Lowest Common Ancestor

Гусев Илья, Булгаков Илья

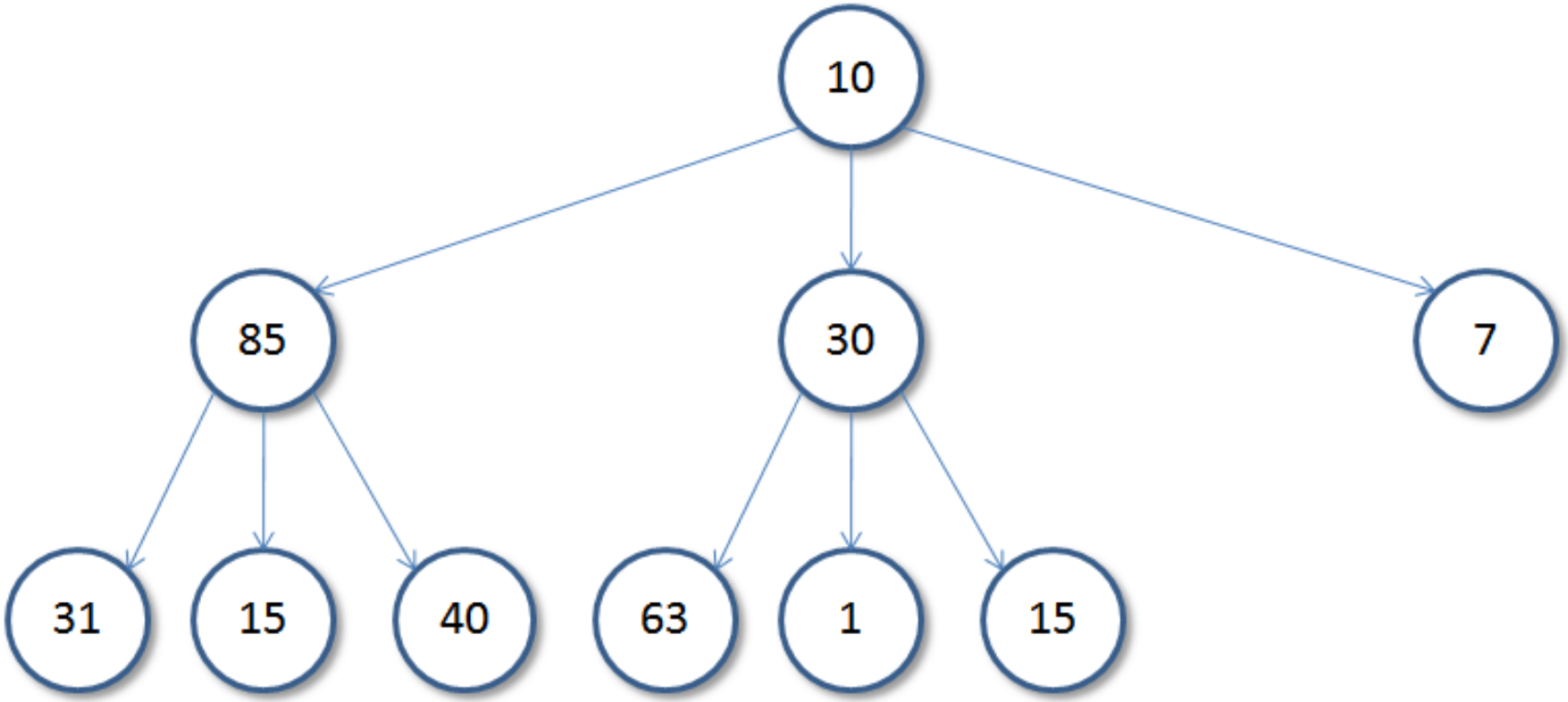
Московский физико-технический институт

Москва, 2019

Содержание

Небинарные деревья

Деревья бывают не только бинарные!
Мы раньше работали в основном с **бинарными деревьями**, но это частный случай более общего вида дерева, которое может ветвиться больше, чем на две дочки.



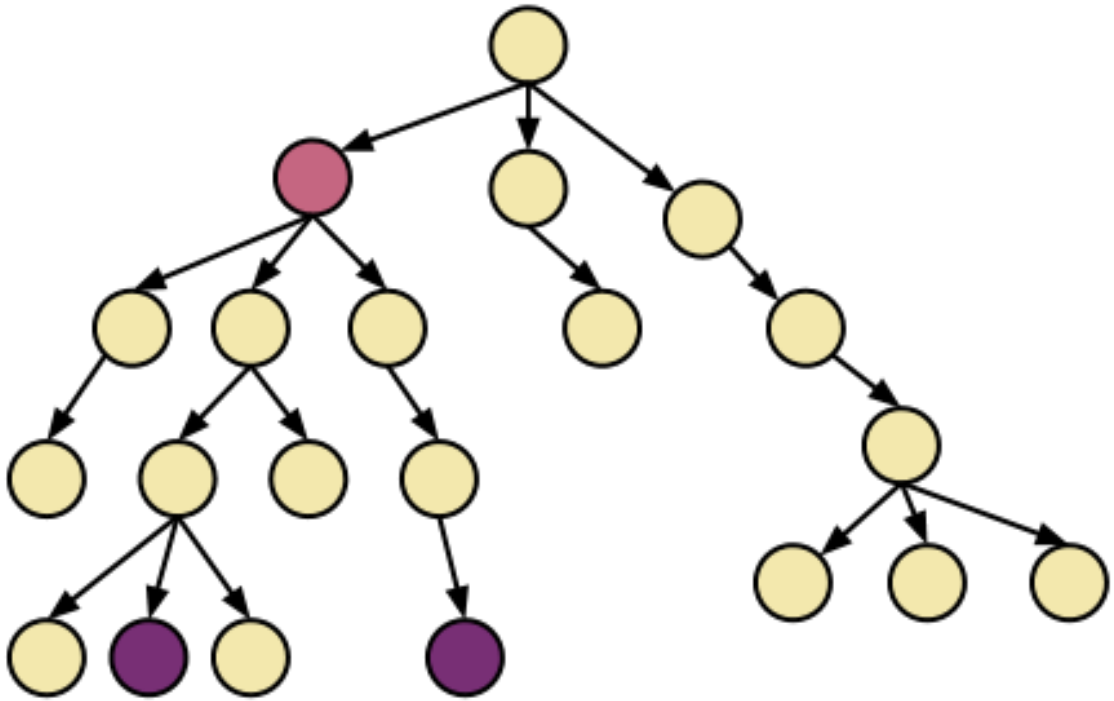
Как реализовать небинарное дерево?

Два способа представления небинарных деревьев:

1. Структура узла: {data, parent, next, first}
   Минусы: new на узел, требуется много памяти.
   Плюсы: удобно работать
2. Структура узла: {data, parent}
   Минусы: неудобно делать обход в глубину
   Плюсы: хранится компактно, для некоторых задач этого достаточно (например, LCA – об этом ниже)

Задача LCA

LCA - Lowest (least) Common Ancestor
Задача нахождения наименьшего общего предка (нижайший общий предок) вершин u и v в корневом дереве T.



LCA - тривиальное решение

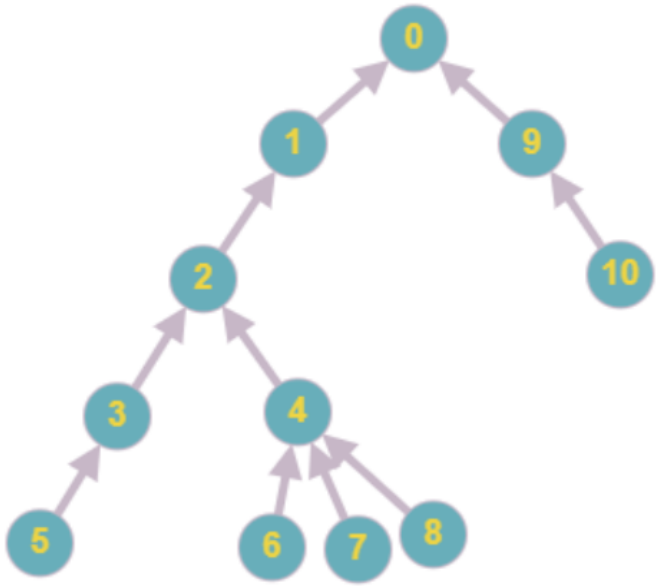
Тривиальное решение задачи LCA

```
LCA(u, v):
    h1 := depth(u)
    h2 := depth(v)

    while h1 != h2:
        if h1 > h2:
            u := parent(u)
            h1 := h1 - 1
        else:
            v := parent(v)
            h2 := h2 - 1

    while u != v:
        u := parent(u)
        v := parent(v)

    return u
```

Сложность?

Метод двоичного подъема для LCA

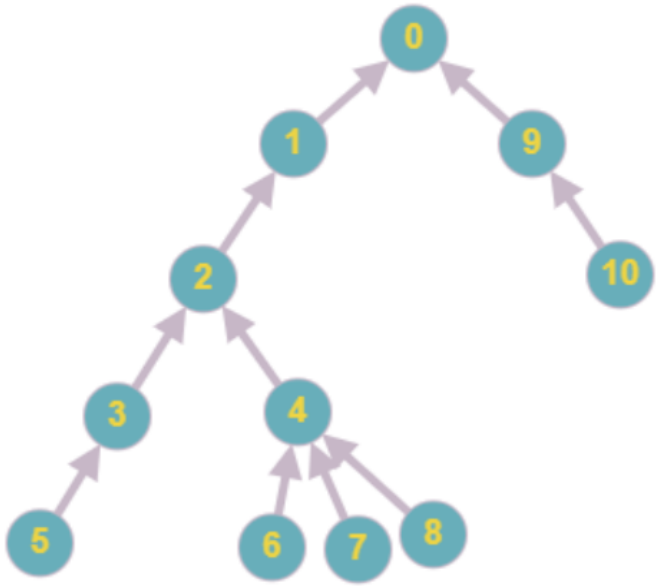
Мотивация: решение за O(n) долгое, хотим быстрее.
Идея метода: выполним препроцессинг, в ходе которого вычислим предков на некоторых глубинах – вычислим для каждой вершины её 1-го предка, 2-го предка, 4-го, 8-го.



Метод двоичного подъема для LCA

Что мы вычисляем в препроцессинге:

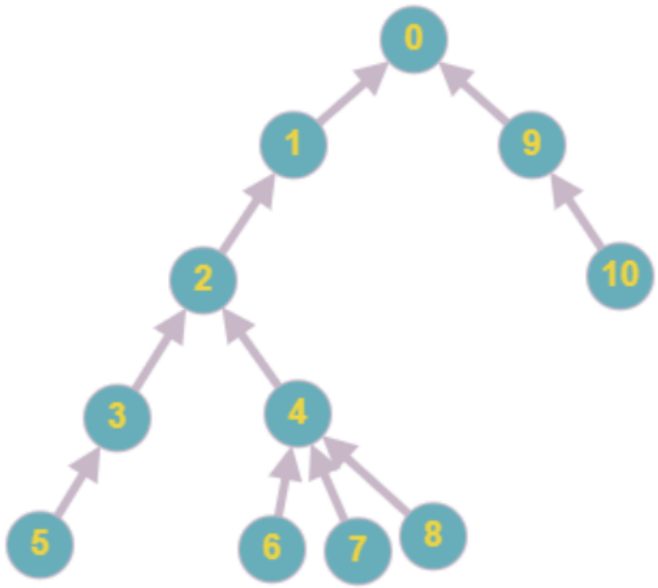
- Массив предков P, т.е. $P[i][j]$ - это 2^j-й предок вершины i, $i = 1..N, j = 0..logN$
  Например,
  P[5][0] = 3
  P[5][1] = 2
  P[5][2] = 0



Метод двоичного подъема для LCA

Что мы вычисляем в препроцессинге:

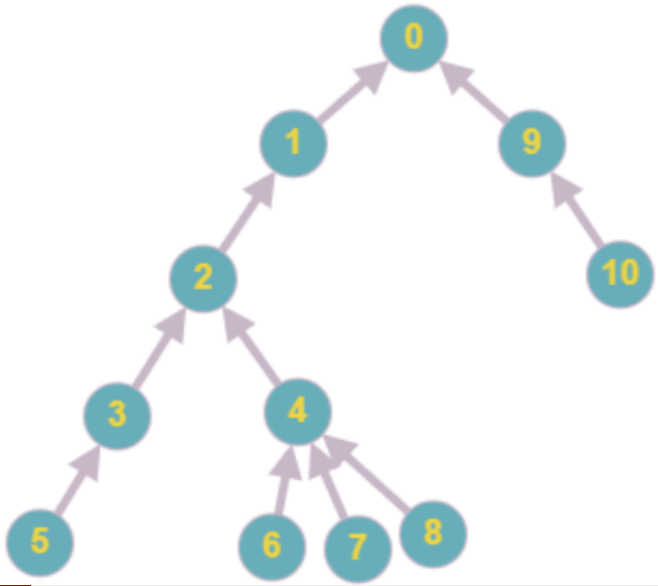
- Для каждой вершины сохраняем пометки входа и выхода при обходе в глубину. Это удобно для определения является ли одна вершина родителем другой: если интервал входа/выхода одной вложен в другой, значит, является обрамляющая является родителем.



Метод двоичного подъема для LCA

Как мы делаем препроцессинг?

- Запускаем DFS. Вычислим для каждой вершины позицию входа и выхода в массиве входов/выходов при обходе в глубину.
  (0 1 2 3 5 5 3 4 6 6 7 7 8 8 4 2 1 9 10 10 9 0)
  Первое упоминание - вход, втрое - выход
- Удобно вычислить родителей 0 уровня, т.е. $P[0]$, прямо тут.
- Динамикой довычисляем все уровни от 1 до log(n)
  P[v][i] = P[P[v][i-1]][i-1]; // Подъем на i это два подъема на i-1. Подъем на 8 - это два подъема на 4.



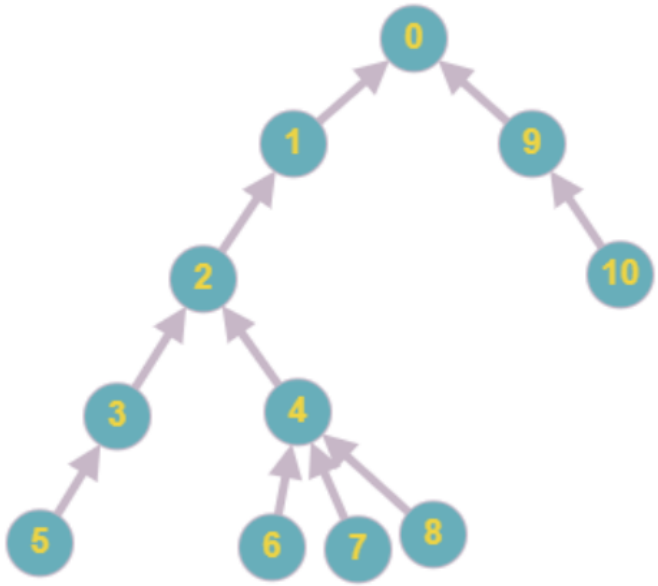
Метод двоичного подъема для LCA

Как искать ответ?

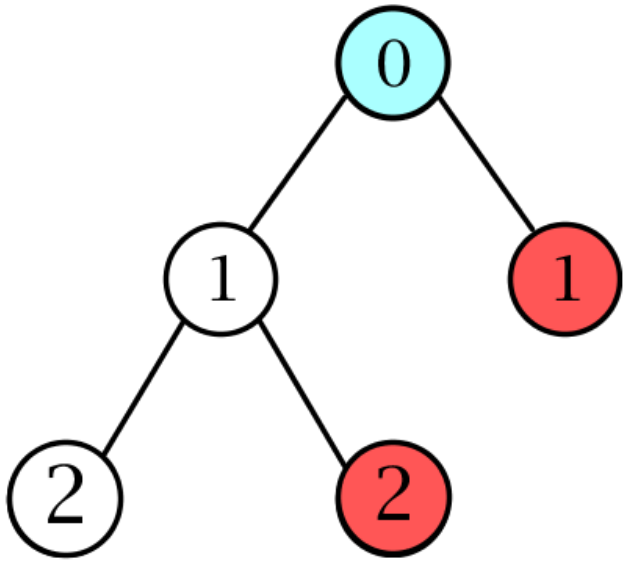
- Проверим, не является ли одна вершина предком другой - в таком случае она и является результатом.
- Будем подниматься по предкам A, пока не найдем самую высокую (т.е. наиболее близкую к корню) вершину, которая ещё не является предком (не обязательно непосредственным) B. Т.е. такую вершину X, что X не предок B, а P[X][0] - предок B.
  Как: пусть сначала I = L. Если P[A][I] не является предком B, то присваиваем A = P[A][I], и уменьшаем I. Если же P[A][I] является предком B, то просто уменьшаем I. Очевидно, что когда I станет меньше нуля, вершина A как раз и будет являться искомой вершиной - т.е. такой, что A не предок B, но P[A][0] - предок B.

Метод двоичного подъема для LCA

Пример. Вычислим общего предка для 5, 10
l = log2 (11) по числу вершин
A = 5, B = 10, l=4. P[A][l] == 0, 0 является предком B
A = 5, B = 10, l=3. P[A][l] == 0, 0 является предком B
A = 5, B = 10, l=2. P[A][l] == 0, 0 является предком B
A = 5, B = 10, l=1. P[A][l] == 2, 2 не является предком B
A = 2, B = 10, l=0. P[A][l] == 1, 1 не является предком B
A = 1, B = 10, l=-1. Останов.



LCA → RMQ



DFS по глубинам (d): 012121010
DFS по вершинам (vtx): 012131040
LCA(3,4) = vtx[rmq(d, l(3), l(4))] = vtx[rmq(d, 4, 7)] = vtx[argmin(2101) + 4] = vtx[6] = 0

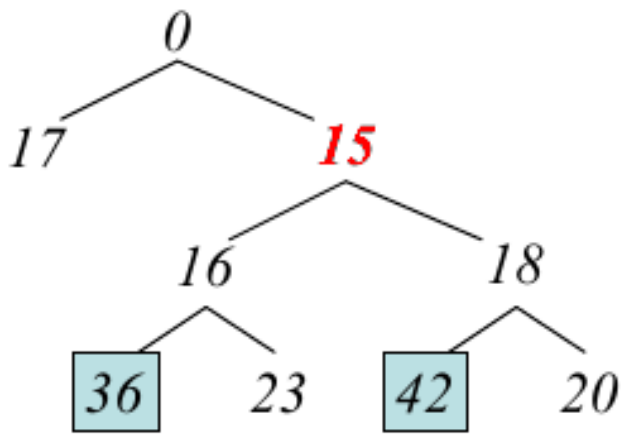
Декартово дерево по неявному ключу

Декартово дерево - бинарное дерево поиска слева направо и куча сверху вниз
Неявный ключ - количество элементов в нашей структуре, находящихся левее нашего элемента
Эквивалентное определение:

- Корнем дерева является элемент массива, имеющий минимальное значение A, скажем A[i]
- Левым поддеревом является декартово дерево по неявному ключу на массиве A[1..i-1]
- Правым поддеревом является декартово дерево по неявному ключу на массиве A[i+1..N]

RMQ → LCA

$$A = [17, 0, 36, 16, 23, 15, 42, 18, 20]$$



RMQ(i, j) = LCA(A[i], A[j])

Полезные ссылки I

E-maxx: метод двоичного подъёма
http://e-maxx.ru/algo/lca_simpler

Викиконспекты: LCA2RMQ
http://bit.ly/2vGza6e

Викиконспекты: RMQ2LCA
http://bit.ly/2vDI2tp