Lowest Common Ancestor

Дефиниция и примери

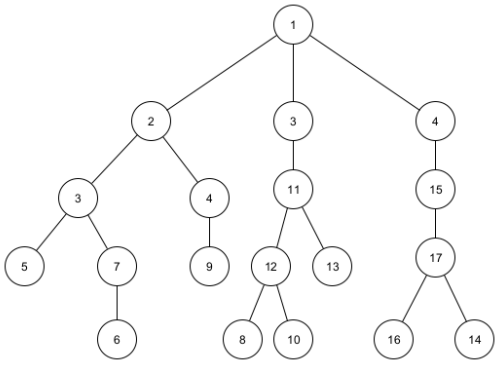
Дефиниция

Нека имаме кореново дърво T с корен $root$. Връх u се нарича предшественик на връх v, ако u лежи на единствения път между $root$ и v. Ще бележим с $depth(u)$ дълбочината на връх u или с други думи казано - разстоянието от $root$ до u. При тази конвенция, най-близкият общ предшественик на u и v, който ще бележим с $lca(u, v)$, ще бъде върхът x с максимална дълбочина, който е предшественик едновременно на u и v.

Примери

В този случай коренът е връх 1.

Фигура 1



- $lca(5, 7) = 3$;

- $lca(4, 9) = 4$;
- $lca(3, 9) = 2$;
- $lca(2, 13) = 1$;
- $lca(4, 14) = 4$;
- $lca(2, 4) = 1$;

Свойства на lca(u, v)

- $lca(u, v) = lca(v, u)$
- $lca(u, v) = u$ тогава и само тогава, когато u е предшественик на v
- $lca(u, lca(v, w)) = lca(lca(u, v), w)$.

  По тази причина можем да записваме обобщено $lca(u, v, w)$. Това е полезно, ако например имаме нужда да пресмятаме lca на цели интервали от масив. Поради това свойство знаем, че задачата има смисъл и може да се реши със сегментно дърво. От следващото свойство ще научим, че има и по-добър начин.

- $lca(u, v, v) = lca(u, v)$

  Това значи, че добавянето на "излишни"върхове към тези, на които търсим lca не променя отговора. Тост ако искаме да сметнем lca на група върхове можем да си ползволим да сметнем lca на група, която съдържа тези върхове, но по няколко пъти. Това ни дава възможност да пресмятаме lca на интервал от върхове чрез sparse table. Повече по темата ще научите по-късно.

- $lca(u, v) = lca(u, v')$, където v' се получава, когато "качим"връх v на нивото на връх u (разбира се, приемаме, че $depth(u) \leq depth(v)$)
- $depth(lca(u, v)) \leq depth(u)$ и $depth(lca(u, v)) \leq v$.

Можете да се убедите с примери от Фигура 1:

- u = 5, v = 6, v' = 7
- u = 2, v = 13, v' = 3
- u = 11, v = 14, v' = 15
- u = 2, v = 9, v' = 2

Тук ще покажем не съвсем формално доказателство на свойството. Ясно е, че след като само "покачваме"върхове то няма как lca да "слезе". Затова само трябва да проверим, че $lca(u, v)$ е предшественик на u и v'. Нека запишем $l = lca(u, v)$, тривиално е, че l предшества u. Сега трябва да видим дали предшества v'. Допускаме противното: нека l не предшества v'. Ние знаем, че l предшества v. Тъй като ние само сме се качвали нагоре, то единственият вариант l вече да не предшества v' е да сме го "подминали тоест l да лежи на пътя между v и v'. Ясно е също и че $l \neq v'$. Това обаче е противоречие, понеже $depth(l) \leq depth(u) = depth(v') \leq depth(v)$ и излиза, че няма как l да лежи на пътя между v и v'.

LCA с binary lifting

Използвайки тези свойства можем да изготвим алгоритъм, който намира $lca(u, v)$ за $O(logN)$.

Забелязваме, че ако имаме два върха u и v на равна дълбочина, то ако започнем постепенно да ги "качваме"едновременно, то стойностите ще се различават до един момомент и по-конкретно, първата стойност на съвпадение ще бъде точно $lca(u, v)$. Нека имаме функция $rise(u, n)$, която връща връх, отговарящ на връх u след като бъде "качен"с n стъпки. Тоест така излиза, че трябва да намерим най-малкото n, за което $rise(u, n) = rise(v, n)$ и тази стойност ще бъде $lca(u, v)$. По-късно обаче, ще стане ясно, че е по-добре да намерим последното n, за което $rise(u, n) \neq rise(v, n)$ и така $lca(u, n)$ ще бъде $parent(rise(u, n))$, където $paret(x)$ е родителят на x.

От тези наблюдения става ясно, че е доста удобно да имаме бърз начин да "качваме"върхове. По тази причина е измислена техниката binary lifting. Идеята е следната: за всеки връх x пазим $rise(x, 1)$, $rise(x, 2)$, $rise(x, 4)$, ..., $rise(x, 2^k)$, ... (разбира се няма нужда да пазим тези стойности до безкрай, а само до достатъчно голяма степен на 2). Ключовото наблюдение тук е, че тези стойности могат да се пресмятат доста лесно. Например: $rise(x, 1) = parent(x)$. Също така $rise(x, 2) = parent(parent(x)) = rise(rise(x, 1), 1)$. Забелязваме също и че: $rise(x, 2^k) = rise(rise(x, 2^{k-1}), 2^{k-1})$. Това е вярно, понеже искаме да направим 2^k стъпки нагоре, които можем да разделим на два скока по 2^{k-1}. Разбира се, че ако се опитаме да скочим прекалено нагоре, няма да има връх в дървото, който да отговаря на този скок. Например $rise(2, 2)$ в дървото от Фигура 1 не съществува. Затова за такива случаи записваме, че стойността е -1. Също така приемаме, че $rise(-1, 2^k) = -1$.