

Дървета


Трифон Трифонов

Структури от данни и програмиране, спец. Компютърни науки, 2 поток, 2024/25 г.

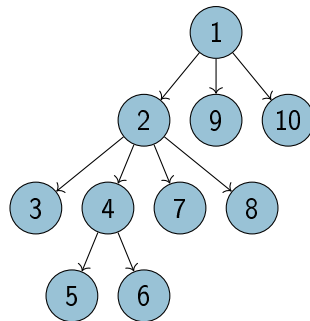
15–22 декември 2022 г.

Този презентация е достъпна под лиценза Creative Commons Признание-Некомерсиално-Споделяне на споделеното 4.0 Международен 



Or bluebudgie (<https://www.needpix.com/photo/1269899/wordcloud-trees-knowledge-tagcloud-text-cloud-word-info-design>) 

Пример: кореново дърво



АТД: Дърво

Йерархична структура, в която на всеки елемент е съпоставено множество от подчинени елементи.

Дефиниция

Кореново дърво е списък (X, T_1, \dots, T_n) , където

- X е данна (корен)
- T_1, T_2, \dots, T_n са коренови дървета (поддървета)

АТД: Дърво

Йерархична структура, в която на всеки елемент е съпоставено множество от подчинени елементи.

Дефиниция

Кореново дърво е списък (X, T_1, \dots, T_n) , където

- X е данна (корен)
- T_1, T_2, \dots, T_n са коренови дървета (поддървета), $n \geq 0$

АТД: Дърво

Йерархична структура, в която на всеки елемент е съпоставено множество от подчинени елементи.

Дефиниция

Кореново дърво е списък (X, T_1, \dots, T_n) , където

- X е данна (корен)
- T_1, T_2, \dots, T_n са коренови дървета (поддървета), $n \geq 0$

Често отъждествяваме кореновите дървета с корените им (**възли**).

АТД: Дърво

Йерархична структура, в която на всеки елемент е съпоставено множество от подчинени елементи.

Дефиниция

Кореново дърво е списък (X, T_1, \dots, T_n) , където

- X е данна (корен)
- T_1, T_2, \dots, T_n са коренови дървета (поддървета), $n \geq 0$

Често отъждествяваме кореновите дървета с корените им (**възли**).

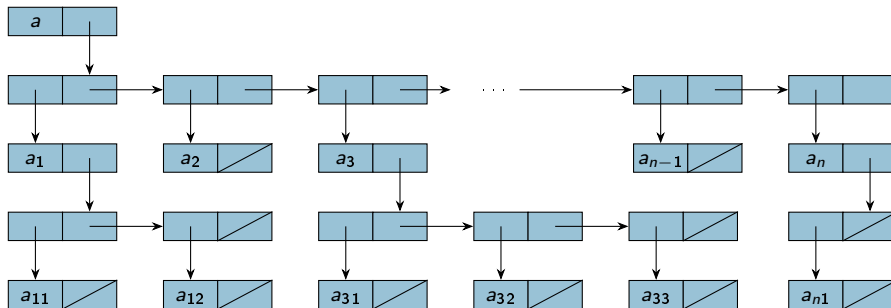
Операции:

- `create(x)` — създаване на дърво с корен x
- `addSubtree(t)` — добавяне на поддърво t
- `root()` — достъп до корена
- `subtrees()` — достъп до поддърветата

Свързано представяне

Свързана структура от възли, където всеки възел се състои от:

- стойността на корена
- свързан списък от възли, представлящи поддърветата (деца)



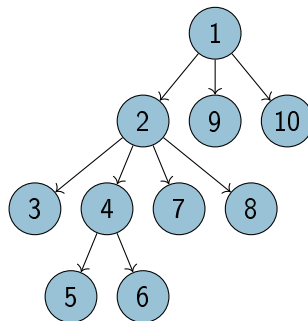
Последователно представяне

Последователност от тройки (корен, най-ляво дете, десен брат)

	0	1	2	3	4	5	6	7	8	...
корен	a	a_1	$a_{1,1}$	a_2	$a_{1,2}$	a_3	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$...
най-ляво дете	1	2	-1	-1	-1	6	-1	-1	-1	...
десен брат	-1	3	4	5	-1	9	7	8	-1	...

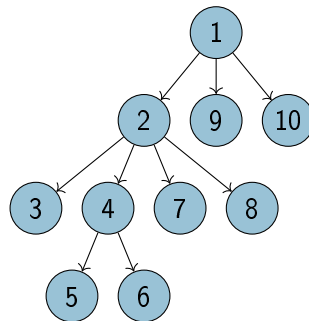
Дефиниции за дърво

- **дете** — коренов възел на поддърво
- **родител** — възел, сред чиито деца е коренът
- **листо** — възел без деца
- **път** — редица от възли, в която всеки следващ дете на предходния
- **разстояние** — дължина на пътя от корена до възел
- **ниво** — множество от възлите на еднакво разстояние
- **височина (дълбочина)** — максималното разстояние в дървото
- **широчина (разклоненост)** — максималният брой деца на възел



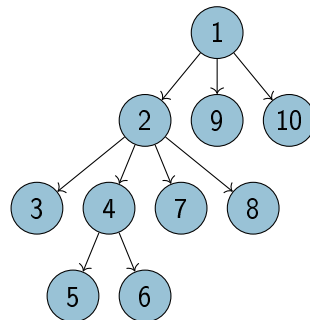
Схеми за обхождане

- префиксно обхождане
 - първо посещаваме корена
 - след това обхождаме децата



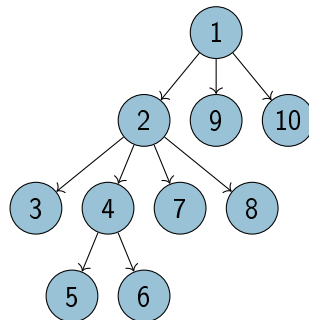
Схеми за обхождане

- префиксно обхождане
 - първо посещаваме корена
 - след това обхождаме децата
 - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10



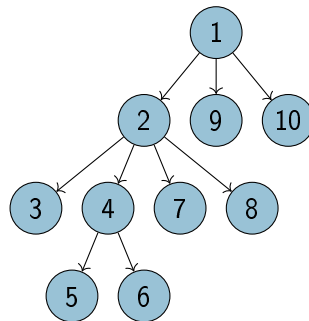
Схеми за обхождане

- префиксно обхождане
 - първо посещаваме корена
 - след това обхождаме децата
 - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- постфиксно обхождане
 - първо обхождаме децата
 - накрая посещаваме корена



Схеми за обхождане

- префиксно обхождане
 - първо посещаваме корена
 - след това обхождаме децата
 - 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
- постфиксно обхождане
 - първо обхождаме децата
 - накрая посещаваме корена
 - 3, 5, 6, 4, 7, 8, 2, 9, 10, 1



Дървета с фиксиран брой поддървета

Ако искаме всеки възел на дървото да има един и същ (максимален) брой деца, използваме алтернативна дефиниция

Дървета с фиксиран брой поддървета

Ако искаме всеки възел на дървото да има един и същ (максимален) брой деца, използваме алтернативна дефиниция

Дефиниция (n -арно дърво)

- \perp (празното дърво)
- наредена $n + 1$ -торка (X, T_1, \dots, T_n) , където
 - X е данна (корен)
 - T_1, T_2, \dots, T_n са n -арни дървета (поддървета)

Дървета с фиксиран брой поддървета

Ако искаме всеки възел на дървото да има един и същ (максимален) брой деца, използваме алтернативна дефиниция

Дефиниция (n -арно дърво)

- \perp (празното дърво)
- наредена $n + 1$ -торка (X, T_1, \dots, T_n) , където
 - X е данна (корен)
 - T_1, T_2, \dots, T_n са n -арни дървета (поддървета)

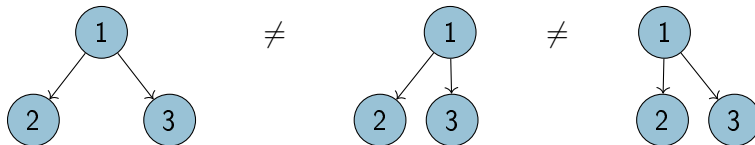
Операции:

- `empty_tree()` — създаване на празно дърво
- `create_tree(x, t1, ..., tn)` — създаване на n -арно дърво с корен x и поддървета t_1, \dots, t_n
- `root()` — достъп до корена
- `subtree(i)` — достъп до i -тото поддърво

Наредба на поддърветата

Дефинициите за коренови дървета и n -арни дървета не са еквивалентни!

Пример: троични дървета

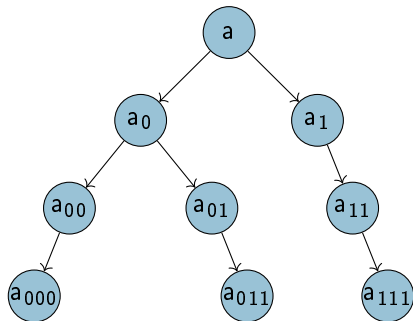


Едно и също кореново дърво може да бъде представено като n -арно дърво по няколко различни начина.

Двоично дърво

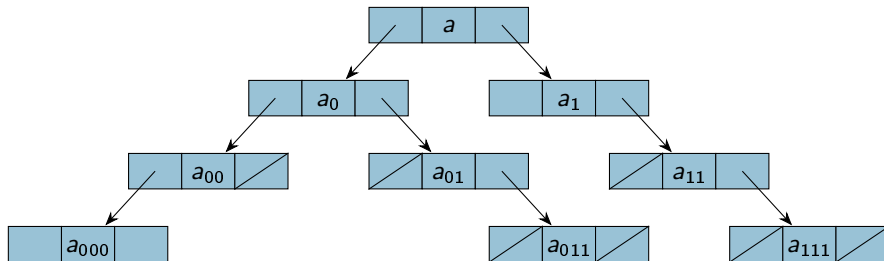
Дефиниция (двоично дърво)

- \perp (празното дърво)
- наредена тройка (X, L, R) , където
 - X е данна (корен)
 - L е ляво поддърво
 - R е дясно поддърво



Свързано представяне

Свързана структура от тройни кутии



Последователно представяне

Последователност от тройки (корен, ляво, дясно)

	0	1	2	3	4	5	6	7	8	...
корен	a	a_0	a_1	a_{00}	a_{01}	a_{11}	a_{000}	a_{011}	a_{111}	...
ляво дете	1	3	-1	6	-1	-1	-1	-1	-1	...
дясно дете	2	4	5	-1	7	8	-1	-1	-1	...

Позиция в двоично дърво

Ще въведем абстракция за позиция в двоичното дърво.

Операции:

- `root()` — инициализация в корена на дървото
- `left()` — преместване наляво
- `right()` — преместване надясно
- `up()` — преместване нагоре
- `get()` — достъп до елемент на дадената позиция
- `valid()` — проверка за валидност

Позиция в двоично дърво

Ще въведем абстракция за позиция в двоичното дърво.

Операции:

- `root()` — инициализация в корена на дървото
- `left()` — преместване наляво
- `right()` — преместване надясно
- `up()` — преместване нагоре
- `get()` — достъп до елемент на дадената позиция
- `valid()` — проверка за валидност

Не използваме термина “итератор”, понеже не абстрахираме начина на обхождане, а само позицията.

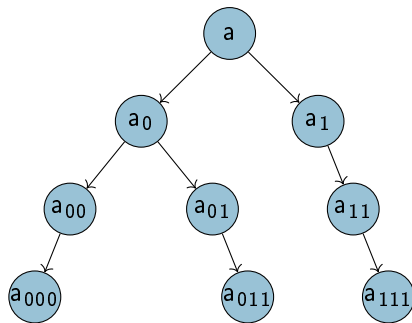
Визуализация на дървета

GraphViz (graphviz.org) е набор от инструменти за визуализация на графи.

- Изключително разнообразни възможности за настройка на графичния изход
- Разнообразни алгоритми за автоматично разполагане на графи
- Използва DOT формат за описание на графите
- `digraph<име-на графа>{ { <родител> -> <дете>; } }`
- `dot -Tpng graph.dot > graph.png`
- Можем да го използваме за бърза визуализация на дървета

Схеми за обхождане

- префиксни
 - КЛД
 - КДЛ
- инфиксни
 - ЛКД
 - ДКЛ
- постфиксни
 - ЛДК
 - ДЛК



Задачи за двоично дърво

Задача. Да се намери дълбочината на дадено дърво.

Задачи за двоично дърво

Задача. Да се намери дълбочината на дадено дърво.

Задача. Да се провери дали две дадени дървета съвпадат.

Задачи за двоично дърво

Задача. Да се намери дълбочината на дадено дърво.

Задача. Да се провери дали две дадени дървета съвпадат.

Задача. Да се построи дърво на аритметичен израз със скоби.

Задачи за двоично дърво

Задача. Да се намери дълбочината на дадено дърво.

Задача. Да се провери дали две дадени дървета съвпадат.

Задача. Да се построи дърво на аритметичен израз със скоби.

Задача. Да се премсетне аритметичен израз, записан в дърво.