0.01], ...].	$3 \cdot O(n)$
Masyvo elementai išrikiuojami 3 kartus (pagal x, pagal y ir pagal z koordinatę) naudojant *stabilų* rikiavimo algoritmą, kurio sudėtingumas O(n log n).	$3 \cdot O(n \log n)$
Po *stabilaus* rikiavimo pasikartojantys masyvo elementai bus gretimi. Nuosekliai lyginant kiekvieną išrikiuoto masyvo elementą su ankstesniu, atgal į *scan3D.xyz* failą skirtingose eilutėse reikia įrašyti tik nepasikartojančių masyvo elementų x, y ir z reikšmes, atskirtas tarpu.	$3 \cdot O(n)$

Šio algoritmo sudėtingumas lygus $3 \cdot O(n) + 3 \cdot O(n \log n) + 3 \cdot O(n) = O(n \log n)$ ir yra geresnis negu kvadratinis $O(n^2)$.