3. Графи. Дървета. Обхождане на графи.

1. **Краен ориентиран и неориентиран (мулти)граф**

   Деф: Краен неориентиран граф

   Краен неориентиран граф наричаме наредената двойка множества $G = (V, E)$, където $V \neq \emptyset$ е крайно множество от върхове, а E е множество с дву-елементни подмножества на V, които наричаме ребра. $E \subseteq \{X \subseteq V : |X| = 2\}$

   Деф: Мултиграф

   Мултиграф е наредена тройка $G = (V, E, f_G)$, където V е непразно множество, чиито елементи се наричат върхове, E е множество, чиито елементи се наричат ребра, $V \cap E = \emptyset$ и $f_G: E \to \{X \subseteq V : |X| = 2\}$ е *свързваща функция*

   Деф: Краен ориентиран граф

   Ориентиран граф е наредена двойка $G = (V, E)$, където V е непразно множество, чиито елементи се наричат върхове, E е множество, чиито елементи се наричат ребра, като

   $$E \subseteq (V \times V) \setminus \{(u, u) \mid u \in V\}$$

   Деф: Краен ориентиран мултиграф

   Ориентиран мултиграф е наредена тройка $G = (V, E, f_G)$, където V е непразно множество, чиито елементи се наричат върхове, E е множество, чиито елементи се наричат ребра, $V \cap E = \emptyset$ и

   $f_G: E \to V \times V$ е свързващата функция

2. **Път (цикъл) в ориентиран и неориентиран мултиграф**

   Деф: Път

   Нека $G = (V, E, f_G)$ е мултиграф. Ориентиран път в G наричаме всяка алтернираща редица от върхове и ребра, за някое $t \geq 0$:

   $$p = (u_{i_0}, e_{k_0}, u_{i_1}, e_{k_1}, u_{i_2}, \ldots, u_{i_{t-1}}, e_{k_{t-1}}, u_{i_t}),$$

   Където $u_{i_p} \in V$ за $0 \leq p \leq t$, $e_{k_p} \in E$ за $0 \leq p \leq t - 1$ и освен това е изпълнено $f_G(e_{k_p}) = (u_{i_p}, u_{i_{p+1}})$ за $0 \leq p \leq t - 1$.

   Връх u_{i_0} се нарича *начало на пътя*, а връх u_{i_t} се нарича *край на пътя.* Останалите върхове са *вътрешни върхове на пътя*. *Дължината* на пътя е броят на ребрата в него, бележим с $|p|$

   Деф: Цикъл

   Нека $G = (V, E, f_G)$ е мултиграф и p е ориентиран път в G, където

   $$p = (u_{i_0}, e_{k_0}, u_{i_1}, e_{k_1}, u_{i_2}, \ldots, u_{i_{t-1}}, e_{k_{t-1}}, u_{i_t})$$

   Казваме, че p е ориентиран цикъл, ако $u_{i_0} = u_{i_1}$. Казваме, че p е прост ориентиран цикъл, ако p е ориентиран цикъл с поне едно ребро и освен това, всички елементи освен $u_{i_0} = u_{i_1}$ са уникални.

   Ако $G = (V, E, f_G)$ е ориентиран мултиграф, то горедефинираните структури наричаме съответно ориентиран път и ориентиран цикъл.

3. **Свързаност и свързани компоненти на граф**

   Деф: Свързаност в граф. Свързан граф

   Нека $G = (V, E)$ е граф. За всеки два върха $u, v \in V$ казваме, че u и v са *свързани*, ако съществува $u - v$ път. G е *свързан граф*, ако всеки два върха в него са свързани.

   Деф: Слабо свързан граф

   Ориентиран граф е слабо свързан, ако между всеки два върха има път в поне една от двете посоки

   Деф: Силно свързан граф

   Ориентиран граф е силно свързан, ако между всеки два върха има път между върховете и в двете посоки

Деф: Релация на достижимост

Нека $G = (V, E)$ е граф. *Релация на достижимост върху* G наричаме $Q_G \subseteq V \times V$, т.ч. $\forall u, v \in V: uQv \leftrightarrow u$ и v са свързани

Деф: Свързани компоненти

Нека $G = (V, E)$ е граф и Q_G е релацията на достижимост върху G. Подграфите на G, индуцирани от класовете на еквивалентност на Q_G се наричат *свързаните компоненти* на G

Друго определение: Свързаните компоненти на граф са максималните по включване свързани подграфи.

4. Дефиниция на дърво и кореново дърво. Всяко кореново дърво е дърво и $|V| = |E| + 1$. Покриващо дърво на граф

Деф: Дърво

Дърво е всеки граф, който е свързан и ацикличен (няма цикли)ю

Деф: Кореново дърво

- База: всеки тривиален граф $T = (\{u\}, \emptyset)$ е кореново дърво с корен u и множество от листа $\{u\}$
- ИС: Нека $\mathrm{T} = (V, E)$ е дърво с корен r и листа $W = e_1, \dots, e_k$. Нека $v \in V$ и $u \notin V$. Тогава $\mathrm{T}' = \big(V \cup \{u\}, E \cup (v, u)\big)$ е дърво с корен r и листа $\big(\big(W \setminus \{v\}\big) \cup \{u\}\big)$
- Няма други коренови дървета

Твърдение: Всяко кореново дърво е дърво

Д-во:

Индукция по построението на кореново дърво:

- База: За $\mathrm{T} = (\{r\}, \emptyset)$ е в сила, че е свързан граф без цикли, защото има само един възел
- ИХ: Нека кореновото дърво $D = (V, E)$ е свързан граф без цикли.
- Стъпка: Нека $T' - (V \cup \{u\}, E \cup \{(v, u)\}), v \in V, u \notin V$.

  Ще покажем, че T' е дърво. Нека v_1 и v_2 са произволни от $V' = V \cup \{u\}$
  - Ако $v_1 = v_2$, то имаме път - тривиален път с дължина нула от v_1 до v_2.
  - Ако $v_1, v_2 \in V$, то v_1, v_2 са върхове в графа T. Съгласно ИХ T е свързан граф. Следователно има път в D от v_1 до v_2 и той се запазва в T'.
  - Ако $v_1 \neq v_2 \in V'$, то или $v_1 = u$, или $v_2 = u$. БОО нека $v_2 = u$. Тогава v_1, v са върхове от T и между тях има ацикличен път в T, който е ацикличен път и в T', т.е. $v_1 = w_1, w_2, \dots, w_k = v$. Така $v_1 = w_1, w_2, \dots, w_k = v, u$ е път в T' и значи T' е свързан. Ако допуснем, че има цикъл в него, т.е. $w_{i_1}, \dots, w_{i_k}$, то всеки връх в цикъла участва като край на две различни ребра $\big(\mathrm{w}_{i-1}, \mathrm{w}_\mathrm{i}\big), (\mathrm{w}_\mathrm{i}, \mathrm{w}_{\mathrm{i}+1})$. Новият връг u е край на едно единствено ребро, така че няма как да участва в цикъла, а пътя $w_1, \dots, w_k$ е ацикличен път в T. Противоречие! Следователно T' е свързан ацикличен граф.

Теорема: Нека $T = (V, E)$ е кореново дърво. Тогава $|V| = |E| + 1$

Д-во: Индукция по построението на кореново дърво

- База: За $T = (\{r\}, \emptyset)$, то $|V| = |E| + 1 = 0 + 1 = 1$
- ИХ: За $\mathrm{T} = (V, E)$ е в сила
- Стъпка: За $T' = \big(V \cup \{u\}, E \cup \{(v, u)\}\big), v \in V, u \notin V$:

$$|V \cup \{u\}| = |V| + 1 \underbrace{=}_{\text{ИХ}} |E| + 1 + 1 = \big|E \cup \{(v, u)\}\big| + 1$$

$$|E \cup \{(v, u)\}| = |E| + 1 \underbrace{=}_{\text{ИХ}} |V|$$

Деф: Покриващо дърво

Нека $G = (V, E)$ е граф. Покриващо дърво на G наричаме дърво $D = (V, E')$, където $\mathrm{E}' \subseteq \mathrm{E}$.

Теорема: Граф $G = (V, E)$ има покриващо дърво т.с.т.к. е свързан граф.

5. **Обхождания на графи**

Деф: Стек (индуктивно)

База: Празният стек означаваме с {}. За него дефинираме $pop(\{\}) = \{\}$, top$(\{\}) = \{\}$.

Нека S е стек и x е елемент. Тогава $S' = push(S, x)$ е стек, като $top(S') = x$, $pop(X') = S$.

Обхождане в дълбочина (DFS):

Идея: "Докато можем, вървим напред. Когато няма как да продължим - връщаме се една стъпка назад"

Нека $G = (V, E)$ е произволен граф с n върха и $v_0 \in V$ е начален връх. Ще построим списък DFS $= \Big((v_0, \emptyset), \big(v_1, s(v_1)\big), \dots, \big(v_n, s(v_n)\big)\Big)$, който включва всички върхове $v_1, \dots, v_n$ и техните непосредствени предшественици в покриващото дърво с корен v_0: $D = (V, E')$, където $\mathrm{E}' = \Big\{\big(v_i, s(v_i)\big) \mid i = 1, \dots, n\Big\}$.

Ще използваме помощен стек S и текущ връх t. В началото $DFS = \big((v_0, \emptyset)\big)$ и $S = \{\}$, $t = v_0$ и v_0 е обходен.

Докато има необходен връх във V:

- Ако има необходен съсед v на t, обявяваме v за обходен, добавяме (v, t) към списък $DFS = DFS \cup \{(v, t)\}$, добавяме t към стека $S \coloneqq push(S, t)$, текущ връх става $t \coloneqq v$.
- Ако няма необходен съсед на t, връщаме се една стъпка назад: $t = top(S), S \coloneqq pop(S)$

Деф: Опашка (индуктивно)

База: Празната опашка означаваме с {}. За нея $pop(\{\}) = top(\{\}) = \{\}$.

Нека Q е опашка и x е елемент. Тогава $Q' = push(Q, x)$ е опашка.

$$top(Q') = \begin{cases} x, & \text{ако } Q = \{\} \\ top(Q), & \text{ако } Q \neq \{\} \end{cases}$$

$$pop(Q') = \begin{cases} \{\}, & \text{ако } Q = \{\} \\ push(pop(Q), x), & \text{ако } Q \neq \{\} \end{cases}$$

Обхождане в широчина (BFS):

Идея: "Докато можем върви в страни. След това минаваме едно ниво надолу".

Нека $G = (V, E)$ е произволен граф с n върха и $v_0 \in V$ е начален връх. Ще построим списък BFS $= \Big((v_0, \emptyset), \big(v_1, s(v_1)\big), \dots, \big(v_n, s(v_n)\big)\Big)$, който включва всички върхове $v_1, \dots, v_n$ и техните непосредствени предшественици в покриващото дърво с корен v_0: $D = (V, E')$, където $\mathrm{E}' = \Big\{\big(v_i, s(v_i)\big) \mid i = 1, \dots, n\Big\}$.

Ще използваме помощна опашка Q и текущ връх t. В началото $BFS = \big((v_0, \emptyset)\big)$ и $Q = \{\}, t = v_0$ и v_0 е обходен.

Докато има необходен връх във V:

- Ако има необходен съсед v на t, обявяваме v за обходен, добавяме (v, t) към списък $BFS = BFS \cup \big\{(v, t)\big\}$, добавяме v към опашката $Q \coloneqq push(Q, v)$
- Ако няма необходен съсед на t, текущият връх става първият връх на опашката: $t = top(Q), Q \coloneqq pop(Q)$

Двете обхождания могат да бъдат модифицирани да намират път между два върха в граф като зададем единият връх да бъде началния и модифицираме условието (докато) да бъде докато се срещне вторият връх.

Теорема: Обхождане в широчина намира най-късите пътища от началния връх до всички останали върхове в графа.

6. **Ойлерови обхождания на мултиграф. Теореми за съществуване на Ойлеров цикъл и Ойлеров път.**

Деф: Ойлеров граф. Ойлеров цикъл

Ойлеров път в свързан мултиграф G наричаме път, минаващ само по веднъж през всяко ребро в графа. Ако началния и крайния връх в графа съвпадат, то имаме Ойлеров цикъл и графът се нарича Ойлеров граф.

Теорема: Нека $G = (V, E)$ е свързан граф. G е Ойлеров т.с.т.к. всеки връх в G има четна степен.
Д-во:

$\Rightarrow$Нека G е Ойлеров и нека $v = v_0, v_1, \dots, v_k = v$ е Ойлеров цикъл в G. Всяко срещане на връх u в редицата $v_0, \dots, v_{k-1}$ съответства на две ребра през u:

- Ако $u = v_0$, ребрата са $\{v_0, v_1\}$ и $\{v_{k-1}, v_0\}$
- Ако $u = v_i$ и $i > 0$, ребрата са $\{v_1, v_{i+1}\}$ и $\{v_{i-1}, v_i\}$.

Тъй като всички ребра в графа се срещат точно по веднъж в цикъла, степента на произволен връх $u \in V$ е равна на два пъти броя на срещанията на u в цикъла. Така $d(u)$ е четно число.

$\Leftarrow$ Нека сега G е свързан граф, в който всеки връх има четна степен и $u \in V$. Ще построим Ойлеров цикъл C през u.

Обявяваме u за текущ връх t, $C = (u)$

1. Докато има необходено ребро $e = \{t, v\}$ през t, добавяме t към C, обявяваме e за обходено и v за текущ връх t.

   На всяко преминаване през 1. броят на необходените ребра намалява, цикълът ще завърши след краен брой стъпки k и $C = (u = v_0, v_1, \dots, v_k)$.

   Да допуснем, че $u \neq v_k$. Алгоритъмът завършва, всички ребра през v_k са обходени и на $k +$ 1-ва стъпка няма необходени ребра през v_k. На стъпка, на която v_k става текущ, обхождаме едно ребро през v_k и броят на обходените ребра през v_k е нечетен. На следващата стъпка, ако v_k престане да е текущ, обхождаме второ ребро през v_k, броят на обходените ребра през v_k става четен. Така на последната стъпка, на която v_k става текущ, броят на обходените ребра през v_k е нечетен, но тогава алгоритъмът завършва, защото няма как да се избере следващ текущ връх, всички ребра през v_k са обходени. Следователно v_k има нечетен брой ребра. Противоречие!

2. Следователно $v_k = u$ и C е цикъл. Ако C съдържа всички ребра, C е Ойлеров цикъл. Иначе C съдържа поне един връх v_i, който е край на необходено ребро (от свързаността на графа). Повтаряме цикъла 1. с начален връх v_i, като строим нов цикъл
   $C' = (v_i = u_0\, u_1, \dots, u_m = v_i)$.
   Заменяме C с цикъла:
   $C = (v_0, \dots, v_{i-1}, v_i = u_0, u_1, \dots, u_m = v_i, v_{i+1}, \dots, v_k)$ и отново се връщаме на 2.

Вторият цикъл също ще завърши, защото при всяко преминаване през него, броят на необходимите ребра намалява. Така в крайна сметка C съдържа Ойлеров цикъл.

Следствие: Свързаният мултиграф G съдържа Ойлеров път т.с.т.к. всички върхове са от четна степен, като само два са от нечетна.