КАТЕДРА: КОМПЮТЪРНИ СИСТЕМИ И ТЕХНОЛОГИИ
ДИСЦИПЛИНА: АЛГОРИТМИ СТРУКТУРИ ОТ ДАННИ
ДИСЦИПЛИНА: СИНТЕЗ И АНАЛИЗ НА АЛГОРИТМИ

ЛАБОРАТОРНО УПРАЖНЕНИЕ № 8

ТЕМА: Дървета. Представяне на дървета.

ЦЕЛ:

Целта на упражнението е студентите да се запознаят с основните понятия в дърветата, както и начините за представяне в дърветата. След упражнението студентите би следвало да могат да алгоритми за статично представяне на дървета, както и образуват елементарни структури за динамично представяне на дърветата.

I. ТЕОРЕТИЧНА ЧАСТ

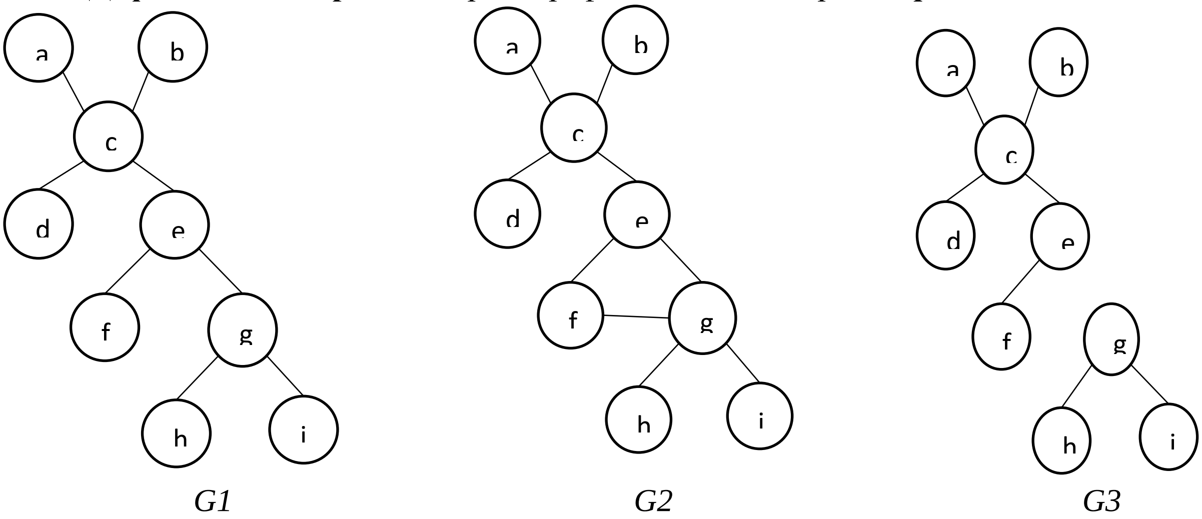
1. Въведение.

Дървото като абстрактна информационна структура се използва масово в операционните системи, компилаторите, системите за управление на бази данни, алгоритмите и много други компютърни приложения. Както и графите от общ тип, дърветата се прилагат за решаване на разнообразни задачи от области, които на пръв поглед нямат нищо общо с теорията на графите и с информатиката. Дърветата се използват за представяне на модели на реални обекти с йерархична структура, като родословни дървета, дървета на решенията, игри, организации, книги, математически формули и други.

2. Основни понятия и дефиниции:

Дефиниция за дърво: ***Дърво*** се нарича всеки граф ***T(V,E)***, който е свързан и не съдържа цикли. Дървото ***T(V,E)***, се нарича ***празно*** и се бележи с T_0, ако броя на върховете му е равен на нула $(|V|=0)$. Ако, то съдържа само един връх, дървото се нарича ***изродено*** и се бележи чрез T_1

Дефиниция за гора: Несвързан граф без цикли се нарича ***гора***.



Фиг.1. Пример за дърво и гора

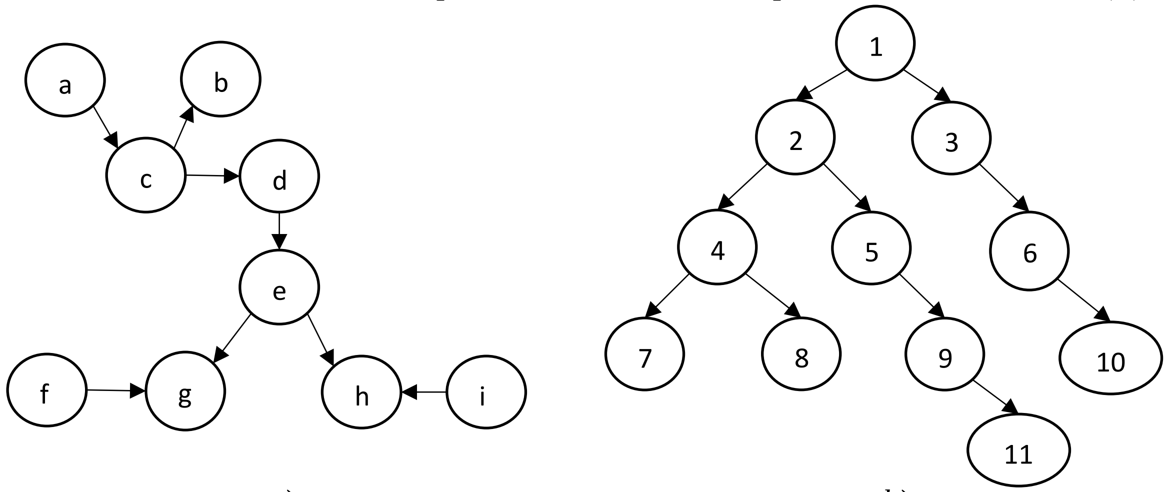
На фиг.1. граф *G1* е дърво, защото няма цикли и е свързан. Граф *G2* не е дърво, защото въпреки че е свързан има цикъл *{e, f, g, e}*. Графът *G3* е гора, защото не е свързан и всяка от компонентите му *{a, b, c, d, e, f}* и *{g, h, i}* не съдържа цикъл.

За дърветата е известно че $|V|=|E|+1$. Това се вижда и от фиг.1. Граф *G1* е дърво с *9* върха и *8* ребра ($|V|=9$, $|E|=8$). Графът *G3* е гора с компоненти *{a, b, c, d, e, f}* и *{g, h, i}*, които съдържат съответно $|V|=6$, $|E|=5$ и $|V|=3$, $|E|=2$. Също така е известно че ако от едно дърво се свържат с два несвързани върха с ребро, то се получава цикъл.

Дефиниция за кореново дърво: ***Кореново дърво*** се нарича дърво ***T(V,E)***, за което е изпълнено:

- Съществува точно един връх v_0 (корен), който няма предшественици (бащи), но може да има върхове наследници (синове). Връзката на корена с неговите наследници се задава чрез ребро;
- Всички останали върхове имат точно един връх – предшественик и евентуално върхове – наследници.

При ориентирано кореново дърво $d^+(v_o)=0$ за корена v_0, където $d^+(v_i)$ е броят на входящите дъги за съответният връх. Всички останали върхове $v_i \in V$, $v_i \neq v_0$ имат $d^+(v_i)=1$.



a) *b)*

фиг.2. Кореново дърво

На фиг.2. граф *(a)* не е кореново дърво, тъй като $d^+(g)=2$ и $d^+(h)=2$. Граф *(b)* е кореново дърво, тъй като за всеки връх $d^+(i)=1$ и за корена $d^+(1)=0$.

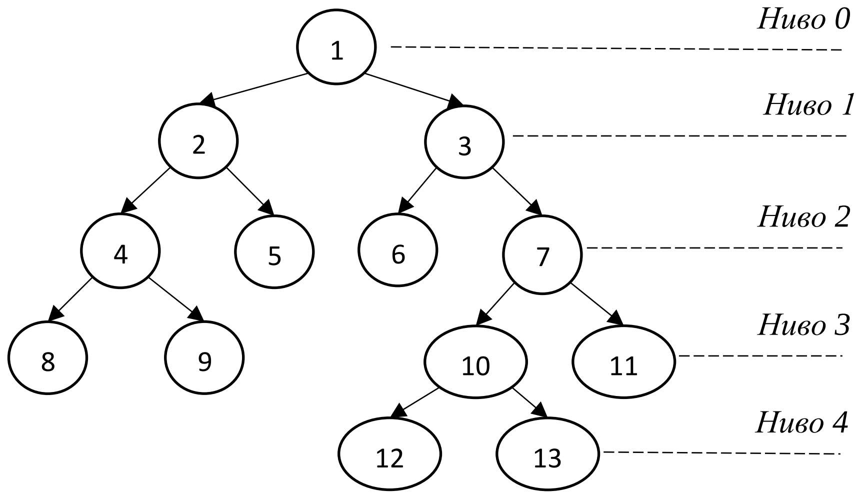
По дефиниция кореновите дървета са неориентирани графи. Ориентацията може да се пропусне, защото изборът на връх за корен неявно ги задава и имаме неявна ориентация.

Ребрата на дървото се наричат ***клони***. Висящите върхове, които нямат наследници, се наричат ***листа*** на дървото.

Дефиниция за височина и разклоненост на връх: Дължината на единствения път между връх v_i и корен ***r*** в кореново дърво се нарича ***височина*** (още ***дълбочина***) на върха и се бележи с $d(v_i)$. Броят на синовете на върха v_i се нарича ***разклоненост*** на върха и се бележи с $p(v_i)$.

Дефиниция за ниво: Върховете от едно дърво с еднаква височина се наричат ***ниво***. Коренът ***r*** на дървото има височина 0 и се намира на нулево ниво.

Максималната височина на върховете в едно кореново дърво определя височината (дълбочината) на дървото и се бележи с ***h***. Максималната разклоненост на върховете в едно кореново дърво се нарича разклоненост на дървото и се бележи с ***m***.



Фиг.3. Височина на кореново дърво

Дървото на фиг.3. има $h=4$ и $m=2$. Ако всеки връх има еднакъв брой синове $\boldsymbol{m}$, с изключение на листата, дървото се нарича ***пълно m-ично дърво***. Дървото на фиг. 3 е пълно двоично дърво, докато дървото от фиг.2.b) е двоично кореното дърво, но не е пълно.

Всяко пълно $\boldsymbol{m}$ – ично дърво с $\boldsymbol{k}$ на брой вътрешни върхове (всички върхове които имат наследници, т.е. – не са листа), съдържа общо $\boldsymbol{n=m*k+1}$ върхове. Всяко дърво с височина $\boldsymbol{h}$ и разклоненост $\boldsymbol{m}$ има не повече от $\boldsymbol{m^h}$ листа. Пълно $\boldsymbol{m}$ – ично дърво, на което всички листа са с еднаква височина, се нарича ***перфектно m – ично дърво***.

3. Представяне на дървета:

Дърветата по същество са графи, затова може да се използват методи за представяне графи (чрез матрици на съседство и на инцидентност, списъци на съседство и др.). Поради факта, обаче че при дърветата е изпълнено равенството |V|=|E|+1, то представянето чрез матрици не би било оптимален вариант понеже те биха били силно разредени.

Съществуват по – ефективни начини за представяне на дървета:

- Списък на бащите;
- Списък на синовете.

3.1.Представяне чрез списък на бащите

- ***Вариант 1:***

Всяко кореново дърво ***T(V, E)*** с корен $\boldsymbol{r}$ може да бъде представено със списък на бащите. Той представлява едномерен масив *p* с **|*V*|=*n*** елемента, като ***p[v]*** е върхът , към който е присъединен върхът $\boldsymbol{v}$ (бащата на $\boldsymbol{v}$). По дефиниция ***p[r]****=0*. Обикновено преименуваме върховете с числа от *1* до *n*, така че те да отговарят на индексите за съответния масив чиито стойности започват да се броят от 1.

Табл.1. Представяне на списъка на бащите за вариант 1, на произволно дърво

p[v]	*4*	*6*	*2*	*6*	*1*	*0*	*4*	*6*	*1*
v	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*

На ред $\boldsymbol{v}$ от таблицата са отбелязани върховете на произволно дърво, а на първия ред от таблицата ***p[v]*** – съответно бащата за конкретния връх. Така за конкретния пример корена на дървото би трябвало да има стойност *0* за ***p[v]***, т.е. корена е връх *6* за това произволно дърво.

- ***Вариант 2:***

Този метод се използва за конструиране на пълно двоично дърво T(V,E) с корен r от вектор, представящ списък на бащите, където върхът в позиция x е родител на върховете в позиции 2x+1 и 2x+2. Евентуално се допуска последният най – десен връх на дървото да има разклоненост 0, 1, 2. По дефиниция първият елемент във вектора е корена на дървото.

Табл.2. Списъка на бащите за вариант 2

v	*A*	*B*	*C*	*D*	*E*	*F*
index	*0*	*1*	*2*	*3*	*4*	*5*

Този метод може да се използва и за непълни двоични дървета като на позицията която липсва съсед се оставя празно поле.

3.2.Списък на синовете

При двоичното търсене е възможно описание чрез списък на синовете. Във всяко двоично дърво може условно да се въведе наредба на синовете. Тогава за всеки връх на дървото T(V, E) се задава наредена двойка от синовете му (ляв син, десен син), като отсъствието на син се задава с празна стойност или -1. Имената на върховете се заменят с числата от 1 до |V|. Това описание не е много ефективно, защото може да има доста празни полета, ако дървото не е пълно.

Табл. 3. Представяне на произволен граф чрез списък на синовете

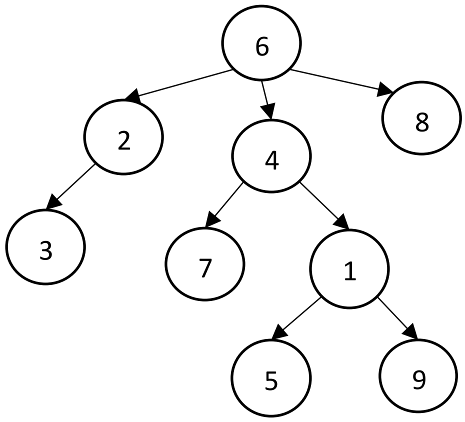
Връх	*Ляв син*	*Десен Син*
1	*5*	*8*
2	*3*	*-1*
3	*-1*	*-1*
4	*1*	*7*
5	*-1*	*-1*
6	*2*	*4*
7	*-1*	*-1*
8	*-1*	*-1*

II. ПРАКТИЧЕСКА ЧАСТ

В практическата част са показани примери, които показват как се използват линейните структури от данни.

ЗАДАЧА1: *Да се създаде дървото представено чрез списъка на бащите по вариант1 произволното дърво визуализиран в таб.1.*

РЕШЕНИЕ:



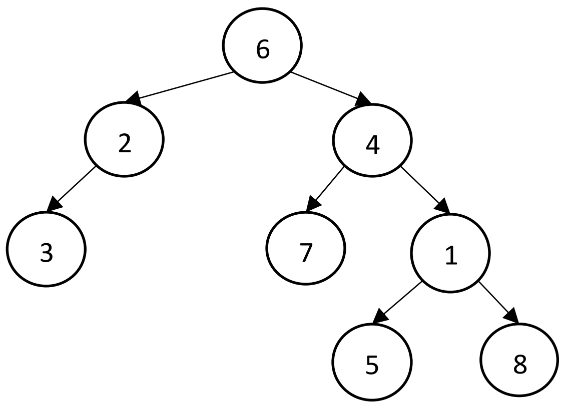
Фиг.4. Решение на зад.2.

ПОЯСНЕНИЕ

Първоначално намираме корена на дървото. Той трябва да има p[v]=0. От списъка в табл.1. се вижда че това е връх 6. След това търсим тези върхове които имат стойност p[v]=6. Пак от таблицата се вижда че това са 2 4 и 8. Т.е. на тези върхове бащата им е връх 6. Продължаваме по същия начин докато няма повече неоткрити върхове в списъка. Решението се вижда на фиг.4.

ЗАДАЧА2: *Да се създаде дървото представено чрез списъка на синовете на произволното дърво визуализирано в таб.3.*

РЕШЕНИЕ:



Фиг.5. Решение на зад.2.

ПОЯСНЕНИЕ

Търси се този връх от списъка който има синове, а той самия не е син на никой от другите върхове. Този връх ще е корена на дървото в случая това е връх 6. След това се продължава като се гледат синовете на съответния връх от списъка и се поставят на съответното място (ляво или дясно). Върховете, чиито синове имат само стойност -1 са листа на дървото. На фиг.5. е представено решението на задачата.

ЗАДАЧА3: *Създайте фрагмент от програма чрез динамична структура от данни реализираща дърво за което всеки връх, който не е листо има по 2 наследника и левия клон на дърветата е винаги по голям от десния*

РЕШЕНИЕ:

```
…………..
void add1(int n, elem* &t)
{
    if (t == NULL)
    {
        t = new elem;  t->key = n;
        t->left = t->right = NULL;
    }
    else
    {
        if (t->left == NULL && t->right == NULL)
            add1(n, t->left);
        else
            if (t->left != NULL && t->right != NULL)
                add1(n, t->left);
            else
                if (t->left == NULL)
                add1(n, t->left);
                else
                add1(n, t->right);
    }
}
……………………………
```

III. Задача за самостоятелна работа.

1. Да се намери дърво за дадения списък на бащите. Да се определи корена на дървото, неговата разклоненост и листата му.

Табл. 4. Списък на бащите за дърво

p[v]	*4*	*6*	*1*	*0*	*1*	*4*	*1*	*4*	*5*	*2*	*13*	*8*	*1*	*8*
v	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*

2. Да се създаде програмен алгоритъм, който решава зад.1.
3. Изпълнете задача 3 от практическата част и я тествайте като задавате последователно следните елементи за дървото:6,4,3,2,5,8,7.
4. Модифицирайте програмата от задача 3 така че накрая да разпечатва височината на дървото.