Երևանի Պետական Համալսարան

Ինֆորմատիկայի և Կիրառական մաթեմատիկայի

ֆակուլտետ

Դիսկրետ Մաթեմատիկայի և Տեսական Ինֆորմատիկայի Ամբիոն



Կուրսային Աշխատանք

Թեմա՝ Մատրիցներ։ Հակադարձ Մատրից

Ուսանողներ՝ Ամիրբեկյան Էդգար, Խաչատրյան Արամ

Ղեկավար՝ Ղազարյան Սերժիկ

# Բովանդակություն

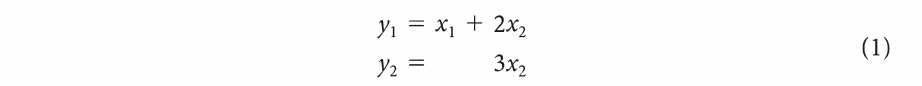
Ծանոթություն -------------------------------------------------------2  
Գործողություններ մատրիցների հետ --------------------------3  
Մատրիցների հանրահաշիվ --------------------------------------7

Հակադարձ մատրից ------------------------------------------------9  
  
Գործնական մաս՝  
Մատրիցների հաշվիչ ծրագրի ռեալիզացիա։  
Նկարագրություն,   
օգտագործված տեխնոլոգիաների սթեքը ---------------------14  
Աշխատանքի մասին -----------------------------------------------15  
Source Կոդը ----------------------------------------------------------16  
  
Օգտագործված գրականություն  
և այլ աղբյուրներ ---------------------------------------------------

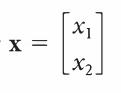
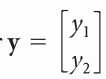
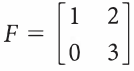
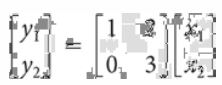
# Ծանոթություն։ Մատրիցան գործողության մեջ

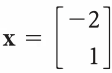
Մատրիցները օգտագործվում են տեղեկատվություն գրանցելու և օժանդակելու, գծային հավասարումների համակարգերի հաշվարկների պարզեցման և այլնի համա, սակայն մատրիցները ունեն իրենց սեփական հանրահաշվական հատկությունները, ինչը մեզ հնարավորություն է տալիս հաշվարկներ կատարել նրանց հետ ենթարկվելով մատրիցների հանրահաշվի կանոններին։

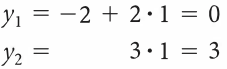
Մատրիցները ստատիկ օբյեկտներ չեն որոնք գրանցում են ինֆորմացիա և տվյալներ․ ավելի շուտ դրանք ներկայացնում են ֆունկցիաների որոշակի տեսակներ որոնք գործում են վեկտորների վրա՝ դրանք վերափոխելով այլ վեկտորների։ Այս մատրիցային վերափոխումները առանցքային դեր ունեն գծային հանրահաշվի ուսումնասիրության մեջ։ Մատրիցները ծագում են բազմաթիվ ձևերից և դիտարկվում են տարբեր տիպերի մատրիցներ ու գործողություններ նրանց հետ և դրանց կիրառությունները որոնք մենք կուսումնասիրենք այս աշխատության ընթացքում։  
Այս բաժնում մենք կքննարկենք մի քանի պարզ օրինակներ ցուցադրելու համար թե ինչպես մատրիցները կարող են վերափոխել վեկտորները։  
  
Դիտարկենք հետևյալ հավասարումները՝

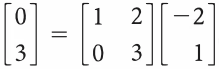


Մենք կարող ենք ցուցադրել այս հավասարումները որպես վեկտորի վերափոխման նկարագրություն՝

Վեկտոր x-իցվեկտոր y-ի ։ Եթե նշանակենք աջ մասի գործակիցների մատրիցը F-ով, ապա՝  և արդեն կարող ենք նկարագրել վերափոխումը հետևյալ կերպ  կամ ավելի համառոտ y = Fx։

Այսպիսով եթե , ապա (1) հավասարումից կստացվի

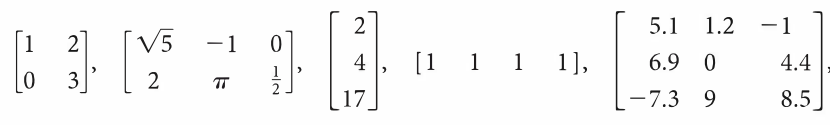
 կամ 

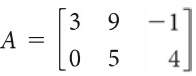
և կարող ենք գրել այս արտահայտությունը որպես ։

# Գործողություններ մատրիցների հետ

Մատրիցը դա թվերի (որոնք կոչվում են մատրիցի տարրեր) ուղղանկյունի զանգված է:

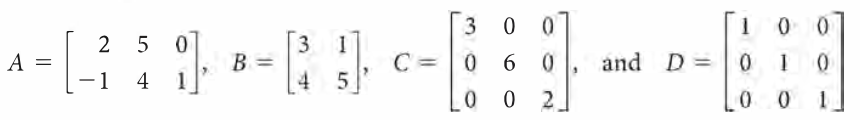
Սրանք մատրիցների օրինակներ են՝



Մատրիցի չափսը դա նրա տողերի և սյուների նկարագրությունն է (m X n)։ Բերված օրինակների չափսերը այսպիսով կլինեն 2x2, 2x3, 3x1, 1x4, 3x3 հաջորդաբար։ 1 x m չափսի մատրիցը կոչվում է տողային մատրից կամ տողային վեկտոր, իսկ n x 1 չափսինը՝ սյունակ մատրից կամ սյունակ վեկտոր։ Մատրիցի յուրաքանչյուր տարր ունի իր համարը որը որոշվում է նրա զբաղեցրած դիրքով մատրիցի տողերի և սյուների մեջ։ A մատրիցի a տարրը գտնովում է նրա i-րդ տողում և j-րդ սյունակում՝   
Օրինակ՝ եթե  ապա = -1 իսկ = 5 (այս գրվածքը կարող է ներկայացվել նաև որպես )։

Մատրիցի անկյունագծային տարրերն են , , ․․․ և եթե մատրիցի տողերը և սյուները հավասար են իրար՝ m = n ապա մատրիցը համարվում է քառակուսի մատրից։ Քառակուսի մատրիցը որի բոլոր ոչ անկյունագծային տարրերը ունեն 0 արժեք կոչվում է անկյունագծային մատրից։ Անկյունագծային մատրիցը որի անկյունագծային տարրերը հավասար են իրար կոչվում է սկայար մատրից։ Եթե սկալյար մատրիցի անկյունագծային տարրերը հավասար են 1-ի այն կկոչվի նույնականացման մատրից։

Օրինակի համար դիտարկենք հետևյալ մատրիցները՝

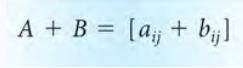


A մատրիցի անկյունագծային տարրերը 2 և 4 են, բայց A-ն քառակուսի չէ։ B-ն քառակուսի մատրից է 2x2 չափսի որի անկյունագծային տարրերը 3 և 5 են։ C-ն անկյունագծային մատրից է, իսկ D-ն 3x3 չափսի նույնականացման մատրից։ n X n չափսի նույնականացման մատրիցը հաճախ նշանակում են որպես ։ Եթե մատրիցի բոլոր տարրերը ունեն զրո արժեք այն անվանվում է զրոյական մատրից և նշանակվում O կամ :  
Այսպիսով մենք կարող ենք դիտարկել մատրիցները որպես վեկտորների ընդհանրացում և ակնհայտ է որ վեկտորների հետ կատարվող գործողությունները կարող ենք տեղափոխել և իրականացնել մատրիցների համար։

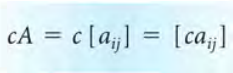
Երկու մատրիցներ համարվում են հավասար եթե նրանք ունեն նույն չափսը և նրանց համապատասխանաբար բոլոր տարրերը հավասար են միմյանց։  
Այսինքն եթե ունենք A = և B = ապա A = B միմիայն այն դեպքում երբ  
m = r, n = s և բոլոր i և j համար։

## Մատրիցների գումարում և սկալյար բազմապատկում

Ընդհանրացնելով վեկտորների գումարման օրենքները մենք սահմանում ենք մատրիցների գումարումը ըստ բաղադրիչների։  
Եթե A = և B = մատրիցները ունեն նույն m X n չափսը ապա նրանց գումարը կլինի կրկին նույն m X n չափսի այլ մատրից որի տարրերը հավասար կլինեն A և B մատրիցների համապատասխան տարրերի գումարին։ Եթե մատրիցների չափսերը հավասար չեն միմյանց գումարման գործողություն սահմանված չէ։



Սկալյար Բազմապատկումը իր բնույթվ նման է գումարմանը քանի որ այն նույնպես հաջորդական է մատրիցի բոլոր տարրերի համար։ Մատրիցը բազմապատկելու համար սկալյար c արժեքով անհրաժեշտ է նրա բոլոր տարրերը բազմապատկել c-ով։



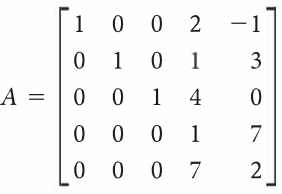
## Մատրիցների բազմապատկում

Ի տարբերություն մատրիցի գումարման և սկալյար բազմապատկման սահմանումների երկու մատրիցների արտադրյալը բաղադրիչային չէ։ Իհարկե մեզ ոչինչ չի խանգարում սահմանելու մատրիցների բազմապատկումը որպես բաղադրիչային, սակայն նմանատիպ սահմանումը ունի շատ քիչ կիրառություններ և այնքան բնական չէ որքան ներքոնշյալ սահմանման դեպքում։  
Եթե A-ն m X n չափսի մատրից է և B-ն n X r չափսի, ապա նրանց արտադրյալը կլինի m X r չափսի մատրից, իսկ նրանց (i, j) տարրերը կհաշվարկվեն հետևյալ բանաձևով

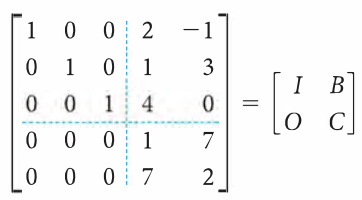
Հարկ է նշել որ մատրիցները պարտադիր պետք է բավարարեն վերոնշյալ չափսային պայմանին դրանք բազմապատկելու համար այլապես բազմապատկման գործողություն սահմանված չէ։

## Մատրիցի տրոհում

Հաճախ հարմար կլինի մատրիցը համարել մի շարք այլ ավելի փոքր ենթամատրիցներից կազմված։ Ներկայացնելով ուղղահայաց և հորիզոնական գծերը մատրիցում մենք կարող ենք տրոհել մատրիցը բլոկների։ Մատրիցները տրոհելու բնական եղանակներ կան որոնք առաջանում են որոշակի կիրառություններում։ Օրինակի համար դիտարկենք հետևյալը՝



Թվում է բնական A-ի հետևայալ տրոհումը



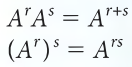
Եթե բազմապատկենք տրոհված ենթամատրիցները կտեսնենք որ մենք կստանանք նույն A մատրիցը քանի որ I, B, O, C մատրիցների չափսերը համապատասխանաբար 3 x 3, 3 x 2, 2 x 3, 2 x 2 են։ Այս դեպքում մենք կարող ենք դիտարկել A-ն որպես այնպիսի մատրից որի տարրերը նույնպես մատրիցներ են։ Մատրիցները տրոհելու կարիք առաջանում է այնպիսի մատրիցներում, որոնք ունեն մեծ թվով 0-ներ և/կամ մեծ թվով տարրեր՝ մատրիցների բազմապատկումը ավելի արագ դարձնելու նպատակով, քանի որ տրոհված մատրիցների բազմապատկումը կատարվում է նույն ձևով ինչ որ սովորական մատրիցներինը։

## Մատրիցի աստիճանային գործողություններ

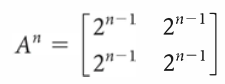
Երբ A և B մատրիցները n X n չափսի քառակուսի մատրիցներ են նրանց արտադրյալը նույնպես կլինի նույն չափսի մատրից։ Հատուկ դեպք է առաջանում այն ժամանակ երբ A = B։ Նման իրավիճակում մենք կստանանք AxA = և ընդհանուր ձևով սահմանենք -ն՝

  
 k հատ

եթե k-ն դրական ամբողջ թիվ է։ Հետևաբար կարելի է սահմանել նաև հետևյալ դեպքերը և ։  
Տեղի ունեն նաև հետևյալ դեպքերը երբ A-ն քառակուսի մատրից է և r-ն ու s-ը ոչ բացասական ամբողջ թվեր են՝



A մատրիցի n աստիճանի համար կարող ենք դուրս բերել հետևյալ բանաձևը   
երբ



Այս պնդումը հեշտությամբ ապացուցվում է մաթեմատիկական ինդուկցիայի մեթոդով։

## Մատրիցի տրանսպոզ

ՄԻնչ այս, մեր կողմից սահմանված բոլոր մատրիցային գործողությունները նման են և ունեն իրենց անալոգիան իրական թվերի գործողությունների հետ։ Մատրիցի տրանսպոզը չունի նման անալոգիա։

m X n չափսի A մատրիցի տրանսպոզը n X m չափսի մատրիցն է՝ ստացված A մատրիցի տողերն ու սյունակները փոխանակելով։  
Հետևաբար -ի i-րդ սյունակը նույնն է ինչ A-ի i-րդ տողը բոլոր i-երի համար և նրանց տարրերի համար բոլոր i-երի և j-երի համար տեղի կունենա հետևյալը՝



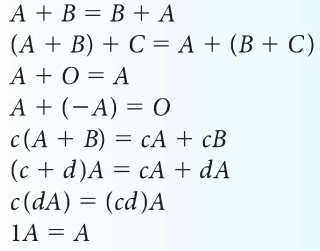
Տրանսպոզը օգտագործվում է նաև քառակուսի մատրիցի ևս մեկ շատ կարևոր տեսակ սահմանելու համար որը կոչվում է սիմետրիկ մատրից։  
Քառակուսի մատրիցը սիմետրիկ է այն դեպքում երբ = A:  
Սիմետրիկ մատրիցը ունի իր սեփական հայելային պատկերը իր հիմնական անկյունագծի երկու մասերում։  
Մատրիցը կարող է համարվել սիմետրիկ միայն ու միայն այն դեպքում երբ բոլոր i-երի և j-երի համար։

# Մատրիցների հանրահաշիվ

Որոշ առումներով, մատրիցների թվաբանությունը ընդհանրացնում է վեկտորների թվաբանությունը։ Մենք չենք սպասում անակնկալների չհաշված գումարման և սկալյար բազմապատկման, և իսկապես դրանք չկան, և սա մեզ թույլ կտա մատրիցների վրա տարածել մի շարք կոնցեպտներ որոնք բխում են վեկտորներից։  
Մասնավորապես գծային կոմբինացիաները, գծային անկախությունը և բազմությունների համախմբերը առանց դժվարության տարածվում են մատրիցների վրա։  
Այնուամենայնիվ, մատրիցներն ունեն իրենց գործողությունները, ինչպիսիք են մատրիցների բազմապատկումը, որոնք վեկտորները չունեն։ Մենք չպետք է ակնկալենք որ մատրիցների բազմապատկումը պետք է վարվի այնպես ինչպես իրական թվերի բազմապատկումը քանի դեռ դա չենք ապացուցել, բայց փաստացի այն այդպես չի վարվում։ Այս Բաժնում մենք ամփոփում ենք մատրիցների գործողությունների հիմնական հատկությունները և մշակում մատրիցների հանրահաշիվ։

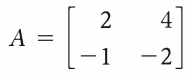
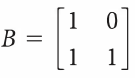
## Գումարման և սկալյար բազմապատկման հատկությունները

Գումարման և սկալյար բազմապատկման բոլոր հատկությունները որոնք գործում են վեկտորների համար ճիշտ են նաև մատրիցների համար։ Ամփոփելով այս ամենը, երբ A-ն, B-ն և C-ն նույն չափսի մատրիցներ են իսկ c-ն ու d-ն սկալյարներ ապա՝

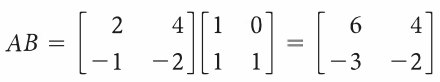


## Մատրիցների բազմապատկման հատկությունները

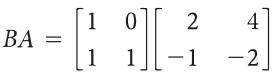
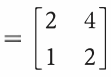
Չնայած որ շատ դեպքերում մատրիցների բազմապատկումը կարող է վարվել իրական թվերի բազմապատկման նման, հարկ է նշել որ դրանք իրականում նույնը չեն և ունեն էական տարբերություններ։  
Դիտարկենք A և B մատրիցները՝

, 

Նրանց AB և BA արտադրյալից կստանանք համապատասխանաբար՝

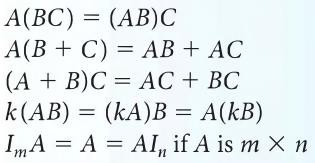


և

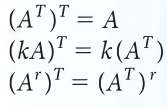
Այսպիսով , այսինքն մատրիցների բազմապատկումը կոմուտատիվ չէ ի տարբերություն իրական թվերի՝ արտադրիչների հաջորդականությունը ազդում է արդյունքի վրա։

Մատրիցների բազմապատկման հատկությունները A, B, C մատրիցների (որոնց չափսերը բավարարում են մատրիցների բազմապատկման պայմանին) և k սկալյարի համար կլինեն․

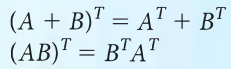


## Մատրիցի տրանսպոզի հատկությունները

Դիցուք A և B մատրիցների (որոնց չափսերը այնպիսին են որ գործողությունը կարող է տեղի ունենալ) և k սկալյարի համար կունենանք․



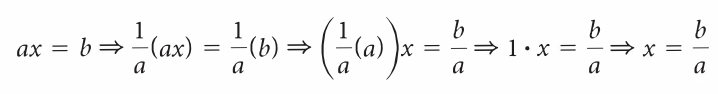
և



Եթե A-ն քառակուսի մատրից է ապա -ն սիմետրիկ մատրից է և յուրաքանչյուր մատրիցի համար և -ն սիմետրիկ մատրիցներ են։

# Հակադարձ մատրից

Այս բաժնում մենք վերադառնում ենք գծային հավասարումների համակարգի Ax = b-ի մատրիցային նկարագրությանը և համակարգի լուծման համար կփնտրենք մատրիցային հանրահաշվարկի օգտագործման եղանակներ։ Անալոգիայի համար կարող ենք հաշվի առնել ax = b հավասարումը, որտեղ a,b և x-ը ներկայացնում են իրական թվեր և մենք ուզում ենք լուծել այն x-ի համար։ Մենք կարող ենք արագ պարզել որ x = b/a որպես լուծում, բայց հարկ է նշել որ դա միայն այն դեպքում երբ   
, այս դեպքում մենք կհասնենք լուծման հետևյալ կերպ․



Այս պրոցեսը որպես մատրիցային Ax=b հավասարումվերարտադրելու համար մեզ հարկավոր է գտնել մատրից (նման 1/a-ին) այնպիսին որ նույնականացման մատրիցին (որը նման է 1-ին)։ Եթե նման մատրից գոյություն ունի (նման պայմանին), ապա մենք կարող ենք իրականացնել հետևյալ հաշվարկների հաջորդականությունը․



Մեր վերոնշյալ նպատակը ճշգրիտ որոշելն է երբ մենք կարող ենք գտնել նման մատրից։ Մենք կպնդենք նաև որ ոչ միայն , բայց նաև ։ Այս պնդումից կհետևի նաև որ մատրիցները քառակուսի են։

Եթե A-ն n X n չափսի քառակուսի մատրից է, և նրա հակադարձ n X n մատրիցը, որը բավարարում է , որտեղ -ը n X n նույնականացման մատրից է ու այդպիսի գույություն ունի, ապա -ն հակադարձելի մատրից է։

Հարկ է նշել որ չնայած այն բանի, որ մենք տեսանք որ մատրիցների բազմապատկումը կոմուտատիվ չէ հիմնականում, բայց A’ (եթե այն գոյություն ունի) պետք է բավարարի A’A = AA’ պայմանին:

Այս ամենը առաջ է քաշում մի քանի հարցեր․  
― Ինչպե՞ս կարելի է իմանալ, թե երբ մատրիցը ունի հակադարձ և երբ ոչ  
― Եթե մարիցն ունի հակադարձ ինչպե՞ս կարելի է գտնել այն  
― Քանի՞ հակադարձ կարող է ունենալ մատրիցը

Հաջորդիվ կպատասխանենք վերոնշյալ հարցերին։

ԹԵՈՐԵՄ։ Եթե A-ն հակադարձելի մատրից է, ապա նրա հակադարձը միակն է։

Ապացույց։ Մենք կօգտվենք ունիկալության ապացուցման ստանդարտ ձևերից մեկից և կենթադրենք, որ միաժամանակ գոյություն ունի մեկ A մատրիցի երկու A’, A’’ հակադարձներ։ Այստեղից հետևում է որ․  
 և  այսպիսով՝ հետևաբար  և այստեղից համոզվում ենք որ հակադարձը ունիկալ է (A մատրիցի հակադարձը նշանակվում է , սակայն մենք այն չենք կարող նեկայացնել որպես 1/A քանի որ մատրիցների համար սահմանված չէ բաժանման գործողություն):

ԹԵՈՐԵՄ։ Եթե A-ն հակադարձելի n X n մատրից է, ապա Ax = b կողմից տրված գծային հավասարման համակարգը ունի ունիկալ լուծում x = , յուրաքանչյուր b պատկամնում է :

Ապացույց։ Թեորեմը ֆորմալացնում է բաժնի սկզբում արված դատողությունը։ Մենք կդիտարկենք այն ավելի մանրամասն այս անգամ։  
Անհրաժեշտ է ապացուցել երկու պնդում․ այն որ Ax = b-ն ունի լուծում և այն միակն է։   
Դա ցույց տալու համար կպահանջվի ապացուցել որ  տեղի ունի։ Ստուգենք՝ Այսպիսով ֊ն բավարարում է հավասարմանը և հետևաբար այն ունի ամենաքիչը մեկ լուծում։  
Ցույց տալու համար որ այն նաև ունիկալ է ենթադրենք գոյություն ունի ևս մեկ՝ y լուծում․ ապա Ay = b և բամապատկելով հավասարման երկու մասերը ֊ով ձախ մասում կստանանք առընչությունների շղթա և՝



Կհետևի որ y-ը նույն լուծումն է ինչը մենք արդեն ստացել էինք, և կասենք որ լուծումն ունիկալ է։

Եվ այսպես վերադառնալով հակադարձվող մատրիցի հակադարձը գտնելուն՝ կսահմանենք ընդհանրացված պրոցեդուրա։ Իրավիճակը առանձնացնենք 2 x 2 մատրիցի համար քանի որ այն բավականաչափ պարզ է օրինափաչությունը հասկանալու համար։

ԹԵՈՐԵՄ։ Եթե ապա A-ն հակադարձելի է եթե՝ ad – bc 0 ինչի դեպքում՝ ։  
Եթե ab – bc = 0, ապա A-ն հակադարձելի չէ։

ad – bc արտահայտությունը կոչվում է A մատրիցի դետերմինանտ և նշանակվում է det(A): Վերոնշյալ A մատրիցի հակադարձը (երբ այն գոյություն ունի) հավասար է 1/det(A)-ի և այն մատրիցի արտադրյալին որը ստացվում է հիմնական անկյունագծի տարրերի տեղափոփոխության և մյուս երկու տարրերի նշանները փոխելուց։

Ընդհանրացված բանաձև դուրս բերելու համար կատարենք հետևյալ քայլերը․  
Գիտենք որ 2x2 չափսի A մատրիցը հակադարձելի է միայն ու միայն այն դեպքում երբ det(A) 0։ Մատրիցի դետերմինանտը քարակուսի մատրիցների հատկություն է և այն միշտ կարող է որոշվել, սակայն այն ռեկուրսիվ գործողություն է և չկա պարզեցված բանաձև շատ մեծ քարակուսի մատրիցների համար, այսինքն դրանք պետք է բաժանվեն 2x2 չափսի մատրիցների և հետ վերադառնալ ռեկուրսիայով։

Ապացույց։ Ենթադրենք որ det(A) = ad – bc 0, ախա՝



Նմանապես՝ 

Քանի որ det(A) 0 կարելի է բազմապատկել յուրաքանչյուր հավասարման երկու մասերը 1/det(A)։

Կստանանք  և 

Հետևաբար  մատրիցը բավարարում է հակադարձելիության պայմանը և A-ն հակադարձելի է։ Քանի որ A-ի հակադարձը ունիկալ է մենք պետք է ունենանք՝



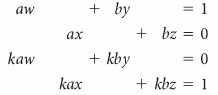
Ընդհակառակը, ենթադրենք որ ad – bc = 0 և առանձին քննարկենք երկու դեպքերը երբ a 0 և a = 0:  
Եթե a 0, ապա d = bc/a, որտեղից մատրիցը կարող ենք ներկայացնել հետևյալ կերպ՝



Որտեղ k = c/a, կամ այլ կերպ ասած A-ի երկրորդ տողը առաջինի բազմապատիկն է, համարելով որ A-ն ունի հակադարձ , ապա



Եվ համապատասխան գծային հավասարումների հետևյալ համակարգը չունի լուծում



Քանի որ եթե a = 0, ապա ad – bc = 0 ֊ն ենթադրում է որ bc = 0, բայց քանի որ ոչ b-ն ոչ c-ն 0 չէ՝ հետևաբար A-ն ունի հետևյալ տեսքերից մեկը

 կամ 

Առաջին դեպքում՝ , նմանապես ֊ը նույնպես չի կարող ունենալ հակադարձ։  
Հետևաբար՝ եթե ad – bc = 0, ապա A-ն հակադարձելի չէ։

Գծային հավասարումների համակարգը հակադարձի մեթոդով լուծելը թվում է բավականաչափ էֆֆեկտիվ սակայն բացի 2x2 չափսի մատրիցներից գրեթե միշտ ավելի արագ է օգտագործել Գաուսիան կամ Գաուս֊Ջորդանի դուրսգրման մեթդը որը էֆֆեկտիվ է 2x2 բազմապատիկ չափսերի մատրիցների և այլ հատուկ կարգի մատրիցների համար։ Հարկ է նշել նաև որ հակադարձի մեթոդը կաշխատի միայն այն դեպքում երբ մատրիցը քառակուսի է և հակադարձելի, այն ինչ Գաուսիան դուրսգրման մեթոդը միշտ կարող է կիրառվել։

## Հակադարձելի մատրիցների հատկություննեը

Եթե A-ն հակադարձելի մատրից է ապա -ը նույնպես հակադարձելի է և՝



Եթե A-ն հակադարձելի մատից է և c-ն ոչ զրոյական սկալյար, ապա cA-ն հակադարձելի մատրից է․



Եթե A-ն և B-ն նույն չափսի հակադարձելի մատրիցներ են, ապա AB-ն նույնպես հակադարձելի է և



Եթե A-ն հակադարձելի մատրից է, ապա նույնպես հակադարձելի է․



Եթե A-ն հակադարձելի մատրից է, ապա -ը հակադարձելի է բոլոր ոչ 0 n ամբողջ թվերի համար․



Ցույց տալու համար որ -ը հակադարձելի է ցույց տանք այնպիսի X մատրիցի գոյությունը որի դեպքում՝ ։ -ը բավարարում է այս պայմանին և ելնելով նրանից որ հակադարձը միակն է կարող ենք ասել որ

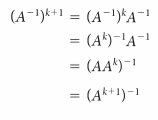
Այս հատկությունն ապացուցվում է մաթեմատիկական ինդուկցիայի մեթոդով, սակայն դիտարկենք n = 2 դեպքը կստանանք



Նմանապես այստեղից՝ , որն էլ -ու հակադարձն է։ Հատուկ դեպք է n = 0 դեպքը երբ ինչը նույնն է ինչ որ ցույց տալը թե I-ն հակադարձելի է որն արդեն տեսել ենք անցած բաժնում։  
Ենթադրենք որ արդյունքը ճիշտ է երբ n = k, որտեղ k-ն հատուկ ոչ բացասական ամբողջ թիվ է։ Ըստ ինդուկցիայի մեթոդի կենթադրենք որ -ն հակադարձելի է և ըստ դրա՝



Հաջորդ քայլով ապացուցենք որ -ը հակադարձելի է և որ ։ Մենք 3-րդ կետում արդեն ցույց ենք տվել որ -ն հակադարձելի է քանի որ A-ն և ըստ ենթադրության -ն։ Ավելին՝



Հետևաբար -ը հակադարձելի է բոլոր ոչ բացասական ամբողջ n-երի և ինդուկցիայի համաձայն։

# Գործնական մաս՝ Մատրիցների հաշվիչ ծրագրի ռեալիզացիա։

## Նկարագրություն, օգտագործված տեխնոլոգիաների սթեքը

Ծրագիրը ստեղծելու համար օգտագործվել է Python ծրագրավորման լեզուն (3-րդ վերսիա), նրա սիսեմային բառարանը և ներկառուցված տվյալների կառուցվածքները։ Լեզվում ներկառուցված կամ երրորդական բառարաններում առկա ալգորիթմները և Մատրիցի տվյալների կառուցվածքը չեն օգտագործվել դրա փոխարեն նախագծվել են տվյալ խնդրի համար առավել օպտիմիզացված իմ սեփական ալգորիթմները և սեփական Մատրիցի տվյալների կառուցվածքը։

Ծրագիրը կառուցված է հետևյալ ֆայլերից՝

*a)-`\_\_init \_\_. py`  
b)- `calculator.py`  
c)- `main.py`  
d)- `Matrix.py`  
e)- `my\_algorithms.py`  
f)- `parser.py`*

a) դատարկ ֆայլը պահանջում է Python լեզվի ինտերպրետատորը անհրաժեշտ մոդուլիզացիան ապահովելու համար ֆայլային կառուցվածքում:  
b) ֆայլում ներառված է ամբողջ հաշվողական լոգիկան և ֆունկցիոնալությունը:  
c) այս ֆայլը ծառայում է որպես օգտագործողի հետ ինտերակցիայի միջոց և կատարման կետ։  
d) Մատրից կլասսի միջոցով նրա տվյալների կառուցվածքը իմպլեմենտացված է այս ֆայլում։  
e) Ծրագրի համար անհրաժեշտ ալգորիթմները ներառված են այս ֆայլում։  
f) Օգտագործողի Ինտերֆեյսից ստացված տվյալները վերլուծվում և փոխանցվում են բուն ծրագրին այս ֆայլի միջոցով։

Ծրագիրը կարող է հաշվել մատրիցների գումարում, հանում, բազմապատկում, սկալյար բազմապատկում, տրանսպոզիցիա, դետերմինանտը և հակադարձը։

Այն սկսելու համար անհրաժեշտ է կոնսոլով տպել՝

*python3 /ճանապարհը/դեպի/անհրաժեշտ/թղթապանակ/Matrix-Calculator/*

Ծրագիրը շնորհիվ իր Օգտագործողի Ինտերֆեյսի կուղեկցի մինչև ծրագրի ավարտը։  
Ծրագիրը ընդունում է նաև լողացող թվով կամ կոտորակային թվեր։

## Աշխատանքի մասին

Ավարտելով Python պրոյեկտի ձևավորման նախապատրաստական աշխատանքները որոշեցի սկսել Օգտագործողի Ինտերֆեյսի մասից և նախագծեցի parser.py մոդուլը, որի մեջ իմպլեմենտացված են մեթոդներ օգտագործողին հարցնելու, ստանալու և վերամշակելու այնպիսի ինֆորմացիա ինչպիսին մատրիցի կազմն ու կիրառվող օպերացիաներն են, ապա ռեալիզացրեցի Մատրիցի տվյալների կառուցվածքը Matrix.py ֆայլի Matrix կլասում ապահովելով անհրաժեշտ մեթոդներով և հատկություններով։ Այնուհետև վերջացնելով տարրական օպերացիաները ինչպիսին սկալյար բազմապատկումն ու գումարումն են, նախագծեցի LU դեկոմպոզիցիայի լոգիկան, որից հետո շատ ավելի հեշտ էր դետերմինանտի հաշվումը ռեալիզացնելը։ LU դեկոմպոզիցիայի ֆունկցիոնալության մեջ իրականացրել եմ քառակուսի մատրիցի դեկոմպոզիցիան երկու եռանկյունի մատրիցներ L-ի և U-ի (որտեղից էլ անունը LU դեկոմպոզիցիա), որոնց դետերմիանտները հեշտությամբ որոշվում են հիմնական անկյունագծից (վերևի ձախ անկյունից դեպի նեքևի աջ) և քանի որ A = L \* U, A-ի դետերմինանտը որոշվում է det(L) \* det(U) -ից։ LU դեկոմպոզիցիան նաև շատ հարմար էր որոշելու մատրիցի հակադարձը նույն ձևով քանի որ : Այնուհետև նախապատրաստեցի Python լեզվի ներկառուցված ալգորիթմների իմ տարբերակների վերսիան որը շատ ժամանակ չխլեց և հիմնականում հեշտ էր իրականացնել։  
Հաջորդիվ ձեռնամուխ եղա ապահովելու ծրագրի աշխատանքը նաև Python version 2 -ի հետ համատեղելիության համար որը նույնպես հեշտությամբ հաջողվեց։ Եվ վերջում իրականացրեցի կոդի դոկումենտացիա և ռեֆակտորինգ դարձնելով այն ավելի մաքուր, ընթեռնելի և հեշտ հասկացվող։

## Source Կոդը

### main.py

*from* .parser *import* parseMatrix, parseOperator, parseScalar, askToContinue  
*from* .calculator *import* \*  
*import* sys  
  
*# Դարձնել այս մոդուլը Pyrhon 2 համատեղելի.  
if* sys.version\_info[0] == 2:  
 input = raw\_input  
  
  
*def* \_\_handle\_operator(operator, matrix):  
 *"""Կատարել օգտվողի կողմից տրված գործողությունը մատրիցի հետ"""  
  
 # Օգտվողը ուզում է հաշվել դետերմինանտը  
 if* operator == "det":  
 ans = matrixDeterminant(matrix)  
 *if* ans != 0 *and not* ans:  
 *print*("Դետերմինանտը անորոշ է կամ գոյություն չունի տրված մատրիցի համար")  
 *else*:  
 *print*("Դետերմինանտն է՝: " + *str*(ans))  
  
 *# Օգտվողը ուզում է հաշվել հակադարձը.  
 if* operator == "inverse" *or* operator == "-1":  
 matrix2 = matrixInverse(matrix)  
 *if not* matrix2:  
 *print*("Մատրիցը հակադարձելի չէ")  
 *else*:  
 matrix = matrix2  
 *print*("Մատրիցի հակադարձումը հաջողված է")  
  
 *# Օգտվողը ուզում է հաշվել բազմապատկել մատրիցը սկալյարով.  
 if* operator == "scalar":  
 n = parseScalar()  
 matrix.multiplyScalar(n)  
 *print*("Մատրիցի բազմապատկումը սկալյարով հաջողված է")  
  
 *# Օգտվողը ուզում է հաշվել տրանսպոզը  
 if* operator == "transpose":  
 matrix = matrixTranspose(matrix)  
 *print*("Մատրիցի տրանսպոզիցիան գտնելը հաջողված է")  
  
 *# Օգտվողը ուզում է հաշվել մատրիցների գումարը  
 if* operator == "+":  
 matrix2 = parseMatrix()  
  
 *# Կվերադարձնի None եթե մատրիցների գումարումն անհնար է* resultMatrix = matrixAddition(matrix, matrix2)  
  
 *if not* resultMatrix:  
 *print*("Մատրիցի գումարումը ձախողված է")  
 *else*:  
 matrix = resultMatrix  
 *print*("Մատրիցի գումարումը հաջողված է")  
  
 *# Օգտվողը ուզում է հաշվել մատրիցների տարբերությունը  
 if* operator == "-":  
 matrix2 = parseMatrix()  
 resultMatrix = matrixSubstraction(matrix, matrix2)  
 *if not* resultMatrix:  
 *print*("Մատրիցի բազմապատկումը ձախողված է")  
 *else*:  
 matrix = resultMatrix  
 *print*("Մատրիցի բազմապատկումը հաջողված է")  
  
 *# Օգտվողը ուզում է հաշվել մատրիցների արտադրյալը  
 if* operator == "\*":  
 matrix2 = parseMatrix()  
 resultMatrix = matrixMultiplication(matrix, matrix2)  
 *if not* resultMatrix:  
 *print*("Մատրիցի բազմապատկումը տապալված է")  
 *else*:  
 matrix = resultMatrix  
 *print*("Մատրիցի բազմապատկումը հաջողված է")  
  
 *# Տպել արդյունքում ստացված մատրիցը օգտվողին  
 print*(matrix)  
 *return* matrix  
  
  
*def* main():  
 *"""Կապել հայտարարված ֆունկցիաները միմյանց"""  
  
 # Հարցնել առաջին մատրիցը* matrix = parseMatrix()  
  
 *# Կատարել օպերացիան կամ կանգնել երբ այն None է  
 while* matrix:  
 *# Հարցնել ինչ օպերացիա է օգտվողն ուզում կիրառել* operator = parseOperator()  
  
 *# Կատարել գործողությունը մատրիցի(ների) վրա* matrix = \_\_handle\_operator(operator, matrix)  
  
 userWantsToContinueWithCurrentMatrix = askToContinue()  
 *if not* userWantsToContinueWithCurrentMatrix:  
 *# Հարցնել նոր մատրից եթե Օգտագործողը ոչ մի տարր չի հայտարարել  
 # կամ եթե մատրիցը None արժեք ունի  
 # ծրագիրը կկանգնի* matrix = parseMatrix()  
  
 *print*("")  
 *print*("Ցտեսություն!")

### parser.py

*from* .my\_algorithms *import* my\_split, my\_strip, my\_lower  
*from* .calculator *import* frac\_reduc  
*from* .Matrix *import* Matrix  
*import* sys  
  
*# Դարձնել այս մոդուլը Pyrhon 2 համատեղելի.  
if* sys.version\_info[0] == 2:  
 input = raw\_input  
  
  
*def* \_\_ask\_input(string):  
 *"""Հարցնել օգտվողի ինֆորմացիա 'string' փոփոխականի միջոցով."""  
 try*:  
 input\_data = input(string)  
 *except* (*KeyboardInterrupt*, *EOFError*):  
 *print*("\nՑտեսություն!")  
 sys.exit(0)  
 *return* input\_data  
  
  
*def* \_\_is\_number(n):  
 *"""Պարզել տրված արգումենտը թիվ է թե ոչ"""  
 try*:  
 *# Տողերը կարող են փոխարկվել լողացող թվով թվերի եթե ճիշտ ձևով են գրված  
 float*(n)  
 *return True  
 except*:  
 *return False  
  
  
def* \_\_parse\_float(string):  
 *"""Փորձել ստանալ լողացող թվով թիվ տողից"""* elem = my\_split(string, ".")  
  
 *# Տողը լող․ կետով թվի փոխարկելի ձևով չի գրված  
 if len*(elem) == 1:  
 elem = my\_split(string, ",")  
  
 *# Տողը լող․ կետով թվի փոխարկելի ձևով է գրված  
 if len*(elem) == 2:  
 *# Ամբողջ կամ տասնավոր մասը թիվ չէ  
 if not* (\_\_is\_number(elem[0])) *or not* (\_\_is\_number(elem[1])):  
 *return None* numerator = *int*(elem[0]) \* 10 \*\* *len*(elem[1])  
 *if* numerator >= 0:  
 numerator += *int*(elem[1])  
 *else*:  
 numerator -= *int*(elem[1])  
  
 denominator = 10 \*\* *len*(elem[1])  
 frac = (numerator, denominator)  
  
 *return* frac\_reduc(frac)  
  
 *return None  
  
  
def* \_\_parse\_fraction(string):  
 *"""Փոխարկել տողի մի մասը. Վերադարձնել none եթե տվյալը մեկ մասից չէ բաղկացած"""* elem = my\_split(string, "/")  
  
 *if len*(elem) == 2:  
 *# Ստուգել տողի կոտորոկային կամ լող․ կետով թվ․ փոխարկելու հնարավորությունը* frac\_1 = \_\_parse\_float(elem[0])  
 frac\_2 = \_\_parse\_float(elem[1])  
  
 *if not* frac\_1:  
 *if not* \_\_is\_number(elem[0]):  
 *return None* frac\_1 = (*int*(elem[0]), 1)  
  
 *if not* frac\_2:  
 *if not* \_\_is\_number(elem[1]):  
 *return None* frac\_2 = (*int*(elem[1]), 1)  
  
 numerator = frac\_1[0] \* frac\_2[1]  
 denominator = frac\_1[1] \* frac\_2[0]  
 frac = (numerator, denominator)  
 *return* frac\_reduc(frac)  
  
 *return None  
  
  
def* \_\_parse\_values(row):  
 *"""Փոխարկել տողը զանգվածի"""* values = []  
 args = my\_split(row, " ")  
  
 *for* i *in range*(*len*(args)):  
 elem = \_\_parse\_fraction(args[i])  
  
 *if not* elem: *# elem-ը մի ամբողջություն չէ* elem = \_\_parse\_float(args[i])  
  
 *if not* elem: *# elem-ը լող․ կետով թիվ չէ  
 if not* \_\_is\_number(args[i]): *# elem-ը թիվ չէ  
 return None* elem = (*int*(args[i]), 1)  
  
 values.append(elem)  
  
 *return* values  
  
  
*def* parseMatrix():  
 *"""Խնդրել օգտվողին ներդնել մատրից գործողություն կատարելու և նոր մատրից վերադարձնելու համար"""* rows = []  
  
 *print*("")  
 *print*("Ներմուծեք մատրիցի տարրերը տող առ տող․ enter-ը նշանակում է տողի վերջ.")  
  
 row = my\_strip(\_\_ask\_input("տող: "))  
  
 *# Ցիկլով անցնել մինչև որ դատարկ տողի հանդիպելը.  
 while* row:  
 newRow = \_\_parse\_values(row)  
  
 *# Օգտվողը տվել է սխալ ինֆորմացիա  
 if not* newRow:  
 *print*("")  
 *print*("Օգտվողը տվել է սխալ ինֆորմացիա")  
 *print*("Խնդրում եմ, գրիր տողերը ճշգրիտ")  
 *print*("Թույլատրելի է միայն ամբողջ թվեր, լողացող կետով կամ կոտորակային թվեր "  
 "առանց փակագծերի և այլ նշանների")  
  
 *# Օգտվողը փորձում է ներմուծել ավելի շատ արժեքներ քան տողի չափսն է  
 elif len*(rows) > 0 *and len*(newRow) != *len*(rows[0]):  
 *print*("")  
 *print*("Դուք ներմուծել եք տարրերի սխալ քանակություն")  
 *print*("Ուշադրություն դարձրեք տեղերի չափսի և տվյալների քանակության վրա")  
  
 *else*:  
 rows.append(newRow)  
  
 row = my\_strip(\_\_ask\_input("տող: "))  
  
 *# Տողեր տրված չեն  
 if len*(rows) == 0:  
 *return None* ret = Matrix(rows, *len*(rows), *len*(rows[0]))  
 *return* ret  
  
  
*def* parseOperator():  
 *"""հարցնել գործողության տիպը"""  
  
 print*("")  
  
 *print*("Ի՞նչ գործողություն եք ցանկանում իրականացնել հաջորդիվ")  
 *print*("Կախված ձեր ընտրությունից հնարավոր է ուրիշ մատրից մուտքագրելու անհրաժեշտություն")  
  
 *print*("")  
  
 *print*("+: Ավելացնել մեկ այլ մատրից ստացվածին")  
 *print*("-: Հանել ուրիշ մատրից ստացվածից")  
 *print*("\*: Բազմապատկել ուրիշ մատրից ստացվածին")  
 *print*("det: Հաշվել ստացված մատրիցի դետերմինանտը")  
 *print*("inverse: Հակադարձել ստացված մատրիցը եթե հնարավոր է")  
 *print*("scalar: Բազմապատկել ստացված մատրիցը սկալյարով")  
 *print*("transpose: Հաշվել ստացված մատրիցի տրանսպոզիցիան")  
 *print*("print: Տպել ստացված մատրիցը")  
  
 *print*("")  
  
 operator = my\_lower(my\_strip(\_\_ask\_input("Operator: ")))  
 *while* operator *not in* ["\*",  
 "-",  
 "+",  
 "det",  
 "scalar",  
 "inverse",  
 "invert",  
 "-1",  
 "print",  
 "transpose"]:  
 *print*("Խնդրում եմ ընտրեք վալիդ գործողություն")  
 operator = my\_lower(my\_strip(\_\_ask\_input("Գործողություն: ")))  
  
 *return* operator  
  
  
*def* parseScalar():  
 *"""Հարցնել օգտվողին սկալյար արտադրիչ"""  
 print*("")  
  
 scalar = \_\_ask\_input("Ներմուծել սկալյար մեծություն ամբողջ թվով: ")  
  
 *# Օգտվողը պեպտք է մուտքագրի ամբողջ թիվ որպես սկալյար  
 while not* \_\_is\_number(scalar):  
 scalar = \_\_ask\_input("Ներմուծել սկալյար մեծություն ամբողջ թվով: ")  
  
 *if* scalar // 1 == scalar:  
 *return int*(scalar)  
 *return* scalar  
  
  
*def* askToContinue():  
 *"""Հարցնել ստացված մատրիցի հետ գործողություն կատարելու որոշման մասին"""  
  
 print*("")  
 *print*("Ուզում ե՞ք կիրառել այլ գործողություն ստացված մատրիցի հետ")  
  
 a = my\_lower(\_\_ask\_input("Այո/Ոչ մուտքագրել Y կամ N: "))  
 *print*("")  
  
 *while*(a != "y" *and* a != "n"):  
 a = my\_lower(\_\_ask\_input("Այո/Ոչ մուտքագրել Y կամ N: "))  
  
 *if* a == "y":  
 *return True  
 return False*

### Matrix.py

*from* .my\_algorithms *import* my\_range  
  
*class* Matrix:  
 *"""Matrix Կլասս"""  
  
 def \_\_init\_\_*(*self*, rows, n, m):  
 *"""Կառուցել մատրիցի օբյեկտ  
  
 Արգումենտներ:  
 rows -- տողերի զանգվածը  
 n -- տողերի թիվը  
 m -- սյունակների թիվը  
 """  
 self*.rowAmount = n  
 *self*.colAmount = m  
 *self*.rowArray = rows  
 *self*.colArray = [[] *for* i *in* my\_range(n)]  
 *self*.scalar = 1  
  
 *def \_\_str\_\_*(*self*):  
 *"""Տպել մատրիցը օգտվողին ընթեռնելի ձևով"""* output = ""  
  
 *for* row *in self*.rowArray:  
 output += "["  
  
 *for* i *in* my\_range(*len*(row)):  
 cell = row[i]  
 elem = *self*.scalar \* 1.0 \* cell[0] / cell[1]  
  
 *if* elem % 1 == 0:  
 elem = *int*(elem)  
  
 *if* i == *len*(row) - 1:  
 output += *str*(elem)  
 *else*:  
 output += *str*(elem) + " "  
  
 output += "]\n"  
 *return* output[:-1]  
  
 *def* multiplyScalar(*self*, n):  
 *"""Բազմապատկել սկալյարով."""  
 self*.scalar \*= n  
  
 *def* getRowAmount(*self*):  
 *"""Վերադարձնել տողերի քանակը"""  
 return self*.rowAmount  
  
 *def* getRowArray(*self*):  
 *"""Վերադարձնել տողերի զանգվածը"""  
 if self*.scalar == 1:  
 *return self*.rowArray  
 returnArray = [[*self*.scalar \* elem *for* elem *in* row]  
 *for* row *in self*.rowArray]  
 *return* returnArray  
  
 *def* getColAmount(*self*):  
 *"""Վերադարձնել սյունակների թիվը"""  
 return self*.colAmount  
  
 *def* getColArray(*self*):  
 *"""Վերադարձնել սյունակների զանգվածը"""  
 return self*.colArray  
  
 *def* getScalar(*self*):  
 *"""Վերադարձնել սկալյարը"""  
 return self*.scalar  
  
 *def* getCell(*self*, row, col):  
 *"""Վերադարձնել տարրի արժեքը ըստ նրա դիրքի"""  
 return* (*self*.scalar \* *self*.rowArray[row][col][0],  
 *self*.rowArray[row][col][1])  
  
 *def* getRow(*self*, row):  
 *"""Վերադարձնել հարցվող տողը"""  
 if self*.scalar == 1:  
 *return self*.rowArray[row]  
  
 returnRow = [(*self*.scalar \* elem[0], elem[1])  
 *for* elem *in self*.rowArray[row]]  
 *return* returnRow  
  
 *def* genColArray(*self*, col):  
 *"""Ստեղծել հարցվող սյունակը և այն պահել self.colArray-ում"""  
 for* i *in* my\_range(*self*.rowAmount):  
 *self*.colArray[col].append(*self*.rowArray[i][col])  
 *return self*.colArray[col]  
  
 *def* getCol(*self*, col):  
 *"""Վերադարձնել հարցվող սյունակը"""  
 if len*(*self*.colArray[col]) == 0:  
 *self*.genColArray(col)  
  
 *if self*.scalar == 1:  
 *return self*.colArray[col]  
  
 returnCol = [(*self*.scalar \* elem[0], elem[1])  
 *for* elem *in self*.colArray[col]]  
 *return* returnCol

### my\_algorithms.py

*import* sys  
  
*# Python 3 համատեղելիություն  
if* sys.version\_info[0] == 3:  
 *# unicode հայտարարված չէ Python 3-ում սակայն այն անհրաժեշտ է ոչ ascii սիմվոլների հետ աշխատելուց Python 2-ում* unicode = *str  
  
  
def* \_\_is\_number(n):  
 *"""Պարզել արգումենտը թիվ է թե ոչ"""  
 return isinstance*(n, *float*) *or isinstance*(n, *int*)  
  
  
*def* \_\_is\_string(s):  
 *"""Պարզել արգումենտը տող է թե ոչ"""  
 return isinstance*(s, *str*) *or isinstance*(s, unicode)  
  
  
*def* my\_abs(n):  
 *"""Հաշվել մոդուլային արժեքը"""  
 if* n < 0:  
 *return* -n  
 *return* n  
  
  
*def* my\_gcd(n, m):  
 *"""Հաշվել ամենամեծ ընդհանուր բաժանարարը n-ի և m-ի օգտագործելով Շտեյնի ալգորիթմը"""* unit = 1  
 *if* m < 0:  
 unit = -1  
 n = my\_abs(n)  
 m = my\_abs(m)  
  
 *if* n == m == 0:  
 *return* 0  
 *if* n == 0:  
 *return* unit \* m  
 *if* m == 0:  
 *return* n  
 *if* n == m:  
 *return* unit \* n  
  
 *if* n < m:  
 n, m = m, n  
  
 *# Exp-ը ամենամեծ երկուսի աստիճանն է որը բաժանվում է և n-ի և m-ի.* exp = 0  
 *# Երբ n-ը և m-ը երկուսն էլ զույգ են, երկուսն էլ բաժանել 2-ի, և մեկով ավելացնել exp-ը.  
 while* (n | m) & 1 == 0:  
 n >>= 1  
 m >>= 1  
 exp += 1  
  
 *while* n & 1 == 0:  
 n >>= 1  
  
 *while* n != 0:  
 *if* n < m:  
 n, m = m, n  
  
 n -= m  
 *while* n != 0 *and* n & 1 == 0:  
 n >>= 1  
  
 *return* unit \* (m << exp)  
  
  
*def* my\_max(x, \*args):  
 *"""Վերադարձնել մաքսիմումը տրված թվերից կամ զանգվածից"""  
  
 # Եթե ոչ դատարկ զանգված է տրված  
 if isinstance*(x, *list*) *and len*(x) > 0:  
 greatest = x[0]  
 i = 0  
 *while* i < *len*(x):  
 *# Եթե զանգվածում առկա են միայն թվեր  
 if not* (\_\_is\_number(x[i])):  
 *raise TypeError*('Սպասվում է ամբողջ թիվ կամ լողացող կետով թիվ')  
 *if* x[i] > x[0]:  
 greatest = x[i]  
 i += 1  
 *return* greatest  
  
 *# Եթե տրված է թիվ  
 if* \_\_is\_number(x):  
 greatest = x  
 i = 0  
  
 *while* i < *len*(args):  
 *if not* \_\_is\_number(args[i]):  
 *raise TypeError*('Սպասվում է ամբողջ թիվ կամ լողացող կետով թիվ')  
 *if* args[i] > x:  
 greatest = args[i]  
 i += 1  
 *return* greatest  
  
 *raise TypeError*(  
 'Սպասվում է ամբողջ թվերի կամ լողացող կետով թվերի զանգված կամ հաջորդականություն')  
  
  
*def* my\_range(x, \*args):  
 *"""Վերադարձնել ամբողջ թվերի զանգված"""  
 # Ներքևի սահման* lower = 0  
 *# Սկզբի սահման* upper = x  
  
 *if len*(args) == 1:  
 lower = x  
 upper = args[0]  
  
 ret = []  
 i = lower  
 *while* i < upper:  
 ret.append(i)  
 i += 1  
  
 *return* ret  
  
  
*def* my\_reversed(x):  
 *"""Շրջել զանվածը"""* ret = []  
 i = *len*(x) - 1  
 *while* i >= 0:  
 ret.append(x[i])  
 i -= 1  
 *return* ret  
  
  
*def* my\_split(s, char):  
 *"""Բսժանել տողը զանգվածի"""  
 if not* \_\_is\_string(s):  
 *raise TypeError*('Expected a string.')  
  
 result = []  
 current\_string = ""  
 *for* i *in* my\_range(*len*(s)):  
 *# i:s-րդ սիմվոլը* c = s[i]  
  
 *if* c == char:  
 result.append(current\_string)  
 current\_string = ""  
 *continue* current\_string += c  
 result.append(current\_string)  
 *return* result  
  
  
*def* my\_strip(s):  
 *"""Ջնջել բացատները տողից"""  
 if not* \_\_is\_string(s):  
 *raise TypeError*('Expected a string.')  
  
 *#i-ն առաջին ոչ բացատ սիմվոլի ինդեքսն է* i = 0  
 *while* i < *len*(s) *and* s[i] == ' ':  
 i += 1  
  
 *# j-ն վերջին ոչ բացատ սիմվոլի ինդեքսն է* j = *len*(s) - 1  
 *while* j >= 0 *and* s[j] == ' ':  
 j -= 1  
  
 *# արդյունքը կլինի տեղի այս մասը s[i] + ... + s[j].* result = ""  
 *for* char *in* my\_range(i, j+1):  
 result += s[char]  
  
 *return* result  
  
  
*def* my\_lower(s):  
  
 *if not* \_\_is\_string(s):  
 *raise TypeError*('Expected a string.')  
  
 uppers = u"ABCDEFGHIJKLMNOPQRSTUVWXYZ"  
 lowers = u"abcdefghijklmnopqrstuvwxyz"  
  
 result = ""  
 *for* i *in* my\_range(*len*(s)):  
 char = s[i]  
 *for* j *in* my\_range(*len*(uppers)):  
 *# Եթե մեծատառ է դարձնել փոքրատառ  
 if* char == uppers[j]:  
 char = lowers[j]  
 result += char  
  
 *return* result

### calculator.py

*from* .Matrix *import* Matrix  
*from* .my\_algorithms *import* my\_gcd, my\_abs, my\_range, my\_reversed  
  
  
*def* frac\_add(frac\_a, frac\_b):  
 *"""Վերադարձնել բաղադրիչների գումարը"""  
 if* frac\_a[1] == frac\_b[1]:  
 *return* (frac\_a[0] + frac\_b[0], frac\_a[1])  
  
 numerator = frac\_a[0] \* frac\_b[1] + frac\_b[0] \* frac\_a[1]  
 denominator = frac\_a[1] \* frac\_b[1]  
 *return* (numerator, denominator)  
  
  
*def* frac\_sub(frac\_a, frac\_b):  
 *"""Վերադարձնել բաղադրիչների տարբերությունը"""  
 if* frac\_a[1] == frac\_b[1]:  
 *return* (frac\_a[0] - frac\_b[0], frac\_a[1])  
  
 numerator = frac\_a[0] \* frac\_b[1] - frac\_b[0] \* frac\_a[1]  
 denominator = frac\_a[1] \* frac\_b[1]  
 *return* (numerator, denominator)  
  
  
*def* frac\_mult(frac\_a, frac\_b):  
 *"""Վերադարձնել բաղադրիչների արտադրյալը"""* numerator = frac\_a[0] \* frac\_b[0]  
 denominator = frac\_a[1] \* frac\_b[1]  
 *return* (numerator, denominator)  
  
  
*def* frac\_div(frac\_a, frac\_b):  
 *"""Վերադարձնել frac\_a բաժանած frac\_b-ի"""* numerator = frac\_a[0] \* frac\_b[1]  
 denominator = frac\_a[1] \* frac\_b[0]  
 *return* (numerator, denominator)  
  
  
*def* frac\_abs(frac):  
 *"""Վերադարձնել բաղադրիչների մոդուլով արժեքը"""  
 return* (my\_abs(frac[0]), my\_abs(frac[1]))  
  
  
*def* frac\_ge(frac\_a, frac\_b):  
 *"""Համեմատել բաղադրիչները"""  
 return* frac\_a[0] \* frac\_b[1] > frac\_b[0] \* frac\_a[1]  
  
  
*def* frac\_reduc(frac):  
 *"""գտնել ամենամեջ ընդհանուր բաժանարարը"""* syt = my\_gcd(frac[0], frac[1])  
  
 *# Խուսափում ենք վերջում 0/0-ից  
 # gcd(0, 0) == 0.  
 if* syt == 0:  
 syt = 1  
  
 *return* (*int*(frac[0] / syt), *int*(frac[1] // syt))  
  
  
*def* matrixAddition(A, B):  
 *"""Գումարել մատրիցները եթե գումարման պայմանը տեղի ունի"""  
  
 if not* A *or not* B:  
 *return None  
 # Գումարումը սահմանված չէ  
 if* A.getColAmount() != B.getColAmount():  
 *return None  
 if* A.getRowAmount() != B.getRowAmount():  
 *return None* C = []  
 *for* rowIndex *in* my\_range(A.getRowAmount()):  
 resultRow = []  
 *for* colIndex *in* my\_range(A.getColAmount()):  
 cellOfA = A.getCell(rowIndex, colIndex)  
 cellOfB = B.getCell(rowIndex, colIndex)  
 result = frac\_add(cellOfA, cellOfB)  
 resultRow.append(frac\_reduc(result))  
 C.append(resultRow)  
  
 *return* Matrix(C, A.getRowAmount(), A.getColAmount())  
  
  
*def* matrixSubstraction(A, B):  
 *"""Հանել մատրիցները եթե պայմանները տեղի ունեն."""  
  
 if not* A *or not* B:  
 *return None  
  
 if* A == B:  
 *# Վերադարձնել 0-ական մատրից այս դեպքում  
 return* Matrix(  
 [[(0, 1) *for* i *in* my\_range(A.getColAmount())]  
 *for* j *in* my\_range(A.getRowAmount())],  
 A.getRowAmount(),  
 A.getColAmount())  
  
 *# Բազմապատկել -1 ով քանի որ A-B == A+(-1\*B).* B.multiplyScalar(-1)  
 resultMatrix = matrixAddition(A, B)  
  
 *# Շտկել B արժեքը* B.multiplyScalar(-1)  
  
 *return* resultMatrix  
  
  
*def* matrixScalarMultiplication(A, scalar):  
 *"""Բազմապատկել սկալյարով"""* A.multiplyScalar(scalar)  
 *return* A  
  
  
*def* matrixMultiplication(A, B):  
 *"""Բազմապատկել մատրիցները եթե բազմապատկման պայմանը տեղի ունի"""  
  
 if not* A *or not* B:  
 *return None  
 # Բազմապատկում սահմանված չէ  
 if* A.getColAmount() != B.getRowAmount():  
 *return None* n = A.getRowAmount()  
 m = A.getColAmount()  
 p = B.getColAmount()  
  
 C = [[(0, 1) *for* i *in* my\_range(p)] *for* j *in* my\_range(n)]  
  
 *for* i *in* my\_range(n):  
 *for* j *in* my\_range(p):  
 cellValue = (0, 1)  
  
 *for* k *in* my\_range(m):  
 toAdd = frac\_mult(A.getCell(i, k), B.getCell(k, j))  
 cellValue = frac\_add(cellValue, toAdd)  
  
 cellValue = frac\_reduc(cellValue)  
 C[i][j] = cellValue  
  
 *return* Matrix(C, n, p)  
  
  
*def* matrixTranspose(A):  
 *"""Հաշվել տրանսպոզիցիան"""* n = A.getRowAmount()  
 m = A.getColAmount()  
 result = [[A.getCell(j, i) *for* j *in* my\_range(n)] *for* i *in* my\_range(m)]  
 *return* Matrix(result, m, n)  
  
  
*def* \_\_pivot(A):  
 *"""Պտտել A-ն, որպեսզի յուրաքանչյուր սյունակի ամենամեծ տարրը գտնվի անկյունագծում:  
  
 Ավելի կոնկրետ ՝ A- ն այն ձևափոխված է,  
 որպեսզի յուրաքանչյուր անկյունային տարր ունենա այնպիսի արժեք,  
 որն ունի առնվազն նույնքան մեծ բացարձակ արժեք, որքան դրա ներքևի յուրաքանչյուր բջիջ:"""* n = A.getRowAmount()  
  
 *# P-ն նույնականացման մատրից է* P = [[(*int*(i == j), 1) *for* i *in* my\_range(n)] *for* j *in* my\_range(n)]  
  
 totalPivots = 0  
  
 *for* j *in* my\_range(n):  
 greatest = (0, 1)  
 swapWith = j  
  
 *for* row *in* my\_range(j+1, n):  
 *if* frac\_ge(frac\_abs(A.getCell(row, j)), greatest):  
 greatest = frac\_abs(A.getCell(row, j))  
 swapWith = row  
  
 *if* swapWith != j:  
 P[j], P[swapWith] = P[swapWith], P[j]  
 totalPivots += 1  
  
 *return* (Matrix(P, n, n), totalPivots)  
  
  
*def* \_\_LUP\_decomposition(A):  
 *"""Հաշվել LUP դեկոմպոզիցիան"""* n = A.getRowAmount()  
  
 *# Ձևափոխել L և U ինչպես զրոյական մատրից* L = [[(0, 1) *for* i *in* my\_range(n)] *for* j *in* my\_range(n)]  
 U = [[(0, 1) *for* i *in* my\_range(n)] *for* j *in* my\_range(n)]  
  
 pivot\_info = \_\_pivot(A)  
 P = pivot\_info[0]  
  
 mult = (-1) \*\* pivot\_info[1]  
  
 Prod = matrixMultiplication(P, A)  
  
 *# Օգտագործելով հետևյալ ալգորիթմը  
 # https://rosettacode.org/wiki/LU\_decomposition հաշվել բջիջների արժեքները L-ի և U-ի համար  
  
 for* j *in* my\_range(n):  
 *# Calculate U[i][j].  
 for* i *in* my\_range(j+1):  
 the\_sum = (0, 1)  
  
 *for* k *in* my\_range(i):  
 toAdd = frac\_mult(U[k][j], L[i][k])  
 the\_sum = frac\_add(the\_sum, toAdd)  
  
 the\_sum = frac\_reduc(the\_sum)  
 *# U\_ij == Prod\_ij - the\_sum* U[i][j] = frac\_sub(Prod.getCell(i, j), the\_sum)  
  
 *# Calculate L[i][j].  
 for* i *in* my\_range(j, n):  
 the\_sum = (0, 1)  
  
 *for* k *in* my\_range(j):  
 toAdd = frac\_mult(L[i][k], U[k][j])  
 the\_sum = frac\_add(the\_sum, toAdd)  
  
 the\_sum = frac\_reduc(the\_sum)  
 L[i][j] = frac\_sub(Prod.getCell(i, j), the\_sum)  
 L[i][j] = frac\_div(L[i][j], U[j][j])  
  
 L = Matrix(L, n, n)  
 U = Matrix(U, n, n)  
  
 *return* (L, U, P, mult)  
  
  
*def* matrixDeterminant(A):  
 *"""Հաշվել դետերմինանտը"""  
  
 # Դետերմինանտը սահմանված չէ ոչ քառակուսի մատրիցների համար  
 if* A.getRowAmount() != A.getColAmount():  
 *return None  
  
 # Կազմալուծել մատրիցը* decomposition = \_\_LUP\_decomposition(A)  
  
 U = decomposition[1]  
  
 *# Դետերմինանտը U-ի անկյունագծային արժեքների արտադրյալն է* ans = (1, 1)  
 *for* i *in* my\_range(U.getRowAmount()):  
 ans = frac\_mult(ans, U.getCell(i, i))  
 ans = frac\_reduc(ans)  
  
 *if* ans[1] == 0:  
 *return* 0  
  
 det\_of\_P = decomposition[3]  
 det = ans[0] \* det\_of\_P \* 1.0 / ans[1]  
 *if* det // 1 == det:  
 *return int*(det)  
 *return* det  
  
  
*def* \_\_forward\_substitution(L):  
 *"""Հակադարձել L-ը առաջ փոխարինմամբ  
  
 http://en.wikipedia.org/wiki/Triangular\_matrix#Forward\_and\_back\_substitution  
 """* m = L.getRowAmount()  
 inverse = [[(0, 1) *for* i *in* my\_range(m)] *for* j *in* my\_range(m)]  
  
 *for* a *in* my\_range(m):  
 bVector = [(0, 1) *for* i *in* my\_range(m)]  
 bVector[a] = (1, 1)  
  
 xVector = []  
 *for* x *in* my\_range(m):  
 the\_sum = (0, 1)  
  
 *for* i *in* my\_range(x):  
 toAdd = frac\_mult(L.getCell(x, i), xVector[i])  
 the\_sum = frac\_add(the\_sum, toAdd)  
  
 the\_sum = frac\_reduc(the\_sum)  
 value = frac\_sub(bVector[x], the\_sum)  
 value = frac\_mult(value, L.getCell(x, x))  
 value = frac\_reduc(value)  
 xVector.append(value)  
  
 *for* i *in* my\_range(m):  
 inverse[i][a] = xVector[i]  
  
 *return* Matrix(inverse, m, m)  
  
  
*def* \_\_backward\_substitution(U):  
 *"""Հակադարձել U-ն հետ փոխարինմամբ"""* m = U.getRowAmount()  
 inverse = [[(0, 1) *for* i *in* my\_range(m)]  
 *for* j *in* my\_range(m)]  
  
 *for* a *in* my\_range(m):  
 bVector = [(0, 1) *for* i *in* my\_range(m)]  
 bVector[a] = (1, 1)  
  
 xVector = []  
 *for* x *in* my\_reversed(my\_range(m)):  
 the\_sum = (0, 1)  
  
 *for* i *in* my\_reversed(my\_range(x+1, m)):  
 toAdd = frac\_mult(U.getCell(x, i), xVector[m-1 - i])  
 the\_sum = frac\_add(the\_sum, toAdd)  
  
 the\_sum = frac\_reduc(the\_sum)  
 value = frac\_sub(bVector[x], the\_sum)  
 value = frac\_div(value, U.getCell(x, x))  
 value = frac\_reduc(value)  
 xVector.append(value)  
  
 *for* i *in* my\_range(m):  
 inverse[m-1-i][a] = xVector[i]  
  
 *return* Matrix(inverse, m, m)  
  
  
*def* matrixInverse(A):  
 *"""Հակադարձել A մատրիցը"""  
 # եթե մատրիցը 0-ական է  
 if* matrixDeterminant(A) == 0:  
 *return None  
  
 # հաշվել A-ի LUP դեկոմպոզիցիան of A. PA = LU* decomposition = \_\_LUP\_decomposition(A)  
  
 *# Հակադարձել L-ը առաջ փոխարինմամբ* L\_inv = \_\_forward\_substitution(decomposition[0])  
 *# Հակադարձել U-ն հետ փոխարինմամբ* U\_inv = \_\_backward\_substitution(decomposition[1])  
  
 P = decomposition[2]  
  
 *# PA = LU  
 # -> (PA)^-1 = (LU)^-1  
 # -> A^-1 \* P^-1 = U^-1 \* L^-1  
 # -> A^-1 = U^-1 \* L^-1 \* P* C = matrixMultiplication(U\_inv, L\_inv)  
 *return* matrixMultiplication(C, P)

# Օգտագործված գրականություն և այլ աղբյուրներ

Այս նյութը կազմելու համար օգտվեցի՝

Դ․ Փուլ – Գծային հանրահաշիվ, Ա․Գ․ Կուրոշ – Բարձրագույն հանրահաշվի դասընթաց, Ռ․Ն․ Տոնոյան – Գործույթների հետազոտման մաթեմատիկական խնդիրներ