3. *Графи. Дървета. Обхождане на графи.*

* Краен ориентиран мутиграф

Нека V е крайно множество, $V = \{ v_1, v_2, \dots, v_n \}$, елементите на V наричаме **върхове**. Нека E е крайно множество, $E = \{ e_1, e_2, \dots, e_m \}$, елементите на E наричаме **ребра**.
Краен ориентиран мултиграф се нарича тройката $G (V, E, f_G)$ с функция $f_G : E \rightarrow V \times V$ (на всяко ребро се съпоставя наредена двойка върхове). f_G наричаме **свързваща функция**.

За по нагледно представяне на графите ще използваме диаграми – върховете означаваме като точки и всяко ребро означаваме със стрелка от един врък към друг.

Ребрата $e \in E$, такива че $f_G (e) = (v, v)$, $v \in V$ наричаме **примки**.

Нека $G (V, E, f_G)$ е краен ориентиран мултиграф и f_G е инекция. Тогава G се нарича **краен ориентиран граф**. Ясно е, че в такъв случай множеството E може да бъде определено като подмножество на $V \times V$. И така при задаване на краен ориентиран граф G свързваща функция не е необходима – графът се означава с $G (V, E)$ и се задава с множество от върхове V и множество от ребра $E \subseteq V \times V$.

* Краен неориентиран мутиграф

Нека $G (V, E)$ е краен ориентиран мултиграф и релацията $E \subseteq V \times V$ е рефлексивна и симетрична. В такъв случай G се нарича **краен неориентиран граф** или просто **граф**. При изобразяване на крайни неориентирани графи не рисуваме примките и стрелките между два върха заменяме с една отсечка.

Ако в краен неориентиран граф допуснем многократни ребра, то отново ще е необходима свързваща функция и в този случай графът наричаме **краен неориентиран мултиграф**.

Нека $G (V, E)$ е граф. Върховете v_i, v_j наричаме **съседи**, ако $(v_i, v_j) \in E$. Ако графът G е ориентиран, казваме че v_i е **баща** на v_j или, че v_j е **син** на v_i. Още казваме, че v_i и v_j са **краища** на реброто (v_i, v_j).

Нека $G (V, E)$ е краен неориентиран граф и $v \in V$. Дефинираме **степен** на върха v - $d(v)$ = броят на ребрата, на които v е край.
Нека $G (V, E)$ е краен ориентиран граф и $v \in V$. Дефинираме **полустепен на изхода** на върха v – $d^- (v)$ = броят на ребрата, излизащи от v и **полустепен на входа** на върха v – $d^+ (v)$ = броят на ребрата, завършващи във v.

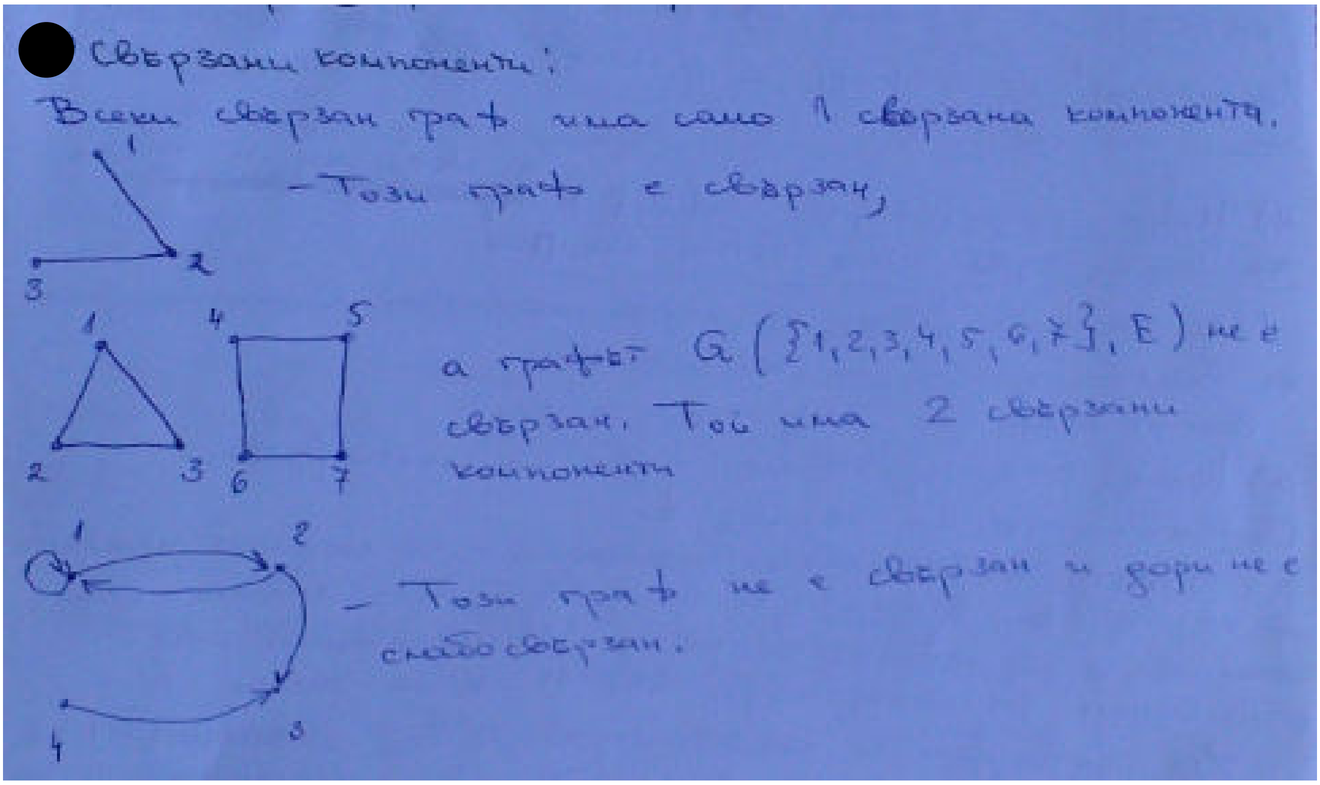
Нека $G (V, E, f_G)$ е краен ориентиран мултиграф. Редицата $v_{i_0}, v_{i_1}, \dots, v_{i_L}$, $L \geq 0$, такава че за всяко $j \in \{ 0, 1, \dots, L-1 \}$ съществува $e \in E$, такова че $f_G (e) = (v_{i_j}, v_{i_{j+1}})$, наричаме **маршрут** от v_{i_0} до v_{i_L} в графа G с дължина L. При $v_{i_0} = v_{i_L}$, маршрутът наричаме **контур**.

Нека G (V, E) е краен неориентиран граф.
Редицата $v_{i_0}, v_{i_1}, \dots, v_{i_L}$, $L \geq 0$, такава че за всяко $j \in \{0, 1, \dots, L-1\}$, $(v_{i_j}, v_{i_{j+1}}) \in E$ и за всяко $j \in \{1, \dots, L-1\}$, $v_{i_{j-1}} \neq v_{i_{j+1}}$, наричаме **път** от v_{i_0} до v_{i_L} в графа G с дължина L. При $v_{i_0} = v_{i_L}$ и $L \geq 3$, маршрутът наричаме **цикъл**. Ясно е, че по тази дефиниция считаме, че има

тривиален път с дължина 0 от всеки връх v_i до v_i. Ще отбележим, че с дефинирането на тривиален път отчитаме примките в неориентирания граф, но им даваме тежест нула при образуване на дължината на пътищата.

* **Свързаност**

Графът G (V, E) се нарича **свързан**, ако за всеки $v_i, v_j \in V$ съществува път от v_i до v_j. Дефиницията е приложима и за ориентирания случай, както и за случая с мултиграф, със замяната на понятието път с понятието маршрут, но тъй като изискването е прекалено силно, ще въведем алтернативно понятие.
Ориентираният граф G (V, E) наричаме **слабо свързан**, ако за всеки $v_i, v_j \in V$ съществува път от v_i до v_j или от v_j до v_i.



* Казваме, че графът D (V, E) е **дърво**, ако D е свързан граф без цикли.

Ще направим индуктивна дефиниция на **кореново дърво** (с корен r):

1. База – графът D ({ r }, ∅) е кореново дърво с корен r и единствен **лист** r.
2. Предположение – нека D (V, E) е кореново дърво с корен r и листа $l_1, l_2, \dots, l_k$.
3. Стъпка – нека $u \in V$, $w \notin V$. Тогава $D' (V', E') = D' (V \cup \{w\}, E \cup \{(u, w)\})$ е също дърво с корен r. Ако $u = l_i$ за $i \in \{1, 2, \dots, k\}$, то неговите листа са $l_1, \dots, l_{i-1}, w, l_{i+1}, \dots, l_k$, в противен случай, те са $l_1, \dots, l_k, w$.

Операцията в индукционната стъпка наричаме **присъединяване на връх**.

Теорема: Всяко кореново дърво е дърво.
Доказателство: Индукция по построението.

1. База – кореновото дърво D ({ r }, ∅) е свързан граф, тъй като съществува тривиален път от r до r и няма цикли, тъй като няма ребра, така че D ({ r}, ∅) е дърво.
2. Предположение – нека кореновото дърво D (V, E) е дърво.
3. Стъпка – нека $u \in V$, $w \notin V$. Ще покажем, че кореновото дърво D′ (V′, E′) = D′ (V ∪ { w }, E ∪ { (u, w)}) е дърво.
   Нека $v_i, v_j \in D'$. Ако $v_i, v_j \in D$, то по индукционното предположение има път от v_i до v_j. Ако $v_i = v_j = w$, то съществува тривиален път от v_i до v_j. Ако $v_j = w$, $v_i \neq w$, то от индукционното предположение съществува път от v_i до u и като добавим реброто (u, w) получаваме път от v_i до w – не е възможно w да съвпада с върха преди u, тъй като $w \notin V$.
   И така D′ е свързан граф. Да допуснем, че в D′ има цикли. Тъй като по индукционното предположение в D няма цикли, то реброто (u, w) със сигурност участва в цикъл, което е противоречие, тъй като w е край единствено на реброто (u, w) и следователно, съгласно дефиницията на път, w не може да участва в цикъл. Така D′ е свързан граф без цикли, т.е. D′ е дърво.

Ще отбележим, че всяко дърво можем да предефинираме като кореново, ако изберем произволен негов връх за корен.

Теорема: Нека D (V, E) е дърво с корен r. Тогава $|V| = |E| + 1$.
Доказателство: Индукция по построението.

1. База – за кореновото дърво D ({ r }, ∅) имаме $|V| = 1$, $|E| = 0$ и тогава $|V| = |E| + 1$.
2. Предположение – нека за кореновото дърво D (V, E) имаме $|V| = |E| + 1$.
3. Стъпка – нека $u \in V$, $w \notin V$. Тогава за кореновото дърво D′ (V′, E′) = D′ (V ∪ { w }, E ∪ { (u, w)}) имаме $|V'| = |V| + 1$, $|E'| = |E| + 1 = |V| + 2 \Rightarrow |V'| = |E'| + 1$.

Нека G (V, E) е граф, а D (V, E′), $E' \subseteq E$ е дърво. Тогава D се нарича **покриващо дърво** на графа G.

Теорема: Графът G (V, E) притежава покриващо дърво тогава и само тогава, когато G е свързан.
Доказателство: Ясно е, че ако G притежава покриващо дърво D, то G е свързан, тъй като в D, а тогава и в G ($E' \subseteq E$) има път от всеки връх до всеки друг.

Нека G е свързан граф. Ще опишем алгоритъм, който построява покриващо дърво на G:

1. H = G.
2. докато в H има цикли
   H = H – ребро от цикъла.

Твърдим, че след изпълнение на алгоритъма H е покриващо дърво на G. Действително, алгоритъмът винаги завършва, тъй като в G има краен брой ребра, а съществуването на цикъл предполага наличието на поне три ребра. Ясно е, че H е граф с върхове V и ребра $E' \subseteq E$. От самия алгоритъм се вижда, че H няма цикли.
Ще покажем, че на всяка стъпка H остава свързан граф.
Да допуснем, че на някоя стъпка премахването на реброто (v_i, v_j) води до несвързан граф. Нека v_k, v_l са върхове между които няма път. Тъй като на предната стъпка графът е бил свързан, то съществува път между v_k и v_l, който минава през реброто (v_i, v_j), но в такъв случай можем да използваме остатъка от цикъла (или подходяща негова част) и да получим път между v_k и v_l в новия граф, което е противоречие. И така H (V, E′) е свързан граф без цикли, т.е. покриващо дърво на G.

* Под **обхождане на граф** ще разбираме процедура, която по определени правила "посещава" всички върхове на графа.

Даден е краен неориентиран граф G (V, E). В резултат на **обхождането в ширина** V се разбива на **нива на обхождане** $L_0, L_1, \ldots, L_k$ по следния начин:

1. избираме начален връх на обхождането $r \in V$. "Обхождаме" r. Нека $L_0 = \{ r \}$, $i = 0$.
2. Образуваме нивото $L_{i+1} = \{ v \mid v \in V, v$ – необходен, съществува $w \in L_i$, така че $(w, v) \in E \}$. "Обхождаме" всички върхове в L_{i+1}.
3. Ако $L_{i+1} \neq \varnothing$, тогава i++, премини към 2. Иначе край.

Ще отбележим, че обхождането ще посети всички върхове точно тогава, когато графът е свързан.
Ще приложим тази алгоритмична схема, за да построим покриващо дърво на свързания граф G (V, E):

1. Избираме начален връх на обхождането $r \in V$.
   Образуваме $D = (V_D, E_D)$, $V_D = \{ r \}$, $E_D = \varnothing$.
   Нека $L_0 = \{ r \}$, $i = 0$.
2. Образуваме нивото $L_{i+1} = \{ v \mid v \in V, v \notin V_D$, съществува $w \in L_i$, такъв че $(w, v) \in E \}$. $E_D = E_D \cup \{ (w, v) \mid w \in L_i, v \in L_{i+1}\}$, като за всяко $v \in L_{i+1}$ се добавя точно едно (w, v), такова че $w \in L_i$, $V_D = V_D \cup L_{i+1}$.
3. Ако $L_{i+1} \neq \varnothing$, тогава i++, премини към 2. Иначе край.

След края на изпълнението $V_D = V$, тъй като G е свързан и получаваме кореново дърво D (V, E_D), $E_D \subseteq E$, което е покриващо дърво на графа G.

Даден е краен неориентиран граф G (V, E). При схемата **обхождане в дълбочина** основни понятия са **текущ връх** t и **баща** на върха t – p (t).

1. Избираме начален връх $r \in V$. "Обхождаме" r.
   Обявяваме $t = r$, p (t) е неопределен.

2. Търсим необходен съсед v на текущия връх t.
   Ако има такъв v, тогава p (v) = t, t = v, “обхождаме” v, премини към 2.
   Ако няма такъв v и $t \neq r$, t = p (t), премини към 2.
   Ако няма такъв v и t = r, край.

И в тази схема ще се обходят всички върхове точно тогава, когато графът е свързан. Ще приложим алгоритмичната схема за решаване на задачата за построяване на покриващо дърво на свързания граф G (V, E):

1. Избираме начален връх $r \in V$, образуваме D (V_D, E_D), $V_D = \{ r \}$, $E_D = \varnothing$, t = r, p (t) – неопределен.
2. Търсим необходен съсед v на текущия връх t.
   Ако има такъв v, тогава p (v) = t, t = v, $V_D = V_D \cup \{ v\}$, $E_D = E_D \cup \{ (t, v)\}$, премини към 2.
   Ако няма такъв v и $t \neq r$, t = p (t), премини към 2.
   Ако няма такъв v и t = r, край.

След края на изпълнението $V_D = V$, тъй като G е свързан и получаваме кореново дърво D (V, E_D), $E_D \subseteq E$, което е покриващо дърво на графа G.

И двата алгоритъма (в ширина и в дълбочина) строят коренови дървета и тогава те могат да бъдат използвани за превръщане на произволни дървета в коренови.

● Ойлерови обхождания на мулти граф.

Деф: Ойлеров път в свързания мулти-граф G(V, E) наричаме път, който минава еднократно през всяко ребро на мултиграфа. Ако Ойлеровият път има начало и край, които съвпадат, тогава той се нарича Ойлеров цикъл. Мулти граф, ребрата на който ~~съвпадат~~ образуват Ойлеров цикъл, наричаме Ойлеров.

В сила е следната

Th: Свързаният мулти граф G(V, E) е Ойлеров тогава и само тогава, когато всеки връх на G е с четна степен

Д-во:

1) Нека ребрата на $G(V,E)$ образуват Ойлеров цикъл. Тогава всеки връх има четна степен, защото при обхождане на мултиграфа по Ойлеровия цикъл, на всяко ребро, "влизащо" във връх v_i съответства ребро (различно от първото, според дефиницията за път), "излизащо" от връх v_i, а цикълът съдържа всички ребра точно по един път.

2) Нека мултиграфът $G(V,E)$ е свързан и такъв, ~~образуват Ойлеров~~ че всеки връх има четна степен.

Ще опишем процедура, която построява Ойлеров цикъл от всички ребра на мултиграфа.

Започваме със следните стъпки:

а) Нека τ е произволен връх на мултиграфа. Обявяваме го за текущ t.

б) Докато е възможно, строим път по следното правило: от върха t, в който се намираме, преминаваме към съседен v по необходено ребро. Използваното ребро обявяваме за обходено, а върхът v за текущ.

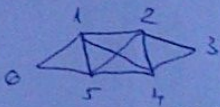
Да допуснем, че когато правилото от стъпка (б) не е вече приложимо, то $t \neq \tau$. Тогава ще се окаже, че степента на t е нечетна, тъй като, влизайки за последен път в t сме отбелязали като обходено едно ребро, а при всяко предно минаване през този връх сме отбелязвали като обходени по 2 негови ребра. Това е противоречие. Следователно, когато правилото от стъпка (б) вече не е приложимо, $t = \tau$ и всички обходени ребра образуват цикъл C.

в) Ако цикълът C съдържа всички ребра - край на обхождането. Построен е Ойлеров цикъл.

г) В противен случай премахваме от мултиграфа ребрата на намерения цикъл. Полученият мулти-граф е такъв, че всеки връх е с четна степен. Търсим в построения цикъл връх τ', от който излиза поне едно необходено ребро. Ако допуснем, че такъв връх няма, при условието, че не всички ребра са обходени, ще получим противоречие със свързаността на мултиграфа.

Повтаряме действията от стъпка (б) с начален връх $t = t'$ и построяваме нов цикъл. Сега разкъсваме двата цикъла в общия им връх t' и отъждествяването на получените краища два по два, получаваме нов цикъл C и преминаваме към (в).

Алгоритъмът ще завърши работа, когато всички ребра се изчерпят и построява Ойлеров цикъл. Следователно мултиграфът ще е Ойлеров.



Следствие:

Свързаният мулти граф $G(V, E)$ съдържа Ойлеров път тогава и само тогава, когато има точно 2 върха с нечетна степен

Д-во:

1) Нека v_i и v_j са върховете с нечетна степен.

Добавяме ребро (v_i, v_j) в графа. Полученият мулти граф е Ойлеров и следователно можем да построим Ойлеров цикъл. Отстраняваме добавеното ребро и получаваме път от v_i и v_j, който съдържа всички ребра на графа $G(V, E)$ точно по веднъж $\Rightarrow$ е Ойлеров път.

2) Нека ребрата на $G(V, E)$ образуват Ойлеров път от v_i до v_j.

Добавяме реброто (v_i, v_j) и този път се превръща в Ойлеров цикъл за новополученият мулти граф и $\Rightarrow$ в него всички върхове са с четна степен. Добавянето на реброто (v_i, v_j) е увеличило с 1 само степените на v_i и v_j. Следователно в G $\forall$ върхове са с четна степен, с изключение на тези два върха.