Tango tree

Структура данных

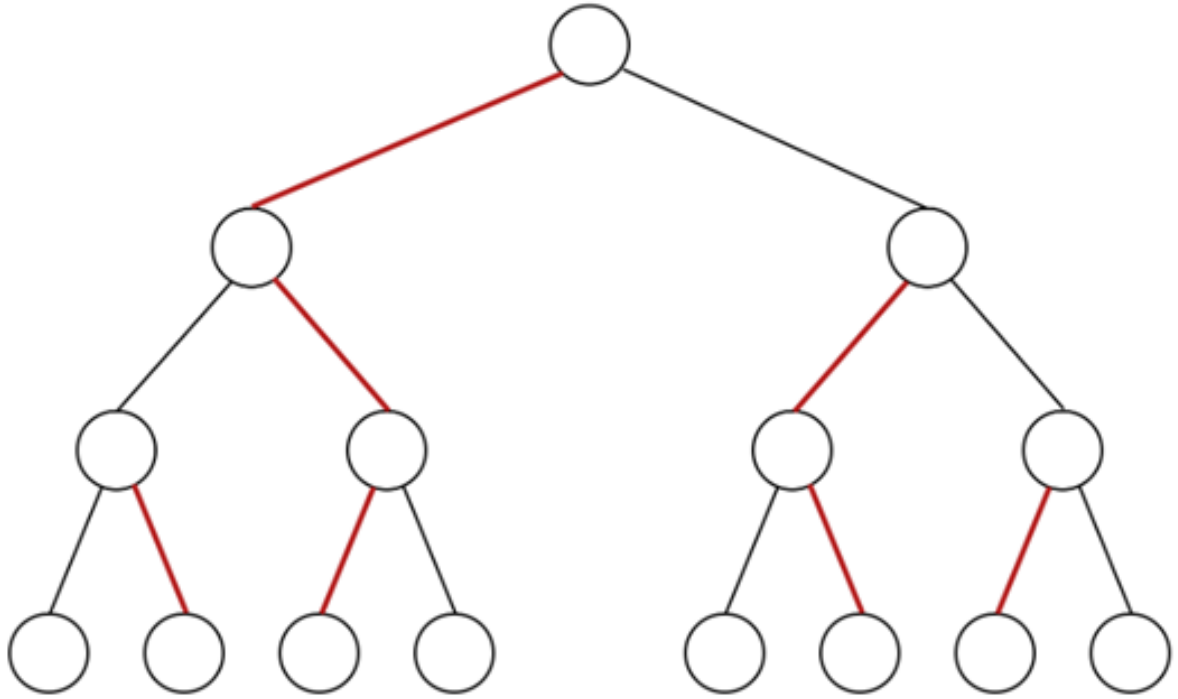
История

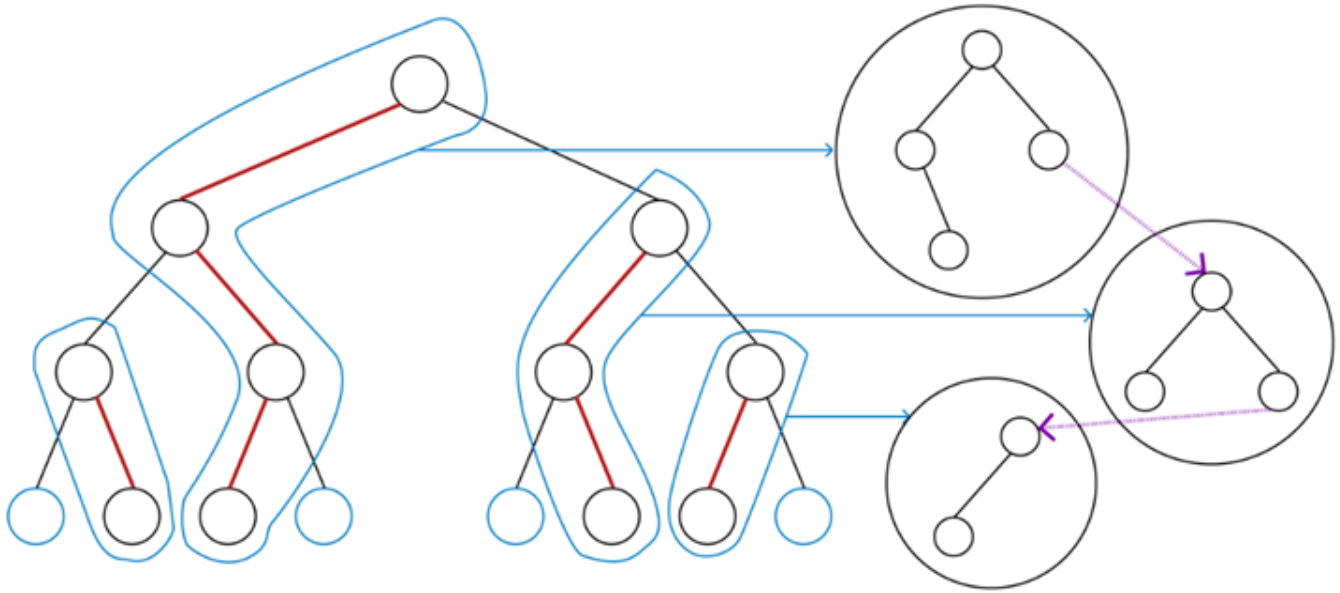
В 2004 году четверо ученых, Эрик Д. Демейн, Дион Хармон, Джон Яконо и Михаи Патраску, разработали Tango дерево. Назвали его в честь города Буэнос-Айреса, символом которого является танго.

Строение дерева

Рассмотрим бинарное дерево поиска. Дочерний элемент к которому обращались в последнюю становится предпочтительным

Путь из этих элементов назовем предпочтительным путем (обозначен красным на рисунке)





Предпочтительные пути образуют цепочки по всему дереву, и мы можем группировать узлы в цепочки и сохранять каждую цепочку как самостоятельное дерево с указателями между ними, от одного дерева к другому.

Вспомогательные деревья

Вспомогательное дерево представляет собой модифицированное бинарное дерево поиска в узлах которого хранятся:

- Ключ
- Глубина узла
- Минимальная глубина вспомогательного дерева в Tango.

Операции во вспомогательном дереве

1. Поиск элемента по ключу во вспомогательном дереве
2. Разрезание вспомогательного дерева на два вспомогательных дерева.
3. Объединение двух вспомогательных деревьев.

Операция поиска в Tango дереве

Поиск в дереве Tango очень похож на таковой в бинарном дереве поиска.

1. Сначала находим предпочтительный путь, который начинается в корне дерева, просматриваем его вспомогательное дерево.
2. Если узел не был найден в заданном вспомогательном дереве, то ищем вспомогательное дерево следующего пути и так далее до перебора всего дерева.

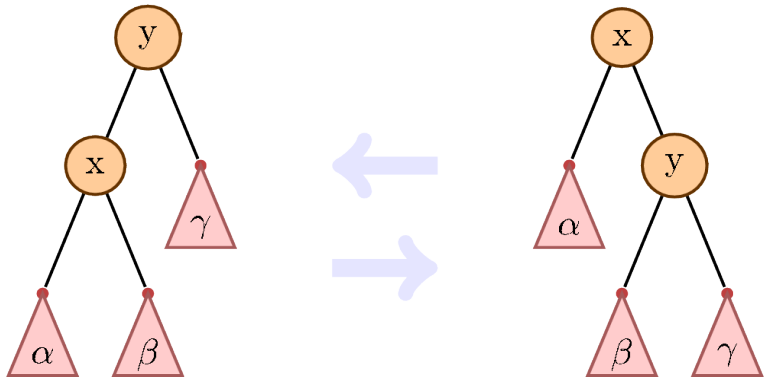
Реализация

Splay-дерево

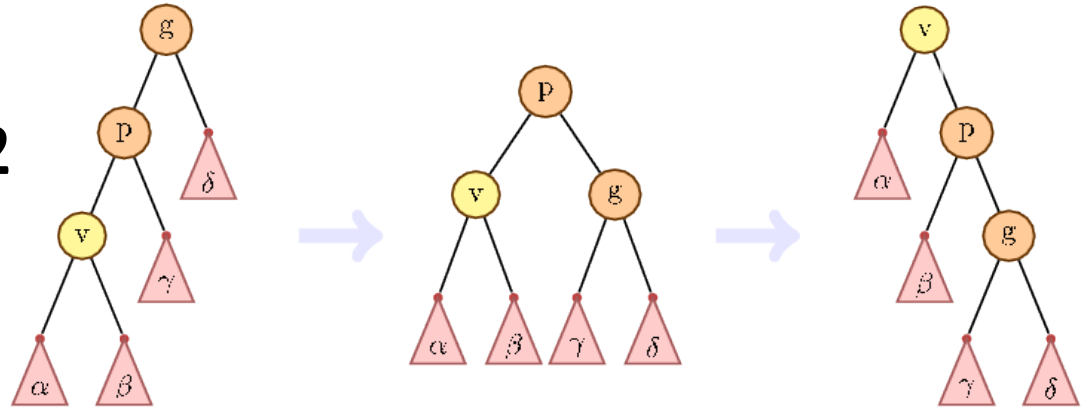
Splay-дерево — это самобалансирующееся бинарное дерево поиска. Дереву не нужно хранить никакой дополнительной информации, что делает его эффективным по памяти. После каждого обращения, в нашем случае после поиска, splay-дерево меняет свою структуру.

Строение splay-дерева

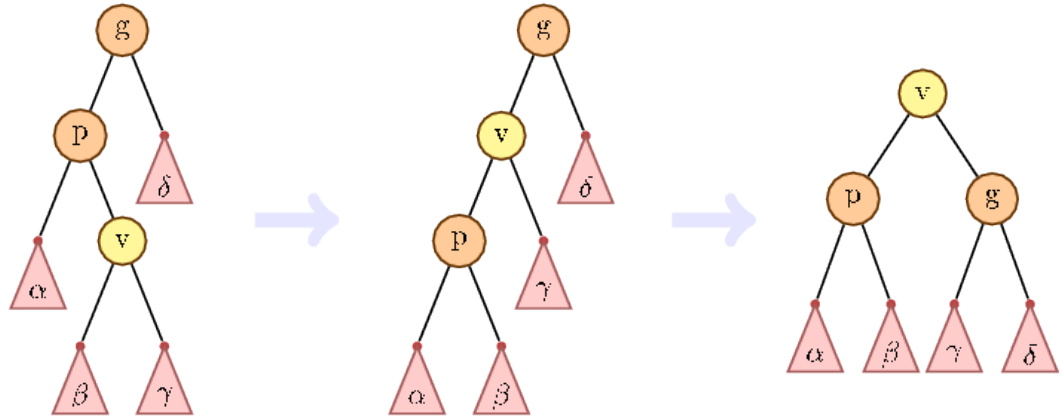
1



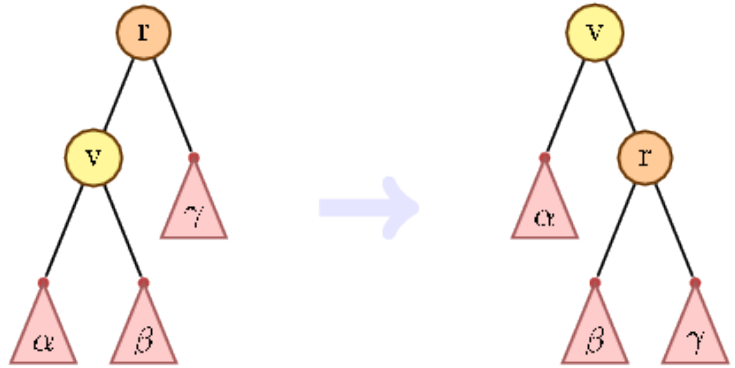
2



3

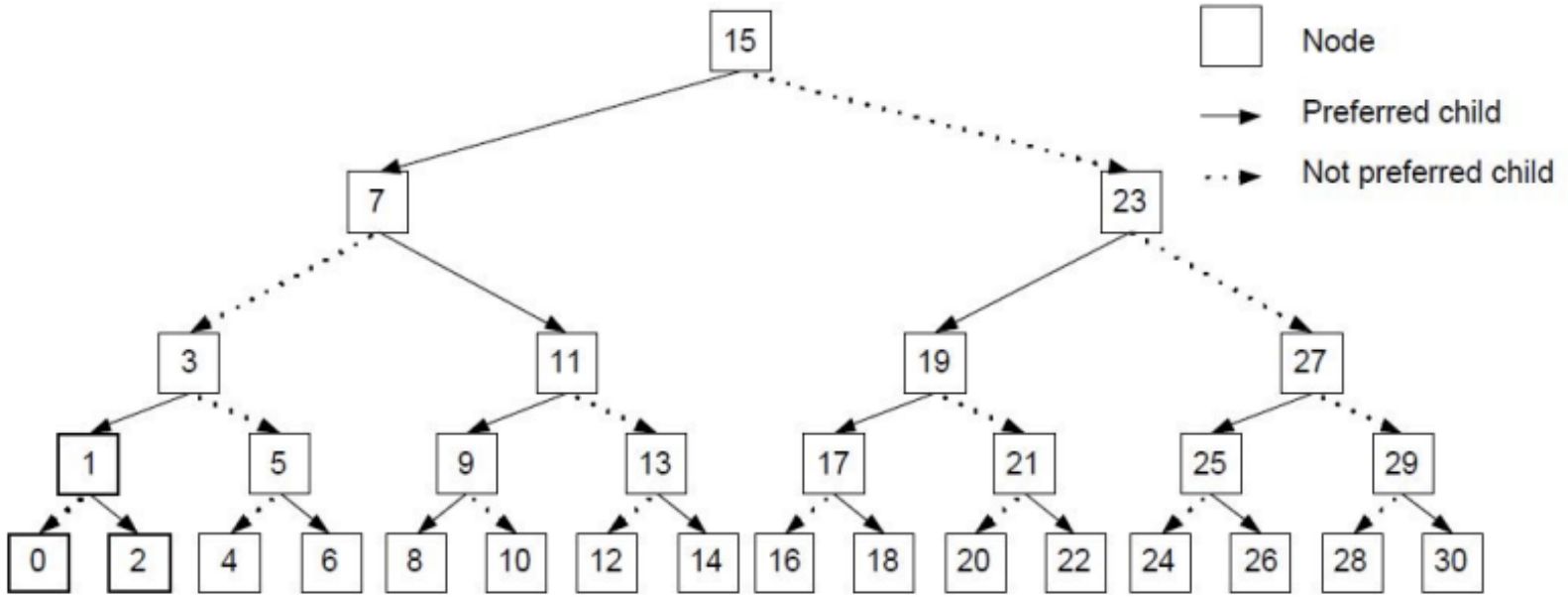


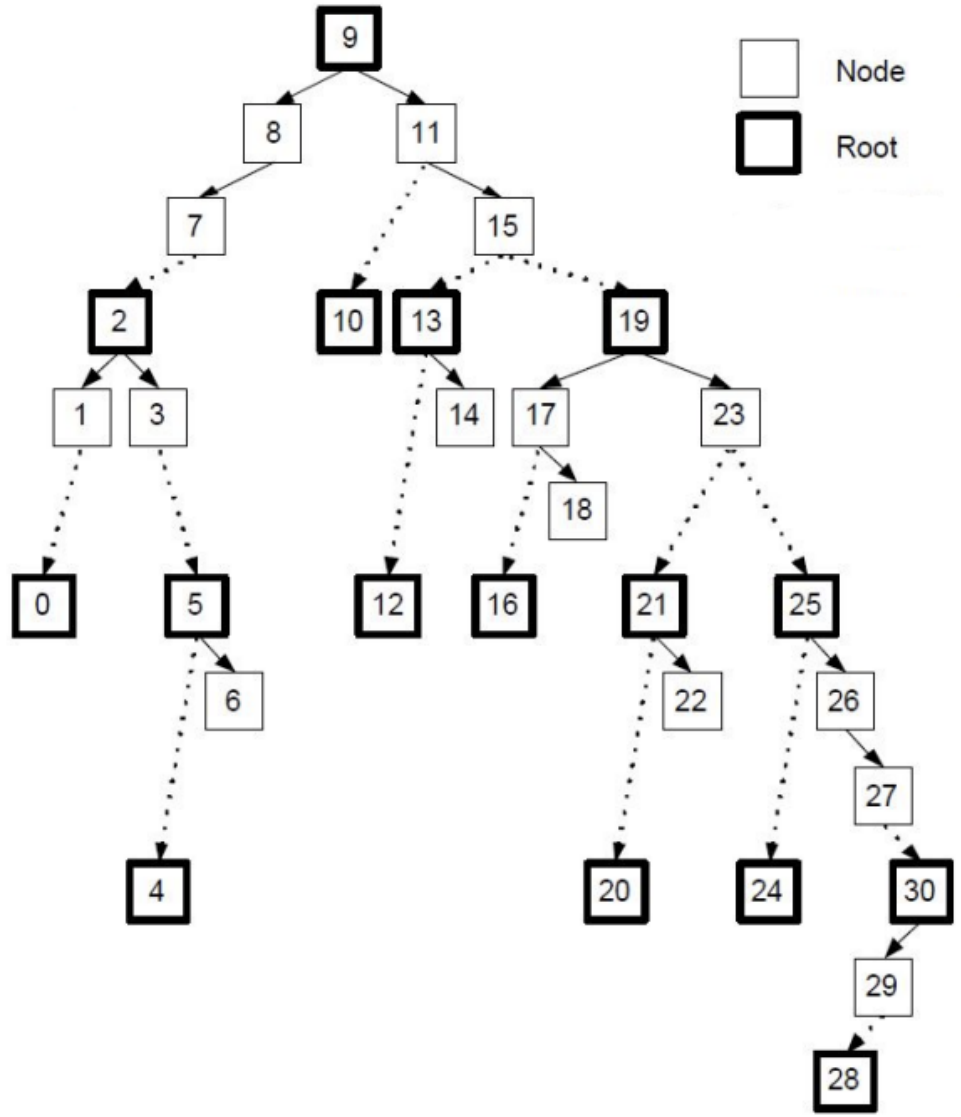
4



Описание реализации

Реализация по большей части не отличается от ранее описанного теоретического варианта.





Дерево Tango на рисунке имеет либо сплошные, либо пунктирные линии для обозначения путей. Набор узлов, соединенных сплошными линиями, — вспомогательное Splay дерево.

Функции, используемые в реализации

- **explore()** — вспомогательная функция, которая отображает все дерево.
- **treeFor()** - функция создания дерева.
- **rotate()** - функция для выполнения операции вращения.
- **splay()** - функция для выполнения операции расширения.
- **switchPath()** — функция для установки значений depth и minDepths в узлах.
- **refParent()** — функция для возврата первого дочернего элемента, значение minDepth которого больше значения depth.
- **expose()** - функция для переноса текущего узла в корень всего дерева.
- **query()** - функция для доступа к элементу в дереве Tango

Обращение к элементу (поиск)

Мы вызываем функцию query() для доступа к необходимому узлу в Tango Tree. Затем функция "поднимает" узел вверх по дереву до тех пор, пока он не достигнет корня Tango Tree.

Псевдокод query():
Ищем, пока не найдем либо ключ, либо NULL
Операция поиска аналогична поиску в бинарном дереве.

```
curr = current node
prev = parent of curr
if curr == NULL:
    expose(prev)
    return false
expose(curr)
return true
```

Тестирование

Оценка производительности выполнена на 3 наборах данных разной размерности (500, 50000, 5000000).

	Время создания	Время поиска
Порядок возрастания		
1. 500	15ns	100ns
2. 50000	1500ns	31000ns
3. 5000000	180000ns	17300000ns
Обратный порядок		
1. 500	15ns	150ns
2. 50000	1500ns	33000ns
3. 5000000	180000ns	12200000ns
Случайный порядок		
1. 500	15ns	350ns
2. 50000	1500ns	83000ns
3. 5000000	180000ns	28500000ns
Одно число		
1. 500	15ns	7ns
2. 50000	1500ns	600ns
3. 5000000	180000ns	59000ns

Результаты тестирования

Самый длительный поиск – это поиск случайного элемента, это предсказуемо потому что к данному типу в этом дереве никаких оптимизаций не применено.
Обратный поиск примерно равен по времени поиску в порядке возрастания.
Поиск одного и того же числа оказался самым быстрым, потому что еще при первой операции поиска дерево перестроилось и это число стало корнем

Результаты

Представлено в репозитории на github:

- Реализация алгоритма
- Отчёт
- Набор тестов
- Презентация

Заключение

Tango tree является самым оптимальным по времени бинарным деревом поиска, но из-за своей статичной структуры, то есть невозможности добавлять, удалять, а также изменять элементы, практического применения не имеет.