Дерева

Модуль 2 Лекція 5

План

- Основні поняття та властивості дерев
- Бінарні дерева пошуку
- Обхід бінарних дерев
- Остовні дерева
- Мінімальні остовні дерева

Умовні позначення

- визначення

- приклад

- примітка

- важливо!

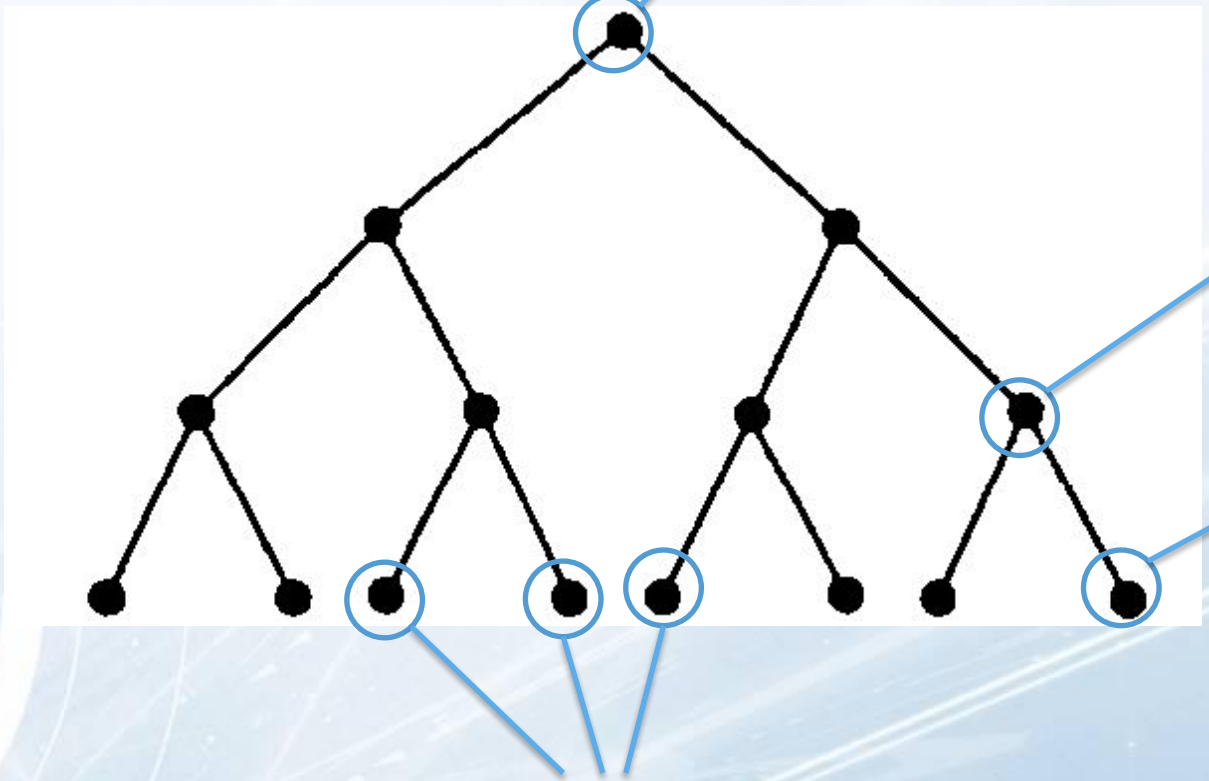
- теорема

Основні поняття та властивості дерев

Дерево - це зв'язний граф без циклів.

Орієнтоване дерево – це вільний від петель орієнтований граф, співвіднесений граф якого є деревом.

Вершина в самій верхній частині називається ***коренем*** дерева. Вершину v орієнтованого дерева називають ***потомком*** вершини u, якщо існує шлях з u в v. В цьому випадку вершину u називають ***предком*** вершини v, а якщо довжина шляху з u в v дорівнює 1, то вершину v називають ***сином*** вершини u, яка при цьому називається ***батьком*** вершини v. Вершина, що не має потомків називається ***листом***.



План

ТЕОРЕМА 8.1. Наступні твердження еквівалентні:

а) граф G – дерево

б) граф G – зв'язний і $v = e + 1$, де v – кількість вершин, а e – кількість ребер графа

в) для кожної пари різних вершин a та b існує єдиний шлях з a в b

г) граф G – ациклічний і $v = e + 1$.

ТЕОРЕМА 8.2. Для орграфа G еквівалентні твердження:

а) G – кореневе орієнтоване дерево

б) G має єдиний такий елемент v_0, що для будь-якої вершини a графа G існує єдиний орієнтований шлях з v_0 в a.

в) співвіднесений граф графа G зв'язаний і G містить єдиний елемент v' такий, що indeg $(v') = 0$, і для будь якої іншої вершини a графа G маємо indeg $(a) = 1$

г) співвіднесений граф графа G зв'язаний, і G містить єдиний елемент v_0 такий, що для будь-якої вершини a графа G існує єдиний шлях з v_0 в a.

В орієнтованому дереві ***рівень вершини*** v – це довжина шляху від кореня дерева до цієї вершини.

Висота орієнтованого дерева – це довжина найдовшого шляху від кореня до листа.

m-арним орієнтованим деревом називається таке орієнтоване дерево, в якому $\text{outdeg}(v) \le m$ для кожної його вершини v. Предок має не більше m потомків.

Повним m-арним орієнтованим деревом називається таке орієнтоване дерево, в якому $\text{outdeg}(v) = m$ для кожної вершини v, що не є листом, і кожний лист знаходиться на одному й тому ж рівні. Таким чином кожен предок має m потомків.

m-арне орієнтоване дерево висоти h називається ***збалансованим*** (повним або майже повним), якщо рівень кожного листа дорівнює h або h-1.

ТЕОРЕМА 8.3. Якщо повне m-арне орієнтоване дерево має n вершин і i внутрішніх вершин, то $n = mi + 1$, $i = \frac{n-1}{m}$

ТЕОРЕМА 8.4. Якщо повне m-арне орієнтоване дерево має n вершин і i внутрішніх вершин та l листів, то $l = (m\text{-}1)i + 1$.

$i = \frac{l-1}{m-1}$

ТЕОРЕМА 8.5. Повне m-арне орієнтоване дерево висоти h має $\frac{m^{h+1}-1}{m-1}$ вершин та m^h листів. В частості, повне бінарне орієнтоване дерево висоти h має $2^{h+1} - 1$ вершин та 2^h листів.

ТЕОРЕМА 8.6. а) Якщо повне m-арне дерево висоти h має l листів, то $h = \log_m (l)$

б) Якщо m-арне дерево висоти h має l листів, то $h \geq \log_m (l)$.

в) Якщо повне бінарне дерево висоти h має v вершин, то $h = \log_2 (v + 1) - 1$.

г) Якщо бінарне дерево висоти h маж v вершин, то $h \geq \log_2(v + 1) - 1$

Мають місце твердження

а) Якщо $e \in E$, то $f(e) \in E'$ ($f(E) \subseteq E'$)

б) Якщо $v \in V$, то $f(v) \in V'$ ($f(V) \subseteq V'$)

в) Якщо вершини u та v інцидентні e в G, то $f(u)$ і $f(v)$ інцидентні ребру $f(e)$ в G′

Два корневих бінарних дерева T(E, V) і T′(E′, V′) ***ізоморфні***, якщо існує ізоморфізм f з T в T′ такий, що

а) v_i – лівий син вершини v_j тоді і тільки тоді, коли $f(v_i)$ – лівий син вершини $f(v_j)$.

б) v_i – правий син вершини v_j тоді і тільки тоді, коли $f(v_i)$ – правий син вершини $f(v_j)$.

в) f відображає корінь r дерева T в корінь r' дерева T′.

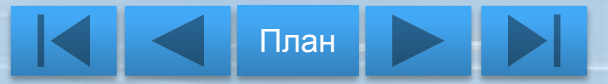
Бінарні дерева пошуку

Побудувати бінарне дерево.

Петерсон, Джонсон, Сміт, Вейл, Спенсер, Рассел.

Алгоритм вставки елемента

1. Починаємо з кореня
2. Якщо елемент < об’єкта в вершині, переходимо до лівого сина
3. Якщо елемент > об’єкта в вершині, переходимо до правого сина
4. Повторяємо кроки 2 і 3, доки не досягнемо вершини, яка не визначена
5. Якщо досягнута вершина не визначена, то визначаємо вершину і вставляємо елемент



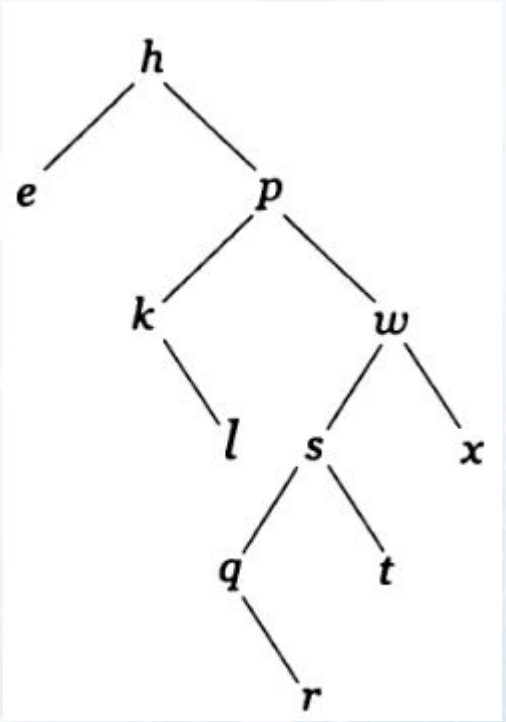
Алгоритм пошуку елемента

1. Починаємо з кореня
2. Якщо елемент < об’єкта в вершині, переходимо до лівого сина
3. Якщо елемент > об’єкта в вершині, переходимо до правого сина
4. Якщо елемент = об’єкту в вершині, то елемент знайдено; виконуємо відповідні дії і виходимо.
5. Повторяємо кроки 2, 3 і 4 доки не досягнемо вершини, яка не визначена.
6. Якщо досягнута вершина не визначена і в дереві немає шуканого елемента, то виконуємо відповідні дії і виходимо.

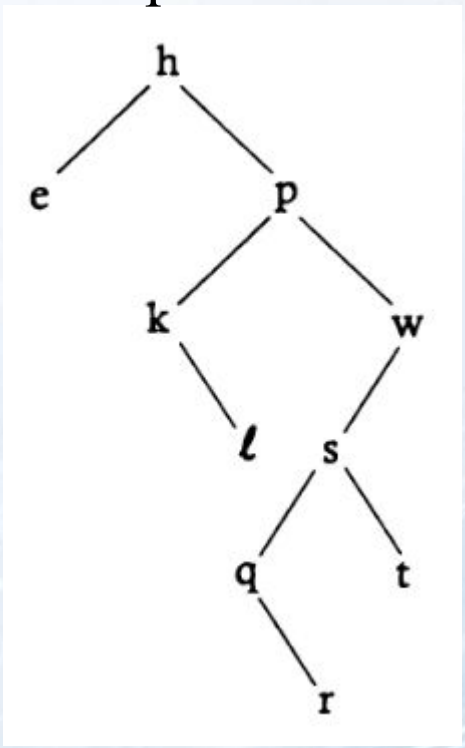
Алгоритм видалення елемента

1. Якщо вершина v_0 не має синів, просто видаляємо її.
2. Якщо вершина v_0 має одного сина, видаляємо v_0 і заміняємо її сином.
3. Якщо v_0 має двох синів, знаходимо правого сина v_1 вершини v_0, а потім знаходимо лівого сина вершини v_1 (якщо він існує). Продовжуємо вибирати лівих синів кожної знайденої вершини, доки не знайдеться така вершина v, у якої не буде лівого сина. Замінимо v_0 на v і зробимо правого сина вершини v лівим сином батька вершини v.

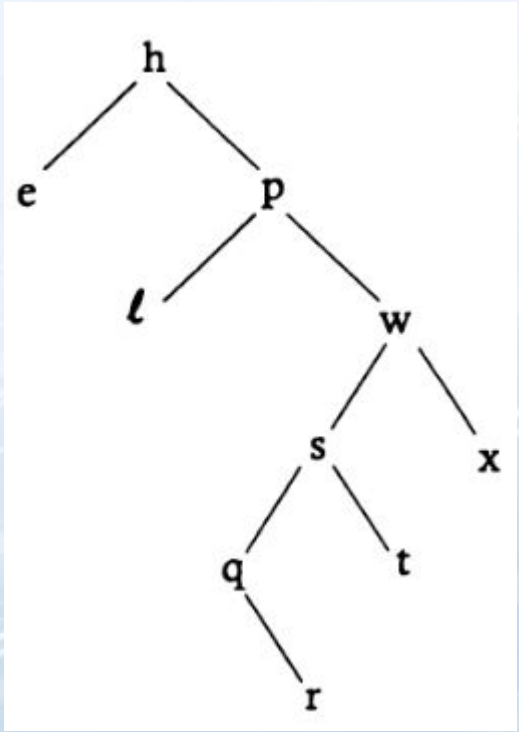
Дане дерево



Дерево, після видалення вершини **x**



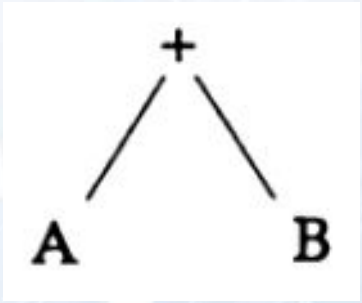
Дерево, після видалення вершини **k**

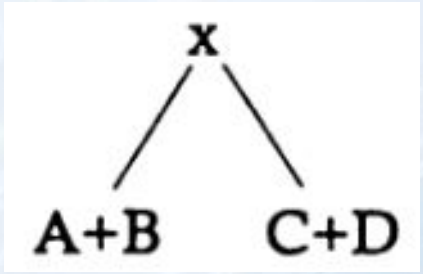


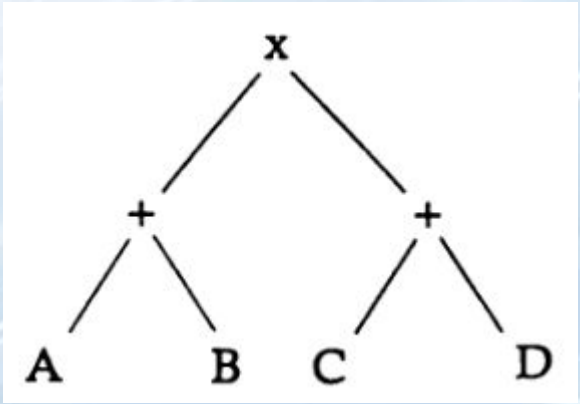
Обхід бінарних дерев

Наступні дерева зображають арифметичні операції

$A + B$ $(A + B) \times (C + D)$ $(A + B) \times (C + D)$



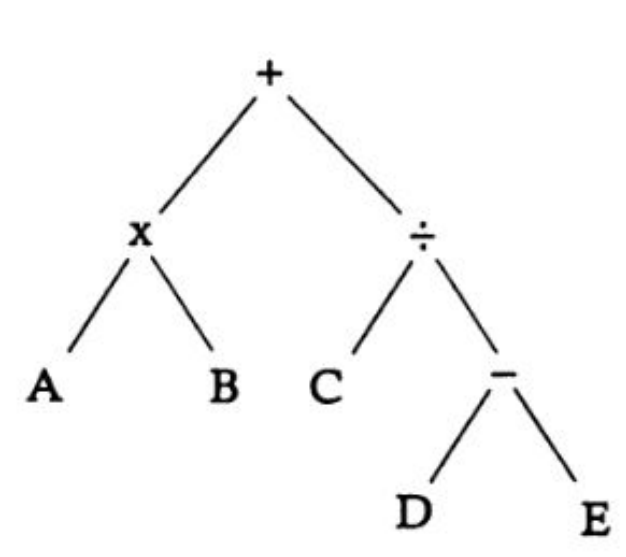




Алгоритм обходу дерева в центрованому порядку – ОЦП(корінь)

1. Якщо ЛівийСин(корінь) існує, то ОЦП(ЛівийСин (корінь))
2. Обробити(корінь)
3. Якщо ПравийСин(корінь) існує, то ОЦП(ПравийСин (корінь))

В результаті обходу дерева в центрованому порядку отримуємо:
A × B + C ÷ D – E

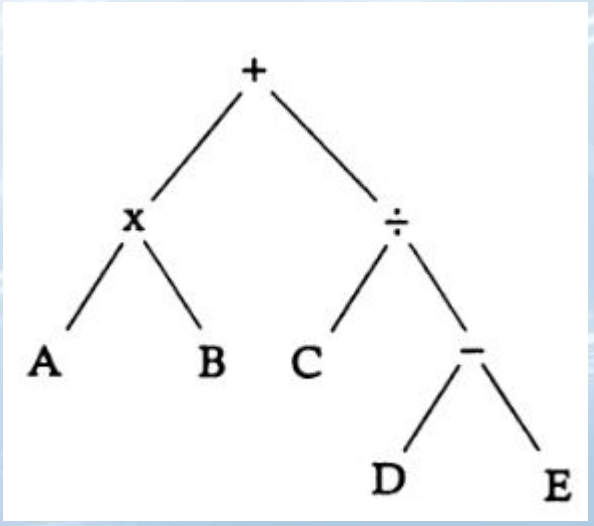


Алгоритм обходу дерева в прямому порядку – ОПП(корінь)

1. Обробити(корінь)
2. Якщо ЛівийСин(корінь) існує, то ОПП(ЛівийСин (корінь))
3. Якщо ПравийСин(корінь) існує, то ОПП(ПравийСин (корінь))

Для даного дерева результат алгоритму:

+ × AB ÷ C – DE

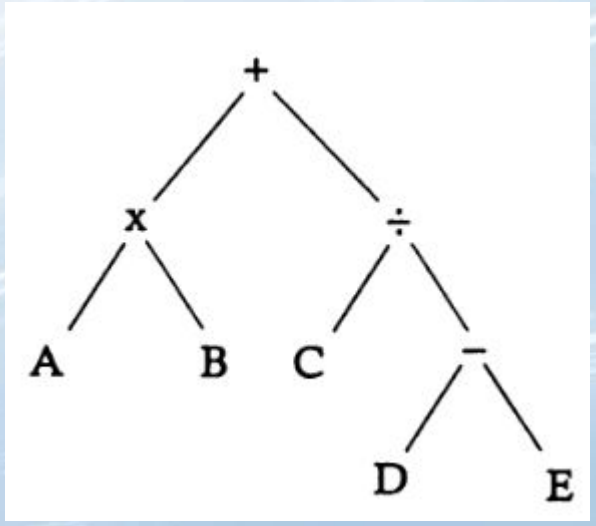


Алгоритм обходу дерева в оберненому порядку – ООП(корінь)

1. Якщо ЛівийСин(корінь) існує, то ООП(ЛівийСин (корінь))
2. Якщо ПравийСин(корінь) існує, то ООП(ПравийСин (корінь))
3. Обробити(корінь).

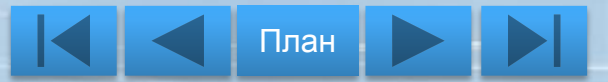
Результат алгоритму:

AB × CDE – ÷ +



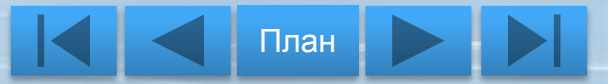
Алгоритм перевірки ізоморфності бінарних дерев – ІБД(r_1, r_2)

1. Обробити та r_2.
2. Якщо наявність(r_1) = Y і наявність(r_2) = N або наявність(r_1) = N і наявність(r_2) = Y, то Ізо = F.
3. Якщо Ізо = F, то завершити ІБД(r_1, r_2)
4. Якщо наявність(r_1) = Y і наявність(r_2) = Y, то ІБД (ЛівийСин(r_1), ЛівийСин(r_2))
5. Якщо наявність(r_1) = N і наявність(r_2) = N, то ІБД (ПравийСин(r_1), ПравийСин(r_2))



Основні дерева. Алгоритм пошуку остовного дерева в ширину – ПОДШ(G)

1. Вибрати довільний елемент v_0 графа G. Нехай $v_0 \in V^T$ та $L(v_0) = 0$
2. Для всіх $v \in V - V^T$ таких, що v суміжна з v_0, покласти $v \in V^T$, $\{v_0, v\} \in E^T$ і $L(v) = 1$
3. Нехай $i = 1$.
4. Вибрати $v_j \in V^T$ таке, що $L(v_j) = i$
5. Вибрати $v \in V - V^T$. Якщо v суміжна з v_j, покласти $v \in V^T$, $\{v_0, v\} \in E^T$ і $L(v) = i + 1$
6. Продовжувати крок 5, доки всі елементи множини $V - V^T$ не будуть розглянуті.
7. Повторювати кроки 4, 5, 6 доки всі v_j такі, що що $L(v_j) = i$, не будуть вибрані.
8. Покласти $i = i + 1$
9. Повторювати кроки 4-8 до $V = V^T$



Алгоритм пошуку остовного дерева в глибину – ПОДГ (G)

1. Помітимо кожну вершину графа G символом *n*.
2. Виберемо довільний елемент v_0 графа G і назвемо його коренем дерева
3. Замінимо мітку вершини v_0 з «нова» на «використовується» і покладемо $v = v_0$
4. Доки існують вільні невибрані вершини, суміжні з v, виконувати наступні дії:
   1. Вибрати вершину w, суміжну з v
   2. Якщо w має мітку «нова», додати (v, w) в РЕБРА ДЕРЕВА, змінити мітку w на «використовується», покласти $w = v$ і повторити крок 4.
   3. Якщо w має мітку «використовується» і не являється батьком v, додати (v, w) в зворотні ребра і повторити крок 4.
5. Якщо $v \neq a$, покласти $v = (v)$ і повторити крок 4.

Ліс остовних дерев називається ***остовним лісом.***

ТЕОРЕМА 8.7. Якщо T – глибинне остовне дерево графа G(V, E) і $\{a, b\}$ – ребро графа G(V, E), то або a являється потомком b, або b – потомком a.

ТЕОРЕМА 8.8. Нехай T – глибинне остовне дерево графа G(V, E). Вершина $a \in V$ являється точкою зчленування графа G(V, E) тоді і тільки тоді, коли вершина a 1) або являється коренем дерева T і має більше ніж одного сина, або 2) вершина a не являється коренем дерева T, і існує такий син c, що між c, або одним з його потомків, і власним предком вершини a не існує зворотнього ребра.

ТЕОРЕМА 8.9. (Формула Келі для дерева) Число остовних дерев для n розмічених вершин дорівнює n^{n-2}

Алгоритм пошуку точок зчленування – ПТЗ(v)

1. Помітити v символом «використовується»
2. Присвоїти i значення $i + 1$
3. Покласти $OP(v) = ЗС(v) = 1$
4. Поки існує невибрана вершина, суміжна з v, виконувати наступне:
    1. Вибрати вершину w, яка суміжна з v.
    2. Якщо w має мітку «нова», то:
        1. Додати (v, w) в РЕБРА ДЕРЕВА
        2. Покласти $v = parent(w)$
        3. Визвати ПТЗ(w)
        4. Якщо $OP(w) \geq ЗС(v)$, то v – точка зчленування. Ребра, нижче v, які охоплюють компоненту, видаляються
        5. Покласти $OP(v) = \min(OP(v), OP(w))$
    3. Якщо w має мітку «використовується» і w не являється $parent(w)$, то $OP(v) = \min(OP(v), ЗС(w))$

План

Алгоритм переведення дерева в послідовність ДвП(Т) для n = 3

1. Для вибора a_1 взяти лист v_j з найменшим значенням j. Завжди існує єдине k таке, що $\{v_j, v_k\}$ являється ребром дерева. Видалити це ребро і покласти $a_1 = k$.
2. Якщо вибрано a_{i-1}, то для вибора a_i з дерева, що залишилося, взяти лист v_j з найменшим значенням j. Завжди існує єдине k таке, що $\{v_j, v_k\}$ являється ребром дерева. Видалити це ребро і покласти $a_i = k$.
3. Продовжувати до тих пір, поки не буде оброблено a_{n-2}.

Алгоритм переводу послідовності в дерево – ПвД(a_1, a_2, …, a_{n-2}) для n ≥ 3

1. Для кожного v_j, якщо i з'являється в послідовності n_i разів, покласти $\deg(v_i) = n_i + 1$.
2. Прочитати a_1 і сформувати ребро $\{v_{a1}, v_j\}$, де j – найменша мітка, така що $\deg(v_j = 1)$.
3. Зменшити $\deg(v_{a1})$ і $\deg(v_j)$ на 1.
4. Припустимо, що a_{i-1} прочитано, читати a_i і формувати ребро $\{v_{ai}, v_j\}$, де де j – найменша мітка, така що $\deg(v_j = 1)$.
5. Зменшити $\deg(v_{ai})$ і $\deg(v_j)$ на 1.
6. Після того, як a_{n-2} прочитано, створити ребро між двома вершинами степені 1, що залишились.

Мінімальні остовні дерева

Вага остовного дерева зваженого графа G дорівнює сумі вагів, приписаних ребрам остовного графа. ***Мінімальним остовним деревом*** називається таке остовне дерево графа G, що вага T меньше або дорівнює вазі будь-якого іншого остовного дерева графа G.

Алгоритм побудови мінімального остовного дерева зваженого графа. **Алгоритм Крускала.**

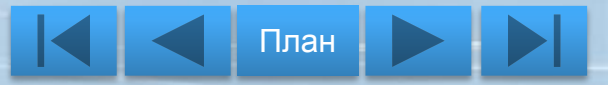
1. Вибрати в графі G ребро *e* мінімальної ваги, що не належать множині E і таке, що його додавання в E не створює цикл в дереві T.
2. Додати його ребро до множини ребер E.
3. Продовжувати, доки є ребра, що мають вказані властивості.

Алгоритм Прима знаходження мінімального остовного дерева

1. Вибрати вершину v_0 графа G і ребро з найменшою вагою e_1, для якого v_0 – одна з вершин, і сформувати дерево T_1.
2. Для заданого дерева T_k з ребрами $e_1, e_2, \dots, e_k$, якщо є вершина, що не належить T_k, вибрати ребро з найменшою вагою, суміжне з ребром дерева T_k і має вершину поза деревом T_k. Додати це ребро в дерево T_k, формуючи дерево T_{k+1}
3. Продовжувати, доки є вершини графа G, що не належать дереву.

Матричний алгоритм Прима

1. Створити вагову матрицю W.
2. Додати додатковий рядок і стовпець, щоб створити матрицю W^*.
3. В рядку 1 матриці W^* помістити * в останньому стовпцю. В стовпці й замінити всі числа на 0 і помістити U в останньому рядку.
4. Вибрати найменше число, так що рядок з цим числом має * в стовпці n+1, а стовпець з цим числом не містить U в рядку n+1.
5. Якщо число обрано в рядку i і стовпці j, то помістити * в останній стовпець рядку j, замінити решту чисел в стовпці j на 0, помістити U в рядку n+1 стовпця j і додати ребро (v_i, v_j) в остовне дерево.
6. Продовжувати виконання кроків 4 і 5, доки не залишиться чисел, які можна вибирати



Література до лекції

- Андерсон Д.А. Дискретная математика и комбинаторика: Пер. с англ.. – М.: Изд. дом «Вильямс», 2003. – 960 с
- Хаггарти Р. Дискретная математика для программистов. Москва: Техносфера, 2005. – 400 с.
- Белоусов А.И., Ткачев С.Б. Дискретная математика: Учеб. для вузов. 3-е изд. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2004. – 744 с.

Дякую за увагу