1. Ներածություն

Աշխատանքը նվիրված է քառակուսային ցանցում կմախքային ծառերի և ցիկլերի քանակի հետազոտմանը: Աշխատանքը բաղկացած է ներածությունից, 4 մասերից և գրականությունից, պարունակում է 24 նկար: Երկրորդ մասը բաղկացած է երկու ենթամասից: Ենթամասերում տրվում են քառակուսայինց ցանցի սահմանումը և նրա պարզագույն հատկությունները: Երրորդ մասը բաղկացած է չորս ենթամասից: Առաջին ենթամասում տրվում են հետևյալ պնդումները ցանցի կմախքային ծառերի քանակի համար.

$$s_2(n) = \frac{\sqrt{3}}{6}\left((2+\sqrt{3})^n-(2-\sqrt{3})^n\right):$$

$$s_3(n) \approx 0.09759\left(12.543754^n+0.079721^n-1.830109^n-0.546416^n\right):$$

$$s_m(n) < \frac{e}{nm}\,4^{nm-1}$$

Երկրորդ ենթամասում հետազոտվում է ցանցի համիլտոնյան ցիկլերի մետրիկական բնութագրերը: Երկու ցիկլերի տարբերվող կողերի քանակը հանդիսանում է հեռավորություն տվյալ ցանցի համիլտոնյան ցիկլերի բազմության մեջ: Ցանցի համիլտոնյան ցիկլերի մինիմալ հեռավորության համար տրվում են հետևյալ պնդումները.

$$d_{min}(3, 2n) = 6(n-1),\ d_{min}(5, 4n+4) = 6n+10,\ d_{min}(5, 4n+6) = 6n+14,$$

$$d_{min}(4m, 4n) = 4mn+4,\ d_{min}(4m, 4n+1) = 4mn+2m+4:$$

$$d_{min}(4m, 4n+2) = 4mn+3m+4,\ d_{min}(4m, 4n+3) = 4mn+5m+4:$$

Երրորդ ենթամասում տրվում են հետևյալ պնդումները ցանցի համիլտոնյան ցիկլերի քանակի համար.

$$h_3(n) = 2^{n/2-1},\ h_4(n) \approx 0.346124(2.538616)^{n-1}-0.148311(-1.276209)^{n-1}:$$

$$h_5(n) \approx 0.115148(11.01648)^{n-1}: \quad nm\, h_m(n) < s_m(n) < h_{2m}(2n):$$

Չորրորդ ենթամասում հետազոտվում է ցիկլերի սիմետրիկությունը: Տրվում են լեմաններ ցանցում հորիզոնական, ուղղահայաց, կենտրոնական սիմետրիկ և ուղղահայաց ցիկլերի գոյության համար: n × 2 ցանցի ոչ սիմետրիկ ցիկլերի համար.

$$\hat{h}_3(n) = \begin{cases} (2^{n/2-1}+2^{[n/4]+1})/4, & \text{երբ } 3n/2 \text{ կենտ է,} \\ (2^{n/2-1}+2^{[n/4]})/4, & \text{հ. դ.:} \end{cases}$$

Չորրորդ մասում տրվում է ստացված արդյունքների կիրառությունները: Ցանցում համիլտոնյան ցիկլերի քանակը լայն կիրառում ունի ֆիզիկայի պոլիմերներում:

Հինգերրորդ մասը բաղկացած է երեք ենթամասից: Ենթամասերում տրվում են համապատասխանաբար՝ ալգորիթմներ կմախքային ծառերի և համիլտոնյան ցիկլերի հաշվման համար, ծրագրով ստացված մաքսիմալ արդյունքներ և ծրագիր:

Աշխատանքի ընթացքում օգտագործված են գրքեր, հոդվածներ և ինտերնետային սայթեր, որոնք ներկայացված են գրականությունում:

2. Քառակուսային ցանց, հիմնական սահմանումներ և հատկություններ

2.1. *Համիլտոնյան, կիսահամիլտոնյան գրաֆ, կմախքային ծառ, արտադրյալ գրաֆ*

Դիտարկենք այս աշխատանքում օգտագործվող գրաֆների տեսության հիմնական սահմանումները։ Դիցուք $G = (V, E)$ գրաֆ է, որտեղ V - ն գրաֆի գագաթների բազմությունն է, իսկ E - ն՝ կողերի։ Այդ գրաֆը կնշանակենք նաև $G = (V(G), E(G))$ [2, 6, 9, 10]։

G գրաֆում կդիտարկենք համիլտոնյան շղթաներ, համիլտոնյան ցիկլեր և կմախքային ծառեր։

Դիցուք G - ն կապակցված գրաֆ է։ Այդ գրաֆում համիլտոնյան շղթա է հանդիսանում այն պարզ շղթան, որն անցնում է գրաֆի յուրաքանչյուր գագաթով և համիլտոնյան ցիկլ է կոչվում այն պարզ ցիկլը, որն անցնում է գրաֆի յուրաքանչյուր գագաթով։ Գրաֆը կոչվում է համիլտոնյան, եթե նա պարունակում է համիլտոնյան ցիկլ և՝ կիսահամիլտոնյան, եթե նա համիլտոնյան ցիկլ չի պարունակում, բայց պարունակում է համիլտոնյան շղթա։

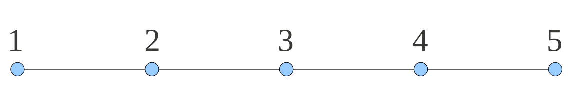
G գրաֆի ենթագրաֆը կոչվում է կմախքային, եթե նա պարունակում է G գրաֆի բոլոր գագաթները։ Պարզ է, որ համիլտոնյան շղթան և համիլտոնյան ցիկլը ներկայացնում են իրենցից կմախքային ենթագրաֆներ։

G գրաֆի այն կմախքային ենթագրաֆը, որը հանդիսանում է ծառ, կոչվում է կմախքային ծառ։

P_n - ով նշանակենք n գագաթ պարունակող պարզ շղթան։

Դիտարկենք $P_n = (V, E)$ պարզ շղթան, որտեղ $V = \{v_1, v_2, \dots, v_n\}$, իսկ $E = \{ \{\boldsymbol{v_i}, \boldsymbol{v_{i+1}}\} \,/\, v_i \in V,\ i = 1, 2, \dots, n\text{-}1 \}$։ Պարզ շղթան գրաֆ է, որի բոլոր գագաթները և կողերը կարելի է պատկերել մի ուղղի վրա։ Հետագայում հենց այդպես էլ կանենք. P_n - ի v_i-րդ գագաթը նշանակելու ենք i-ով և համապատասխանեցնելու ենք ուղղի վրա հավասարահեռ դասավորված n կետերից i - րդը։ Իսկ $x = \{v_i, v_{i+1}\}$ կողը պատկերելու ենք $i,i+1$ հատվածով։ Այսպես, օրինակ, նկ. 2.1 - ում պատկերված է P_5 պարզ շղթան.

P_5

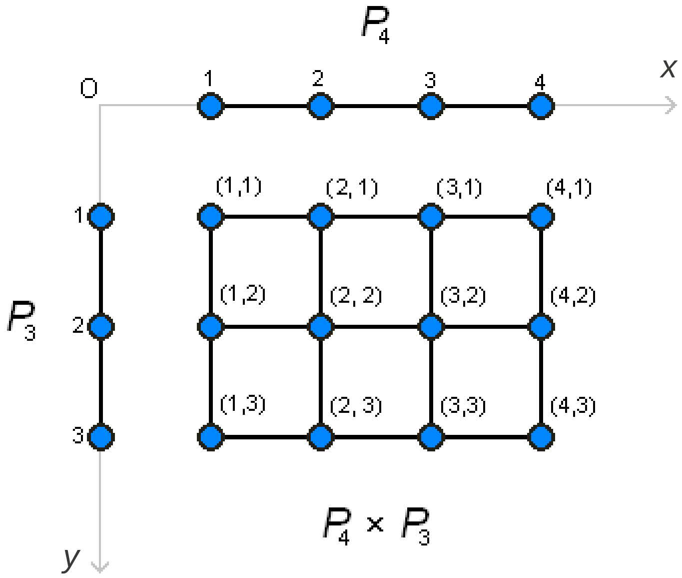


Նկ. 2.1

Դիցուք ունենք P_n և P_m պարզ շղթաները: Դիտարկենք $T_{n,m} = P_n \times P_m$ արտադրյալ գրաֆը: $T_{n,m}$ գրաֆի գագաթների բազմությունը հանդիսանում է $V(P_n)$ և $V(P_m)$ բազմությունների դեկարտյան արտադրյալը, այսինքն այդ գրաֆի գագաթները (ν, μ) կարգավոր զույգեր են, որտեղ ν - ն առաջին բաղադրիչի գագաթն է, μ - ն` երկրորդ: (ν, μ) և (ν', μ') գագաթները կից են այն և միայն այն դեպքում, երբ

$$| \nu - \nu' | + | \mu - \mu' | = 1;$$

$\nu, \nu' = 1,2, ..., n, \mu, \mu' = 1,2, ..., m$: (Նկ. 2.2):



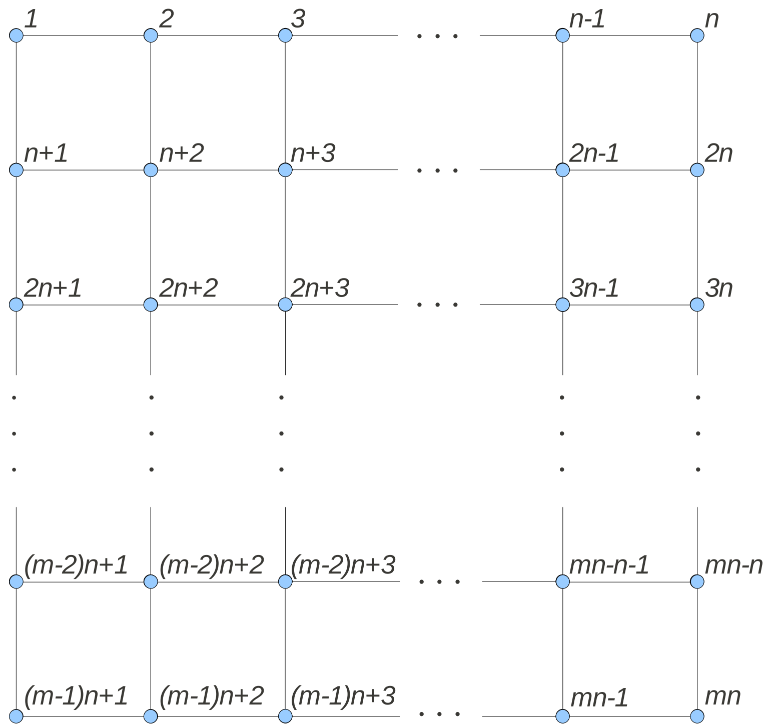
Նկ. 2.2

2.2. *Ցանց, տրման եղանակները և պարզագույն հատկությունները*

$T_{n,m} = P_n \times P_m$ արտադրյալ գրաֆը կանվանենք $n \times m$ *քառակուսային* ցանց: Ակնհայտ է, որ քառակուսային ցանցի գագաթների քանակը nm է, իսկ կողերինը` $n(m - 1) + m(n - 1)= 2nm - n - m$: $T_{n,m}$ ցանցը պատկերելու ենք հարթության վրա քառակուսային ցանցի միջոցով: Համարակալենք ցանցի գագաթները հետևյալ եղանակով: (ν, μ) գագաթի համարն է $\varphi(\nu, \mu) = (\mu - 1)n + \nu$,

$\nu = 1,2, ..., n, \mu = 1,2, ..., m$:

(Նկ. 2.3):



Նկ. 2.3

m թիվը կանվանենք ցանցի *լայնություն*, իսկ n - ը` *երկարություն*: $T_{n,m}$ ցանցը կանվանենք *քառակուսի*, եթե $m = n$:

Որոշ դեպքերում հարմար է ցանցերը նկարագրել *կցությունների մատրիցի* միջոցով։

$$\alpha_{i,j} = \begin{cases} 1, \text{ եթե } |\,i - j\,| = 1 \text{ կամ } |\,i - j\,| = n \\ 0, \text{ հակառակ դեպքում} \end{cases}$$

i, j = 1, 2, ..., mn: Նկ.2.4 - ում պատկերված է $T_{m,n}$ ցանցի հարևանության մատրիցը։

	1	2	3	...	n-1	n	n+1	n+2	n+3	...	2n-1	2n	2n+1	2n+2	2n+3	...	3n-1	3n	...	mn-n+1	mn-n+2	mn-n+3	...	mn-1	mn
1	0	1	0	...	0	0	1	0	0	...	0	0	0	0	0	...	0	0	...	0	0	0	...	0	0
2	1	0	1	...	0	0	0	1	0	...	0	0	0	0	0	...	0	0	...	0	0	0	...	0	0
3	0	1	0	...	0	0	0	0	1	...	0	0	0	0	0	...	0	0	...	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
n-1	0	0	0	...	0	1	0	0	0	...	1	0	0	0	0	...	0	0	...	0	0	0	...	0	0
n	0	0	0	...	1	0	0	0	0	...	0	1	0	0	0	...	0	0	...	0	0	0	...	0	0
n+1	1	0	0	...	0	0	0	1	0	...	0	0	1	0	0	...	0	0	...	0	0	0	...	0	0
n+2	0	1	0	...	0	0	1	0	1	...	0	0	0	1	0	...	0	0	...	0	0	0	...	0	0
n+3	0	0	1	...	0	0	0	1	0	...	0	0	0	0	1	...	0	0	...	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2n-1	0	0	0	...	1	0	0	0	0	...	0	1	0	0	0	...	1	0	...	0	0	0	...	0	0
2n	0	0	0	...	0	1	0	0	0	...	1	0	0	0	0	...	0	1	...	0	0	0	...	0	0
2n+1	0	0	0	...	0	0	1	0	0	...	0	0	0	1	0	...	0	0	...	0	0	0	...	0	0
2n+2	0	0	0	...	0	0	0	1	0	...	0	0	1	0	1	...	0	0	...	0	0	0	...	0	0
2n+3	0	0	0	...	0	0	0	0	1	...	0	0	0	1	0	...	0	0	...	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3n-1	0	0	0	...	0	0	0	0	0	...	1	0	0	0	0	...	0	1	...	0	0	0	...	0	0
3n	0	0	0	...	0	0	0	0	0	...	0	1	0	0	0	...	1	0	...	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
mn-n+1	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	...	0	0	...	0	1	0	...	0	0
mn-n+2	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	...	0	0	...	1	0	1	...	0	0
mn-n+3	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	...	0	0	...	0	1	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
mn-1	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	...	0	0	...	0	0	0	...	0	1
mn	0	0	0	...	0	0	0	0	0	...	0	0	0	0	0	...	0	0	...	0	0	0	...	1	0

Նկ. 2.4

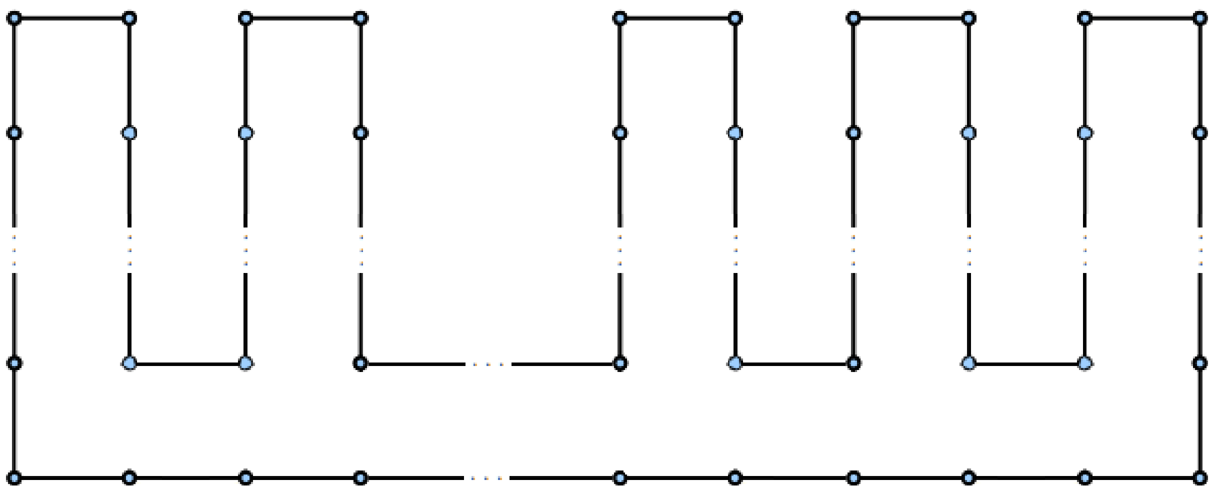
Այսուհետ կդիտարկենք միայն այնպիսի ցանցեր, որոնց համար *n > 1*, *m > 1*, և հաճախ չենք նշի այդ պայմանը: Պարզ է, որ $T_{n,m}$ ցանցում 2 աստիճան ունեցող գագաթները 4 - ն են, կանվանենք *անկյունային գագաթներ,* 3 աստիճան ունեցողները՝ *2*(*n - 2 + m - 2*), կանվանենք եզրային գագաթներ և 4 աստիճան ունեցողները՝ (*n - 2*) (*m - 2*), կանվանենք ներքին գագաթներ:

Դժվար չէ նկատել, որ քառակուսային ցանցը հանդիսանում է երկկողմանի գրաֆ, հետևաբար ցանցի ցանկացած ցիկլի երկարություն կլինի զույգ:

Լեմմա 1: $T_{n,m}$ ցանցը համիլտոնյան է այն և միայն այն դեպքում, երբ նրա գագաթների քանակը զույգ է:

↦ *Անհրաժեշտությունը.* Ենթադրենք ցանցը համիլտոնյան է, հետևաբար ցանցում գոյություն ունի համիլտոնյան ցիկլ: Ցանցի գագաթների քանակը *nm* հատ է, ուստի, և համիլտոնյան ցիկլի երկարությունը ևս կլինի *nm*: Քանի որ ցանցի ցանկացած ցիկլի, այդ թվում և համիլտոնյան ցիկլի երկարությունը զույգ է՝ ցանցի գագաթների քանակը կլինի զույգ:

Բավարարությունը. Ենթադրենք ցանցի գագաթների քանակը զույգ է: Ցույց տանք, որ ցանցը պարունակում է համիլտոնյան ցիկլ: Ցանցի գագաթների քանակը *nm* է, այսինքն *n, m* - ից գոնե մեկը զույգ է: Առանց ընդհանրությունը խախտելու կարող ենք ենթադրել, որ *n* - ն է զույգ: Նկ.2.5 - ում պատկերված է որևիցէ համիլտոնյան ցիկլ: ↵



Նկ. 2.5

3. Ցանցի կմախքային ծառերի և կմախքային ցիկլերի հատկությունները

3.1. Ցանցի կմախքային ծառերի քանակը

Պարզ է, որ ցանցում միշտ գոյություն ունի կմախքային ծառ, քանի որ ցանցը կապակցված է: $T_{n.m}$ ցանցի կմախքային ծառերի քանակը նշանակենք $s_m(n)$ - ով: Ընդհանուր դեպքում ցանկացած գրաֆում, ինչպես նաև ցանցում կմախքային ծառերի քանակը կարող ենք հաշվել ըստ Կիրխհորֆի թեորեմի: Դիտարկենք Կիրխհորֆի մատրիցը.

$$M(G) = B(G) - A(G),$$

որտեղ $A(G)$ - ն գրաֆի կցությունների մատրիցն է: $B(G)$ մատրիցի գլխավոր անկյունագծի վրա գրաֆի գագաթների աստիճաններն են, իսկ մնացած տարրերը 0 են:
Կիրխհորֆի թեորեմ: $M(G)$ մատրիցի բոլոր հանրահաշվական լրացումները իրար հավասար են և նրանց արժեքը $G = (V, E)$ գրաֆի կմախքային ծառերի քանակն է:

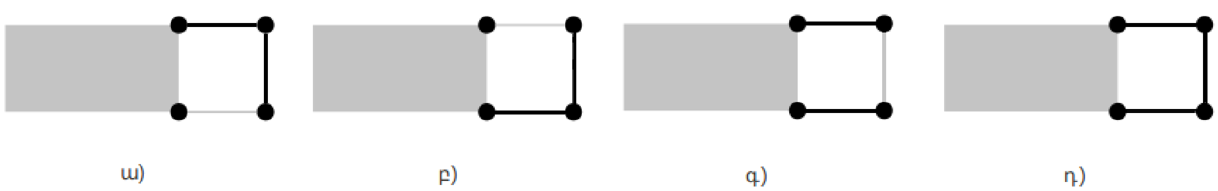
Պարզ է, որ $s_m(n) = s_n(m)$, $s_1(n) = 1$:

Պնդում 3.1: $s_2(n) = 4\, s_2(n-1) - s_2(n-2)$,
$n = 2, 3, \ldots$, որտեղ $s_2(0) = 0$, $s_2(1) = 1$:

↦ $T_{n-1.2}$ - ի կմախքային ծառից $T_{n.2}$ - ի կմախքային ծառ կարող ենք ստանալ *4* եղանակով (Նկ. 3.1) : $T_{n.2}$ - ի ա), բ), գ), դ) տեսքով վերջացող կմախքային ծառերի քանակը նշանակենք համապատասխանաբար $f_1(n)$, $f_2(n)$, $f_3(n)$, $f_4(n)$: Պարզ է, որ.

$$s_2(n) = f_1(n) + f_2(n) + f_3(n) + f_4(n),$$

$$f_1(n) = f_2(n) = f_3(n) = s_2(n - 1):$$



Նկ. 3.1

Եթե $T_{n\text{-}1.2}$ - ի կմախքային ծառը վերջանում է ա), բ), դ) տեսքով, ապա կհեռացնենք նրա աջ եզրային կողը և կավելացնենք դ) տեսքը՝ կստանանք $T_{n.2}$ - ի կմախքային ծառ: Իսկ եթե $T_{n\text{-}1.2}$ - ի կմախքային ծառը վերջանում է գ) տեսքով՝ դ) տեսքով չենք շարունակի կառուցել: Պարզ է, որ նման ձևով կստանանք $T_{n.2}$ - ի բոլոր կմախքային ծառերը, որոնք վերջանում են դ) տեսքով.

$$f_4(n) = f_1(n - 1) + f_2(n - 1) + f_4(n - 1) = s_2(n - 1) - f_3(n - 1) = s_2(n - 1) - s_2(n - 2):$$

$$s_2(n) = f_1(n) + f_2(n) + f_3(n) + f_4(n) = 4\, s_2(n - 1) - s_2(n - 2):$$

Գտնենք $s_2(n)$ - ի ստացված անդրադարձ առնչությանը բավարարող $s_2(n) = \lambda^n$ $(\lambda \neq 0)$ տեսքի հաջորդականություն[], այդ դեպքում.

$$\lambda^n = 4\lambda^{n-1} - \lambda^{n-2},\ \ n = 2, 3, \ldots,$$
$$\lambda^2 - 4\lambda + 1 = 0:$$

Հավասարման լուծումներն են $\lambda_1 = 2 + \sqrt{3}, \lambda_2 = 2 - \sqrt{3},$ հետևաբար $c_1\lambda_1^n + c_2\lambda_2^n, n = 0, 1, 2, \ldots$ հաջորդականությունը ցանկացած c_1 և c_2 թվերի համար բավարարում է անդրադարձ առնչությանը: Ընտրենք այդ թվերն այնպես, որ բավարվեն սկզբնական արժեքները.

$$\begin{cases} c_1 + c_2 = 0, \\ c_1\lambda_1 + c_2\lambda_2 = 1, \end{cases}$$ լուծելով ստանում ենք $c_1 = \frac{\sqrt{3}}{6},\ c_1 = \frac{-\sqrt{3}}{6}:$

Հետևաբար.

$$s_2(n) = \frac{\sqrt{3}}{6}\left((2+\sqrt{3})^n - (2-\sqrt{3})^n\right):$$

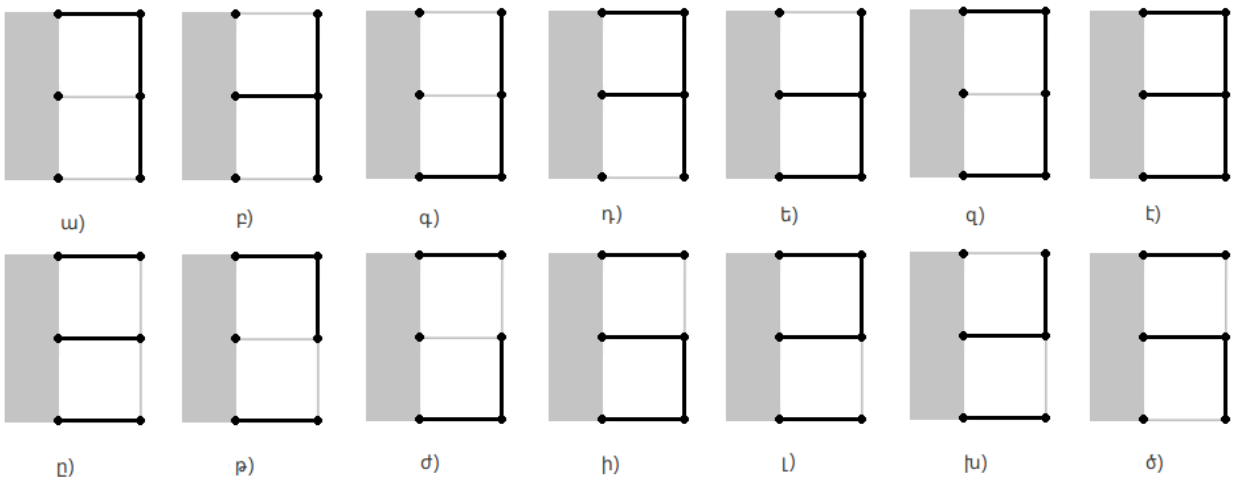
Պնդում 3.2: $s_3(n) = 15\, s_3(n-1) - 32\, s_3(n-2) + 15\, s_3(n-3) - s_3(n-4),$
$n = 4, 5, \ldots$, որտեղ $s_2(0) = 0,\ s_2(1) = 1,\ s_2(2) = 15,\ s_2(3) = 192:$

↦ $T_{n\text{-}1.3}$ - ի կմախքային ծառից $T_{n.3}$ - ի կմախքային ծառ կարող ենք ստանալ *14* եղանակով (Նկ. 3.2): $T_{n.3}$ - ի ա), բ), գ), ..., խ), ծ) տեսքով վերջացող կմախքային ծառերի քանակը նշանակենք համապատասխանաբար $f_1(n), f_2(n), f_3(n), \ldots, f_{13}(n), f_{14}(n)$: Պարզ է, որ.

$$f_1(n) = f_2(n) = f_3(n) = f_8(n) = f_9(n) = f_{10}(n) = f_{13}(n) = f_{14}(n) = s_3(n - 1),$$

$$f_4(n) = f_5(n) = f_{11}(n) = f_{12}(n),$$

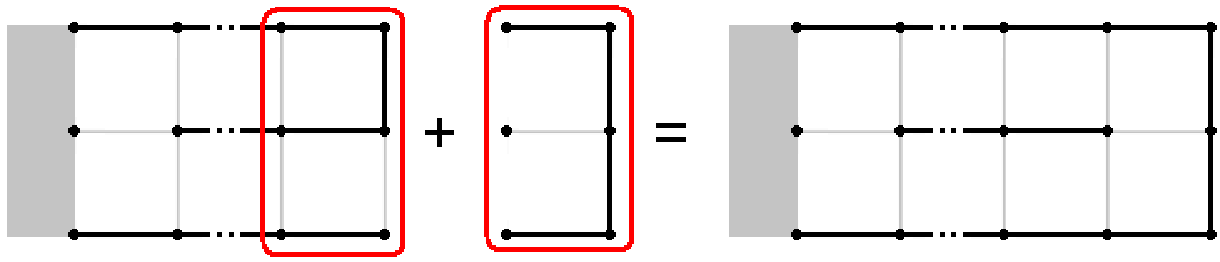
$$s_3(n) = f_1(n) + f_2(n) + f_3(n) + \ldots + f_{13}(n) + f_{14}(n) = 8\, s_3(n - 1) + 4\, f_4(n) + f_6(n) + f_7(n):$$



Նկ. 3.2

$T_{n\text{-}1.3}$ - ի կմախքային ծառին նման տեսքեր ավելացնելիս հնարավոր է, որ $T_{n.3}$ ցանցում առաջանան այնպիսի ցիկլեր, որոնցից գոնե մեկը չի անցնի $T_{n\text{-}1.3}$ - ի աջ եզրային կողերով, այդպիսի դեպքերը հաշվի չենք առնի, իսկ եթե առաջանան ցիկլեր կամ ցիկլ, որոնք անցնում են աջ եզրային կողերով, պարզապես կհեռացնենք այդ կողերը, կարող է լինել *2* դեպք եզրային կողերի հեռացնելիս, որոնք երկուսն էլ հաշվի կառնենք, օր. ա) տիպի շղթային հաջորդի գ) տիպի շղթան: Այն դեպքում, երբ առաջացած ցիկլը բացի աջ եզրային կողից կանցնի $T_{n\text{-}1.3}$ - ի ցանցի այլ կողերով, եզրային կողը հեռացնելիս հնարավոր է լինեն *2* վատ դեպքեր.

1. կրկնվող դեպքեր(օր. ժ)դ) և թ)դ) այդպիսի դեպքերը չենք հաշվի),
2. ստացված պատկերը չլինի ծառ(օր. լ)զ) - ի այն դեպքը, երբ $T_{n\text{-}1.3}$ - ի ցանցի ծառը աջ կողմի վերին և ստորին անկյունային գագաթները կմիացնի առանց աջ եզրային կողերով,նկ.3.3, այդպիսի դեպքերը ևս հաշվի չենք առնի)



Նկ. 3.3

Արդյունքում կստանանք.

$$f_4(n) = f_1(n-1) + f_2(n-1) + f_3(n-1) + f_4(n-1) + f_5(n-1) + f_6(n-1) + f_7(n-1) + f_9(n-1) + f_{12}(n-1) + f_{13}(n-1) = s_4(n-1) - 3\, s_4(n-2) - f_4(n-1),$$

$$f_6(n) = 2\, f_1(n-1) + 2\, f_2(n-1) + 2\, f_3(n-1) + 2\, f_4(n-1) + 2\, f_5(n-1) + 2\, f_6(n-1) + 2\, f_7(n-1) + \left(f_{11}(n-1) - s_4(n-3) - s_4(n-4) - \ldots - s_4(1)\right) + \left(f_{12}(n-1) - s_4(n-3) - s_4(n-4) - \ldots - s_4(1)\right) + f_{13}(n-1) + f_{14}(n-1) = 2\, f_4(n) - 2\left(s_4(n-2) + s_4(n-3) + \ldots + s_4(1)\right),$$

$$f_7(n) = f_1(n-1) + f_2(n-1) + f_3(n-1) + f_4(n-1) + f_5(n-1) + f_6(n-1) + f_7(n-1) = 2\, f_4(n) - s_4(n-1) + s_4(n-2):$$

Տեղադրելով ստացված արդյունքները $s_3(n)$ - ում և պարզեցնելով, կստանանք.

$$8\, f_4(n) = s_3(n) - 7\, s_3(n-1) + s_3(n-2) + 2\left(s_4(n-3) + s_4(n-4) + \ldots + s_4(1)\right):$$

Քանի որ $f_4(n) = s_4(n-1) - 3\, s_4(n-2) - f_4(n-1)$, հետևաբար.

$$s_3(n) = 14\, s_3(n-1) - 18\, s_3(n-2) - 3\, s_3(n-3) - 4\left(s_4(n-4) + s_4(n-5) + \ldots + s_4(1)\right) =$$

$$= 15\, s_3(n-1) - 32\, s_3(n-2) + 15\, s_3(n-3) - s_4(n-4) +$$

$$\left(s_3(n-1) - 14\, s_3(n-2) + 18\, s_3(n-3) + 3\, s_3(n-4) + 4\left(s_4(n-5) + s_4(n-6) + \ldots + s_4(1)\right)\right) =$$

$$= 15\, s_3(n-1) - 32\, s_3(n-2) + 15\, s_3(n-3) - s_4(n-4): ↵$$

Գտնենք $s_3(n)$ - ի ստացված անդրադարձ առնչությանը բավարարող $s_3(n) = \lambda^n$ $(\lambda \neq 0)$ տեսքի հաջորդականություն, այդ դեպքում.

$$\lambda^n = 15\lambda^{n-1} - 32\lambda^{n-2} + 15\lambda^{n-3} - \lambda^{n-4}, \quad n = 1, 2, \ldots,$$
$$\lambda^4 - 15\lambda^3 + 32\lambda^2 - 15\lambda + 1 = 0:$$

Հավասարման լուծումներն են.

$$\lambda_1 \approx 12.543754,$$
$$\lambda_2 \approx 1.830109,$$
$$\lambda_3 \approx 0.546416,$$
$$\lambda_4 \approx 0.079721,$$

հետևաբար $c_1\lambda_1^n + c_2\lambda_2^n + c_3\lambda_3^n + c_4\lambda_4^n$, $n = 1, 2, \ldots,$ հաջորդականությունը ցանկացած c_1, c_2, c_3, c_4 թվերի համար բավարարում է անդրադարձ առնչությանը: Ընտրենք այդ թվերն այնպես, որ բավարվեն սկզբնական արժեքները.

$$\begin{cases} c_1+c_2+c_3+c_4=0, \\ c_1\lambda_1+c_2\lambda_2+c_3\lambda_3+c_4\lambda_4=1, \\ c_1\lambda_1^2+c_2\lambda_2^2+c_3\lambda_3^2+c_4\lambda_4^2=15, \\ c_1\lambda_1^3+c_2\lambda_2^3+c_3\lambda_3^3+c_4\lambda_4^3=192, \end{cases}$$, լուծելով ստանում ենք $$\begin{matrix} c_1\approx\ 0.09759, \\ c_2\approx-0.09759, \\ c_3\approx-0.09759, \\ c_4\approx\ 0.09759: \end{matrix}$$

Հետևաբար.

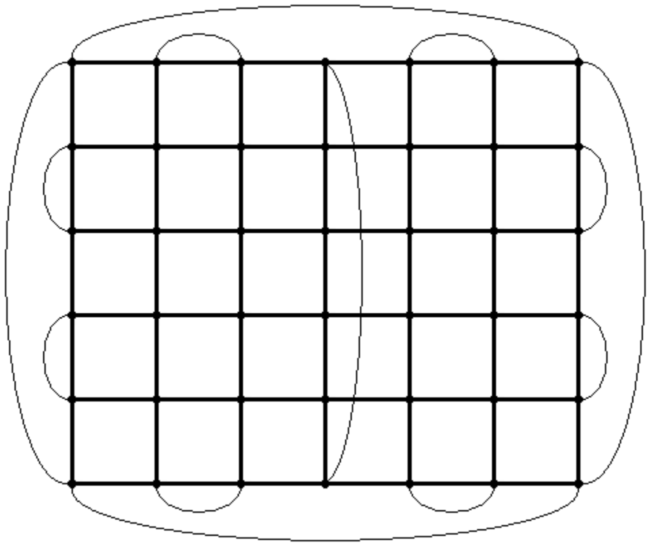
$$s_3(n)\approx\ 0.09759\left(12.543754^n+0.079721^n-1.830109^n-0.546416^n\right):$$

Դիցուք *G* - ն *k* աստիճանի ռեգուլյար գրաֆ է գագաթների *p* քանակով(դա այն գրաֆն է, որի ցանկացած գագաթի ստիճանը *k* է, գրաֆը անվանում են (*k*, *nm*) ռեգուլյար գրաֆ), իսկ s(*G*) - ն նրա կմախքային ծառերի քանակն է։ s(*G*) - ի համար հայտնի է Կելմանս - Բիգսի վերին գնահատականը [12].

$$s(G)\le\frac{1}{p}\left(\frac{pk}{p-1}\right)^{p-1}<\frac{e}{p}k^{p-1}$$

$T_{n.m}$ քառակուսային ցանցից ստանանք *G* = (*4*, *nm*) ռեգուլյար գրաֆ՝ եզրային և անկյունային գագաթների աստիճանները դարձնելով *4,* (Նկ.3.4 - ում ցույց է տրված $T_{7.6}$ ցանցից (*4*, *42*) ռեգուլյար գրաֆ ստանալու եղանակներից մեկը)։ Պարզ է, որ $s_m(n)<s(G)$, հետևաբար.

$$s_m(n)<\frac{e}{nm}4^{nm-1}$$



Նկ. 3.4

3.2. Համիլտոնյան ցիկլերի մետրիկական բնութագրերը

Դիցուք $T_{n,m}$ ցանցը համիլտոնյան է(nm - ը զույգ է)։ $E = \{ e_1, e_2, \dots , e_{2mn-m-1} \}$։ $T_{n,m}$ ցանցի բոլոր համիլտոնյան ցիկլերի բազմությունը նշանակենք $C(T_{n,m})$։ Ենթադրենք C' և C'' ∊ $C(T_{n,m})$, համիլտոնյան ցիկլերին համապատասխանեցնենք ցանցի կողերի բազմություն, որոնցով ցիկլը անցնում է, պարզ է, որ $C_1, C_2 \subseteq E$։

Դիտարկենք C' և C'' բազմությունների $C' \Delta C'' = (C' \cup C'') \setminus (C' \cap C'')$ սիմետրիկ տարբերությունը։ $C'\Delta C''$ բազմության հզորությունը նշանակենք h(C', C''), այսինքն h(C', C'') թիվը ցույց է տալիս թե քանի կողով են նրանք տարբերվում։ Նկատենք, որ h(C', C'') - ը զույգ է, քանի որ ցանցը երկկողմանի է։

Պնդում 3.3: Դիտարկված h(C', C'') *- ը հանդիսանում է հեռավորության.*

- h(C', C'') ≥ 0, և h(C', C'') = 0 ⇔ $C' = C''$,
- h(C', C'') = h(C'', C'),
- h(C', C'') + h(C'', C''') ≥ h(C', C'''), (*եռանկյան անհավասարություն*)

↦ 1-ին և 2-րդ պնդումների ապացույցները պարզ են, հետևում են սահմանումներից։
Ամեն մի ցիկլի համապատասխանեցնենք բուլյան վեկտոր k*=2mn-m-n* երկարությամբ, որտեղ վեկտորի *i*-րդ կոմպոնենտը ցույց է տալիս ցանցի տվյալ կողը կա ցիկլում թե ոչ.

$C' \to \alpha = (\alpha_1, \alpha_2, \dots , \alpha_k)$, որտեղ $\alpha_i = \begin{cases} 1, \text{ եթե } e_i \in C', \ i = 1,2, \dots, k; \\ 0, \text{ հակառակ դեպքում:} \end{cases}$

նման ձևով. $C'' \to \beta = (\beta_1, \beta_2, \dots , \beta_k)$, $C''' \to \gamma = (\gamma_1, \gamma_2, \dots , \gamma_k)$։

Պարզ է, որ $||\alpha|| = ||\beta|| = ||\gamma|| = mn$, $||\alpha\oplus\gamma||$ = h(C', C''), $||\beta\oplus\gamma||$ = h(C'', C'''), $||\alpha\oplus\gamma||$ = h(C', C''')։

h(C', C'') + h(C'', C''') = $||\alpha\oplus\beta|| + ||\beta\oplus\gamma|| = \sum(\alpha_i\oplus\beta_i) + \sum(\beta_i\oplus\gamma_i) = \sum(\alpha_i\oplus\beta_i + \beta_i\oplus\gamma_i)$;
h(C', C''') = $||\alpha\oplus\gamma|| = \sum(\alpha_i\oplus\gamma_i)$;

Դիտարկենք գումարների *i*-րդ բաղադրիչը.

եթե $\beta_i = 0$; $\alpha_i\oplus\beta_i + \beta_i\oplus\gamma_i = \alpha_i + \gamma_i \geq \alpha_i\oplus\gamma_i$;
եթե $\beta_i = 1$; $\alpha_i\oplus\beta_i + \beta_i\oplus\gamma_i = \bar{\alpha}_i + \bar{y}_i \geq \bar{\alpha}_i\oplus\bar{y}_i = \alpha_i\oplus\gamma_i$;

հետևաբար $\sum(\alpha_i\oplus\beta_i + \beta_i\oplus\gamma_i) \geq \sum(\alpha_i\oplus\gamma_i)$:↵

$T_{n,m}$ ցանցի համիլտոնյան ցիկլերի մաքսիմալ և մինիմալ հեռավորությունները նշանակենք համապատասխանաբար $d_{max}(n, m)$ և $d_{min}(n, m)$.

$$d_{max}(n, m) = \max(\ h(C', C'')\ ),$$

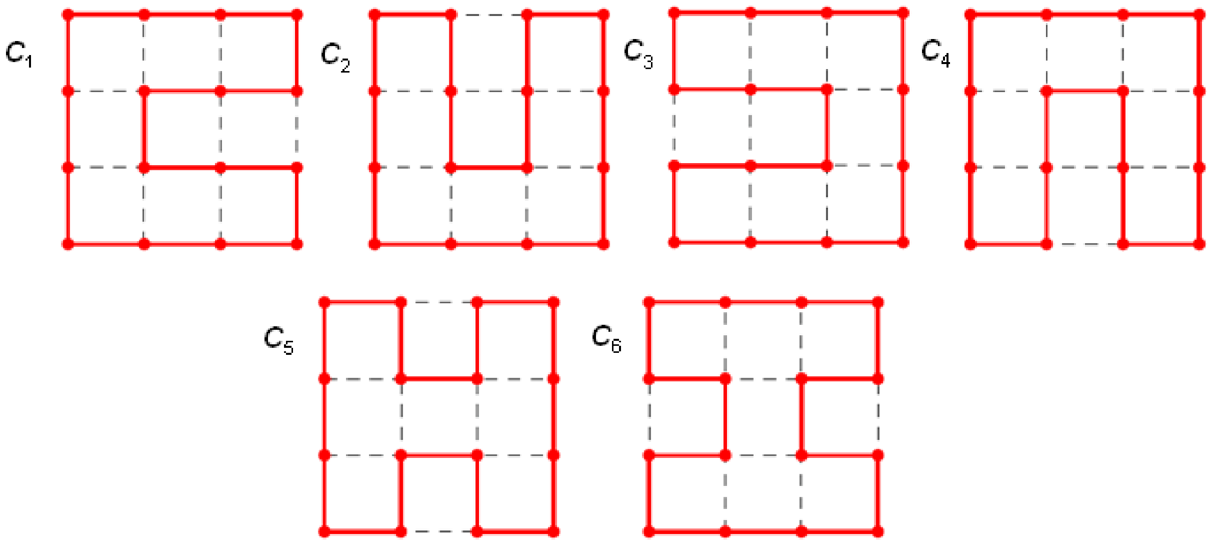
$$d_{min}(n, m) = \min(\ h(C', C'')\ ):$$

C', $C'' \in C(T_{n,m})$: Այսինքն $d_{min}(n, m)$ - ը ցույց է տալիս կողերի մինիմալ քանակը, որքանով որևէ *2* ցիկլեր համնկնում են: Պարզ է, որ

$$2\ d_{min}(n, m) + d_{max}(n, m) = nm:$$

Նկատենք, որ $d_{min}(n, m) \geq 8$, քանի որ ցանկացած համիլտոնյան ցիկլ անցնում է անկյունային գագաթներով:

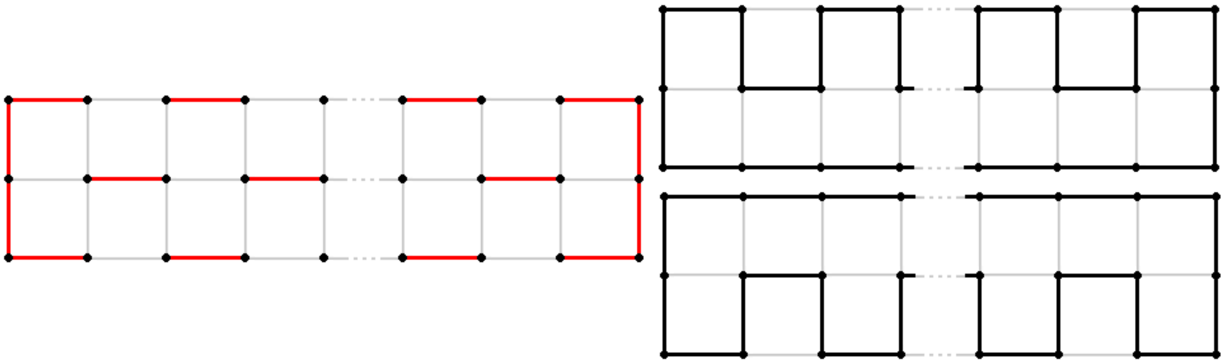
Այսպես, օրինակ նկ. 3.5 - ում պատկեցված են $T_{4,4}$ քառակուսի ցանցի բոլոր 6 համիլտոնյան ցիկլերը: $d_{min}(n, m) = 8$, քանի որ C_5 և C_6 ցիկլերը համնկնում են միայն անկյունային գագաթներին կից կողերով:



Նկ. 3.5

Պնդում 3.4: $d_{min}(3, n) = 3n - 6$, $n = 2k$, $k = 2, 3, \ldots$:

↦ Նկատենք, որ $n \times 3$ քառակուսային ցանցի ցանկացած ցիկլ պարունակում է նկ.3.6 ա)-ում պատկերված կողերը, հետևաբար $d_{min}(3, n) \geq 3n - 6$, մնում է ցույց տալ, որ կան երկու ցիկլեր, որոնք մնացած բոլոր կողերով տարբերվում են, այդպիսի ցիկլերից երկուսը պատկերված նկ.3.6 բ) - ում:↵



Նկ. 3.6

Պնդում 3.5: $d_{min}(4, n) = d_{min}(4, n-1) + d_{min}(4, n-4) - d_{min}(4, n-5)$, $n = 8, 9, 10, ...,$

$d_{min}(4, 3) = 9$, $d_{min}(4, 4) = 8$, $d_{min}(4, 5) = 10$, $d_{min}(4, 6) = 11$, $d_{min}(4, 7) = 13$:

↦ Ունենք, որ(հավելված, ստացված է համակարգչի օգնությամբ),

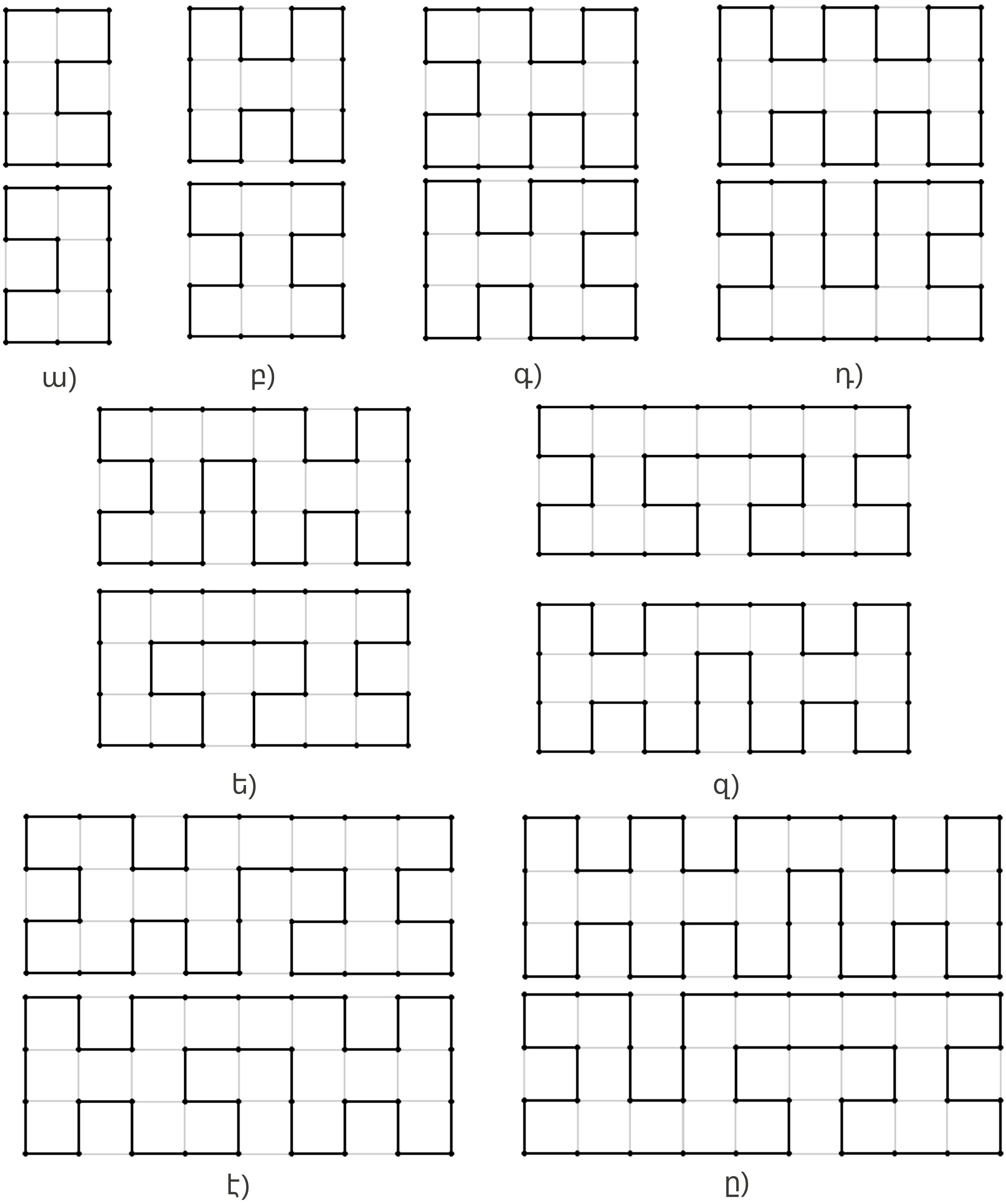
$d_{min}(4, 3) = 9$, $d_{min}(4, 4) = 8$, $d_{min}(4, 5) = 10$, $d_{min}(4, 6) = 11$:

Նկ.3.7 - ի ա), բ), գ), դ), ե), զ), է), ը) - ում պատկերված են համապատասխանաբար 4×3, 4×4, 4×5, 4×6, 4×7, 4×8, 4×9, 4×10 քառակուսային ցանցերում որէիցէ երկու ցիկլեր, որոնց հեռավորությունը մինիմալ է: Նկատենք, որ ե), զ), է), ը) տեսքի ցիկլերը ստացվում են ա), բ), գ), դ) ցիկլերից, նրանց աջից կցելով բ) տեսքի ցիկլեր(հիշենք, որ դրանք 4×4 քառակուսի ցանցի այն ցիկլերն էին, որոնք համնկնում էին միայն անկյունային գագաթներին կից կողերով), այսինքն պրոցեսը կրկնելով և ամեն անգամ ա), բ), գ), դ) տեսքի ցիկլերին $k = 1, 2, ...$ անգամ աջից կցելով բ) տեսքի ցիկլեր, կստանանք համապատասխանաբար $4 \times (4k + 3)$, $4 \times (4k + 4)$, $4 \times (4k + 5)$, $4 \times (4k + 6)$, քառակուսային ցանցի այն ցիկլերը, որոնց հեռավորությունը մինիմալ է. այսինքն.

$$d_{min}(4, 4k + 3) = d_{min}(4, 3) + k\,(d_{min}(4, 4) - 4) = 9 + 4k,$$
$$d_{min}(4, 4k + 4) = d_{min}(4, 4) + k\,(d_{min}(4, 4) - 4) = 8 + 4k,$$
$$d_{min}(4, 4k + 5) = d_{min}(4, 5) + k\,(d_{min}(4, 4) - 4) = 10 + 4k,$$
$$d_{min}(4, 4k + 6) = d_{min}(4, 6) + k\,(d_{min}(4, 4) - 4) = 11 + 4k:$$

$k = 1, 2, ...$: Ընդհանրացնելով ստացված արդյունքները, կարող ենք գրել, որ.

$$d_{min}(4, n) = d_{min}(4, n - 1) + d_{min}(4, n - 4) - d_{min}(4, n - 5),\ n = 8, 9, 10, ..., \hookleftarrow$$



Նկ. *3.7*

Պնդում 3.6: $d_{min}(5, n) = d_{min}(5, n - 2) + d_{min}(5, n - 4) - d_{min}(5, n - 6)$,

$n = 2k$, $k = 5, 6, \ldots$,
$d_{min}(5, 4) = 10$, $d_{min}(5, 6) = 14$, $d_{min}(5, 8) = 16$, $d_{min}(5, 10) = 20$:

↦ Ունենք, որ(հավելված, ստացված է համակարգչի օգնությամբ),

$$d_{min}(5, 4) = 10,\ d_{min}(5, 6) = 14,\ d_{min}(5, 8) = 16,\ d_{min}(5, 10) = 20:$$

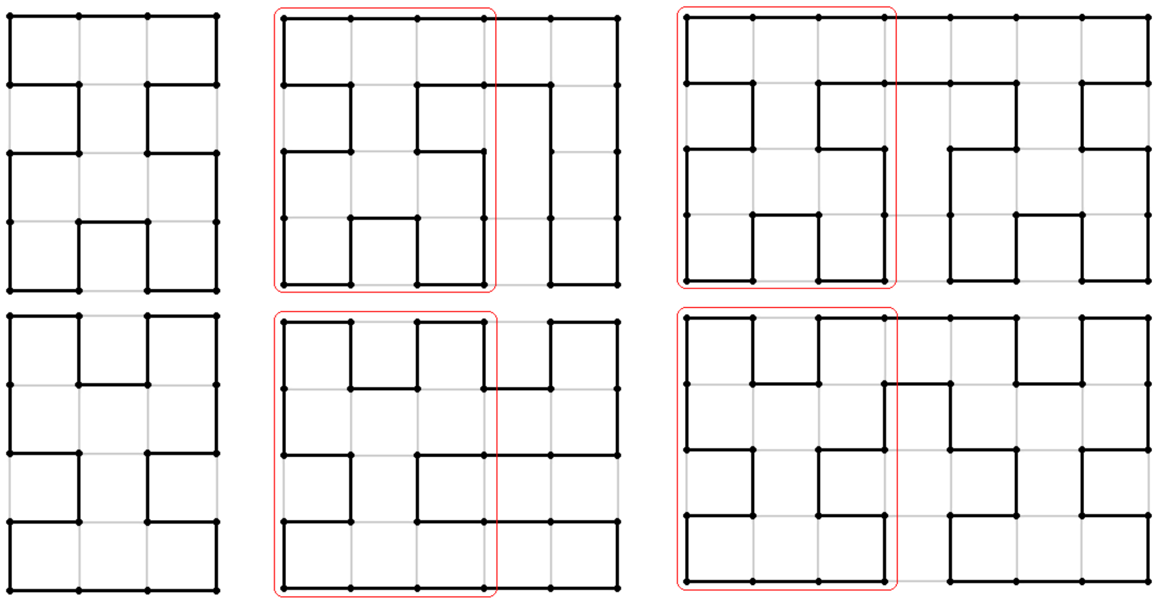
Նկ.3.8 - ի ա), բ), գ) - ում պատկերված են համապատասխանաբար 5×4, 5×6, 5×8 քառակուսային ցանցերում որէիցէ երկու ցիկլեր, որոնց հեռավորությունը մինիմալ է: Նկատենք, որ բ) տեսքի ցիկլը ստացվում է ա) ցիկլերից, նրան աջից միացնելով ինչ որ շղթա, իսկ գ) տեսքի ցիկլերը պարզապես ա) ցիկլերի միացումն է հենց իր տեսքի ցիկլին, այսինքն պրոցեսը կրկնելով և ամեն անգամ ա) տեսքի ցիկլերին $k = 1, 2, \ldots$ անգամ աջից միացնելով ա) կամ բ) տեսքի ցիկլեր կստանանք համապատասխանաբար $5 \times (4k + 4)$, $5 \times (4k + 6)$ քառակուսային ցանցի այն ցիկլերը, որոնց հեռավորությունը մինիմալ է. այսինքն.

$$d_{min}(5, 4k + 4) = d_{min}(5, 4) + k\,(d_{min}(5, 4) - 4) = 10 + 6k,$$

$$d_{min}(5, 4k + 6) = d_{min}(5, 6) + k\,(d_{min}(5, 4) - 4) = 14 + 6k,$$

$k = 1, 2, \ldots$: Ընդհանրացնելով ստացված արդյունքները, կարող ենք գրել, որ.

$$d_{min}(5, n) = d_{min}(5, n - 2) + d_{min}(5, n - 4) - d_{min}(5, n - 6),\ n = 2k,\ k = 5, 6, \ldots,$$ ↵



ա) բ) գ)

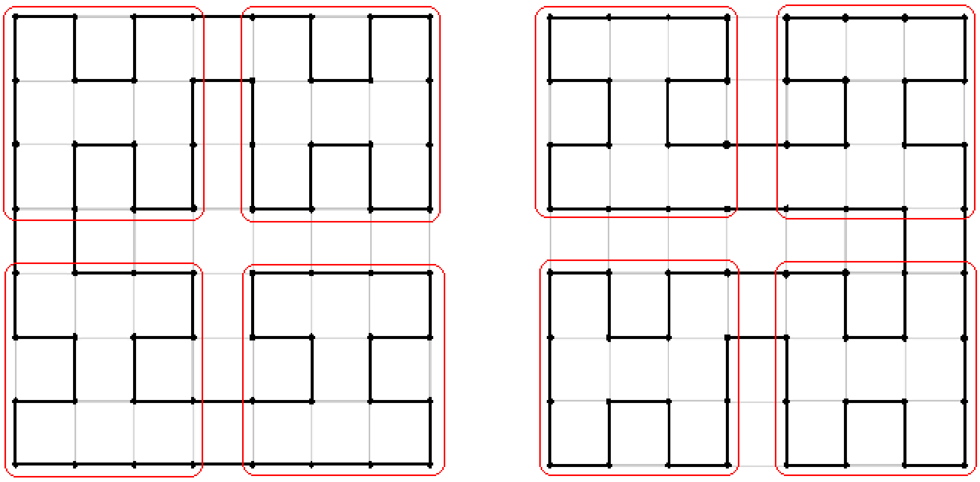
Նկ. 3.8

Պնդում 3.7:

$$d_{min}(4m, n) = md_{min}(4m, n - 1) + md_{min}(4m, n - 4) - md_{min}(4m, n - 5) - 4m + 4;$$

$$n = 8, 9, \ldots, m = 1, 2, \ldots:$$

↦ Պարզ է, որ $d_{min}(4m, n) = md_{min}(m, n) - 4m - 4$(քանի որ $d_{min}(m, n)$ գտնելիս մենք աշխատում էինք հնարավորինս շատ օգտագործել Նկ.3.9 - ի բ) տեսքի ցիկլերից, քանի որ նրանք լավագույնն են և համընկնում են միայն անկյունային գագաթներին կից կողերով: Արդյունքում ստացված ցիկլերը միմյանց միացնելով $4(m - 1)$ կող ևս կտարբերվեն: Նկ.3.9 - ում պատկերված է 8×8 քառակուսային ցանցի այն ցիկլերը, որոնք ստացվում են վերը նկարագրված եղանակով և ունեն մինիմալ հեռավորություն:↵



Նկ. 3.9

Հետևանք:

$$d_{min}(4m, 4n) = 4mn + 4:$$

$$d_{min}(4m, 4n + 1) = 4mn + 2m + 4:$$

$$d_{min}(4m, 4n + 2) = 4mn + 3m + 4 :$$

$$d_{min}(4m, 4n + 3) = 4mn + 5m + 4:$$

↦ Ապացույցը պարզ է, ստացվում է օգտվելով պնդում 3.5 - ում ստացված արդյունքներից և` պնդում 3.7 - ում կառուցման ձևից: ↵

3.3. Ցանցի կմախքային ցիկլերի քանակը

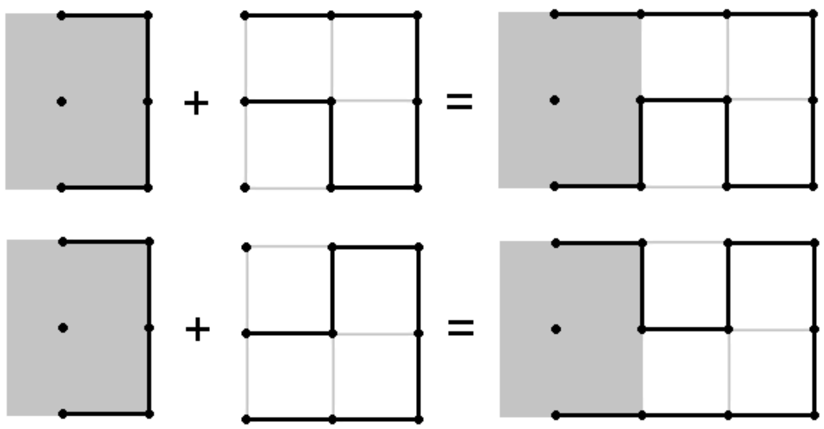
$T_{n,m}$ ցանցի իրարից տարբեր համիլտոնյան ցիկլերի քանակը նշանակենք $h_m(n)$-ով: Պարզ է, որ $h_m(n) = h_n(m)$, $h_1(n) = 0$, $h_2(n) = 1$:

Դիտարկենք $T_{n,3}$ ցանցը, քանի որ $m = 3$ կենտ է, => ցանցը համիլտոնյան է եթե n - ը զույգ է:

Պնդում 3.8: $h_3(n) = 2^{\,n/2-1}$, $n = 2k$, $k = 1, 2, \dots$:

↦ $T_{3.n-2}$ - համիլտոնյան ցիկլից $T_{n,3}$ - ի համիլտոնյան ցիկլ կարելի է ստանալ երկու եղանակով(նկ. 3.10), ուստի.

$$h_3(n) = 2\,h_3(n-2) = 2^{\,n/2-1}\,h_3(2) = 2^{\,n/2-1}: \hookleftarrow$$



Նկ. 3.10

Պնդում 3.9: $h_4(n) = 2\,h_4(n-1) + 2\,h_4(n-2) - 2\,h_4(n-3) + h_4(n-4)$, $n = 3, 4, \dots$,

$h_4(-1) = h_4(0) = h_4(1) = 0$, $h_4(2) = 1$, $h_4(3) = 2$:

↦ $T_{n,4}$ - ի համիլտոնյան ցիկլը կարող է ունենալ նկ. 3.11 ա) - ում պատկերված 5 տեսքերից մեկը, նրանց քանակը նշանակենք համապատասխանաբար $f_1(n)$, $f_2(n)$, $f_3(n)$, $f_4(n)$, $f_5(n)$, պարզ է, որ $h_4(n) = f_1(n) + f_2(n) + f_3(n) + f_4(n) + f_5(n)$: Նկ. 3.11 բ) - ում ցույց է տրված համիլտոնյան ցիկլի կառուցման որոշ եղանակներ(ցիկլի կառուցման կրկնվող եղանակները և պարզ դեպքերը նկարված չեն), արդյունքում կստանանք.

$$f_1(n) = h_4(n - 1),\ f_2(n) = f_3(n) = h_4(n - 2),$$

$$f_4(n) = f_2(n - 2) + f_3(n - 2) + f_4(n - 2) + f_5(n - 2) = h_4(n - 2) - f_1(n - 2) = h_4(n - 2) - h_4(n - 3),$$

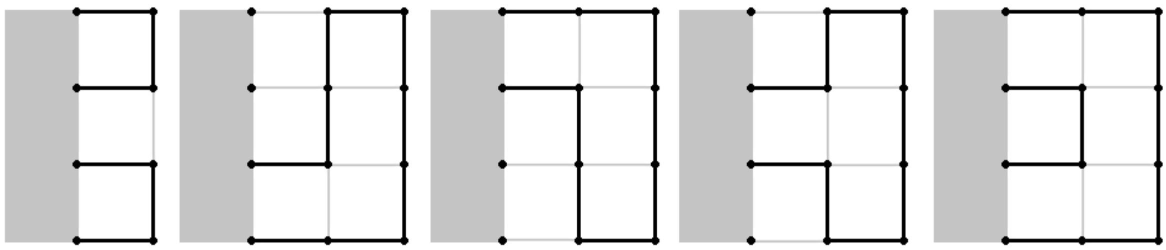
$$f_5(n) = 2\, f_1(n - 2) + f_2(n - 2) + f_3(n - 2) + f_5(n - 2) = h_4(n - 2) + f_1(n - 2) - f_4(n - 2) =$$

$$= h_4(n - 2) + h_4(n - 3) - h_4(n - 4) + h_4(n - 5):$$

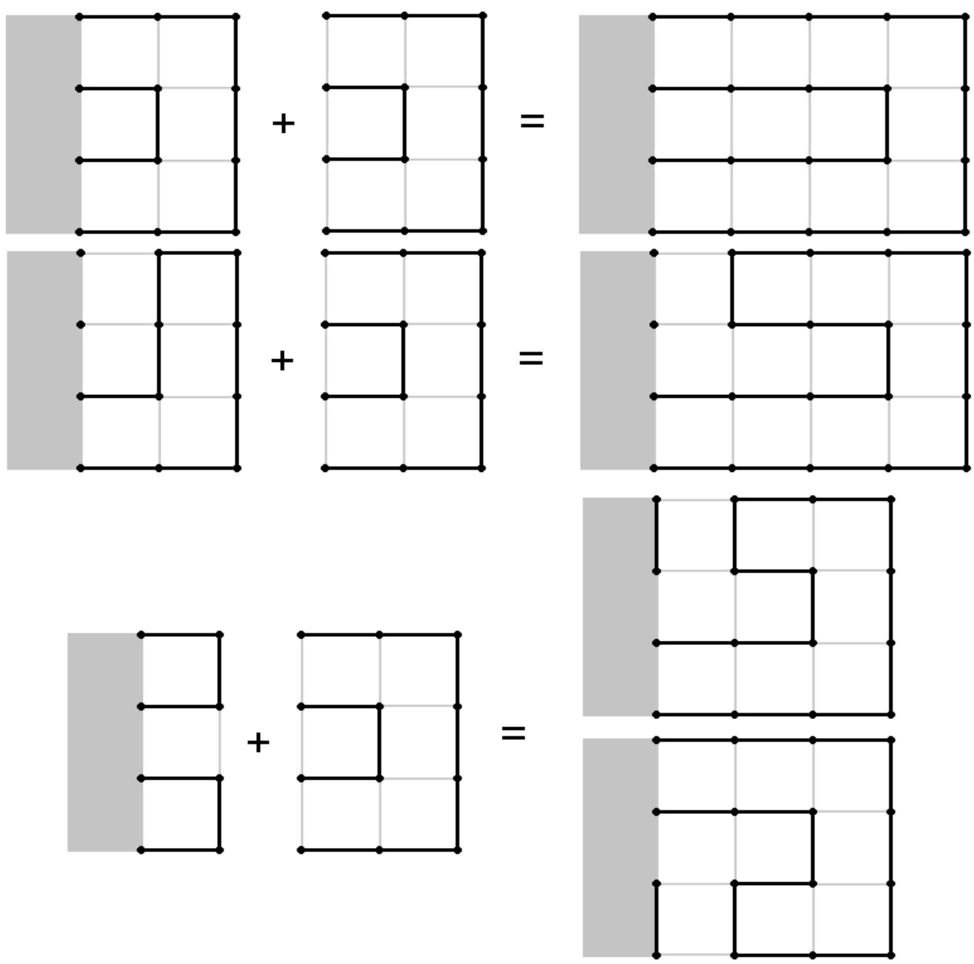
$$h_4(n) = f_1(n) + f_2(n) + f_3(n) + f_4(n) + f_5(n) = h_4(n - 1) + 4\, h_4(n - 2) - h_4(n - 4) + h_4(n - 5) =$$

$$= 2h_4(n-1)+2h_4(n-2) - 2h_4(n-3)+h_4(n-4) - h_4(n-1)+2h_4(n-2)+2h_4(n-3) - 2h_4(n-4)+h_4(n-5):$$

Ստացվածից հետևում է, որ $h_4(n) = 2\, h_4(n-1)+2\, h_4(n-2) - 2\, h_4(n-3)+h_4(n-4)$:



ա)



բ)

Նկ. *3.11*

Գտնենք h$_4$(*n*) - ի ստացված անդրադարձ առնչությանը բավարարող h$_4$(*n*) = λ^{n-1} $(\lambda\neq 0)$ տեսքի հաջորդականություն, այդ դեպքում.

$$\lambda^{n-1}=2\lambda^{n-2}+2\lambda^{n-3}-2\lambda^{n-4}+\lambda^{n-5},\ n=5,\,6,\,\ldots,$$
$$\lambda^4-2\lambda^3-2\lambda^2+2\lambda-1=0:$$

Հավասարման լուծումներն են.

$$\lambda_1\approx\ 2.538616,$$
$$\lambda_2\approx\ -1.276209,$$
$$\lambda_3\approx\ 0.368796-0.415512\mathrm{i},$$
$$\lambda_4\approx\ 0.368796+0.415512\mathrm{i},$$

հետևաբար $c_1\lambda_1^{n-1}+c_2\lambda_2^{n-1}+c_3\lambda_3^{n-1}+c_4\lambda_4^{n-1},\ n=1,\,2,\ldots,$ հաջորդականությունը ցանկացած $c_1,\ c_2,\ c_3,\ c_4$ թվերի համար բավարարում է անդրադարձ առնչությանը: Ընտրենք այդ թվերն այնպես, որ բավարվեն սկզբնական արժեքները.

$$\begin{cases} c_1+c_2+c_3+c_4=0,\\ c_1\lambda_1+c_2\lambda_2+c_3\lambda_3+c_4\lambda_4=1,\\ c_1\lambda_1^2+c_2\lambda_2^2+c_3\lambda_3^2+c_4\lambda_4^2=2,\\ c_1\lambda_1^3+c_2\lambda_2^3+c_3\lambda_3^3+c_4\lambda_4^3=6,\end{cases}$$
, լուծումներն են՝
$$c_1\approx\ 0.346124,$$
$$c_2\approx\ -0.148311,$$
$$c_3\approx\ -0.098906-0.006019\mathrm{i},$$
$$c_4\approx\ -0.098906+0.006019\mathrm{i}:$$

Վերծնելով իրական արժեքները կարող ենք գրել.

h$_4$(*n*) ~ $c_1\lambda_1^{n-1}+c_2\lambda_2^{n-1}\approx 0.346124\,(2.538616)^{n-1}-0.148311\,(-1.276209)^{n-1}:$

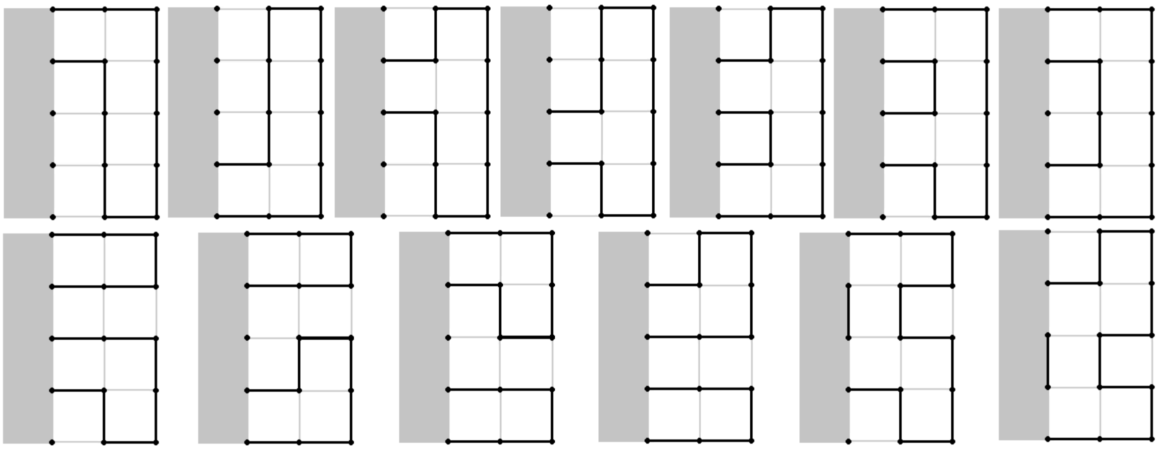
Պնդում 3.10: h$_5$(*n*) = *11* h$_5$(*n-2*) + *2* h$_5$(*n-6*), *n = 2k, k* = 3, 4, ...,

h$_5$(*0*) = h$_5$(*1*) = *0*, h$_5$(*2*) = *1,* h$_5$(*4*) = 14:

↦ $T_{n,5}$ - ի համիլտոնյան ցիկլը կարող է ունենալ նկ. 3.12 - ում պատկերված *13* տեսքերից մեկը, նրանց քանակը նշանակենք համապատասխանաբար f$_1$(*n*), f$_2$(*n*), ..., f$_{12}$(*n*), f$_{13}$(*n*), պարզ է, որ h$_5$(*n*) = f$_1$(*n*) + f$_2$(*n*) + ... + f$_{12}$(*n*) + f$_{13}$(*n*): Նկ. 3.13 - ում ցույց է տրված համիլտոնյան ցիկլի կառուցման որոշ դեպքեր(ցիկլի կառուցման կրկնվող եղանակները և պարզ դեպքերը նկարված չեն), կստանանք.

f$_1$(*n*) = f$_2$(*n*) = f$_9$(*n*) = f$_{10}$(*n*) = h$_4$(*n - 2*),

f$_3$(*n*)=f$_4$(*n*)=f$_8$(*n*)=f$_{11}$(*n*) = f$_1$(*n-2*)+f$_2$(*n-2*)+f$_3$(*n-2*)+f$_4$(*n-2*)+f$_5$(*n-2*)+f$_6$(*n-2*)+f$_7$(*n-2*)+f$_{10}$(*n-2*)+ +f$_{11}$(*n-2*)+f$_{13}$(*n-2*)) = h$_5$(*n-2*) - f$_8$(*n-2*) - f$_9$(*n-2*) - f$_{12}$(*n-2*) = h$_5$(*n-2*) - h$_5$(*n-4*) - f$_3$(*n-2*) - f$_5$(*n-2*)



Նկ. 3.12

$f_5(n)=f_6(n)=f_{12}(n)=f_{13}(n) = f_1(n-2)+f_2(n-2)+f_3(n-2)+f_5(n-2)+f_7(n-2)+f_{10}(n-2)+f_{11}(n-2)+f_{13}(n-2) =$
$= h_5(n-2) - f_4(n-2) - f_6(n-2) - f_8(n-2) - f_9(n-2) - f_{12}(n-2)= h_5(n-2) - h_5(n-4) - 2f_3(n-2) - 2f_5(n-2)$,

$f_7(n) = 2f_1(n-2)+2f_2(n-2)+f_3(n-2)+f_4(n-2)+f_5(n-2)+f_6(n-2)+2f_7(n-2)+f_8(n-2)+f_9(n-2)+f_{10}(n-2)+$
$+f_{11}(n-2)+f_{12}(n-2)+f_{13}(n-2) = 2h_5(n-2) - 2h_5(n-4) - 4f_3(n-2) - 4f_5(n-2)$:
Նկատենք, որ $f_5(n) = 2f_3(n) - h_5(n-2) + h_5(n-4)$, $f_7(n) = 4f_3(n) - 2h_5(n-2) + 2h_5(n-4)$: Տեղադրենք ստացված արտահայտությունները $h_5(n)$-ում և $f_7(n)$-ում.
$h_5(n) = 4h_5(n-2) + 4f_3(n) + 4f_5(n) + f_7(n) = 16f_3(n) - 2h_5(n-2) + 6h_5(n-4)$,
$16f_3(n) = h_5(n) + 2h_5(n-2) - 6h_5(n-4)$
$4f_3(n) - 2h_5(n-2) + 2h_5(n-4) = f_7(n) = h_5(n-2)+f_1(n-2)+f_2(n-2)+f_7(n-2) = h_5(n-2)+2h_5(n-4)+$
$+4f_3(n-2) - 2h_5(n-4) + 2h_5(n-6)$, $16f_3(n) = 16f_3(n-2) + 12h_5(n-2) - 8h_5(n-4) + 8h_5(n-6)$:
Տեղադրենք $16f_3(n)$ - ի արժեքը վերջին արտահայտությունում.
$h_5(n) + 2h_5(n-2) - 6h_5(n-4) = h_5(n-2) + 2h_5(n-4) - 6h_5(n-6)+12h_5(n-2) - 8h_5(n-4) + 8h_5(n-6)$,
$h_5(n) = 11h_5(n-2) + 2h_5(n-6)$:↵
Գտնենք $h_5(n)$ - ի ստացված անդրադարձ առնչությանը բավարարող $h_5(2n) = \lambda^{n-1}$ $(\lambda \neq 0)$ տեսքի հաջորդականություն, այդ դեպքում.

$$\lambda^{n-1}=11\lambda^{n-2}+2\lambda^{n-4},\ n=4, 5, \ldots,$$
$$\lambda^3-11\lambda^2-2=0:$$

Հավասարման լուծումներն են.

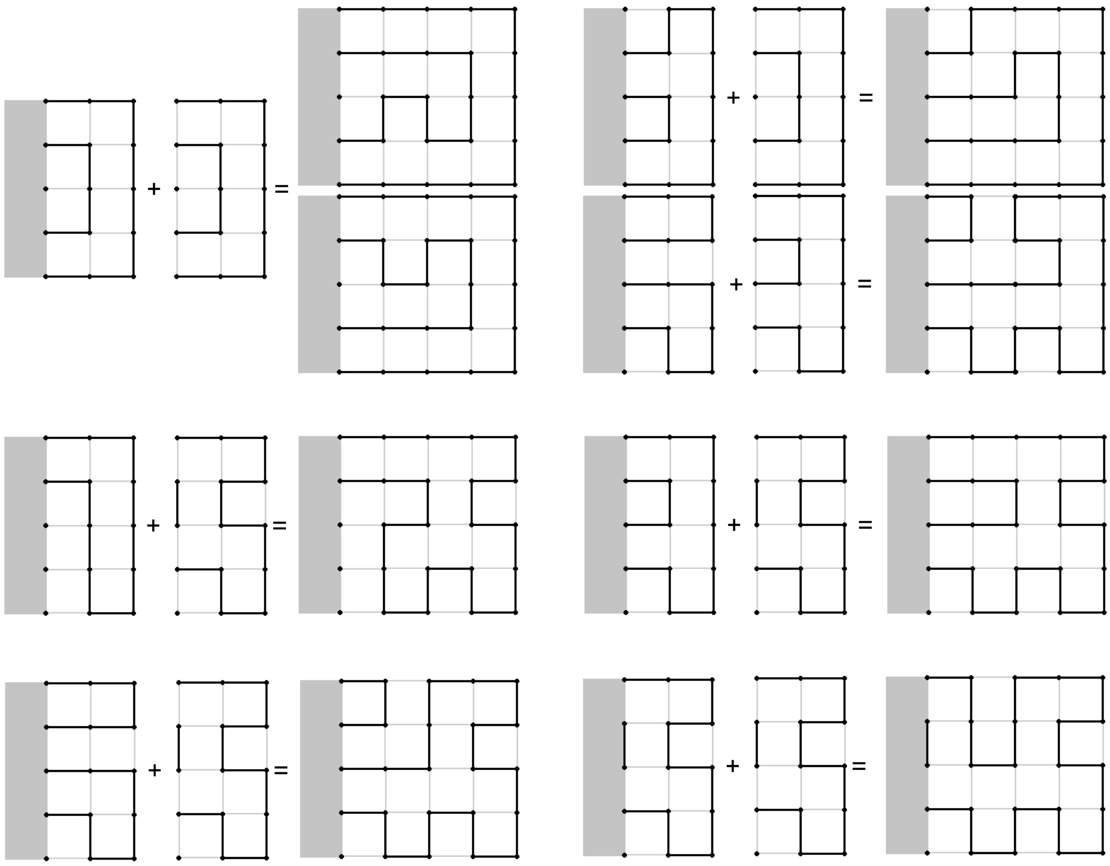
$$\lambda_1 \approx 11.01648,$$
$$\lambda_2 \approx -0.00824-0.426i,$$
$$\lambda_3 \approx -0.00824+0.426i,$$

հետևաբար $c_1\lambda_1^{n-1}+c_2\lambda_2^{n-1}+c_3\lambda_3^{n-1},\ n=1, 2, \ldots,$ հաջորդականությունը ցանկացած $c_1,\ c_2,\ c_3,$ թվերի համար բավարարում է անդրադարձ առնչությանը: Ընտրենք այդ թվերն այնպես, որ բավարվեն սկզբնական արժեքները.

$$\begin{cases} c_1+c_2+c_3=0, \\ c_1\lambda_1+c_2\lambda_2+c_3\lambda_3=1, \\ c_1\lambda_1^2+c_2\lambda_2^2+c_3\lambda_3^2=14, \end{cases}$$, լուծումներն են՝ $$\begin{aligned} c_1 &\approx \ 0.115148, \\ c_2 &\approx \ -0.057574-0.316283i, \\ c_3 &\approx \ -0.057574+0.316283i: \end{aligned}$$

Վերծնելով իրական արժեքը կստանանք.

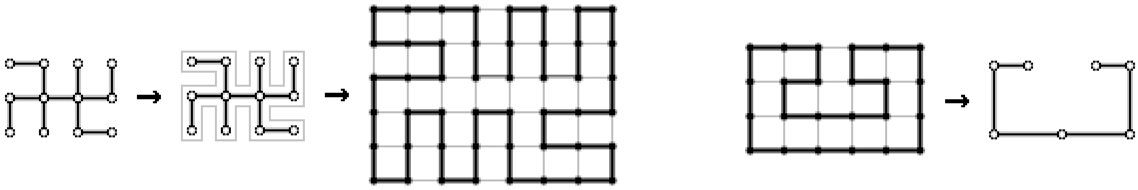
$$h_5(n) \sim \ c_1\lambda_1^{n-1} \approx 0.115148(11.01648)^{n-1}:$$



Նկ. 3.13

Պնդում 3.11: $nm\,h_m(n) < s_m(n) < h_{2m}(2n)$, $m,n = 2, 3, \ldots$:

↦ Պարզ է, որ $T_{n,m}$ ցանցի ցանկացած համիլտոնյան ցիկլից որևէիցէ կող հեռացնելով կստանանք կմախքային ծառ $T_{n,m}$ ցանցում, բայց հակառակը ճիշտ չէ, հետևաբար $nm\,h_m(n) < s_m(n)$ ակնհայտ է: $T_{2n,2m}$ ցանցի համիլտոնյան ցիկլ կստանանք $T_{n,m}$ - ի կմախքային ծառը պարփակելով ցիկլով, պարզ է, որ այդ դեպքում կլինի այնպիսի մի կմախքային ծառ, որ դեպքում ցանցի չափսերը կկրկնապատկվեն(նկ. 3.14 ա)): Այդպես պարզ է, որ ամեն մի կմախքային ծառի միարժեք կհամապատասխանի որևիցէ համիլտոնյան ցիկլ, հետևաբար $h_{2m}(2n) \geq s_m(n)$: Սակայն հակառակը ճիշտ չէ, ամեն մի $T_{2n,2m}$ ցանցի համիլտոնյան ցիկլի չէ, որ նման կառուցման ձևով կհամապատասխանի $T_{n,m}$ - ի կմախքային ծառ, ավելին ստացված պատկերը կարող է նույնիսկ կմախքային ծառ չլինել(Նկ. 3.14 բ)):↵

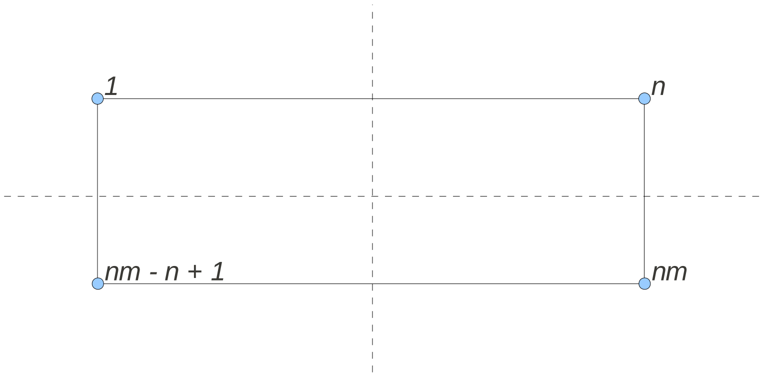


ա) բ)

Նկ. 3.14

3.4. ***Սիմետրիկ ցիկլեր***

Քառակուսային ցանցի կենտրոնից տանենք ուղղահայաց և հորիզոնական առանցքներ (քառակուսային ցանցը նաև ուղղանկյուն է, հետևաբար նրա կենտրոնը կլինի անկյունագծերի հատման կետը), այնպես, որ առանցքները ուղղահայաց և զուգահեռ լինեն ցանցի կողերին (Նկ. 3.15):



Նկ. 3.15

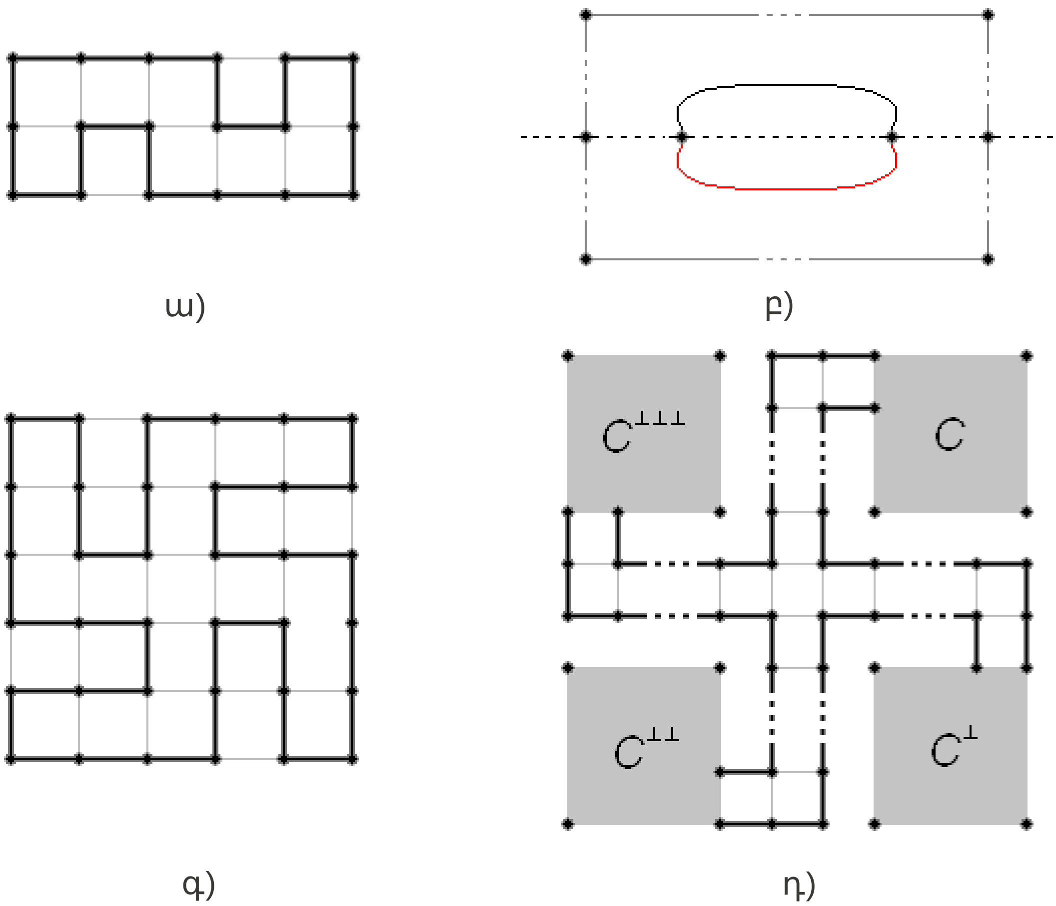
Այն համիլտոնյան ցիկլը, որը ստացվում է *C* ցիկլից նրան հորիզոնական առանցքով սիմետրիկ արտապատկերելիս, կանվանենք *C* ցիկլի *հորիզոնական-սիմետրիա* և կնշ. C^{h}, իսկ այդ ցիկլերին կանվանենք *հորիզոնական սիմետրիկ*: Եթե ցիկլը և նրա հորիզոնական-սիմետրիան համընկնում են, ապա այդ ցիկլը կանվանենք *հորիզոնական սիմետրիկ ցիկլ*: Համանման ձևով կսահմանենք *ուղղահայաց-սիմետրիա, կենտրոնական-սիմետրիա* և կնշանակենք համապատասխանաբար $C^{ու}$ և $C^{կ}$:
Պարզ է , որ $C^{hh} = C^{ուու} = C^{կկ} = C$, $C^{hու} = C^{ուh} = C^{կ}$, $C^{hուh} = C^{ու}$:

Նկ. 3.4 - ում C_3 ցիկլը ստացվում է C_1 ցիկլից նրան ուղղահայաց առանցքով արտապատկերելիս, այսինքն $C_1^{\text{ու}} = C_3$:

Եթե ցիկլը կենտրոնական սիմետրիկ է, ապա չի հետևում, որ այն հորիզոնական կամ ուղղահայաց սիմետրիկ է(նկ. 3.16 ա)):

Լեմմա 2: $T_{n,m}$ ցանցում գոյություն ունի հորիզոնական(ուղղահայաց) սիմետրիկ համիլտոնյան ցիկլ այն և միայն այն դեպքում երբ m(n) - ը զույգ է:

↦ Բավարարությունը ակնհայտ է:
Ենթադրենք ցանցում կա հորիզոնական սիմետրիկ ցիկլ երբ m - ը կենտ է: Պարզ է ,որ այդ դեպքում հորիզոնական առանցքը կպարունակի ցանցի գագաթներ: Դիտարկենք համիլտոնյան ցիկլի այն ամենակարճ պարզ շղթան, որը միացնում է հորիզոնական առանցքի վրա գտնվող երկու գագաթներ: Ստացված պարզ շղթան իր հորիզոնական սիմետրիայի հետ միասին կկազմի ցիկլ որը չի անցնում ցանցի բոլոր գագաթներով, հետևաբար ցիկլը համիլտոնյան չէ (նկ. 3.16 բ)): Համանման ձևով կապացուցենք ուղղահայաց սիմետրիայի համար:↵



Նկ. 3.16

Լեմմա 3: $T_{n,m}$ ցանցում գոյություն ունի կենտրոնական սիմետրիկ համիլտոնյան ցիկլ այն և միայն այն դեպքում երբ կամ n - ը և m - ը միաժամանակ զույգ են կամ $nm/2$ - ը կենտ է:

↦ Բավարարությունը: Այն դեպքում, երբ n - ը և m - ը միաժամանակ զույգ են, ակնհայտ է: Ենթադրենք $nm/2$ - ը կենտ է, համիլտոնյան ցիկլը

կկառուցենք, այնպես, որ նրա պատկերը նման լինի Նկ. 3.16 ա) - ում պատկերված ցիկլին:
Անհրաժեշտությունը: Ենթադրենք *n, m* - ից *m* - ը կենտ է: Այդ դեպքում հորիզոնական առանցը կպարունակի ցանցի գագաթներ: Պարզ է, որ այդ դեպքում կենտրոնական սիմետրիկ համիլտոնյան ցիկլը չի անցնի ցանցի կենտրոնին հարևան երկու գագաթները միացնող կողով, որը գտնվում է հորիզոնական առանցքի վրա, քանի, որ կստացվեր, որ ցիկլի երկարությունը կենտ թիվ է: Համիլտոնյան ցիկլը բաղկացած է *2* պարզ շղթաներից, որոնց եզրերը այդ կենտրոնական կողի եզրերն են: Քանի որ ցիկլը կենտրոնական սիմետրիկ է, հետևաբար այդ երկու պարզ շղթաների երկարությունները հավասար են, նշանակենք *q* - ով, $nm = 2q$, *q* - ն կենտ է, քանի որ կենտրոնական կողը ավելացնելիս այդ պարզ շղթաները կդառնան ցիկլեր, որոնց երկարությունները զույգ է:↵

Միայն քառակուսի և համիլտոնյան ցանցերի համար սահմանենք ցիկլերի ուղղահայացությունը:

Այն համիլտոնյան ցիկլը, որը ստացվում է *C* ցիկլից նրան + 90° պտտելիս, կանվանենք այդ ցիկլին ուղղայաց ցիկլ և կնշանակենք $C^{\perp}$, եթե $C = C^{\perp}$, ապա ցիկլը կկոչվի ուղղահայաց: (Նկ. 3.4 - ում C_5 և C_6 ցիկլերը միմյանց ուղղահայաց են, իսկ Նկ. 3.16 գ) - ում ցիկլը ուղղահայաց է): Պարզ է , որ $C^{\perp\perp\perp\perp} = C$, $C^{\perp\perp} = C^{\text{հոր}}$:

Լեմմա 4: $T_{n,n}$ ցանցում գոյություն ունի ուղղահայաց համիլտոնյան ցիկլ այն և միայն այն դեպքում երբ *n/2* - ը կենտ է:

↦ Բավարարությունը: Նկ. 3.16 դ) - ում պատկերված է ուղղահայց ցիկլը, որտեղ *C* - ն ցանկացած ցիկլ է $(n/2 - 1) \times (n/2 - 1)$ ցանցում:
Անհրաժեշտությունը: Ենթադրենք ցանցում կա ուղղահայաց համիլտոնյան ցիկլ, պարզ է, որ համիլտոնյան ցիկլը չի անցնում ցանցի կենտրոնին մոտ *4* կողերով (օրինակ Նկ. 3.16 գ)), ցիկլը բաղկացած է 4 պարզ շղթաներից, ամեն մի այդպիսի կողի եզրերը այդ շղթաներից մեկի եզրերն են, շղթաներից մեկի վերջը մյուսի սկիզբն է, մեկը մյուսից կարելի է ստանալ նրան պտտելով 90° աստիճանով, այսինքն այդ պարզ շղթաների երկարությունները միմյանց հավասար են: Նշանակենք *q* - ով: *q* - ն կենտ թիվ է, քանի որ նրանցից յուրաքանչյուրին միայն մեկ կող ավելացնելիս կստանանք ցիկլ: Հետևաբար.

$$p = \frac{n^2}{4} = \frac{n}{2}\frac{n}{2}: \quad ↵$$

C_1 և C_2 ցիկլերը կանվանենք *սիմետրիկ,* եթե նրանք հորիզոնական, ուղղահայաց կամ կենտրոնական սիմետրիկ են, և ցիկլը կկոչվի սիմետրիկ, եթե սիմետրիան հենց ինքն է: Քառակուսի ցանցերի համար «սիմետրիա» հասկացողության մեջ կհասկանանք նաև ցիկլերի ուղղահայացությունը, ինչպես նաև ուղղահայաց պատկերի սիմետրիաները: Պարզ է, որ ամենաշատը կարող են լինել *4* հատ ցիկլեր` *C*, $C^{\text{հ}}$,$C^{\text{ու}}$, $C^{\text{կ}}$, որոնցից ցանկացած երկուսը լինեն միմյանց սիմետրիկ (քառակուսային ցանցերի համար *8* հատ): Հետևաբար ցանցի ցիկլերի բազմությունը տրոհվում է միմյանց հետ չհատվող ենթաբազմությունների, այնպես, որ միևնույն ենթաբազմությանը պատկանող բոլոր ցիկլերը միմյանց սիմետրիկ են, իսկ տարբեր ենթաբազմություններից ցանկացած ցիկլեր սիմետրիկ չեն։

$T_{n,m}$ ցանցի զույգ առ զույգ ոչ սիմետրիկ ցիկլերի քանակը նշ. $\hat{h}_m(n)$-ով։

Պնդում 3.12: $\hat{h}_3(n) = \begin{cases} (2^{n/2-1}+2^{\lfloor n/4 \rfloor+1})/4, & \text{երբ } 3n/2 \text{ կենտ է,} \\ (2^{n/2-1}+2^{\lfloor n/4 \rfloor})/4, & \text{հ. դ.։} \end{cases}$

↦ Քանի որ ցանցի ցիկլերի բազմութունը տրոհվում է միմյանց հետ չհատվող ենթաբազմությունների, որոնցից յուրաքանչյուրի հզորությունը 4 - ից շատ չէ, պարզ է, որ 4 - ից փոքր կլինի այն դեպքում, երբ այդ բազմությունը կպարունակի սիմետրիկ ցիկլ։ Հաշվենք այդ ցիկլերի քանակը։ Քանի որ ցանցի լայնությունը՝ *3* - ը, կենտ է, հետևաբար ցանցում գոյություն չունի հորիզոնական սիմետրիկ ցիկլ(լեմմա 2)։ Օգտվելով $T_{n,3}$ ցանցի համիլտոնյան ցիկլերի կառուցման եղանակից (նկ.3.10), ուղղահայաց սիմետրիկ ցիկլ կստանանք, եթե նման եղանակով ցանցի աջ կեսում կառուցենք պատկեր և միացնենք այդ պատկերը նրա ուղղահայաց առանցքով սիմետրիկ արտապատկերված պատկերին։ Կառուցման այդպիսի եղանակով ցիկլերի քանակը կլինի $2^{[n/4]}$։ Այն դեպքում, երբ *3n/2* կենտ է, ստացված պատկերը միացնելով նրա կենտրոնական սիմետրիային կստանանք կենտրոնական սիմետրիկ ցիկլ, նրանց քանակը նույնպես կլինի $2^{[n/4]}$։ Ստացված արժեքները գումարելով ցանցի համիլտոնյան ցիկլերի քանակին և բաժանելով *4* - ի կստանանք տրոհված ենթաբազմությունների և հետևաբար ոչ սիմետրիկ ցիկլերի քանակը։↵

$T_{n,m}$ ցանցի վրա *i* և *j* գագաթների հեռավորությունը կիամարենք այդ գագաթները միացնող այն շղթայի երկարությունը, որը ամենակարճն է և կնշանակենք $d_T(i, j)$,պարզ է որ.

$$d_T(i, j) = | \{i / n\} - \{j / n\} | + | \{i / m\} - \{j / m\} |։$$

Նույն *i* և *j* գագաթների հեռավորությունը համիլտոնյան ցիկլի վրա կիամարենք համիլտոնյան ցիկլի ուղղությամբ շարժվելիս *i* -ից դեպի *j* գագաթը միացնող շղթայի երկարությունը և կնշանակենք $d_C(i, j)$։

$T_{n,m}$ ցանցի *C* ցիկլի համար սահմանենք հետևյալ ֆունկցիան.

$$f_C^T(k) = \sum_{d_C(i, j)=k} d_T(i, j) \,; \quad k = 1, 2, ..., nm-1$$

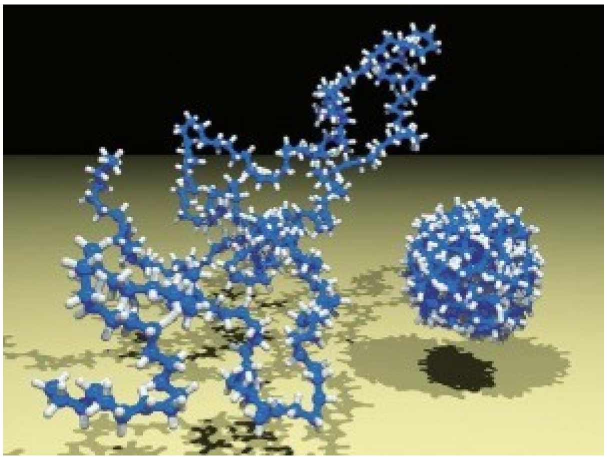
Ֆունկցիան կանվանենք ցիկլը բնութագրող ֆունկցիա։ Պարզ է, որ $f_C^T(1) = mn$, $f_C^T(2) = 2mn$, $f_C^T(k) = f_C^T(mn - k)$։

Պարզ է, որ *C′* և *C″* սիմետրիկ ցիկլերի համար $f_{C'}^T(k) = f_{C''}^T(k)$, *k = 1,2,...,nm-1:*

Հիպոթեզ: Եթե $f_{C'}^T(k) = f_{C''}^T(k)$, *k = 1,2,..., mn-1,* ապա *C′* և *C″* սիմետրիկ են (հատկությունը տեղի ունի $T_{n,m}$ *m = 2,3,4 5,6, n = 1,2,...,27,* բոլոր ցանցերի համար։ Արդյունքները ստացված են համակարգչով. հավելված)։

4. Կիրառությունները

Քառակուսային ցանցում համիլտոնյան ցիկլերի քանակը գտնելու խնդիրը ունի պրակտիկ կիրառում ֆիզիկայի պոլիմերներում: Պոլիմերը դա մոլեկուլային խոշոր կապակցություն է, այսինքն խոշոր մոլեկուլ (ինչպես օրինակ ներկերի մի քանի տեսակներում, պոլիստերոլ և այլն): Ընդհանուր առմամբ պոլիմերները ամենուրեք են: 60 տարի առաջ պոլիմերների ֆիզիկան դարձավ առանձին գիտական ճյուղ: Պոլիմերը հարմար է մոդելավորել պարզ շղթայի տեսքով, որի գագաթները անցնում են եռանկյուն ցանցով:



Նկ. 4.1

Բնականաբար իրական կյանքում բոլոր մակրոմոլեկուլները ուղիղ անկյունով չեն միանա, բայց պարզության համար պոլիմերային մոլեկուլը ամեն դեպքում մոդելավորվում է երկչափ քառակուսային ցանցում, քանի որ եռաչափում խնդիրը չափազանց բարդանում է:

Փաստորեն, ցանցը դա պոլիմերի նկարագրման միջավայրն է, իսկ ինքը պոլիմերը՝ ցանկացած պարզ շղթա այդ ցանցում: Եթե պոլիմերը հարմար, խիտ հավաքված է(նկ.4.1 աջից), ապա շղթան կանցնի ցանցի բոլոր գագաթներով, այսինքն կլինի համիլտոնյան, բացի դրանից շղթան կարող է փակվել ինքն իրենով, կազմելով ցիկլ: Խնդիրը կայանում է ցանցերում համիլտոնյան ցիկլերի քանակը գտնելուն, քանի որ ֆիզիկոսներին հետաքրքրում է կառուցվածքային էնտրոպիան, որի օգնությամբ կարելի է ստանալ զանազան կառուցվածքի հատկություններ: Ավելի կոնկրետ, այդ դեպքում էնտրոպիան հավասար է պոլիմերների բոլոր հնարավոր եղանակներով բաշխման քանակի լոգարիթմին, այսինքն բոլոր ցիկլերի քանակին:

5. Հավելված

5.1. Քառակուսային ցանցում կմախքային ծառերի, ցիկլերի քանակի և նրանց մետրիկական բնութրիչների հաշվման ալգորիթմներ

Աշխատանքում ցիկլերի քանակի հաշվման համար օգտագործված են հետևյալ ալգորիթմները.

1. ըստ խորության փնտրումներ[15]
2. ըստ լայնության փնտրումներ[16]
3. Դինամիկ ծրագրավորման մեթոդ(մատրիցային մեթոդ և առանց մատրիցի)[17]

Ընդհանուր դեպքում ծրագիրը աշխատում է `փնտրում ըստ խորության` և ըստ լայնության ալգորիթմներով։ Սակայն ալգորիթմները ժամանակատար են և հարմար չեն մեծարժեք մուտքային տվյալների համար(օրինակ 4x2200Hz միջուկային պրոցեսորներով համակարգչով ծրագիրը 12x12 քառակուսային ցանցում համիլտոնյան ցիկլերի քանակը, որն է` 1076226888605605706, գտնում է 37000 րոպեում(≈26 օր))։ Հետևաբար օգտագործվում է դինամիկ ծրագրավորման մեթոդը։ Ցանցից կառուցվում է վերջավոր աֆտոմատ այնպես, որ $T_{n,m}$ ցանցի համիլտոնյանց ցիկլերի քանակը լինի աֆտոմատի սկզբնական վիճակից մինչև վերջնական վիճակ n քայլում հասնելու բոլոր հնարավոր եղանակները։ Դրա համար կա երկու տարբերակ, փոխանցման մատրիցով և առանց դրա։ Երկրորդ դեպքում ալգորիթմի բարդությունը $O(6^m\sqrt{m}n)$ է։ Առաջին դեպքում եթե աֆտոմատի մատրիցան A = nxn է, ապա բավական է հաշվել A^n, և վերծնել A(1,n) էլեմենտը։ A^n - ը հաշվվում է զուգահեռ` օգտագործելով CUDA զուգահեռ ծրագրավորման լեզուն։ (Ծրագրի աշխատանքի համար անհրաժեշտ է NVIDIA videocard:) Մատրիցների բազմապատկում գործողությունը հնարավոր է զուգահեռացնել է n^2 անգամ։ Հետևաբար ալգորիթմի ժամանակային բարդությունը կախված համակարգչի հնարավորությունից կլինի $n \log n$ ։

Ծրագիրը բաղկացած է երեք մասից։ Սկզբում այն գտնում է ցանցի բոլոր համիլտոնյանց ցիկլերի բազմությունը, ստացված բազմության համար գտնում մինիմալ հեռավորությամբ ցիկլերը, և գտնում այդ բազմությունից բոլոր այն ցիկլերը, որոնք ունեն մինիմալ հեռավորություն։ Հաշվում է այդ բազմության բոլոր ցիկլերի բնութագրող ֆունկցիաները։ Այն ցիկլերի վրա, որոնց համար այդ ֆունկցիաների արժեքները բոլոր ինդեքսների համար հավասար են, ստուգում է այդ ցիկլերի սիմետրիկությունը։ Գտնում է բոլոր ոչ սիմետրիկ ցիկլերը և որպես ելք տալիս է այդ ցիկլերի բազմությունը, ինչպես նաև ցանցի համար Կիրխհոֆի մատրիցը։ Ծրագրի վերջին մասը աշխատում է Octave գրադարանով։ Հաշվում է Կիրխհոֆի մատրիցի հանրահաշվական լրացման դետերմինանտը(հիշենք, որ այն ցանցի կմախքային ծառերի քանակն է), և արտածում է ոչ սիմետրիկ ցիկլերի բնութագրող ֆունկցիաները(նկ.5.1, $T_{10,3}$ ցանցի համար) և դրանց գրաֆիկները(նկ.5.2)։

```
               Matrix

*--*--*--*--*--*--*--*--*--*
|                          |
*  *--*  *--*  *--*  *--*  *
|  |  |  |  |  |  |  |  |  |
*--*  *--*  *--*  *--*  *--*

 Loop = 1 2 3 4 5 6 7 8 9 10 20 30 29 19 18 28 27 17 16 26 25 15 14 24 23 13 12 22 21 11 1

 f( 1) =  1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1 = 30;
 f( 2) =  2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2 = 60;
 f( 3) =  3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 3+ 3 = 72;
 f( 4) =  4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 4+ 4 = 80;
 f( 5) =  5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 5 = 104;
 f( 6) =  6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6 = 128;
 f( 7) =  7+ 7+ 7+ 7+ 7+ 5+ 3+ 1+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 5+ 7+ 7 = 134;
 f( 8) =  8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 8+ 8 = 140;
 f( 9) =  9+ 9+ 9+ 7+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 5+ 7+ 9+ 9 = 158;
 f(10) = 10+10+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 6+ 8+10+10 = 172;
 f(11) = 11+ 9+ 7+ 5+ 5+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 3+ 1+ 3+ 5+ 5+ 7+ 9+11+ 9 = 174;
 f(12) = 10+ 8+ 6+ 6+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 6+ 6+ 8+10+10+10 = 176;
 f(13) =  9+ 7+ 7+ 5+ 3+ 1+ 3+ 5+ 5+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 5+ 5+ 3+ 1+ 3+ 5+ 7+ 7+ 9+ 9+ 9+ 9 = 180;
 f(14) =  8+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8 = 184;
 f(15) =  9+ 7+ 5+ 3+ 3+ 3+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 9+ 7+ 9+ 7+ 5+ 3+ 3+ 3+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 9+ 7 = 186;
               Matrix

*--*--*--*--*--*--*--*  *--*
|                    |  |  |
*  *--*  *--*  *--*  *--*  *
|  |  |  |  |  |  |        |
*--*  *--*  *--*  *--*--*--*

 Loop = 1 2 3 4 5 6 7 8 18 19 9 10 20 30 29 28 27 17 16 26 25 15 14 24 23 13 12 22 21 11 1

 f( 1) =  1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1 = 30;
 f( 2) =  2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2 = 60;
 f( 3) =  3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 3+ 3 = 74;
 f( 4) =  4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 4+ 4 = 84;
 f( 5) =  5+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 3+ 5+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 5 = 104;
 f( 6) =  6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6 = 124;
 f( 7) =  7+ 7+ 7+ 5+ 5+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 5+ 5+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 5+ 7+ 7 = 134;
 f( 8) =  8+ 8+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 8+ 8 = 144;
 f( 9) =  9+ 7+ 7+ 7+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 5+ 7+ 9+ 7 = 156;
 f(10) =  8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 6+ 8+ 8+ 8 = 164;
 f(11) =  9+ 9+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 7+ 7+ 7+ 7+ 7+ 5+ 7+ 5+ 7+ 5+ 3+ 1+ 3+ 5+ 5+ 7+ 7+ 9+ 9 = 172;
 f(12) = 10+10+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10 = 180;
 f(13) = 11+ 9+ 7+ 5+ 3+ 1+ 3+ 5+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 7+ 7+ 7+ 5+ 5+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 9+11+ 9 = 182;
 f(14) = 10+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10+10 = 184;
 f(15) =  9+ 7+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 7+ 7+ 9+ 9+ 9+ 9+ 9+ 7+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 7+ 7+ 9+ 9+ 9+ 9 = 186;
               Matrix

*--*--*--*--*--*  *--*--*--*
|              |  |        |
*  *--*  *--*  *--*  *--*  *
|  |  |  |  |        |  |  |
*--*  *--*  *--*--*--*  *--*

 Loop = 1 2 3 4 5 6 16 17 7 8 9 10 20 30 29 19 18 28 27 26 25 15 14 24 23 13 12 22 21 11 1

 f( 1) =  1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1 = 30;
 f( 2) =  2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2 = 60;
 f( 3) =  3+ 3+ 3+ 3+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 3+ 3 = 76;
 f( 4) =  4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 4+ 4 = 88;
 f( 5) =  5+ 5+ 5+ 3+ 3+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 3+ 3+ 3+ 5+ 3+ 5+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 5 = 104;
 f( 6) =  6+ 6+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6 = 120;
 f( 7) =  7+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 3+ 1+ 3+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 5+ 7+ 5 = 132;
 f( 8) =  6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6 = 144;
 f( 9) =  7+ 7+ 7+ 7+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 7+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 7+ 7 = 154;
 f(10) =  8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8 = 160;
 f(11) =  9+ 9+ 9+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 5+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 7+ 5+ 3+ 1+ 3+ 5+ 3+ 5+ 7+ 9+ 9 = 172;
 f(12) = 10+10+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+10+10 = 184;
 f(13) = 11+ 9+ 7+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 9+ 7+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 7+ 9+11+ 9 = 184;
 f(14) = 10+ 8+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 8+10+10+10 = 184;
 f(15) =  9+ 7+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 7+ 9+ 9+ 9+ 9+ 9+ 7+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 7+ 9+ 9+ 9+ 9 = 186;
               Matrix

*--*--*--*--*--*  *--*  *--*
|              |  |  |  |  |
*  *--*  *--*  *--*  *--*  *
|  |  |  |  |              |
*--*  *--*  *--*--*--*--*--*

 Loop = 1 2 3 4 5 6 16 17 7 8 18 19 9 10 20 30 29 28 27 26 25 15 14 24 23 13 12 22 21 11 1

 f( 1) =  1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1 = 30;
 f( 2) =  2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2 = 60;
 f( 3) =  3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 1+ 3+ 3+ 3 = 74;
 f( 4) =  4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 4+ 4 = 84;
 f( 5) =  5+ 5+ 5+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 5 = 106;
 f( 6) =  6+ 6+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6 = 128;
 f( 7) =  7+ 5+ 5+ 5+ 5+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 5+ 5+ 5+ 5+ 3+ 5+ 3+ 5+ 3+ 1+ 3+ 5+ 7+ 5 = 134;
 f( 8) =  6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 6+ 6+ 6 = 140;
 f( 9) =  7+ 7+ 7+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 7+ 7+ 7+ 7+ 7+ 5+ 5+ 5+ 5+ 3+ 3+ 3+ 3+ 5+ 5+ 7+ 7 = 154;
 f(10) =  8+ 8+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8 = 164;
 f(11) =  9+ 7+ 7+ 7+ 7+ 5+ 3+ 1+ 3+ 5+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 7+ 7+ 7+ 5+ 3+ 1+ 3+ 5+ 3+ 5+ 7+ 9+ 7 = 170;
 f(12) =  8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 2+ 2+ 4+ 4+ 4+ 6+ 8+ 8+ 8 = 176;
 f(13) =  9+ 9+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 7+ 7+ 9+ 9+ 9+ 9+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 7+ 7+ 9+ 9 = 182;
 f(14) = 10+10+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10+10+10+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10 = 188;
 f(15) = 11+ 9+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 9+11+ 9+11+ 9+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 5+ 7+ 9+11+ 9 = 190;
```

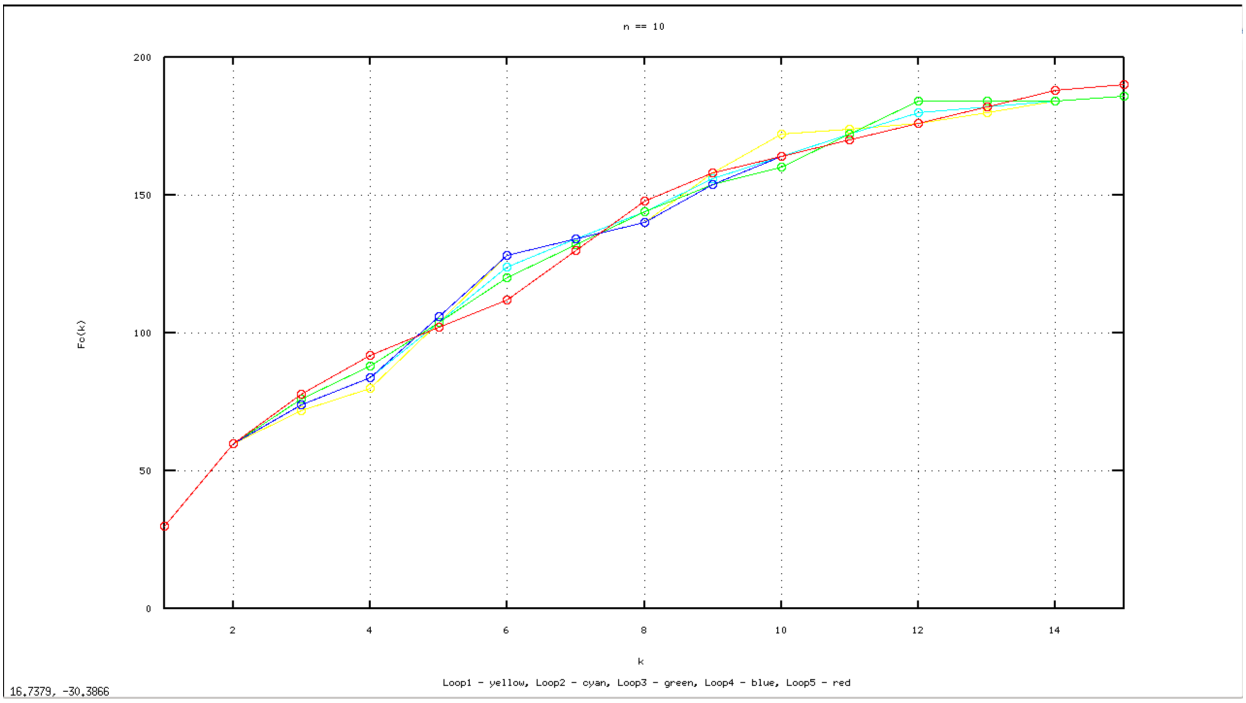
```
                Matrix

*--*--*--*  *--*--*--*  *--*
|        |  |        |  |  |
*  *--*  *--*  *--*  *--*  *
|  |  |        |  |        |
*--*  *--*--*--*  *--*--*--*

 Loop = 1 2 3 4 14 15 5 6 7 8 18 19 9 10 20 30 29 28 27 17 16 26 25 24 23 13 12 22 21 11 1

 f( 1) =  1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1+ 1 = 30;
 f( 2) =  2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2+ 2 = 60;
 f( 3) =  3+ 3+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 3+ 3+ 3+ 3+ 1+ 3+ 1+ 3+ 3+ 3 = 78;
 f( 4) =  4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 2+ 2+ 2+ 2+ 2+ 4+ 4 = 92;
 f( 5) =  5+ 3+ 3+ 3+ 5+ 3+ 5+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 3+ 5+ 3+ 3+ 3+ 5+ 3+ 5+ 3+ 3+ 3+ 3+ 1+ 3+ 5+ 3 = 102;
 f( 6) =  4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 4+ 2+ 2+ 4+ 4+ 4 = 112;
 f( 7) =  5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 5 = 130;
 f( 8) =  6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6 = 148;
 f( 9) =  7+ 7+ 7+ 5+ 7+ 5+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 7+ 7+ 7+ 7+ 5+ 7+ 5+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 7+ 7 = 158;
 f(10) =  8+ 8+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 8+ 8+ 8+ 8+ 6+ 6+ 6+ 6+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 8+ 8 = 164;
 f(11) =  9+ 7+ 7+ 7+ 7+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 9+ 7+ 9+ 7+ 7+ 7+ 7+ 5+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 9+ 7 = 170;
 f(12) =  8+ 8+ 8+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 4+ 6+ 8+ 8+ 8+ 8+ 8+ 8+ 8+ 6+ 4+ 4+ 2+ 2+ 4+ 4+ 6+ 8+ 8+ 8 = 176;
 f(13) =  9+ 9+ 9+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 5+ 7+ 7+ 9+ 9+ 9+ 9+ 9+ 7+ 5+ 3+ 3+ 1+ 3+ 5+ 5+ 7+ 7+ 9+ 9 = 182;
 f(14) = 10+10+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10+10+10+ 8+ 6+ 4+ 2+ 2+ 2+ 4+ 6+ 6+ 6+ 8+10+10 = 188;
 f(15) = 11+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 5+ 7+ 9+11+ 9+11+ 9+ 7+ 5+ 3+ 1+ 3+ 3+ 5+ 7+ 5+ 7+ 9+11+ 9 = 190;
```

Նկ. 5.1



Նկ. 5.2

5.2. *Արդյունքներ*

Քառակուսային ցանցերի համար ստացված են հետևյալ արդյունքները.

Համիլտոնյան ցիկլերի քանակի համար.

4 x n ցանցում - 1, 2, 6, 14, 37, 92, 236, 596, 1517, 3846, 9770, 24794, 62953, 159800, 405688, 1029864, 2614457, 6637066, 16849006, 42773094, 108584525, 275654292, 699780452, 1776473532, 4509783909, 11448608270, 29063617746

5 x 2n ցանցում - 1, 14, 154, 1696, 18684, 205832, 2267544, 24980352, 275195536, 3031685984, 33398506528, 367933962880, 4053336963648, 44653503613184

6 x n ցանցում - 1, 4, 37, 154, 1072, 5320, 32675, 175294, 1024028, 5668692, 32463802, 181971848, 1033917350, 5824476298, 32989068162, 186210666468, 1053349394128, 5950467515104:

10 x n ցանցում - 1, 16, 1517, 18684, 1024028, 17066492, 681728204, 13916993782, 467260456608, 10754797724124, 328076475659033, 8091313110371792:

2n x 2n ցանցում - 1, 6, 1072, 4638576, 467260456608, 1076226888605605706:

Կմախքային ծառերի քանակի համար.

2 x n ցանցում - 1, 4, 15, 56, 209, 780, 2911, 10864, 40545, 151316, 564719, 2107560, 7865521, 29354524, 109552575, 408855776, 1525870529, 5694626340, 21252634831, 79315912984, 296011017105, 1104728155436, 4122901604639, 15386878263120

3 x n ցանցում - 1, 15, 192, 2415, 30305, 380160, 4768673, 59817135, 750331584, 9411975375, 118061508289, 1480934568960, 18576479568193, 233018797965135, 2922930580320960

4 x n ցանցում - 1, 56, 2415, 100352, 4140081, 170537640, 7022359583, 289143013376, 11905151192865, 490179860527896:

5 x n ցանցում - 1, 209, 30305, 4140081, 557568000, 74795194705, 10021992194369, 1342421467113969

n x n ցանցում - 1, 4, 192, 100352, 557568000, 32565539635200, 19872369301840986112,

6. Ծրագիր

makefile

```
CC = gcc
CXX = g++
LINK = g++
MKDIR = mkdir -p
DEL_FILE = rm -f
DEL_DIR = rmdir
SOURCES := $(wildcard src/*.cpp)
OBJECTS := $(patsubst src/%.cpp,obj/%.o,$(SOURCES))
DEPENDS := $(patsubst src/%.cpp,obj/%.dep,$(SOURCES))
bin/program: ./bin $(OBJECTS) bin/out
	echo "./bin/out" > ./bin/program
	echo "octave ./octave/octave.m" >> ./bin/program
	chmod 777 ./bin/program
bin/out:
	$(LINK) $(OBJECTS) -o bin/out
./bin :
	mkdir -p bin
obj/%.o : src/%.cpp
	$(CXX) -c $< -o $@
obj/%.dep : src/%.cpp ./obj
	$(CXX) -MM $< -MT "$@ $(putsubst %.dep,%.o,$@)" -MF $@
./obj:
	mkdir -p obj
.PHONY : clean
clean:
	$(DEL_FILE) bin/*
	$(DEL_DIR) bin
	$(DEL_FILE) obj/*
	$(DEL_DIR) obj
-include $(DEPENDS)
```

main.cpp

```
#include "functions.hpp"
#include <iostream>

int main()
{
	int M, N;
	std::cout << "Enter count of rows    ";
	std::cin >> M;
	std::cout << "Enter count of columns ";
	std::cin >> N;
	std::cout << std::endl
		<< " P" << N << " x P" << M << " grid graph  "
		<< std::endl;
	generate_ham_cycle(M, N, 1);
	return 0;
}
```

functions.hpp

```
#ifndef _FUNCTIONS_HPP_
#define _FUNCTIONS_HPP_

#include <vector>

//print hamiltonian loop in grid
void print_vector_in_grid(std::vector<int> &, int);

//distance between two cycles
int distance(int, int);

//print hamiltonian loop
void print_vector(std::vector<int> &);

//generate hamiltonian cycles
//int generate_ham_cycle_helper(int, int&, std::vector<int> &);
int generate_ham_cycle_helper(int, int&, std::vector<int> &, int** &, const bool* &, int&,
int&);

//generate non simetric matrix for mxn grid
void generate_grid_matrix(int** &, int, int);

//visit generate_ham_cycle_helper
void generate_ham_cycle(int, int, int = 1);


#endif//_FUNCTIONS_HPP_
```

functions.cpp

```
#include "functions.hpp"
#include <iostream>
#include <fstream>
#include <iomanip>
#include <cmath>

std::vector< std::vector<int> > cycles;
std::ofstream fout;

//characteristic function of hamiltonian cycle
long long unsigned int charac_function(std::vector<int> &mas, std::vector<int>
    &f, int column)
{
    int index = 0;
    int result = 0;
    int horilength = 0;
    int vertlength = 0;
    int sum = 0;
    int size = mas.size();
    f.clear();
```

```
    for(int i = 1; i <= size / 2; ++i){ // ciklum i erkarutyamb
        //std::cout << " f(" << setw(2) << i << ") " << "= ";
        for(int j = 0; j < size; ++j) {       // hertov cikli j-rd elementic i
    erkarutyamb
            vertlength = fabs((mas[(j + i)%size] / column) - (mas[j] / column));
            horilength = fabs((mas[(j + i)%size] % column) - (mas[j] %
    column));
            result += vertlength + horilength;
            //std::cout << setw(2) << vertlength + horilength << "+";
        }
        //std::cout << "\b = " << result << ";" << std::endl;
        //fout << ' ' << result << std::endl;
        f.push_back(result);
        sum += result;
        result = 0;
    }
    return sum;
     //std::cout << sum << std::endl;
}
//print hamiltonian loop in grid
void print_vector_in_grid(std::vector<int> &mas, int column)
{
    const char* ch1 = "___";
    char ch2 = '|';
    int size = mas.size();
    int row = size / column;
    std::cout << std::endl << "\t";
    for(int i = 1; i <= size; ++i){
        //std::cout << std::setw(3) << i;
        std::cout << ' ';
        if(i % column == 0) {
            std::cout << std::endl << "\t";
            if( i / column != row ) {
                for(int j = 0; j < column; ++j) {
                    //std::cout << std::setw(3);
                    for(int k = 0; k < size; ++k) {
                        if(mas[k] == i - column + j) {
                            if((k > 0 && mas[k-1] == i + j) ||
                              (k < size-1 && mas[k+1] == i + j) ||
                              (k == 0  && mas[size-1] == i + j)) {
                                std::cout << ch2 << " ";
                            }
                            else {
                                //std::cout << " ";
                          std::cout << "  ";
                            }
                            break;
                        }
                    }
                    //std::cout << "  ";
```

```
                    std::cout << "  ";
                }
            }
            std::cout << std::endl << "\t";
        }
        else {
            //std::cout << ' ';
            for(int j = 0; j < size; ++j) {
                if(mas[j] == i - 1) {
                    if((j > 0   && mas[j-1] == i) ||
                       (j < size-1 && mas[j+1] == i)) {
                        std::cout << ch1;
                    }
                    else {
                        //std::cout << ' ';
                        std::cout << "   ";
                    }
                    break;
                }
            }
        }
    }
    std::cout << std::endl;
}
//distance between two cycles
int distance(int c1, int c2)
{
    int t = cycles[c1].size() - 1;
    if(c1 == c2) {
        return t + 1;
    }
    int c = 2;
    for(int i = 1; i < t; ++i){
        for(int j = 1; j < t; ++j){
            if(cycles[c1][i] == cycles[c2][j]){
                if(cycles[c1][i+1] == cycles[c2][j+1] || cycles[c1][i+1] ==
    cycles[c2][j-1]) {
                    ++c;
                }
                break;
            }
        }
    }
    return c;
}

//print hamiltonian loop
void print_vector(std::vector<int> &myvector)
{
    int t = myvector.size();
```

```
        for(int i = 0; i < t; ++i){
                std::cout << std::setw(3) << myvector[i] + 1 << ' ';
        }
        std::cout << std::endl;
}

//print characteristic function of hamiltonian cycle
void print_function(std::vector<int> &f)
{
      int t = f.size();
        for(int i = 0; i < t; ++i){
                std::cout << " f(" << std::setw(2) << i + 1 << ") " << "= " << f[i] << "   ";
        }
        std::cout << std::endl;
}

//print characteristic function of hamiltonian loop to file functions.in for the
      octave code
void print_function_to_file(std::vector<int> &myvector)
{
      int t = myvector.size();
        for(int i = 0; i < t; ++i){
                fout << std::setw(3) << myvector[i] << ' ';
        }
        fout << std::endl;
}

//Verify function's value for cycles
bool if_equal(std::vector<int> &f1, std::vector<int> &f2)
{
      int t = f1.size();
      bool b = true;
        for(int i = 0; i < t; ++i){
                if(f1[i] != f2[i]){
                        b = false;
                }
        }
      return b;
}

//generate hamiltonian cycles
int generate_ham_cycle_helper(int v, int& w, std::vector<int> &myvector, int**
      &matrix, bool* &visit, int& size, int& column)
{
        //static long long unsigned int cycle_count = 0; // count of hamiltonyan
      cycles
        if(myvector.size() == size && matrix[myvector.back()][w] == 1){
                //visit print_vector for hamiltonyan cycle
                //print_vector(myvector);
                cycles.push_back(myvector);
```

```
            //++cycle_count;
            //std::cout << "--------->  " << cycle_count << "  <---------" << std::endl;
        }
        //visit[v] = true;
        //for(int i = 0; i < size; ++i){            // optimize, not on all size
        //      if(matrix[v][i] && !visit[i]){
        //              myvector.push_back(i);
        //              generate_ham_cycle_helper(i, w, myvector);
        //      }
        //}
        //visit[v] = false;
        //myvector.pop_back();
        visit[v] = true;
    if(v && matrix[v][v - 1] && !visit[v - 1]) {
            myvector.push_back(v - 1);
            //generate_ham_cycle_helper(v - 1, w, myvector);
            generate_ham_cycle_helper(v - 1, w, myvector, matrix, visit, size,
    column);
            myvector.pop_back();
    }
    if(v >= column && matrix[v][v - column] && !visit[v - column]) {
                    myvector.push_back(v - column);
                    //generate_ham_cycle_helper(v - column, w, myvector);
                  generate_ham_cycle_helper(v - column, w, myvector, matrix, visit,
    size, column);
                    myvector.pop_back();
    }
    if(v != size - 1 && matrix[v][v + 1] && !visit[v + 1]) {
            myvector.push_back(v + 1);
            //generate_ham_cycle_helper(v + 1, w, myvector);
            generate_ham_cycle_helper(v + 1, w, myvector, matrix, visit, size,
    column);
            myvector.pop_back();
    }
    if(v + column < size  && matrix[v][v + column] && !visit[v + column]) {
                    myvector.push_back(v + column);
                    //generate_ham_cycle_helper(v + column, w, myvector);
                  generate_ham_cycle_helper(v + column, w, myvector, matrix, visit,
    size, column);
                    myvector.pop_back();
    }
        visit[v] = false;
        //return cycle_count; // return count of hamiltonyan cycles
                            // return (cycle_count / 2)
                          // when the matrix is simetrical;
    return 0;
}

// generate non simetric matrix for mxn grid
void generate_grid_matrix(int** &matrix, int m, int n)
```

```
{
        int size = m * n;
        matrix = new int* [size];
        for(int i = 0; i < size; ++i){
                matrix[i] = new int [size];
        }
        for(int i = 0; i < size; ++i){
                for(int j = 0; j < size; ++j){
                        matrix[i][j] = 0;
                }
      }
        for(int s, e, j = 0; j < m; ++j){
                s = j * n;
                e = j * n + n - 1;
                for(int i = s; i < e; ++i){
                        matrix[i][i + 1] = 1;
                        matrix[i + 1][i] = 1;
                }
        }
        for(int i = 0; i < size - n; ++i){
                matrix[i][i + n] = 1;
                matrix[i + n][i] = 1;
        }
        matrix[0][n] = 0;       // for non simetrical matrix
}

// visit generate_ham_cycles_helper
void generate_ham_cycle(int m, int n, int v)
{
      int size = m * n;
      int column = n;
        int** matrix;
        generate_grid_matrix(matrix, m, n);
        ///////////////////////////////////////////////////////
      fout.open("cuda/matrix.in");
      fout << "  " << size << std::endl;
        for(int i = 0; i < size; ++i){
                for(int j = 0; j < size; ++j){
                        fout << std::setw(3) << matrix[i][j];
                }
                fout << std::endl;
        }
      fout.close();
      fout.clear();
        ///////////////////////////////////////////////////////
      //print adjacency matrix for grid graph
        //for(int i = 0; i < size; ++i){
        //      for(int j = 0; j < size; ++j){
        //              std::cout << matrix[i][j] << ' ';
        //              //std::cout << matrix[i][j] << ", ";
```

```
  //      }
  //      std::cout << std::endl;
  //}
  ///////////////////////////////////////////////////////
  bool* visit; // = {0,};
visit = new bool[size];
  for(int i = 0; i < size; ++i){
        visit[i] = false;
  }

  int w = v - 1;                    // starting node of hamiltonian cycle
  std::vector<int> myvector;
  myvector.push_back(w);     // start of hamiltonian cycle
  std::cout << std::endl;
  generate_ham_cycle_helper(w, w, myvector, matrix, visit, size, column);
  std::cout << " Count of HAMILTONIAN CYCLES  " << cycles.size() <<
std::endl << std::endl;
  //std::cout << "\nCount of HAMILTONIAN CYCLES = "
  //       << generate_ham_cycle_helper(w, w, myvector, matrix, visit, size,
column)
  //       << std::endl << std::endl;
//print all hamiltonian cycles
//for(int i = 0; i < cycles.size(); ++i){
//      print_vector(cycles[i]);
//}
int s = cycles.size();
if(s) {
        int t,c1,c2;
        int min = distance(0, 0);
        long unsigned int sum1 = 0;
        long unsigned int sum2 = 0;
        std::vector<int> f1;
        std::vector<int> f2;
        c1 = 0;
        c2 = 0;
        //variables for non simetrical cycles
        bool b = true;
        unsigned int count = 0;
        fout.open("octave/functions.in");
        for(int i = 0; i < s; ++i){
                //sum and f1 functions of C1 cycle
                sum1 = charac_function(cycles[i], f1, column);
                for(int j = i + 1; j < s; ++j){
                        //sum and f2 functions of C2 cycle
                        sum2 = charac_function(cycles[j], f2, column);
                        if(if_equal(f1, f2)){
                                b = false;
                        // print cycles, which function are equaly
                        //      print_function(f1);
                        //      print_vector_in_grid(cycles[i], column);
```

```
			//	print_vector_in_grid(cycles[j], column);
			}
			t = distance(i,j);
			if(min > t){
				min = t;
				c1 = i;
				c2 = j;
			}
		}
		if(b) {
		//print non charac. functions to file functions.in
			print_function_to_file(f1);
			++count;
		}
		b = true;
	}
	fout.close();
	fout.clear();
	std::cout << " Count of non simetrical HAMILTONIAN CYCLES  "
		<< count << std::endl << std::endl;
	std::cout << " nm   " << size << std::endl;
	std::cout << " min distance   " << min << std::endl;
	std::cout << " max distance   " << 2 * size - 2 * min << std::endl;
	//print_vector(cycles[c1]);
	//print_vector(cycles[c2]);
	print_vector_in_grid(cycles[c1], column);
	print_vector_in_grid(cycles[c2], column);
	//for(int i = 0; i < s; ++i){
	//	for(int j = i + 1; j < s; ++j){
	//		t = distance(i,j);
	//		if(min == t){
	//			std::cout << "////////////////////////////////////////////////////////////" <<
std::endl;
	//			print_vector_in_grid(cycles[i], column);
	//			print_vector_in_grid(cycles[j], column);
	//		}
	//	}
	//}
}
////////////////////////////////////////////////////////////
std::cout << " Generated" << std::endl;
std::cout << " Adjacency matrix to file			cuda/matrix.in" << std::endl;
  std::cout << " Characteristic functions of loops to file  octave/functions.in"
<< std::endl;
//generate Kirchhorff matrix
  for(int i = 0; i < size; ++i){
	for(int j = 0; j < size; ++j){
		matrix[i][j] = -matrix[i][j];
	}
  }
```

```
    matrix[0][n] = -1;
    matrix[0][0] = 2;
    matrix[n - 1][n - 1] = 2;
    matrix[size - n][size - n] = 2;
    matrix[size - 1][size - 1] = 2;
     for(int i = 1; i < n-1; ++i){
            matrix[i][i] = 3;
            matrix[size - i - 1][size - i - 1] = 3;
     }
     for(int i = 1; i < m - 1; ++i){
            matrix[i * n][i * n] = 3;
            matrix[n + i * n - 1][n + i * n - 1] = 3;
     }
     for(int i = 1; i < m - 1; ++i){
            for(int j = 1; j < n - 1; ++j){
                    matrix[i * n + j][i * n + j] = 4;
            }
     }
    fout.open("octave/matrix.in");
     for(int i = 0; i < size; ++i){
            for(int j = 0; j < size; ++j){
                  fout << std::setw(3) << matrix[i][j];
            }
            fout << std::endl;
     }
    fout.close();
    fout.clear();
    std::cout << " Kirchhorff matrix to file              octave/matrix.in" << std::endl
    << std::endl;
     ////////////////////////////////////////////////////////
     for(int i = 0; i < size; ++i){
            delete [] matrix[i];
     }
     delete [] matrix;
}
```

octave.m

```
#number of spanning trees of grid graph
load octave/matrix.in
format long;
d = det(matrix(1:length(matrix)-1,1:length(matrix)-1));
printf("Count of spanning trees of grid graph = %i\n", d);
#det(matrix(2:length(matrix),2:length(matrix)))
#det(matrix(1:length(matrix)-1,2:length(matrix)))
#det(matrix(2:length(matrix),1:length(matrix)-1))
nm = length(matrix);
m = 2;
n = 2;
for i = 3:nm
    if( matrix(1,i) == -1 )
```

```
            n = i - 1;
            m = nm / n;
            break;
    endif
    if( matrix(i,1) == -1)
            m = i - 1;
            n = nm / m;
            break;
    endif
endfor

#print graphics of characteristic functions for hamiltonian loops
figure;
hold on;
grid on;
ylabel("Fc(k)");
tit = strcat("n=", num2str(n));
tit = strcat(tit,", m=");
tit = strcat(tit, num2str(m));
title(tit);
load octave/functions.in
[m n] = size(functions);
axis([1 n]);
color = [" - red,";" - green,";" - blue,";" - magenta,";" - cyan,";" - brown,"];
xlab = "k\n";
if( m < 8 )
    for i = 1:m
            xlab = strcat(xlab, strcat(" loop", num2str(i)));
            xlab = strcat(xlab, color(i,:));
    endfor
else
    str = strcat("number of graphics=", num2str(m));
    xlab = strcat(xlab, str);
endif
xlabel(xlab);

col = "rgbmcb"; # red, green, blue, magenta, cyan, brown
for i = 1:m
    color = col( mod(i - 1, 6) + 1);
    opt = strcat(color, "o-.");
    plot(functions(i,:), opt);
endfor
pause;
```

matrix_mult.cu

```
//Header from standard libraries
#include <fstream>

//size of blocks in grid
#define BLOCK_SIZE 16

//kernel function  to multiply c=a*b
__global__ void Muld(float* a,float* b,float* c,int n)
{
    int i = blockIdx.x * blockDim.x + threadIdx.x;
    int j = blockIdx.y * blockDim.y + threadIdx.y;

    float sum = 0;

    if(i>=n || j>=n){
            return;
    }

    for(int k = 0 ; k < n ; ++k) {
            sum = sum + a[i*n + k]*b[k*n + j];
    }

    c[i*n+j] = sum;
}

//host function  to multiply C=A*B
void Mul(float* A, float* B, int n,float* C)
{
      int size;
      // Load A and B to the device
      float* Ad;
      size = n * n * sizeof(float);
      cudaMalloc((void**)&Ad, size);
      cudaMemcpy(Ad, A, size, cudaMemcpyHostToDevice);
      float* Bd;
      size = n * n * sizeof(float);
      cudaMalloc((void**)&Bd, size);
      cudaMemcpy(Bd, B, size, cudaMemcpyHostToDevice);
      // Allocate C on the device
      float* Cd;
      size = n * n * sizeof(float);
      cudaMalloc((void**)&Cd, size);
      // Compute the execution configuration assuming
      // the matrix dimensions are multiples of BLOCK_SIZE
      dim3 dimBlock(BLOCK_SIZE, BLOCK_SIZE);
      dim3 dimGrid(n / dimBlock.x + (n%dimBlock.x!=0), n / dimBlock.y+(n
    %dimBlock.y!=0));
      // Launch the device computation
      Muld<<<dimGrid, dimBlock>>>(Ad, Bd, Cd , n);
```

```
    // Read C from the device
    cudaMemcpy(C, Cd, size, cudaMemcpyDeviceToHost);
    // Free device memory
    cudaFree(Ad);
    cudaFree(Bd);
    cudaFree(Cd);
}

int main(int argc,char* argv[])
{
   std::ifstream  fin(argv[1]);
   std::ofstream fout("out.txt");
   int n;
   fin >> n;
    float *hA = (float*)malloc(sizeof(float)*n*n);
    float *hB = (float*)malloc(sizeof(float)*n*n);
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            fin >> hA[i*n+j];
        }
    }
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            hB[i*n+j] = 0;
        }
        hB[i*n+i] = 1;
    }
   // in case, when count of input matrix > 1
   // hC = hA * hB;
   // Mul(hA,hB,n,hC);
   for(int i = 2; i <= n; i <<=1) {
        Mul(hA,hA,n,hA);
        if(i & n) {
             Mul(hA,hB,n,hB);
        }
   }
    for(int i = 0; i < n; ++i) {
        for(int j = 0; j < n; ++j) {
            fout << hB[i*n+j] << ' ';
        }
        fout << std::endl;
    }
   free(hA);
   free(hB);
    return 0;
}
```

7. Գրականություն

[1]. Н о в и к о в , Ф . А . Дискретная математика для программистов
Ф. А. Новиков. – СПб.: Питер, 2001. – с.

[2]. К р и с т о ф и д е с , Н . Теория графов. Алгоритмический подход
Н. Кристофидес. – М.: Мир, 1978. – с.

[3]. К о р м е н , Т . Алгоритмы. Построение и анализ
Т. Кормен, Ч. Лейзер, Р. Ривест. МЦНМО, Москва, 1999. – с.

[4]. А х о , А . Структуры данных и алгоритмы
А. Ахо, Дж. Хопкрофт, Дж. Ульман. – М.: Вильямс, 2000. – с.

[5]. Р е й н г о л ь д Э . Комбинаторные алгоритмы / Э. Рейнгольд,
Ю. Нивергельт, Н. Део. – М.: Мир, 1980. – с.

[6]. Харари Ф. Теория графов. — М.: Мир, 1973.

[7]. Беллман Р. Динамическое программирование. — М.: Изд-во иностранной литературы, 1960.

[8]. Лекции по теории графов / Емеличев В.А., Мельников О.И., Сарванов В.И., Тышкевич Р.И. – М.: Наука, 1990.

[10]. Зыков А.А. Основы теории графов. – М.: Наука, 1987.

[11]. Татт У. Теория графов. – М.: Наука, 1988.

[12]. Воблый В.А. Простая верхняя оценка для числа остовных деревьев регулярных графов. Дискретная математика, т. 20, вып. 3, 2008 ,

[13]. http://www.intuit.ru/department/algorithms/algocombi/8/1.html (Алгоритмы рекуррентных соотношений)

[14]. http://www.genfunc.ru/theory/rsol/ (Решение рекуррентных соотношений)

[15]. http://rriai.org.ru/poisk-v-glubinu.html (Поиск в глубину)

[16]. http://rriai.org.ru/poisk-v-shirinu-3.html (Поиск в ширину)

[17]. http://habrahabr.ru/blogs/algorithm/105705/ (Метод динамического программирования)