Vilniaus
universitetas

Informatikos ir informatinio mąstymo mokomoji veikla

Aklavietė

Kuriame Lietuvos ateitį
2014–2020 metų Europos Sąjungos fondų investicijų veiksmų programa

Vilniaus universitetas

Aklavietė

Informatikos ir informatinio mąstymo mokomosios veiklos sukurtos įgyvendinant projektą „Aukštųjų mokyklų tinklo optimizavimas ir studijų kokybės gerinimas Šiaulių universitetą prijungiant prie Vilniaus universiteto“ (Nr. 09.3.1-ESFA-V-738-03-0001), finansuojamą iš Europos socialinio fondo lėšų pagal 2014–2020 metų Europos Sąjungos fondų investicijų veiksmų programos 9 prioriteto „Visuomenės švietimas ir žmogiškųjų išteklių potencialo didinimas“ įgyvendinimo priemonę Nr. 09.3.1-ESFA-V-738 „Aukštųjų mokyklų tinklo tobulinimas“.

Šios veiklos autoriai: Alvida Lozdienė, Viktoras Dagys ir prof. dr. Valentina Dagienė

Redagavo: Viktoras Dagys

Iliustravo: Vaidotas Kinčius

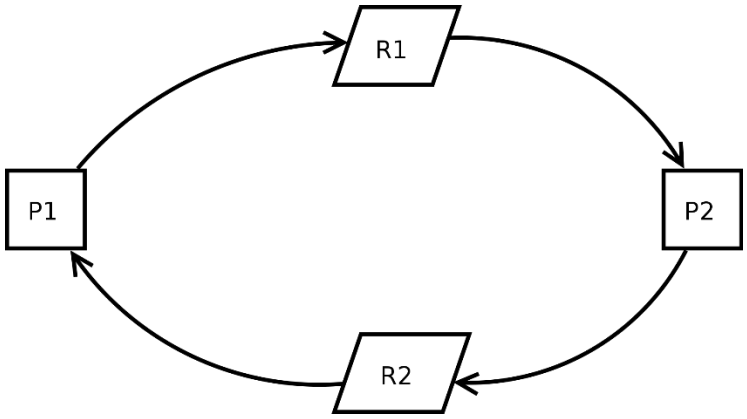
Aklavietė

Aklavietė (anglų k. *deadlock*) – tai būsena, kai kiekvienas sąveikaujančios procesų aibės procesas laukia įvykio, kurį gali pateikti tik tos aibės procesas. Kadangi visi procesai yra laukimo būsenoje, tai nė vienas iš jų tokio įvykio negali pateikti – visi stovi ir laukia.

Lygiagrečiųjų skaičiavimų srityje arba duomenų bazių valdymo sistemoje **aklavietė** yra tokia situacija, kai keli procesai (tam tikros subjektų grupės nariai) laukia resursų, kurie yra užimti vienas kito, ir nė vienas iš jų negali tęsti vykdymo.

Aklavietė yra dažna daugiaprocesorinių sistemų, lygiagrečiųjų skaičiavimų ir paskirstytųjų sistemų problema, nes šiose sistemose dažnai naudojami programiniai arba aparatūriniai užraktai bendriems ištekliams paskirstyti ir procesų sinchronizavimui įgyvendinti.

Jei procesas neribotą laiką negali pakeisti savo būsenos, nes jo prašomus resursus naudoja kitas procesas, kuris pats laukia, tai reiškia, kad sistema yra aklavietėje.



1 pav. Dviejų procesų (P1 ir P2) ir abiejų šių procesų naudojamų resursų (R1 ir R2) aklavietės pavyzdys
https://en.wikipedia.org/wiki/Deadlock#/media/File:Process_deadlock.svg

Abiem procesams reikia išteklių, kad jie būtų toliau vykdomi. Procesui P1 reikia papildomo ištekliaus R1 ir jis naudoja išteklių R2, o procesui P2 reikia papildomo ištekliaus R2 ir jis naudoja R1. Todėl nė vienas procesas negali toliau vykdyti užduoties.

Aklavietės pavyzdys

Žingsnis	*1-as procesas*	*2-as procesas*
0	Nori perimti resursus A ir B, pradeda nuo A	Nori perimti resursus A ir B, pradeda nuo B
1	Perima A resursą	Perima B resursą
2	Laukia atsilaisvinančio B resurso	Laukia atsilaisvinančio A resurso
3	Aklavietė	

Kaip ir kitų sinchronizavimo klaidų, aklavietės klaidų šalinimą apsunkina tai, kad keli procesai yra vykdomi vienu metu (pavyzdžiui, jei 1-am procesui būtų pavykę perimti išteklių B anksčiau nei 2-am procesui, klaida neįvyktų).

Yra algoritmų aklavietei pašalinti. Vykdant aklavietės pašalinimo algoritmus gali atsirasti dinaminė aklavietė – aklavietė formuojama, panaikinama, vėl formuojama, vėl panaikinama ir t. t.

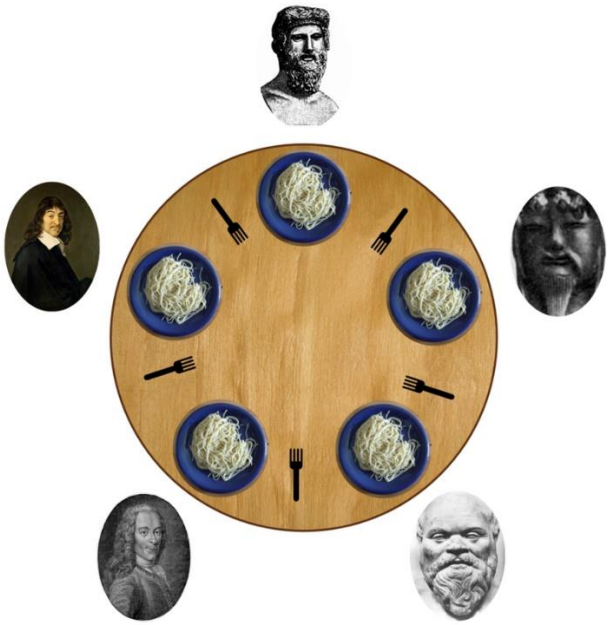
Projektuojant sistemą reikia pasirūpinti aklaviečių pašalinimu. Tai vienintelis daugiau ar mažiau patikimas būdas susidoroti su aklavietėmis. Kai pagrindinė užduoties koncepcija neleidžia išvengti aklaviečių, kraštutiniu atveju reikėtų bent jau suprojektuoti visas resursų užklausas taip, kad tokias aklavietes būtų galima neskausmingai pašalinti.

Vakarieniaujančių filosofų uždavinys

Informatikoje vakarieniaujančių filosofų uždavinys demonstruoja, kaip lygiagrečiųjų algoritmų projektavimo procese parodyti sinchronizavimo problemas ir numatyti aklavietės pašalinimo būdus.

1965 m. filosofų uždavinį Edsgeris Dijkstra (*Edsger W. Dijkstra*) suformulavo studentams kaip egzamino užduotį. Netrukus po to Tony Hoare‘as (*Charles Antony Richard Hoare*) suteikė šiam uždaviniui dabartinę formą.

Penki nebylūs filosofai sėdi prie apvalaus stalo, priešais kiekvieną filosofą – po lėkštę spagečių. Šakutės po vieną padėtos ant stalo tarp kiekvienos greta sėdinčių filosofų poros.



2 pav. Vakarieniaujančių filosofų uždavinio iliustracija
https://en.wikipedia.org/wiki/Dining_philosophers_problem#/media/File:An_illustration_of_the_dining_philosophers_problem.png

Kiekvienas filosofas gali arba valgyti, arba mąstyti. Valgymo neriboja likusių spagečių kiekis – jų pasiūla yra begalinė. Tačiau filosofas gali valgyti tik laikydamas dvi šakutes: vieną dešinėje rankoje, o kitą kairėje. (galima alternatyvi formuluotė, kai spagečių lėkštės pakeičiamos dubenėliais su ryžiais, o šakutės – lazdelėmis).

Kiekvienas filosofas gali paimti tik artimiausią šakutę (jei ji yra) arba ją padėti, jei ją jau turi rankoje. Kiekvienos šakutės paėmimas ir grąžinimas ant stalo yra atskiri veiksmai, kuriuos reikia atlikti vieną po kito.

Užduotis: sukurti elgesio modelį (lygiagretų algoritmą), pagal kurį nė vienas filosofas nebadautų, t. y. amžinai kaitaliotų valgymą ir mąstymą.

Pavyzdžiui, kiekvienam filosofui galima patarti atlikti tokį algoritmą:

- Mąstykite, kol kairioji šakutė bus laisva. Kai šakutė bus laisva, paimkite ją.
- Mąstykite, kol dešinioji šakutė bus laisva. Kai šakutė bus laisva, paimkite ją.
- Valgykite.
- Padėkite kairę šakutę.
- Padėkite dešinę šakutę.
- Pakartokite algoritmą nuo pradžių.

Toks sprendimas yra neteisingas: jis leidžia sistemai pasiekti aklavietę, kai kiekvienas filosofas yra paėmęs jo kairėje esančią šakutę ir laukia, kol bus ant stalo šakutė, esanti jo dešinėje.

Resursų stygiaus (anglų k. *resource starvation*) problema gali kilti, jei vienas iš filosofų negali perimti kairiosios ir dešiniosios šakutės dėl laiko sinchronizavimo. Pavyzdžiui, būtų galima pasiūlyti taisyklę, kad filosofai, palaukę penkias minutes, kol atsiras kita šakutė, turėtų padėti šakutę atgal ant stalo ir palaukti dar penkias minutes ir po to vėl bandyti paimti šakutes. Ši schema pašalina aklavietės galimybę (nes sistema visada gali pereiti į kitą būseną), tačiau vis dar išlieka galimybė, kad sistema užsiciklins (anglų k. *livelock*). Sistema keis savo būseną, bet neatliks jokio naudingo darbo. Pavyzdžiui, jei visi penki filosofai vienu metu pasirodys valgomajame ir kiekvienas iš jų tuo pačiu metu paims kairiąją šakutę, tada lauks penkias minutes, tikėdamiesi paimti dešiniąją šakutę, tada padės kairiąją šakutę ir lauks dar penkias minutes, kol vėl bandys paimti šakutes.

Abipusis pašalinimas (anglų k. *mutual exclusion*) yra pagrindinė filosofų vakarienės uždavinio idėja. Šiame uždavinyje pateikiamas bendras abstraktus scenarijus, kuriuo galima paaiškinti tokio tipo problemas. Filosofų klaidos vaizdžiai parodo, kokių sunkumų kyla realiame programavime, kai kelioms programoms reikia išskirtinės prieigos prie bendrų resursų.

Pirminis Dijkstros tikslas formuluojant filosofų uždavinį buvo parodyti problemas, susijusias su išoriniais kompiuterių įrenginiais. Tačiau šios problemos taikymo sritis yra daug platesnė. Pavyzdžiui dažnai kyla sunkumai, kai keli procesai bando gauti prieigą prie atnaujinamų duomenų.

Sistemose, kuriose vienu metu vyksta daug lygiagrečių procesų (pavyzdžiui, operacinių sistemų branduoliuose), naudojami tūkstančiai aklaviečių ir sinchronizavimo taškų. Norint išvengti aklaviečių, resursų stygiaus ir duomenų iškraipymo, tenka griežtai laikytis taisyklių.

Filosofų uždavinį galima išspręsti pakviečiant prie stalo padavėją. Filosofai, prieš paimdami šakutę, turi gauti padavėjo leidimą. Padavėjas žino, kiek šakučių yra naudojama, todėl jis gali priimti sprendimus dėl šakučių paskirstymo ir taip užkirsti aklavietės susidarymą. Jei keturios šakutės iš penkių jau naudojamos, kitas filosofas, prašantis šakutės, turės laukti padavėjo leidimo, kuris bus gautas tik tada, kai dar viena šakutė atlaisvins. Daroma prielaida, kad

filosofas visada stengiasi pirmiausia paimti kairiąją šakutę, o paskui – dešiniąją (arba atvirkščiai). Padavėjas veikia kaip šviesoforas. Tokią sprendimo koncepciją Dijkstra pristatė 1965 m.

Kaip veikia šis sprendimas? Pažymėkime filosofus raidėmis A, B, C, D, E (pagal laikrodžio rodyklę). Jei valgo du filosofai A ir C, tai užimtos keturios šakutės. Filosofas B sėdi tarp A ir C, todėl nė viena šakutė jam nepasiekiama. Tuo pat metu filosofai D ir E turi prieigą prie vienos nenaudojamos šakutės. Tarkime, kad filosofas D yra alkanas. Aklavietė galima, jei filosofas D iš karto pasiima laisvą šakutę. Tačiau jei jis paprašo padavėjo leidimo ir padavėjas paprašo palaukti, tai reiškia, kad kai tik atsilaisvina pora šakučių, bent vienas filosofas gali paimti dvi šakutes. Aklavietė tampa neįmanoma.

Resursų hierarchija

Tegul resursai (šakutės) sunumeruoti nuo 1 iki 5, o kiekvienas filosofas visada pirmiausia ima šakutę su mažiausiu numeriu, o paskui šakutę su didžiausiu numeriu. Pavalgęs filosofas pirmiausia padeda šakutę su didesniu skaičiumi, o po to su mažesniu. Šiuo atveju, jei keturi iš penkių filosofų vienu metu paims po šakutę su mažiausiu skaičiumi, ant stalo liks šakutė su didžiausiu galimu skaičiumi. Taigi penktasis filosofas negalės paimti jokios šakutės. Be to, tik vienas filosofas galės naudotis šakute su didžiausiu skaičiumi, todėl jis galės valgyti dviem šakutėmis. Baigęs naudoti šakutes, jis pirmiausia padės ant stalo šakutę su didesniu skaičiumi, o tada su mažesniu, taip leisdamas kitam filosofui paimti trūkstamą antrą šakutę ir pradėti valgyti. Šį sprendimą irgi pasiūlė Dijkstra.

Nors išteklių hierarchija padeda išvengti aklavietės, šis sprendimas ne visada praktiškas, ypač kai iš anksto nežinomas reikiamų resursų sąrašas. Pavyzdžiui, jei filosofas turi 3-ią ir 5-ą resursus ir nusprendžia, kad jam reikia 2-o resurso, jis turi atlaisvinti 5-ą resursą, tada 3 resursą, tada perimti 2-ą resursą ir vėl paimti 3-ią ir 5-ą. Kompiuterinės programos, dirbančios su dideliu skaičiumi įrašų duomenų bazėje, veiks neefektyviai, jei prieš perimdamos naują įrašą turės išleisti visus įrašus su aukščiausiais indeksais. Todėl šis metodas yra nepraktiškas.

Uždavinys „Grupinis darbas“

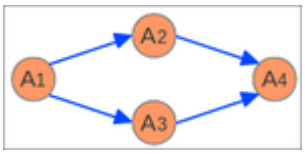
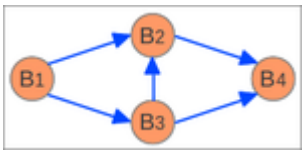
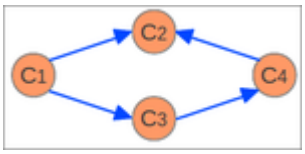
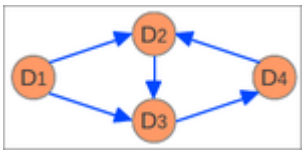
Tai uždavinys iš 2014 metų „Bebro“ konkurso.

Keturios mokinių grupės vykdė projektinius darbus. Tačiau galutinius rezultatus pateikė tik trys iš jų.

Klasės gudručiai Ada ir Vytis pabandė suprasti, kas atsitiko. Jie pastebėjo, kad kai kurie grupės nariai turėjo laukti kito grupės nario darbo rezultatų, kad galėtų atlikti savo darbą.

Grafikai vaizduoja grupių (A, B, C ir D) veiklą: skritulys žymi grupės narį, o rodyklė, nubrėžta nuo nario X1 link nario X2, rodo, jog X2 turi laukti X1 darbo rezultatų, kad galėtų pradėti darbą.

Kuris piešinys vaizduoja grupę, negalinčią užbaigti savo darbo?

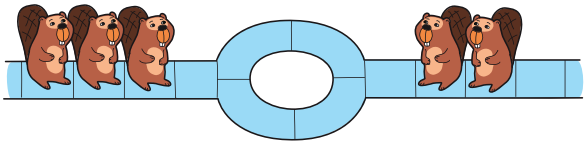
A	B	C	D
			

Paveiksluose pavaizduoti grafai, kuriuose vyksta keturi priklausomi procesai. Jeigu grafe su priklausomais procesais yra ciklų, tai procesai gali būti blokuojami. Vienintelis grafas D turi tokį ciklą (D2, D3, D4). Procesas D2 laukia D4 rezultato, o D4 laukia D3 rezultato ir D3 laukia D2.

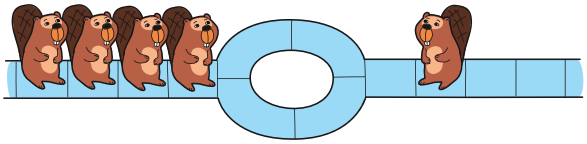
Bebro namas

Bebro name yra keletas kelių. Kadangi Bebras negali eiti (ar važiuoti) atbulas, savo name jis padarė keletą lygiagrečių kelio atkarpų, kad galėtų prasilenkti su kitu bebru. Piešinyje parodytas bebro kelias suskirstytas laukeliais – viename laukelyje gali būti tik vienas bebras. Kurioje situacijoje bebrams bus neįmanoma prasilenkti?

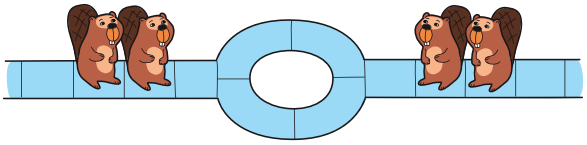
A



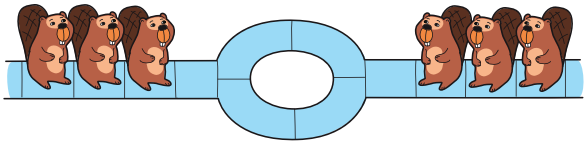
B



C



D



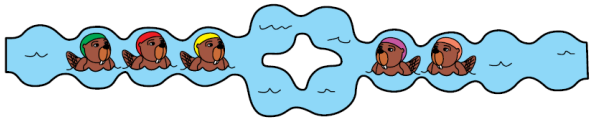
Atsakymas: D

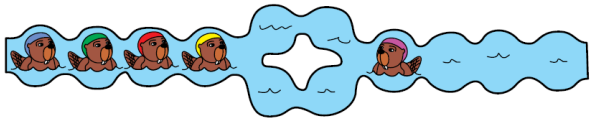
Jei vienoje pusėje laukia vienas ar du bebrai, jie telpa į dvi kelio juostos vietas. Tada priešpriešai važiuojantys automobiliai gali pravažiuoti. Tačiau D situacijoje kiekvienoje pusėje laukia daugiau nei du bebrai, tad visiškai sutriks eismas. Užduotis susijusi su informatikos mokslu: jei du dalyviai laukia, kol kitas dalyvis ką nors padarys pirmas, tuomet negalima imtis jokių veiksmų. Kompiuterijoje tai vadina aklaviete.

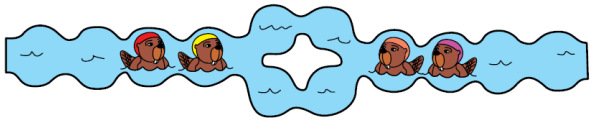
Bebrų upelis

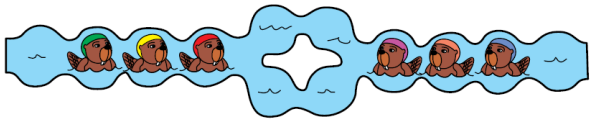
Bebrų žemėje teka siauras upelis, vadinamas Bebro upe. Bebras nenori plaukti atgal, todėl Bebro upėje jis padarė du kanalus, kad galėtų prasilenkti kitos bebrų komandos, plaukiančios priešinga kryptimi. Kiekviename kanale vienu metu daugiausiai gali būti tik du bebrai.

Kuriuo atveju bebrų komandos negalės prasilenkti?

Ⓐ 

Ⓑ 

Ⓒ 

Ⓓ 

Atsakymas: D

Šis uždavinys analogiškas uždaviniui „Bebro namas“.

Bendri įrankiai

Šis uždavinys buvo sprendžiamas Lietuvos „Bebro“ konkurse 2012 metais. Uždavinys labai aiškiai parodo aklavietės situaciją, kai projektuojant lygiagrečius procesus neatsižvelgiama į visas galimas situacijas.

Bebrai Alius ir Bitė kuria žaislus naudodamiesi vienu iš šių įrankių: plaktuku, žirklėmis ar pjūklu. Įrankiai laikomi upės salelėse. Kuriam prireikia įrankio, tas jį ir pasiima. Jei reikiamo įrankio nėra salelėje, bebras laukia, kol tą įrankį grąžins kitas bebras ir turimo įrankio nepadeda.

Kartais abiem bebrams vienu metu reikia to paties įrankio. Jei taip nutinka, jie nutraukia darbą ir eina maudytis.

Kurioje iš pateiktų situacijų bebrai tikrai eis maudytis?

	Alius turi	*Aliui reikia*	*Bitė turi*	*Bitei reikia*
A.				
B.				
C.				
D.				

Atsakymas: B

Paaiškinimas. Jei susiklosto situacijos A, C ir D, vienas iš bebrų gali gauti reikiamą įrankį, baigti kurti žaislą ir grąžinti į salelę abu įrankius. Tada kitas bebras gali gauti reikiamą įrankį ir baigti kurti savo žaislą.

B situacijoje bebrai turės laukti vienas kito amžinai, nes pasiėmė vienas kito reikiamą įrankį.

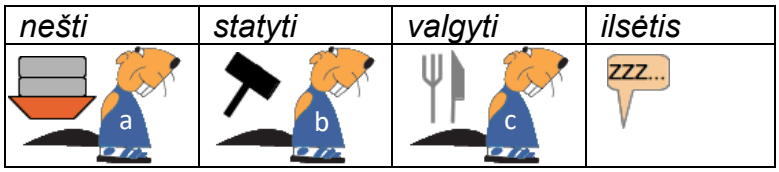
Užtvankos statyba

Šis uždavinys buvo sprendžiamas tarptautiniame „Bebro“ konkurse 2012 metais.

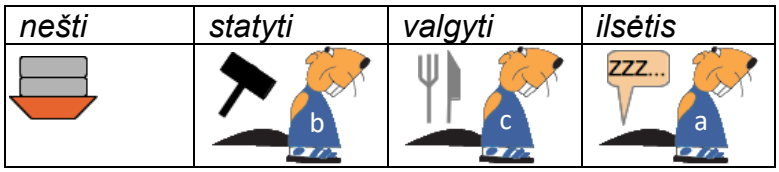
Uždavinio idėja siejasi su bandymu numatyti ir išvengti aklavietės pasinaudojus Filosofų uždavinio galimu sprendimu. Tik šiuo atveju procesus valdo ne padavėjas, bet vyresnysis bebras.

Trys bebrai, kurių vardai „a“, „b“ ir „c“, stato užtvanką, o jiems vadovauja vyresnysis bebras. Yra keturios užtvankos statymo būsenos: „nešti“, „statyti“, „valgyti“ ir „ilsėtis“.

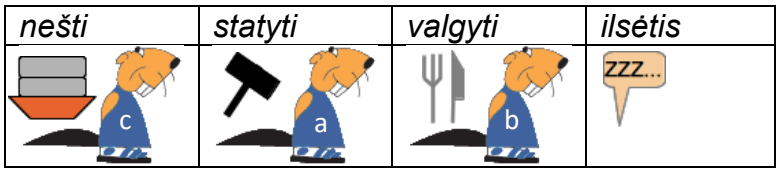
Dabar trijų bebrų būsena yra tokia:



Kiekvienoje būsenoje vienu metu gali būti tik po vieną bebrą. Iš pradžių vyresnysis bebras duoda bebrams tokius nurodymus: „nešti“ → „ilsėtis“. Tai reiškia, kad bebras „a“ iš būsenos „nešti“ turi pereiti į būseną „ilsėtis“. Po šio nurodymo bebrų būsena pasikeičia taip:



Tada vyresnysis bebras duoda tam tikrus nurodymus ir bebrų būsena pasikeičia taip:



Nurodymai vykdomi vienas po kito. Kokius nurodymus davė vyresnysis bebras?

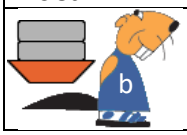
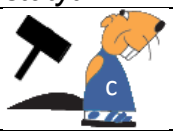
A. „ilsėtis“ → „nešti“, „valgyti“ → „nešti“, „statyti“ → „valgyti“
B. „statyti“ → „nešti“, „valgyti“ → „statyti“, „ilsėtis“ → „valgyti“
C. „valgyti“→ „nešti“, „ statyti“ → „valgyti“, „ilsėtis“ → „statyti“
D. „ilsėtis“ → „nešti“, „ statyti“ → „ilsėtis“, „valgyti“ → „statyti“, „ilsėtis“ → „valgyti“

Atsakymas: C

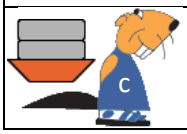
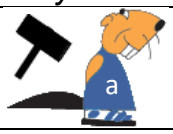
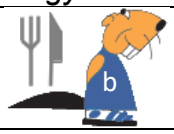
Paaiškinimas

A. Atvejis neįmanomas, nes bebras „a“ ir bebras „c“ būtų vienodoje būsenoje. Tai ir būtų aklavietės atvejis, kai abu bebrai (procesai) naudoja vienu metu vieną resursą. Tai sąlygoje draudžiama.

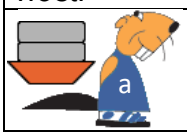
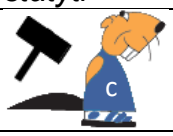
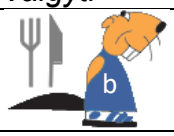
B. Šiuo atveju, įvykdžius vyresniojo bebro instrukcijas, **visi** bebrai būtų skirtingose būsenose, bet **visi** bebrų vardai neatitinka sąlygos. Tai – klaidingas atsakymas.

nešti	*statyti*	*valgyti*	*ilsėtis*
			

C. Atvejis yra **teisingas**, įvykdžius vyresniojo bebro instrukcijas, visų trijų bebrų būsenos yra skirtingos ir sutampa su užduoties sąlyga.

nešti	*statyti*	*valgyti*	*ilsėtis*
			

D. Atvejis neteisingas, nes įvykdžius vyresniojo bebro instrukcijas visų trijų bebrų būsenos yra skirtingos , tačiau tik vienas bebras „b“ bus tokios būsenos, kokia nurodyta sąlygoje. Bebro „a“ ir bebro „c“ būsenos nesutampa su sąlygoje nurodytomis.

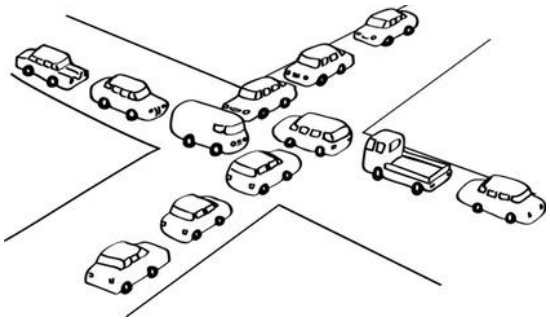
nešti	*statyti*	*valgyti*	*ilsėtis*
			

Apelsinų žaidimas (Maršruto parinkimas ir tinklo aklavietė)

Užduotis parengta pagal leidinio „Informatika be kompiuterio“ (https://classic.csunplugged.org/documents/books/lithuanian/Unplugged-LT.pdf) 10-ą veiklą.

Kompiuterių tinklais informacija (pranešimai, daugiau žr. temą „Įvadas į informacijos teoriją“) perduodama iš kompiuterio į kompiuterį. Prastai projektuojant procesus su informacija gali kilti įvairių problemų.

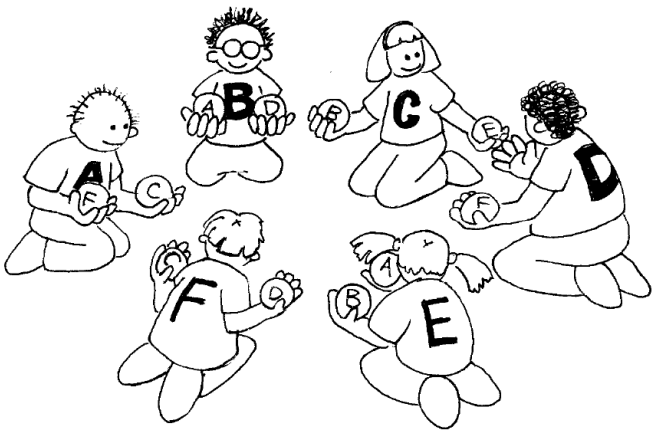
Žaidžiant šį žaidimą su mokinių grupe galima susipažinti su procesų užstrigimu (aklaviete).



3 pav. Aklavietės situacija, kai prastai suprojektuojamas intensyvaus eismo gatvėse reguliavimas https://classic.csunplugged.org/images/activities/routing-and-deadlock/deadlock.jpg

Įvadas

Kai vienu metu daugeliui žmonių prireikia to paties resurso (pavyzdžiui, daug vairuotojų pasirenka tą patį maršrutą), gali susidaryti aklavietė. Šis uždavinys iliustruoja idėją, kad bendradarbiaujant ir susitariant galima išvengti šio nemalonumo.



4 pav. Apelsinų žaidimas

Ugdomi gebėjimai

Problemų sprendimas bendradarbiaujant. Loginis mąstymas

Priemonės

Kiekvienam mokiniui reikės:

Dviejų apelsinų arba teniso kamuoliukų.

Lipnių lapelių su užrašytomis skirtingomis raidėmis. Geriausiai tinka angliškos abėcėlės raidės. Šie lapeliai priklijuojami kiekvienam mokiniui. Atitinkamomis raidėmis pažymimi vaisiai ar teniso kamuoliukai.

Priemonės raidei užrašyti ar priklijuoti ant kiekvieno mokinio ir jo apelsinų ar teniso kamuoliukų.

Svarbu

Vienam mokiniui tenka tik vienas vaisius ar teniso kamuoliukas su jam priskirta ir užrašyta raide.

Alternatyva

Galima žaisti ir mokiniams dėvint skirtingų spalvų marškinėlius ir prie tų spalvų derinant vaisius. Pavyzdžiui, jei geltoni marškinėliai, tai parenkami du bananai ar dvi citrinos, jei žali – žalios paprikos, jei raudoni – raudonos.

Žaisdami šį žaidimą mokiniai bendraudami ir bendradarbiaudami mokosi spręsti užduotį. Kiekvieno žaidėjo galutinis tikslas – savo rankose turėti tik žaidėjui priskirta raide pažymėtus apelsinus.

1. Penki ar daugiau mokinių susėda ratu. Kiekvienas mokinys turi priklijuotą lapelį su jam priskirta raide.
2. Kiekvienam mokiniui duodama po du apelsinus, išskyrus **vieną mokinį** – šiam duodamas vienas apelsinas (tam, kad visada rate būtų vienas delnas be vaisiaus).
3. Žaidimas pradedamas, kai apelsinai išdalijami taip, kad nei vienas mokinys neturėtų nei vienoje rankoje apelsino su jam priskirta raide.
4. Apelsinai siunčiami ratu pagal dvi taisykles:
   a) vienu metu rankoje gali būti tik vienas apelsinas;
   b) apelsinas gali būti perduotas tik mokinio kaimynui iš dešinės ar kairės, jei kaimynas tuo metu rankoje neturi apelsino (mokinys gali perduoti bet kurį iš savo apelsinų).
5. Apelsinai baigiami siųsti ratu, kai kiekvienas mokinys rankose laikys jo raide pažymėtus apelsinus.

Mokiniai greitai pastebės, kad jei jie bus „godūs“ (laikysis savųjų apelsinų, kai tik juos gaus), visai grupei gali nepavykti pasiekti savo tikslo. Gali tekti paaiškinti, kad tai yra komandinis žaidimas ir pavieniui mokiniai negali jo laimėti. Užduotis bus įvykdyta tik tuomet, kai **visi** mokiniai savo rankose turės savuosius apelsinus (su jam priskirtomis raidėmis.

5 pav. Apelsinų žaidimas. Informatikos mokymo simpoziumas. Osaka, Japonija, 2010 m.

6 pav. Užduotis išspręsta. Apelsinų žaidimo variantas su įvairių vaisių poromis. (Vienas mokinys rankoje laiko tik vieną vaisių, o kitos rankos delnas tuščias)

Diskusija

Kokiais būdais mokiniai sprendžia užduotį?

Kokiose realaus gyvenimo situacijose pasitaiko aklaviečių? (Pavyzdžiui, eismo spūstys, žaidžiant krepšinį, žmonių spūstis tarpduryje bandant išeiti iš patalpos.)

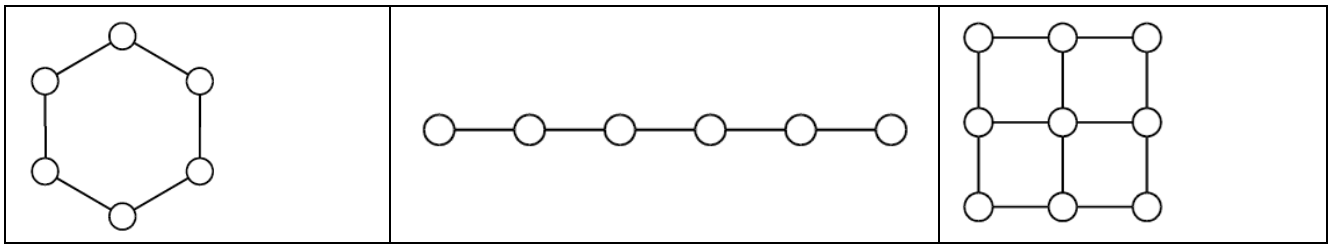
Gudručiams

Mokiniai gali išbandyti žaidimą su daugiau (ar mažiau) mokinių.

Tegul mokiniai pasvarsto apie naujas žaidimo taisykles.

Tegul mokiniai atlieka veiklą nekalbėdami.

Tegul mokiniai išbando pakeistus žaidimus: susėsti viena linija arba turėti daugiau nei du kaimynus, kaip parodyta paveikslėliuose:



7 pav. Sudėtingesni „Apelsinų žaidimo“ variantai

Vaizdo filmuko „Apelsinų žaidimas“ nuoroda:
https://youtu.be/WforXEBMm5k

Šaltiniai

https://www.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/7_Deadlocks.html

https://en.wikipedia.org/wiki/Dining_philosophers_problem

https://www.geeksforgeeks.org/wait-for-graph-deadlock-detection-in-distributed-system/

https://classic.csunplugged.org/activities/routing-and-deadlock/

https://classic.csunplugged.org/books/#lithuanian

https://bebras.lt/3s/informatikos-mokymas-mokytojams/algoritmas-ir-programavimas/aklaviete/