28. Дървета

28. 1. Определение. Примери

В информатиката графите се използват като информационни модели при решаване на разнообразни задачи, в които структурите от данни са твърде сложни. Особено приложение намира един специален клас от графи, наречени дървета.

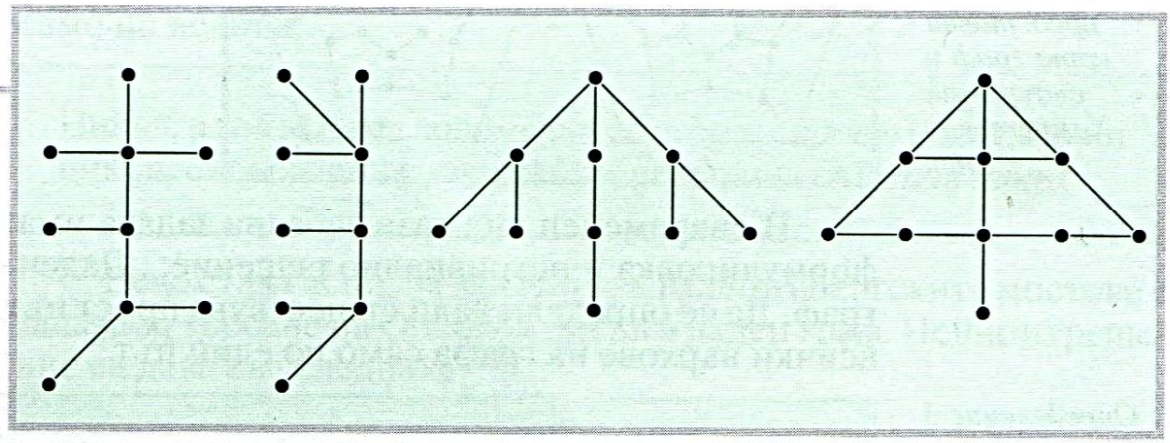
Определение 1

Дървото е неориентиран, свързан, ацикличен граф.

На фиг. 28.1 са дадени примери на графи, от които първите три са дървета. Последният граф не е дърво, защото съдържа цикли.

Фиг. 28.1

Примери на графи. Първите три са дървета.



За всяко дърво е в сила следното

Свойство 1 Ако T е дърво, тогава между всеки два върха на T съществува единствен прост път.

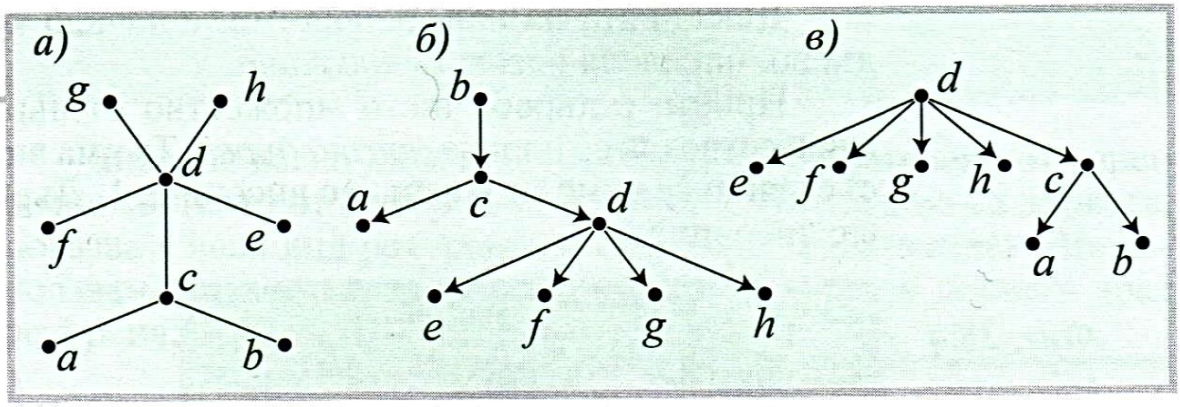
Например в последния граф от фиг. 28.1 съществуват върхове, между които има повече от един прост път.

Вярно е и обратното свойство:

Свойство 2 Ако в един граф между всеки два върха съществува единствен прост път, тогава графът е дърво.

Дървета с корени

В много случаи на използване на дървета един от върховете се избира като специален и се нарича *корен* на дървото. След като е избран корен на едно дърво, тогава може да се определи посоката на всяко от ребрата и те да станат ориентирани.



Фиг. 28.2

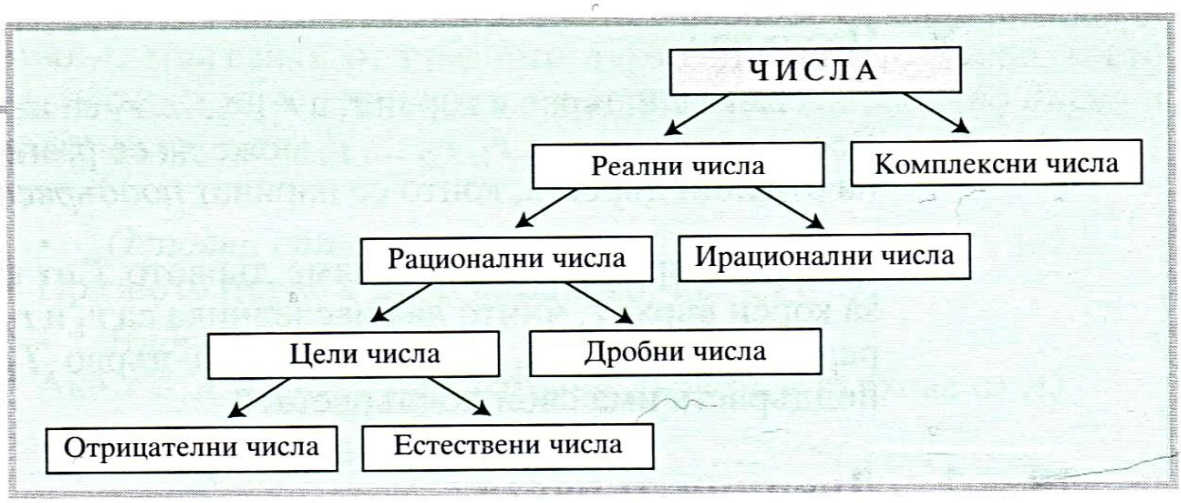
Примери на дървета:

а) без корен
б, в) с корен

На фиг. 28.2а е дадено дърво без корен. В другите два случая дърветата имат корени съответно b и d и техните ребра са вече ориентирани.

Класификационни схеми

Дърветата с корени се използват често за изграждане на класификационни схеми. Например на фиг. 28.3 е дадена класификацията на числата. Непосредствено се вижда, че тя е дърво с корен „Числа“, представляващо най-общото понятие.



Фиг. 28.3

Класификация на числата

Наследник, предшественик и лист

Върховете, които са свързани с корена, се наричат негови *наследници* (*деца*), а коренът – техен *предшественик* (*родител*). Наследниците на корена представляват върховете от 1-во ниво за дървото. Второто ниво е образувано от наследниците на върховете от 1-во ниво. Например върховете „Реални числа“ и „Комплексни числа“ са наследници на „Число“, докато върховете „Рационални числа“ и „Ирационални числа“ са наследници на „Реални числа“. Върховете „Отрицателни числа“ и „Естествени числа“ са наследници на „Цели числа“, но самите те нямат наследници. Такива върхове на дървото се наричат *листа*.

Две важни свойства:

- Коренът на всяко дърво няма предшественик.
- Всеки връх, с изключение на корена, има точно един предшественик.

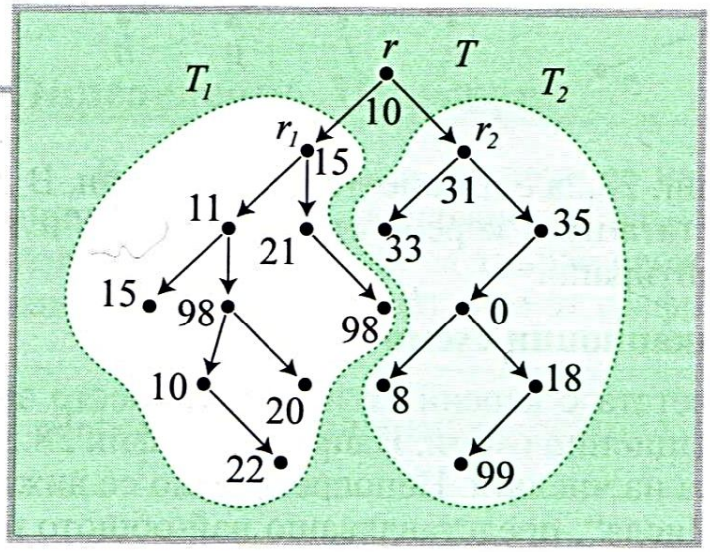
Височина

Дължината на най-дългия път, с начало корена на дадено дърво, определя неговата *височина*.

Прието е дърво, чието множество от върхове е празното множество, да се нарича *празно дърво*. То има височина 0. Дърво, състоящо се само от корен, е с височина 1. Дървото от фиг. 28.4 е с височина 6.

Фиг. 28.4

Дърво и неговите поддървета



Поддърво

Нека T е дърво с корен r, а $r_1, r_2, ..., r_k$ са наследниците на r. Всеки от върховете $r_1, r_2, ..., r_k$ може да се разглежда като корен на отделни дървета, които се наричат *поддървета* на дървото T.

Като пример да разгледаме дървото T от фиг. 28.4. То има за корен върха r, чиито два наследника са r_1 и r_2. Върхът r_1 е корен на дървото T_1, а r_2 е корен на дърво T_2. Всяко от тези поддървета има свои поддървета.

28. 2. Двоични дървета

Определение 2

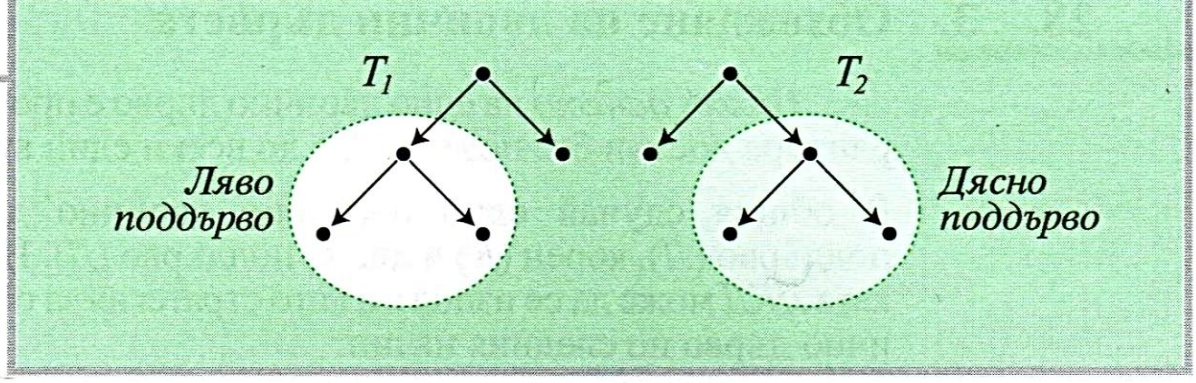
Дърво, всеки връх на което има не повече от два наследника, т.е. всеки връх е от степен не по-голяма от 2 се нарича двоично дърво.

Наредени дървета

Ако поддърветата в едно дърво са наредени, тогава се казва, че дървото е *наредено*. Например нареденото двоично дърво T_1 от фиг. 28.5 е различно от нареденото двоично дърво T_2, защото левите и десните им поддървета са различни.

Фиг. 28.5

Наредени двоични дървета



Това, че всяко дърво се разглежда като съставено от корен и множеството от неговите поддървета, които също са дървета, показва в явен вид рекурсивната природа на дърветата. Ще използваме тази идея, за да дефинираме двоичните дървета чрез рекурсия.

★★★

Определение 3

Двоичното дърво T е крайно множество, което е или празното множество или се състои от един елемент, наречен *корен,* и от две непресичащи се множества, наречени *поддървета* на T, всяко от които е двоично дърво.

Тази дефиниция подсказва как може да се конструира множеството $\mathfrak{T}_A$ на двоичните дървета с върхове от дадено множество A, тръгвайки от празното дърво. Двоичното дърво, което не съдържа нито един връх, се нарича *празно двоично дърво* и се означава с „<>“.

- *Основна стъпка:*

Празното дърво е двоично дърво, т.е. $<> \in \mathfrak{T}_A$.

- *Индуктивна стъпка*:

Ако $x \in A$ и T_L и T_R са двоични дървета с върхове от A,

т.е. $T_L \in \mathfrak{T}_A$ и $T_R \in \mathfrak{T}_A$, тогава

$T = <T_L, x, T_R>$ е двоично дърво, т.е. $T \in \mathfrak{T}_A$.

Тази рекурсивна дефиниция подсказва как могат да се дефинират рекурсивно и някои други понятия. Да дефинираме например понятието височина на произволно двоично дърво $T = <T_L, x, T_R>$, означено с $h(T)$.

$$h(T) = \begin{cases} 0, & \text{ако } T = <>; \\ 1 + max\{h(T_L), h(T_R)\}, & \text{ако } T \neq <>. \end{cases}$$

28. 3. Обхождане на двоични дървета

Обхождането на едно двоично дърво е процедура, която гарантира достъп без повторение до всеки един връх на дървото.

В общия случай едно наредено двоично дърво има ляво поддърво (*Л*), корен (*К*) и дясно поддърво (*Д*). Наредената тройката *ЛКД* може да се използва като стратегия за обхождане на двоично дърво по следния начин:

1. Извършва се обхождане на лявото поддърво в реда *ЛКД* (*рекурсивна стъпка*).
2. Извършва се достъп до корена на дървото.
3. Извършва се обхождане на дясното поддърво в реда *ЛКД* (*рекурсивна стъпка*).

Ако двоичното дърво от фиг. 28.4 се обходи в реда *ЛКД*, последователността от върхове на дървото ще бъде:

15, 11, 10, 22, 98, 20, 15, 21, 98, 10, 33, 31, 8, 0, 99, 18, 35.

Следва алгоритъм, записан на Паскалоподобен език, за обхождане на двоично дърво в реда *ЛКД*, наричан още *инфиксен ред*.

Алгоритъм: Обхождане на двоично дърво в реда *ЛКД*.

Входни данни: Двоично наредено дърво *T*.
Изходен резултат: Списък с върховете на дървото, обходено в реда *ЛКД*.

```
type BinaryTree = ...;                    {Тип данни "Двоично дърво"}
procedure INORDER(T : BinaryTree)
  begin
    if T ≠ <> then
      begin
        INORDER(TL);  {Обхождане на лявото поддърво TL на T}
        Print(T[root]);                   {Процедура, отпечатваща
                                                  корена root на T}
        INORDER(TR)   {Обхождане на дясното поддърво TR на T}
      end;
  end;
```

29. Ⓜ Дървета и изрази

★★★

Едно от многобройните приложения на двоичните дървета са т.нар. *дървета-изрази*. Става дума за възможността изразите от даден тип (числен, булев и др.) да се представят, преобразуват и пресмятат с помощта на двоични дървета.

29. 1. Представяне на изрази чрез дървета

Основната идея за представянето на изразите чрез двоични дървета е следствие от факта, че при наличие на два израза E_1 и E_2 и една бинарна операция $\otimes$ при прилагането ѝ се получава отново израз $E_1 \otimes E_2$. Този израз може да се представи с двоично дърво T, чийто корен е бинарната операция $\otimes$ и поддървета T_1 и T_2 са съответно дадените два израза E_1 и E_2. Естествено те ще бъдат поддървета на дървото-израз. В израза $E_1 \otimes E_2$ E_1 е ляв аргумент (операнд), а E_2 е десен аргумент. Те ще бъдат представени съответно като ляво и дясно поддърво на T. За графичното представяне на дърво-израз ще използваме три графични знака:

- *правоъгълник*, в който се записва стойността на операнд;
- *кръг*, в който се записва операция;
- *стрелка*, с която се указва връзката между два върха (възела) на дървото-израз.

Специално да отбележим, че в случая става дума за наредени двоични дървета, което означава, че специално се указва кое е ляво и кое е дясно поддърво. Това е съществено, защото не всички операции са комутативни и се прави разлика между ляв и десен аргумент. Ще направим и уговорката, че аргументът на израз с унарна операция се представя като дясно поддърво на дървото с корен унарната операция.

Примери:

Нека разгледаме няколко примера за представяне на изрази от числен тип. Предполага се, че всички идентификатори A, B, C, X, Y и Z са от числен (реален) тип данни.

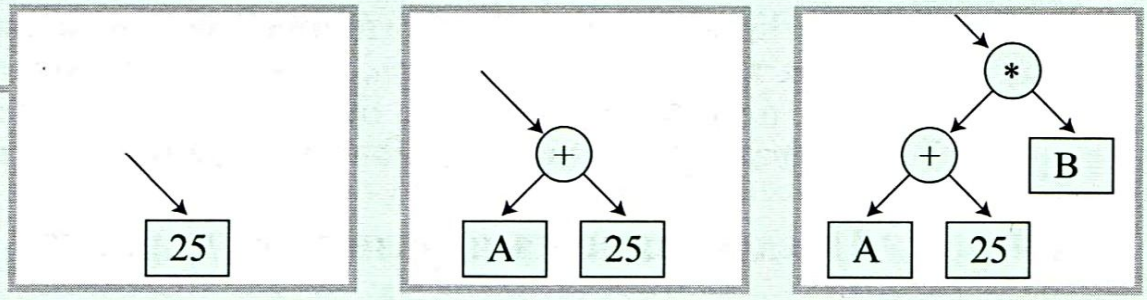
Пример 1 Най-простият вид изрази са константите и променливите от даден тип. Съгласно направените по-горе уговорки константата 25 се представя с дървото-израз от фиг. 29.1. Всъщност това е дърво-израз, което има само корен.

Пример 2 Изразът A+25 съдържа два аргумента и бинарната операция събиране. Съответното дърво-израз ще бъде представено с корен операцията „+“ и поддървета A – ляво поддърво, 25 – дясно поддърво (фиг. 29.2).

Пример 3 Да усложним този израз, като добавим нова бинарна операция: (A+25)*B. Корен на дървото-израз ще бъде операцията умножение, която е с аргументи (A + 25) и B. Те ще бъдат нейните поддървета (фиг. 29.3). В този израз операцията събиране трябва да се изпълни преди операцията умножение, а това означава, че тя не може да бъде корен на дървото-израз. Операцията, която е в корена на дървото, се изпълнява последна.

Фиг. 29.1, 2 и 3

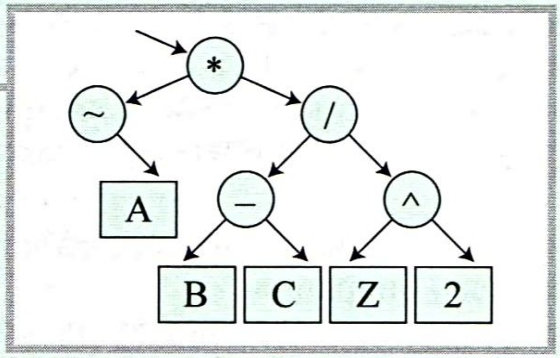
Илюстрации, съответващи на примери 1, 2 и 3



Пример 4 На фиг. 29.4 е представено дървото-израз на израза: $-A*(B-C)/Z^2$. Освен бинарните операции изваждане, умножение и деление присъства и унарната операция „минус“. Тя е означена със знака „~“. Нейният аргумент A е представен като дясно поддърво. Операцията степенуване е означена с „^“.

Фиг. 29.4

Илюстрация към пример 4



От разгледаните примери може да се направят две интересни бележки, които са в сила и в общия случай:

- Листата на дървото-израз са аргументи на израза, а вътрешните върхове са операции.
- Колкото е по-висок приоритетът на една операция, толкова тя е разположена по-ниско като връх в дървото-израз.

29. 2. Операции с дървета-изрази

Обхождане на дърво-израз

Да обходим дървото-израз от фиг. 29.3 в реда ЛКД (инфиксен ред). Получава се:

› A, +, 25, *, B.

Това е първоначалният израз, в който липсват скобите. Ако обходим същото дърво в реда ЛДК (постфиксен ред), резултатът ще е следният:

› A, 25, +, B, *.

В този случай всяка операция е разположена след своите аргументи. Обхождането на дървото-израз от фиг. 29.4 в реда ЛДК ще даде резултата:

› A, ~, B, C, –, Z, 2, ^, /, *.

Пресмятане стойността на дърво-израз

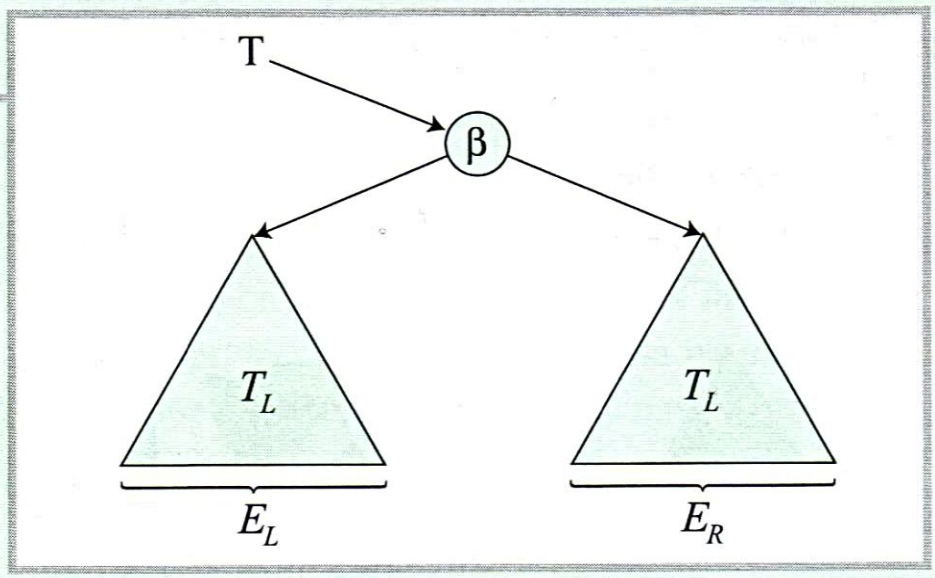
Дърветата са рекурсивни структури и много от операциите с тях се описват по-лесно с рекурсия. Такава е например функцията за *пресмятане стойността на израз*, представен с дърво-израз. Да означим с E произволен числов израз, с T – съответното дърво-израз и с *Val* – функцията за пресмятане стойността на израза E, представен с T. Функцията *Val* може да се дефинира рекурсивно по следния начин:

Основна стъпка:
Ако изразът E е представен с дървото, което се състои само от корен (единствен връх), който съдържа константа v или променлива със стойност v, тогава $Val(E) = v$.

Индуктивна стъпка:

а) Ако изразът E е представен с дървото T, с корен унарната операция α, и с дясно поддърво-израз T_L, тогава $Val(E) = \alpha Val(E_L)$, където E_L е изразът, представен с T_L.

б) Ако изразът E е представен с дървото T, с корен бинарната операция β, и с ляво и дясно поддърво-израз T_L и T_R, тогава $Val(E) = Val(E_L)\ \beta\ Val(E_R)$, където E_L е изразът, представен с T_L, E_R, е изразът, представен с T_R (фиг. 29.5).



Фиг. 29.5

Рекурсивно пресмятане стойностите на израза:

$Val(E) = Val(E_L)\ \beta\ Val(E_R)$