Практична робота №1

ТЕМА: Визначення умовно незалежних події.

МЕТА: Оволодіти методикою визначення ймовірності довільної події по стійкості накопиченої частоти.

ТЕОРІЯ: Для визначення ймовірності події можливо використати наступний "частотний" метод, що грунтується на стійкості послідовності значень появи події. Згідно цього методу слід провести серію дослідів однієї розмірності, в кожному з яких підраховують N_i кількість тих випадків, коли настає подія, де і-номер досліду. Накопичена частота V_n обчислюється за такою формулою.

$$V_n = \sum_{i=1}^{n} \frac{N_i}{i \bullet d}$$

Отримані числа записуються в таблицю.
Отриману послідовність $\{ V_n \}$ дослідимо на предмет наявності властивості стабільності. Ця властивість полягає в тому, що починаючи з номера N для всіх n>N матиме місце нерівність:

$$| V_n - P | < E \quad , \; E=0{,}0001$$

Якщо E-нескінченно мала величена, то маємо рівність.

$$P = \lim_{n \to \infty} V_n$$

$n \to \infty$ -нескінченість

де n-номер серіі досліду, Vn-накопичена частота появи букви після n-тоі серіі.

Задачі:

1.Визначити ймовірність появи літери, що має порядковий номер, який співпадає із порядковим номером вашого прізвища в журналі групи:
   а) вручну для тексту на укр.мові розміром 5кб.
   б) програмно для тексту на укр.мові розміром >50кб.
2.Визначити ймовірність появи складу, який розпочинається літерою з пункту 1 та умовну (байесівську) залежнсть між ними.
3.Побудувати таблицю ймовірностей всіх літер алфавіту.

Приклад..Практичне обчислення ймовірності події присвячене розв'язання задачі №1, для «О», «о» :

1) беремо текст із частини газетної української статті

2) Складаємо таблицю для порції на 100 літер

№порції	1	2	3		47	48	49	50
Кількість в порції «О» , «о»	4	5	6		Числа без зростання при $\epsilon = 0{,}002$			
Накопичена_частота	$\frac{4}{100}$	$\frac{4+5}{100}$	$\frac{4+5+6}{300}$				A=0,058	

3)Висновок щодо стійкості робимо розглядаючи числа (№№50,49,48,47). Якшо в кінці таблиці(50,29,28..) є ближчими до числа А і відзначаються від цього (А) на одну тисячну. А=0,058. За імовірність появи літери візьмемо середнє арифметичне цих послідовних чотирьох чисел, на якій спостерігається стійкість. Хід розв'язання задачі 1 для літери О чи о матиме наступний вигляд:

1) Відкриваемо файл β
2) Поки не кінець файла β виконувати:

Зчитуємо символ m.

Якщр це літера укр. Алфавіту чи пробел то збільшуємо на 1 значення лічильника 0, інакше переходимо до кінця циклу читання файлу β.

Якшо ASCI код символа m співпадає із кодом літери О чи о, то збільшуємо на 1 значення лічильника 1 та переходимо до кінця циклу читання файлу β, інакше переходимо до кінця циклу читання файлу β..

3) Кінець циклу читання файлу β

Практична робота №2

ТЕМА: Подання графа для обробки за допомогою комп'ютера

МЕТА: Вивчити способи подання графів як інформаційних об'єктів для комп'ютерної обробки.

Завдання: Вибрати згідно вашого номера в списку групи відповідний граф та виконати наступне:

1) вручну подати граф за всіма способами;
2) запрограмувати подання довільного скінченого графа.

ТЕОРІЯ:

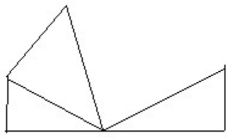
1) Представлення графів в пам'яті залежить від структури данних,які допускае алгоритмична мова та типу ЕОМ.

а) Представлення за допомогою матриці суміжністі,порядок якої співподае із числом вершин, де елемент (i-j) –й дорівнює 1, якщо і-та вершина суміжна із j-ою вершиною, та (i-j)-й елемент рівний 0 в протележному випадку.

Для графів із великою кількістю дуг це досить компактне представлення, а для графів із невеликім числом дугматимо досить розрідженну матрицю. Наведемо приклади таких представлень для неорінтовного графа: G1 та орграфа G2:

x3

G1: x2 x5



x1 x4 x6

$$A(G1) = \begin{matrix} 010100 \\ 01100 \\ 0100 \\ 011 \\ 01 \\ 0 \end{matrix}$$

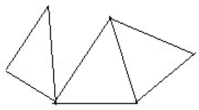
$$A(G2) = \begin{matrix} 011000 \\ 000000 \\ 010110 \\ 000010 \\ 000001 \\ 000100 \end{matrix}$$

x2

x4

x6

G2: x1



x3 x5

б) Представлення за допомогою матриці інцедентностей визначае граф однозначно бо мае порядок n x m , де n-кількість вершин,а m-кількість ребер; елементи матриці визначають наявність чи відсутність відношення інцидентності між вершинами та ребрами.
Використовуеться рідко із-за відсутності алгоритмів обробки працюючіх з такою структурою.

в) Представлення за допомогою списків суміжностей є головною альтернотивою представленю за допомогою матриць. Список суміжностей для вершини v є списком кінцевих дуг, що виходять із цієї v вершини орграфу, або просто списком всіх суміжних із v вершиннеоріентованого графу.
Наведемо приклад спискового представлення графів, що мали наведення вище матричне представлення для неорієнтовоного G1:

x1 : x2, x4; x3 : x2, x4; x4 : x1, x2, x3, x5, x6;
x2 : x1, x2, x4; x5 : x4, x6; x6 : x5, x6;

представлення та орієнтованого графа G2:

x1 : x2, x3; x2 : nil; x3 : x2, x4, x5; x4 : x5; x5 : x6; x6 : x4;

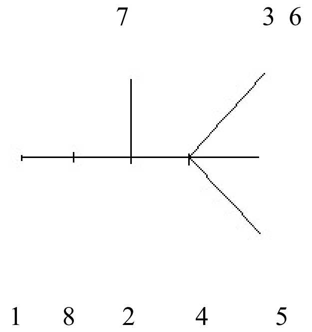
г) Представлення за допомогою списка дуг використовують для збереження різної інформації про дуги. При цьому способі кожній дужзі придають трійку чисел(u, x, y) ,де u=(x ,y), x-початок, y-кінец дуги. Вагу дуги можливо представити як четверте число до цієї трійки.

Код Харари визначаемо за допомогою матриці A(G)- матриці суміжностей шляхом послідовного запису рядків із тих елементів, що розміщені над головною діагональю, один за одним. Таким чином матимемо двійкове число, величина якого залежитеме від нумераці вершин. Найбільше ізціх чисел буде буде кодом Хірарі данного графу G. Нумерція вершин, що відповідае коду Хірарі зветься каноничною.

Код Прюфера використовуеться для представлення дерев. Нехай T-дерево із множиною вершин { v1, v2,…vn }, де номер вершини viдорівнюе i. Припишемо дереву T послідовніст (a1,a2,…an-2)побудовану за наступним правиломЖ

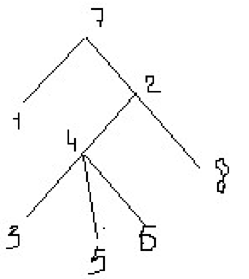
1) i=1;
2) в послідовності 1,2,…n шляхом перегляду зліва на право шукаемо номер першої висячої вершини. Нехай це bi .
3) Щукаемо вершину що суміжна із bi . Нехай це ai.
4) В послідовності із пункту 2) викреслюемо bi .
5) В дереві Твидаляемо вершину bi.
6) i = i+1;
7) якщо i< n-1, то переходимо до 2), інакше видаемо { ai,…an-1 }

Це й буде код Прюфера. Наприклад для дерева:



код Прюфера матемо вигляд (8,4,4,4,2,2).

У випадку ордерева побудова коду Прюфера виконуеться аналогічно.Необхідно тількі на останьому місці писати кореневу вершину та при декодуванні коду недописувати цю вершину. Так для ордерева:



Матимо кодПрюфера рівним(7,4,4,4,2,2,7).

Код Прюфера е оптімальним з точки зору економії пам”яті та доведений теоремі Келі :

числом помічених n-вершиндерев=n^{n-2} .

№2-2 Глобальний аналіз графів.

Означае виділення структури графа та визначення характеристик виделенної структури для розв’язку задачі. Цей аналіз полягае в збіранні інформаціїпро побудову графа шляхом обходу вершин та дуг(ребер) графа. Інформація отримана таким шляхом оформлюеться у виглядіпідходящої нумерації вершин графа.

1 Нумеція ,що виявляе логічну структуру графа.

1.1. Нумераціею F будемо називати приписування вершинам графа G різних чисел (номерів) з множини натуральних чисел N , то F : V(G) <-> N . Звеликого касу нумерації найбільш важливишими е нумерація побудована на пошуку вглибину(базисна нумерація), пряма нумерація та еранжировка. Іноді ці нумерації звуть лінійними.

Пошук в глибину це обхід вершин графа за наступних правил:

1) Знаходячісь у вершині x треба рухатися в любю іншу, раніше не пройдену, якщо така знайдеться, одночасно запам’ятовуючі дугу по якій вперше попали до вершини ;
2) Якщо із вершини x неможливо потрапити до раніше пройденої вершини або такої взагалі немае, то повертаемося до вершини zіз якої вперше попали до x та продовжумо пошук в глубену із вершени z.

При виконанні обходу графа поцім правилам ми намагаемося проникнути в глибь графа наскількі це можливо , потім відступаемо на крок назад і знову намагаемося пройти в перед. При пошуку в глиб орграфа можливопопасти в вершину y з вершини x тількі завдякі наявності дуги (x,y), то ми повинні рухатися вперед тількі в напрямку оріентації дуг, а повертатися в протележному напрямку. Внеорентованому графі таких обмежень немае. Будемо називати М-нумерацію вершин графа ту нумерацію що відповідае порядку їх обходу при пошуку в глубину. Шлях м=(g=x1,x2,…,xn=p) називатиме М-шляхом, якщо для кожної i,i=1,(1)n, виконуеться умова:

М(xi)<М(xi+1), то номер xi меньше номера xi+1; Вершина p зветься М-досяжною із вершини g, якщо існуе М-шлях із g в p

1.2 Алгоритм пошуку в глубину та побудову М-нумерації в оріентованому графі мае наступний вигляд:

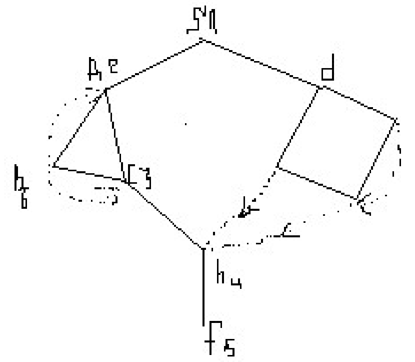
<u>Вхід</u> : Граф G=(V,E) заданий списками суміжностей A(v) , де v – вершина з множини V, A(v) її список суміжних вершин.

<u>Вихід</u>: М-нумерація вершин і розбиття множин E на чотири класи: деревяних дуг T , прямих дуг F , обернених дугB та поперечних дуг C.

```
    початок
    прцедура ПОШУК ( V,N );
    початок;
1.  присвоїти виршині v  номер M(v)=N;
2.  N:=N+1;
3.  Для  w E A(v) //  вершина w вибрана із списку A(v)
    Виконати цікл:
4.  якщо w не мае М-номера то
5.    початок;
6.    добавити дуги ( v,w ) в T ;
7.    Пошук ( w, N ) ;
8.    Кінець;
9.       інакше;
10.       початок
11. ___     якщо  М-номер вершини w більше М-номера вершини v  то доьавити ( v,w ) до F,
12. ______  інакше  якшо існуе М–шлях із w в  v  то
13.         добавити ( v ,w ) до  B;
14.        інакше   добавити ( v ,w ) до C;
15.         кінець // якщо з g)
16.       кінець // цiклу
17.   T:=0; F:=0; B:=0; C:=0; N:=1;
18.  для v E V виконати в циклі
```

19. помітить вершину v як немаючу номера
20. поки існує вершина v без М-номера виконати
21. ПОШУК (v, N);
22. кінец циклу поки. // кінец алгоритму.

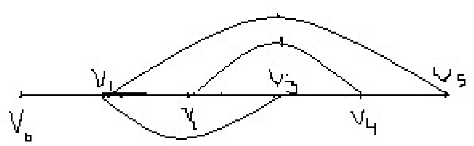
Приклад М- нумерації побудованої пошуком вглиб:



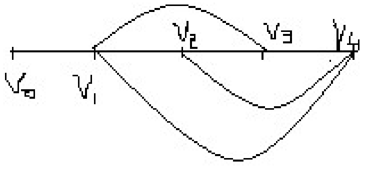
Де жирними дугами помічено деревьяні дуги з множини T, тонкими дугами помічено прямі дуги з множиниF, штриховими лініями поміченно обернені дуги із множини B, а пунктиром поміченно поперечні дуги із множини C.
Використання стеку спрощує реалізацію алгоритма. Присвоення М- номера відбуваеться в той момент коли вершина вводиться в стек; видалення вершини відбуваеться в той момент коли виявляються пройденими всі дуги,що виходять із данної вершини.
N-нумерацією вершин графа, що мае n-вершин, зветься присвоення перший викинутий зі стеку вершині номера n, а другій викінутй вершині присваемое номер n-1.
Пошук в глибину у неорієнтованому графі відризняеться від пошуку в глибину в орграфі тим, що всі ребра розбаваються на три класи:
Клас деревьяних(остовних) ребер, клас ребер дотику та клас прямих ребер, де клас ребер дотику відповідае класамобернених та поперечних дуг в орграфі т.то B+C. Цім вичерпууться зміни в наведеному алгоритмі.
Пошук в глибинуу неорентованому графі перетворюе вхідний графG на орграф G’ , шляхом наведення на кожному ребрі орієнтації у напрямку проходження.

1.3.

<u>Аранжировкою</u> або А-нумерацією будемо називати таку нумерацію коли кожен простий шлях із входом А однохідного графа зветься А-шляхом. Граф зветься аранжируемим якщо він допускае аранжировку.
Приклад аранжуруемого наведемо як граф G1:



А граф G2 е неаранжуемим:



Не факт неаранжуемості вказує наявність двох шляхів із входу s=v0 в вершину v2, один із яких містить вершину v3, а другий ні.

2-4

Побудова аранжировки для ациклічного графа здійснюється за наступним алгоритмом:
Вхід: Граф G=(V,E) заданий стеками суміжностей A(v) , де V0 множина вхідних вершин.
Вихід: Аранжировка (А-нумерація) вершин графа.

1. початок
2. процедура Аранжіровка (v, N);
3. початок
4. присвоїти вершині v номер N ;
5. N:=N+1;
6. Для w E A(v) ЦІКЛ
7. Виконати Аранжировка (w, V), якщо всі попередні вершини w маютьА-номери
8. Кінец
9. N=1
10. Для v E V0 цікл
11. Помітити вершину v як не маючу А-номера; кінець циклу
12. Покі існує вершина v E V0 без А- номера виконати цикл АРАНЖИРОВКА(v ,N);
13. Кінец ціклу поки
14. кінец алгоритму

1.4.

Пряма нумерація виконується на базі М-нумерації та полягає у визначенні формальних циклів графа, де під формальним циклом розуміють граф породженний вершинами простого шляху F(i,. .k) та оберненої дуги (Vk, Vi) вважаемо що від ш-тої вершини до k-тої зростають (i < j < k).

№2 -4 Логічний аналіз графів. Лінійні компоненти.

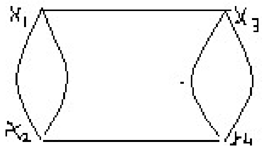
2.1. Матриця досяжності та транзетивне замикання.
У випадку , коли необхідно встановити ліше факт наявності шляхів між вершинами, виникае задача про побудову транзитивного замикання орграфа. Транзетивним замиканням G* графа Gзветься орграф із тіеюже множиною вершин що й G, але в ньому присутня дуга (xi xj) тоді і тількі тоді , коли вершина xj досяжна із xi ,.т. то існує шлях (xi- xy) . Матрицею суміжностей транзетивного замикання G* орграфа G служить матриця досяжності R(G) графа G , т. то квадратна матриця порядка n , n= /V(G)/ , із елементами rij=1,якщо вершина xj досяжна із xi , также rij =0.
Найпростішим алгоритмом побудови матриці R(G) мае вигляд:
Вхід: Матриця A(G) суміжності орграфа G , що мае рядки A1, A2, .. An.
Вихід: Матриця досяжності R(G) ,що мае рядки R1,R2, ..Rn.

Початок алгоритма:
1. для і від 1 до nкрок 1 виконати цікл
2. побудувати множину J <-> { j }, таких індексів ,що aij=1;
3. Ri:=Ai; K:=0;
4. Поки J<-> 0 (J непуста множина) виконати цикл
5. Вибрати j E J;
6. Ri:=Ri+Aj, де + -- логічна операція ’ и ’, тобто обедная матриць одного порядку , тобто 1 або в одній або в другий матриці дае А в результативній матриці.
7. J:=J\{ y };
8. K:=K U { j };
9. Зформувати множену Ji – індексів К таких , що ajk=1;

10. J:=J U (Jj \K);
11. Кінец цикла;
12. Виводимо Ri;
13. Кінец цикла ;

Кінец алгоритму.

Наприклад, для графа G та його матриці A(G) :

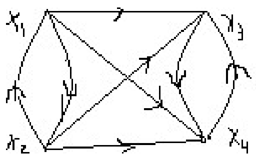


:

$$A(G) = \begin{matrix} 0110 \\ 1000 \\ 0001 \\ 0010 \end{matrix}$$

маемо матрицу R(G) та G* -транзетивне замикання:

$$R(G) = \begin{matrix} 1111 \\ 1111 \\ 0011 \\ 0011 \end{matrix}$$



Більш ефективним е алгоритм Уоршела (Warahall) , якій знаходить матрицу R шляхом обчислення послідовності квадратних матриць $B^0, B^1, \ldots B^n$ порядка n за наступними правилами:

1. B^0 : = A(G) ; // B^e = {B^e ij};
2. Для L =1 до n кроком1 виконати цикл
3. B^L ij = B^{L-1} ij +|B^{L-1} ij & Bij | ; // + логічне додавання діеть множину ребер із числа присутніх або в одному або в другому
4. Кінець циклу
5. B=B^n ;

   Кінець алгоритму.

Елементам B^e ij надати той зміст , що B^e ij=1 тоді ітільки тоді коли ершина xi та xj звязані шляхом , що проходить через вершини xi, x…xj .

Тоді початковий крок B^0 = A(G) означатиме шляхи без прміжних вершин, а останній крок B^n

R:=B означатиме наявність шляхів через любі проміжні вершини. Цикл виконувате провірку факту, що xi та xj звязанні шляхом , проміжні вершини якого належатимуть множині { xi,…xj } та задовільняють випадкам:

1) існує шлях від xi до xj , що проходить через (xi,…x L-1 };
2) існує шлях від xi до x L та від x L до xj , всі проміжні вершини якого належать множині (xi,…x L-1 }.

4-2
2.2.

№ 2-3 <u>Відшукання бікомпонент орграфа.</u>

Бікомпонентою називають найбільший (за включенням) підграф в якому люба пара вершин досяжна із кожної вершини іншої пари вершин. Ефективний алгоритм знаходження біокомпонент на пошуку " в глибину " мае наступний вигляд:

<u>Вхід</u>: Граф G =(V,E), заданий списками суміжностей A(v).
<u>Вихід</u>: Список B з біокомпонент.

Початок алгоритму

1. Процедура БІОКОМП (End (TT)) ;
2. Початок процедури
3. V : = End (TT);
4. Для w E A(v) виконати цикл
5. Якщо A(v) недорівнюють 0 ,то

6-8. якщо S(w) =0 то 1) TT : = w; S(w) =1
2) БІКОМП (w);

9.

10. інакше // w розміщенно в стеку TT

11-12. стягнути хвіст стеку TT почінаючі із вершини w до вершини w' , т.то видаляти, почінаючі із вешини w хвіст TT, запамятовувати видаленні вершини та замість w взяти w' для якої список суміжностей е списком отримань шляхом обеднання списків суміжностей зтягнутих вершин.

13. БІКОМП(w') ;

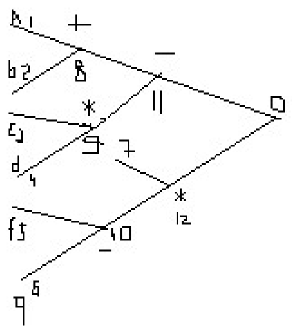
14. інакше 1) видаити v із стеку TT та занести в B, т.то в список біокомпонент вершину v;
2) БІКОМП (End (TT));

15. кінец циклу;
16. повертаемо B;
17. кІнец процедури БІКОМП ( );
18. кінец алгоритму.

7-1
7.2.

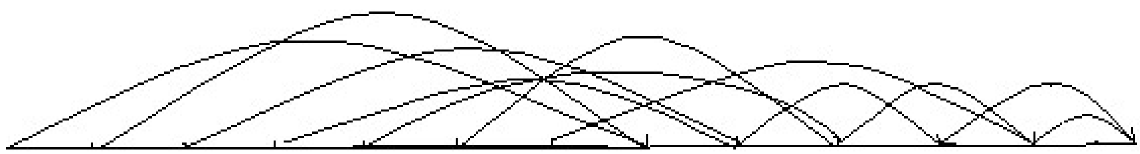
<u>Мінімізація памяті при обчисленні арифметичних виразів.</u>

Арифметичні вирази можуть відображатися як ордерева, в кожну вершину яких входить не більш двох дуг. Початкові данні відповідають висячим вершинам дерева, а проміжні результати обчислень. Кожна внутрішна вершина відображае бінарну операцію над оргументами, представленими вершинами- попередниками. Порядок в якому треба виберати аргументи несуттевий у простих моделях.Наприклад процес обчислення арифметичного виразу ((a +b)-c * d) / (e * (f - g)) зображенна наступним чином:



Вибір можливого порядку операцій означае топологічно впорядкувати дерево ,т.то розтошувати його вершини у ‘ цілочисельних ’ точках числової прямої в такій послідовності де вершина v передуе вершини v’ якщо існуе дуга із v’ в v, т.то дуга (v’, v) . Ця умова еквівалентна вимозі , щоб проміжний результат обчислювався раніше ніж використовувався. Приклад топологічного впорядкування дерева , для наведеного вище виразу мае вигляд:

1 2 3 4 5 6 7 8 9 10 11 12 0



Відзначимо, що величини які не використовуються в якості операндов- аргументів операції , що виконуеться у поточний момент часу , повинні зберігатися в памяті.

На малюнку топологічного впорядкування дерева тиким велечинам відповідають дуги , які проходять над вершиною що позначае поточну операцію.

Таким чином виникае оптимізаційна задача:
Мінімізувти кількістькомірок памяті для організації обчислення арифметичних виразів.

Ця задача зводиться до побудови топологічних впорядкувань ордерева із мінімальною ширеною.

<u>Генерація оптимального коду для арифметичних виразів.</u>

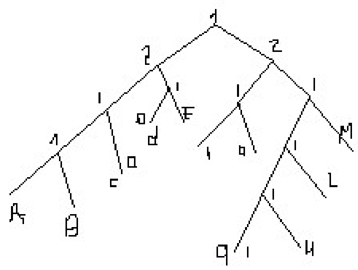
Нехай маемо машину із необмеженою памятю та T-регістрами та допустимими е команди наступних типів:
1) переселання: память → регістр;
2) переселання: регістр → память;
3) Операція : (регістр, память) → регістр;
4) Операція : (регістр, регістр) → регістр.

Зауважимо на те , що операція: (регістр, память) → регістр; е забороненою. Під ціною обчислень будемо розуміти кількість операторів (програмних кроків) потрібних для повного процесу обчислень. Для того, щоб визначити наймеьшу кількість потребних регістрів, а також оптимальну послідовність операцій будемо використовувати розмітку вершин, враховуючі некомутативні оерації. Розмітку виконаемо наступним чином:
1) якщо вершина v висяча та е лівим потоком деякої вершини, то покладемо L(v)=1, якщо вона е правим потоком , то вважатимемо L(v) = 0/
2) якщо вершина v мае потоків із мітками L1, L2; то при L1#L2 покладемо , що L(v) =max (L1,L2) ,а при L1=L2=L вважатимемо , що L(v) =L+1.

Алгоритм розвязку мае наступний вигляд:
Початок алгоритму
1. процедура T (v);
2. початок процедури
3. <u>якщо</u> f(v) то
4-5. <u>якщо</u> v-вісяча вершина ,<u>то</u> заслати значення вершини v у доступний регістр Bm із найменьшим номером; // в цьому випаку вешина v е лівим потоком свого предка //

6. інакше T(&(v)) ;
7-8. інакше якщо min (L(&(v)) , L (p(v))) => N то T (p(v));
// користуемося позначенням &(v) для лівого left(v), а через P(v) позначмо right(v) – правого потомка для v.
9. інакше якщо L(&(v))# L (P(v)) то
10. початок
11. w:= потомок вершини v із найбільшою міткою.
12. T(w);
13. кінец;
14. інакше T(P(w));
15. якщо (L(v)=1) ^(v-не е висячою вершиною) то
16. початок
17. обчислити вершину v , взявши значення лівого потомка із регістру Bm , а значення правого потомка із памяті
18. заслати значення вершини v до регістру Bm;
19. кінец;
20. інакше початок
21. T(&(v));
22. виконати операцію , що відповідае вершині v , беручі значення лівого потомка з регістру Bm , а значення правого потомка з регістру Bm+1;
// на цьому кроці доступними е регістри Bm , Bm+1;
23. якщо (L(v) => N)^(v- правий птомок свого предка) то
24. початок
25. значення вершини v заслати в память;
26. звільнити регистри Bm , Bm+1;
27. кінець;
28. кінець // якщо з номером 3
29. кінец процедури T(v)
30. v = корень дерева T.
кінець

Розглянемо приклад роботи цього алгоритму над виразом
((ab-c) /(d+c))/(g+i)/(j+k)*L-m)).
Дерево для цього виразу мае вигляд:



Для N=2 алгоритм генеруе такий код:
1. f→ B1;
2. B1+k → B1;
3. B1*L → B1;
4. B1-m→ B1;
5. G → B2;
6. B2+I→ B2;
7. B1/B2→ (память);
8. D→B1;
9. B1+e→ B1;
10. A→ B2;
11. B2*B→B2;
12. B2-c→ B2;
13. B1/B2→B1;
14. B1/(память)→ B1;

Практична робота №3

ТЕМА: Лінійний синтез скінчених графів

МЕТА: Отримати навички лінійного синтезу дискретніх об'єктів та аналізу наслідування властивостей.

Завдання. Виконати синтез по всім різним простим ланцюгам, як вручну, так і за допомогою программного засобу, двох наступних графів:

1) заданного графа із додатку 1 та порядковим номером тотожнім номеру Вашого прізвища в журнальному списку;

2) графа $K_{2,3}$ для першої групи, графа K_4 для другої групи

та проаналізувати число досяжності множини вершин кожного синтезованого графа.

Приклад:

1) Рассмотрим граф G (см. рис. 3.) иллюстрирующий

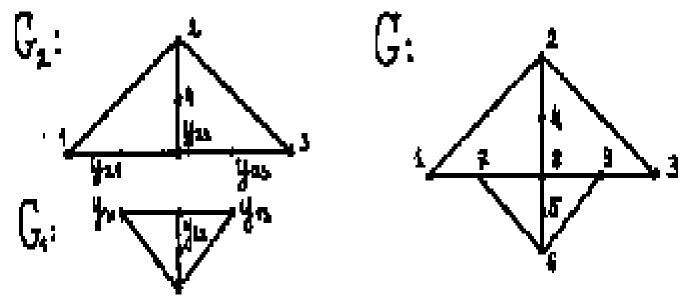


Рис. 3.

Граф G является φ - образом графа $\sum_{i=1}^{2} G_i$, где $G_2 \cong K_{2,3}$ $G_1 \cong K_{2,.3}$ а φ - преобразование задано следующим образом:

$$\varphi(\sum_{i=1}^{2} G_i, \sum_{j=1}^{2}(y_{1j} + y_{2j})) = (G, \{y_j\}_{j=1}^{3})$$

где $y_j = 6 + j, j = 1,2$,

а) $G_i \cong K_{2,3}, i = 1,2$;

б) $G_j(\{y_{ij}\}_{j=1}^{3})$ - простая цепь длинны 2 графа G_i

y_{21}, y_{23} - внутренние точки ребер, $i = 1,2$

в) $G(\{y_i\}_{j=1}^{3})$ - простая цепь длинны 2 графа G.

2) Рассмотрим граф G (рис.4.) иллюстрирующий утверждение 3) теоремы 1.2. в том случае когда $G_0(\{Z_{0j}\}_{j=1}^{n})$ - простой цикл, не являющийся границей внешней грани графа $f(G_0)$, где вложение f реализует $t_G(G^0)$.

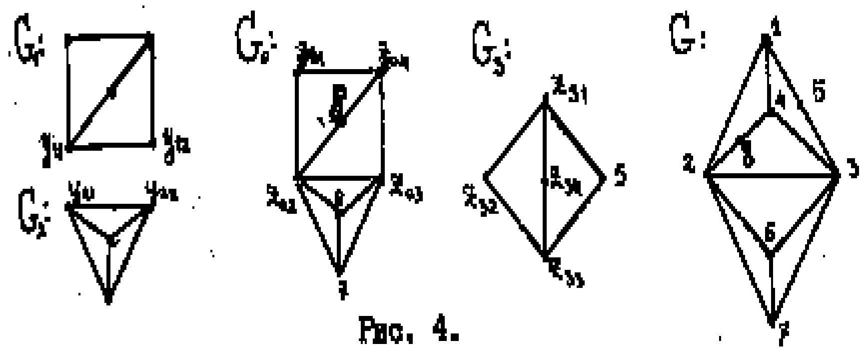


Рис. 4.

а) φ - преобразование графа $\sum_{i=1}^{2} G_1$ в граф G_0 следующим образом:

$$\varphi(\sum_{i=1}^{2} G_i, \sum_{j=1}^{2} y_{1j} + y_{2j}) = (G_0, \sum_{i=2}^{3} Z_{0i}),$$

где $G_j(\{y_{ji}\}_{i=1}^{2})$ - ребро графа G_j, $j = 1,2$,

$G_0(\{Z_{0i}\}) \in G^1$;

б) $G_3 \approx K_{2,3}$;

в) $G_0(\{Z_{0i}\}_{j=1}^{4})$, $G_3(\{Z_{3j}\}_{j=1}^{4})$, $G(\{j\}_{j=1}^{4})$ - простые циклы длинны 4 графов G_0, G_3, G - соответственно.

Практична робота №4

ТЕМА: Нелінійний синтез скінчених графів
МЕТА: Отримати навички нелінійного синтезу дискретніх об'єктів та аналізу наслідування властивостей.
Завдання. Виконати синтез по всім різним простим циклам, як вручну, так і за допомогою программного засобу, двох наступних графів:
1) заданного графа із додатку 1 та порядковим номером тотожнім номеру Вашого прізвища в журнальному списку;
2) графа $K_{2,3}$ для першої групи, графа K_4 для другої групи
та проаналізувати число досяжності множини вершин кожного синтезованого графа.

Приклад. Для иллюстрации в том случае, когда $G_0(\{Z_{0i}\}_{j=1}^{4})$ - цикл с диагональю, являющийся внешней гранью графа $f \mid G_0(G_0)$, где f - вложение реализующее $t_G(G^0)$:

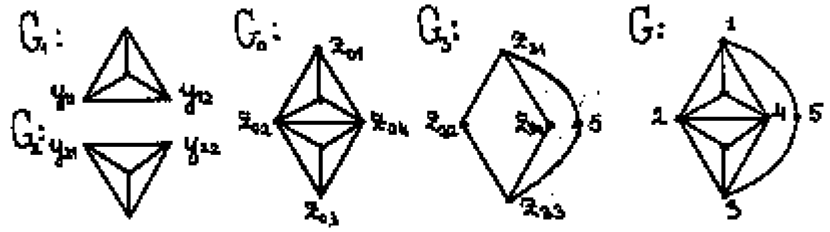


Рис. 5.

Практична робота №5-6

ТЕМА: Алгоритми на скінчених графах

МЕТА: Отримати навички

Завдання: 1) Виконати побудову остовного дерева графа із додатку 1 номер якого співпадає із номером вашого прізвища в списку групи, як вручну, так і за допомогою програмного засобу методом в глибину графа;

2) Виконати побудову множин фундаментальних циклів та множин простих циклів графа із додатку 1 номер якого співпадає із номером вашого прізвища в списку групи, як вручну, так і за допомогою програмного засобу методом в глибину графа.

Додаток 1

Графи 3-минимальні із номерами №1-№18.

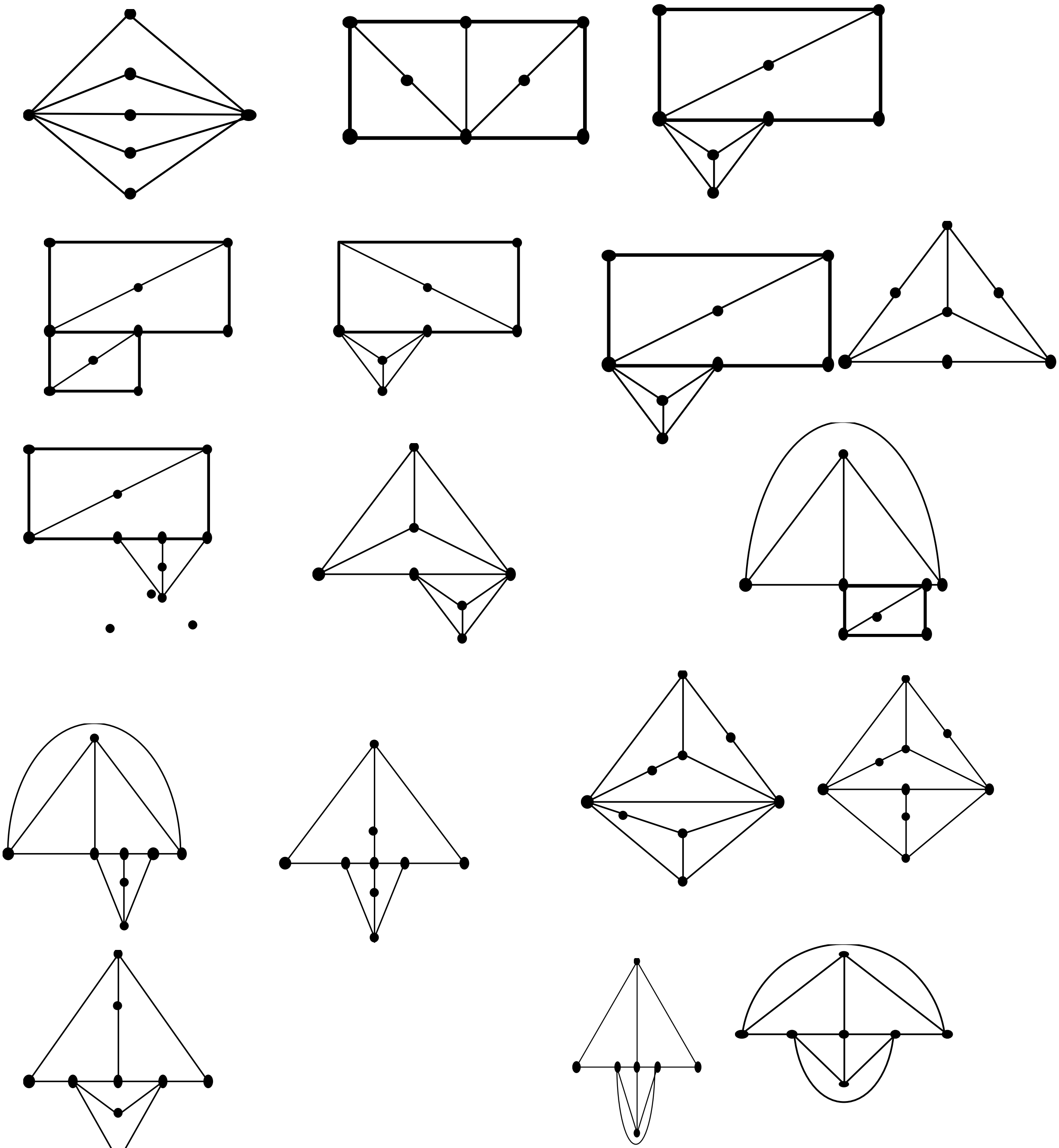


Рис.1. Графи 3-минимальні із номерами №1-№18.