Lowest Common Ancestor and Range Minimum Query

Jarosław Gomułka

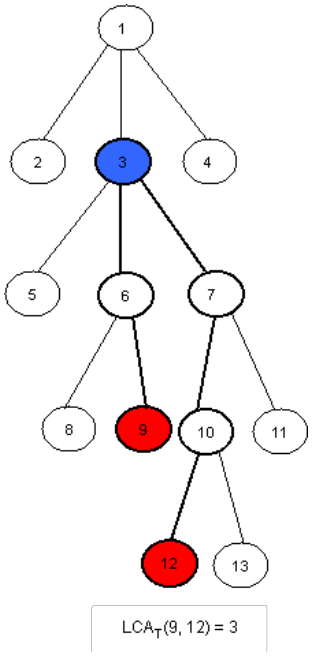
Wrocław, 31 stycznia 2011

Streszczenie

LCA to problem odpowiadania na serie pytań o najniższy wspólny przodek dwóch wierzchołków w ukorzenionym drzewie. RMQ to problem w którym odpowiada się na serie pytań o indeks minimalnego elementu w zadanym fragmencie tablicy. Praca zawiera przegląd algorytmów dla obu problemów zaczynając od najprostszych i najwolniejszych aż do algorytmów optymalnych. Wszystkie opisane algorytmy są analizowane zarówno jeśli chodzi o złożoność czasową jak i o pamięciową.

1 Definicja problemu LCA

Niech K to korzeń drzewa T. Zbiór przodków wierzchołka A to zbiór wierzchołków będących na ścieżce od A do K. W zapytaniu LCA(X,Y) chcemy znaleźć wierzchołek który jest przodkiem wierzchołków X i Y. Jeśli istnieje wiele wspólnych przodków chcemy znaleźć ten który jest najbardziej oddalony od korzenia. Przykładowy wynik zapytania:



Źródło: [1]

Uwaga 1.1 *Przodkowie 9 to: 9, 6, 3, 1*
Przodkowie 12 to: 12, 10, 7, 3, 1
Wspólni przodkowie 9 i 12: 3,1
Najniższy wspólny przodek: 3

1.1 Model złożoności

Złożoność rozwiązania problemu to para $< f(n), g(n) >$. przy czym
$f(n)$ - czas potrzebny na przygotowanie odwiedniej struktury danych
$g(n)$ - czas odpowiedzi na pojedyncze zapytanie

1.2 Algorytm trywialny

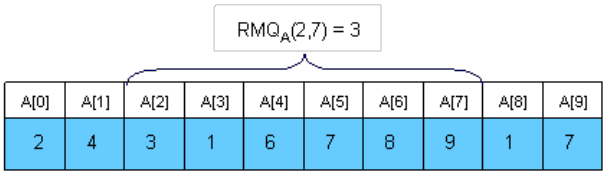
W preprocessingu należy obliczyć dla każdego wierzchołka w drzewie

- Odległość od korzenia
- Następny wierzchołek na ścieżce do korzenia.

Ponieważ rozpatrywana struktura jest drzewem, wszystko można wyliczyć poprzez odwiedzanie dfsem wierzchołków w kolejności pre-order. Odpowiedź na pytanie $LCA(x, y)$ można wyznaczyć poprzez znalezienie ścieżek z x i y do korzenia. Następnie wyznaczyć najbardziej oddalony wierzchołek od korzenia który leży na obu ścieżkach. Złożoność tego rozwiązania to $< O(n), O(n) >$.

2 Definicja problemu RMQ

Zapytanie $RMQ_T(x, y)$ jest zdefiniowane jako indeks minimalnego elementu w tablicy T spośród elementów o indeksach z przedziału $[x, y]$. Formalnie $RMQ_T(x, y) = \{res \in [x, y] | \forall e \in [x, y] \quad T[res] <= T[e]\}$. Należy stworzyć strukture która przetworzy daną tablice i będzie w stanie szybko odpowiadać na pytania RMQ.

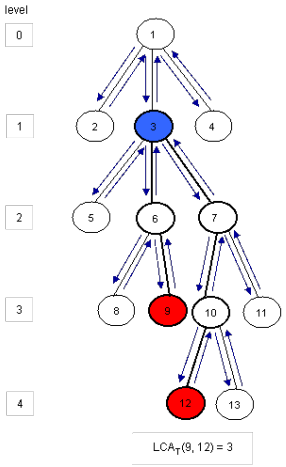


Źródło: [1]

3 Równoważność problemów LCA i RMQ

3.1 RMQ => LCA

Okazuje się, że mając algorytm rozwiązujący problem RMQ w czasie $< f(n), g(n) >$ potrafimy rozwiązać LCA w czasie $< f(2*n)+O(n), g(2*n)+O(1) >$. Poniżej opisuje metode konstrukcji algorytmu rozwiązującego LCA korzystając z RMQ.



Źródło: [1]

Niech

- CE - kolejność odwiedzania wierzchołków na cyklu eulera w drzewie.
- $H[x]$ - wysokość wierzchołka $CE[x]$
- $R[x]$ - indeks pierwszego wystąpienia wierzchołka x w tablicy CE.

Lemat: $LCA = CE[RMQ_H(R[x], R[y])]$. Dowód:

Element reprezentujący LCA znajdzie się między indeksami $R[x]$, $R[y]$

Ciąg elementów pomiędzy indeksami $R[x]$ i $R[y]$ tworzy ścieżke pomiędzy wierzchołkami x i y. Z tego wynika, że wszystkie wierzchołki ze ścieżki $R[x]$ -> $R[y]$ znajdują się między indeksami $R[x]$ i $R[y]$ w tablicy CE. Wobec tego znajduje się tam też $LCA(x, y)$.

Zostanie wybrany wierzchołek będący LCA

Element $LCA(x, y)$ znajdzie się między indeksami $R[x]$, $R[y]$ w tablicy CE. Z tego wynika, że jeśli wierzchołek będący $LCA(x, y)$ nie został wybrany to znaczy, że wśród elementów $CE[R(x)..R(y)]$ znajduje się wierzchołek mający niższą wysokość niż $LCA(x, y)$. Z tego wynika, że przed indeksem $R[y]$, ścieżka opuściła poddrzewo o korzeniu $LCA(x, y)$. Wobec tego wierzchołek y nie znajduje się w podrzewie o korzeniu $LCA(x, y)$. Z tego wynika, że $LCA(x, y)$ nie jest przodkiem y. Sprzeczność.

3.2 LCA => RMQ

Umiemy zrobić LCA, chcemy zrobić RMQ. Definicja CartesianTree:

- CartesianTree jest drzewem binarnym.
- Każdej liczbie w ciągu przypisany jest wierzchołek.
- Do jednego wierzchołka przypisany jest dokładnie jeden element ciągu.
- Przejście drzewa w kolejności in-order i wypisywanie kolejno odwiedzanych wierzchołków powoduje wypisanie oryginalnego ciągu.
- Wartość ojca dla każdego wierzchołka (o ile takiego posiada) jest mniejsza niż wartość rozpatrywanego wierzchołka.

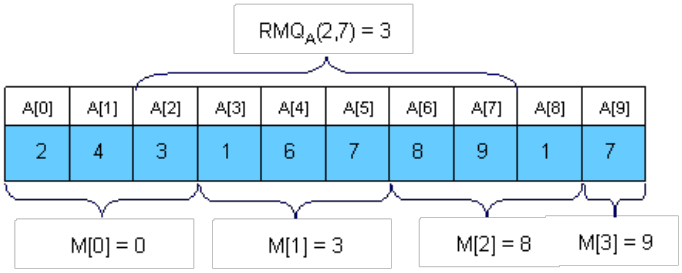
Lemat: $RMQ_T(x, y) = LCA_{CartesianTree(T)}(x, y)$. Dowód jest prosty i można go przeprowadzić po indukcji drzewa.

CartesianTree da się zbudować w czasie liniowym. Zostało to szczegółowo opisane w [2].

4 Rozwiązania problemów LCA i RMQ

4.1 Algorytm <O(n), O($\sqrt{n}$) > rozwiązujący RMQ

Podzielmy tablice T o rozmiarze n na kawałki o rozmiarze $\sqrt{n}$. Dla każdego kawałka obliczmy indeks minimalnego elementu spośród indeksów przykrytych przez dany kawałek.



Źródło: [1]

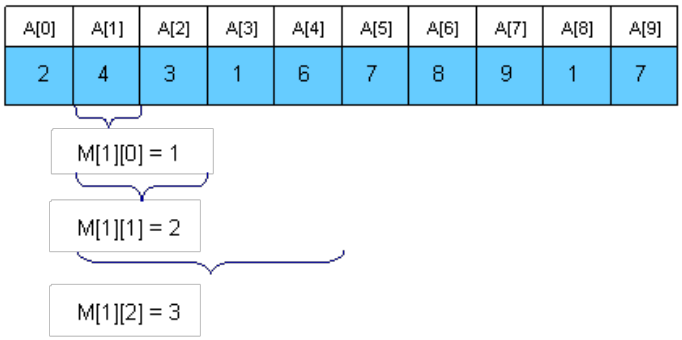
Odpowiedź na pojedyncze zapytanie składa się z 2 części:

- Iteracja po wszystkich kawałkach i rozpatrzeniu minimów spośród kawałków które w całości zawierają się w obszarze zapytania. Złożoność tej części to $O(\sqrt{n})$ ponieważ kawałków jest $\sqrt{n}$.
- Iteracja po wszystkich elementach kawałków które częściowo są zawarte w obszarze zapytania. Ponieważ takich kawałków jest maksymalnie dwa, to złożność tej części to $O(\sqrt{n})$.

Zatem sumaryczna złożoność czasu potrzebnego na odpowiedzenie na pojedyncze pytanie to $O(\sqrt{n})$.

4.2 Algorytm <O($n \log n$), O(1) > rozwiązujący RMQ

Lepszym rozwiązaniem jest algorytm oparty o tablice minimów. W elemencie tablicy M[x][y] pamiętany jest indeks minimalnego elementu spośród elementów o indeksach z przedziału $[x, x + 2^y - 1]$.



Źródło: [1]

Rozmiar tablicy T to $O(n \log n)$ ponieważ pierwszy wymiar zajmuje rozmiar n a drugi $\log n$. Obliczenie tablicy M można wykonać w czasie $O(n \log n)$. Należy skorzystać ze wzoru:

$$M[i][j] = \begin{cases} M[i][j-1] & T[M[i][j-1]] <= T[M[i+2^{j-1}][j-1]] \\ M[i+2^{j-1}][j-1] & \text{w przeciwnym przypadku} \end{cases}$$

Iterowanie najpierw po drugim wymiarze a nastepnie po pierwszym, powoduje że zawsze będziemy odwoływać się do elementów wcześniej już obliczonych.

Odpowiedź na pojedyncze pytanie polega na rozpatrzeniu 2 obszarów, które całkowicie przykryją rozpatrywany fragment tablicy. Niech $k = \lfloor \log(y - x + 1) \rfloor$.

$$RMQ_T(x, y) = \begin{cases} M[x][k] & T[M[x][k]] <= T[M[y-2^k+1][k]] \\ M[y-2^k+1][k] & \text{w przeciwnym przypadku} \end{cases}$$

Rozpatrywane 2 obszary tablicy pokrywają cały rozpatrywany fragment tablicy więc wynik będzie poprawny.

4.3 Algorytm <O(n), O(1) > do obliczania LCA

Niech

- CE - kolejność odwiedzania wierzchołków na cyklu eulera po drzewie.
- $H[x]$ - wysokość wierzchołka $CE[x]$
- $R[x]$ - indeks pierwszego wystąpienia wierzchołka x w tablicy CE.

Zauważmy że kolejne elementy w tablicy H różnią się zawsze o 1. Niech $ZH[x] = H[x] - H[x-1]$. Podzielmy tablice ZH na kawałki o rozmiarze $\frac{\log n}{2}$

Idea obliczania wyniku jest następująca:

$$RMQ_H(x,y) = min \begin{cases} \text{minimum z kawałków które zawierają się w obszarze zapytania} \\ \text{minimum z elementów dwóch kawałków które są częściowo przykryte} \end{cases}$$

Kawałki całkowicie zawierające się w obszarze zapytania

Niech $ZH[i] = RMQ_H(\frac{i \log n}{2}, \frac{(i+1)\log n}{2} - 1)$.
Rozmiar tablicy ZH to $O(\frac{n}{\log n})$. Do obliczania RMQ w tablicy ZH możemy użyć algorytmu <O($n \log n$), O(1) >. W takim wypadku złożoność preprocessingu wyniesie O($\frac{n}{\log n} \log n$) = O(n). Czas odpowiedzi na pytanie pozostanie O(1).

Kawałki częściowo pokryte przez obszar zapytania

Ponieważ kawałek jest reprezentowany przez $\frac{\log n}{2}$ bitów, można stablicować wyniki dla wszystkich możliwych kawałków. Złożoność tej operacji to O($2^{\frac{\log n}{2}} * \frac{\log n}{2}^2$) = O($\sqrt{n}\frac{\log n}{2}^2$) co w szczególności jest O(n).

Podsumowując preprocesing zajął czas liniowy, a odpowiadanie na zapytania w obu przypadkach zajmuje czas stały czyli udało się osiągnąc optymalny algorytm dla LCA i RMQ czyli <O(n), O(1) >.

Literatura

[1] `http://www.topcoder.com/tc?module=Static&d1=tutorials&d2=lowestCommonAncestor`

[2] Gabow, Harold N.; Bentley, Jon Louis; Tarjan, Robert E. (1984), Ścaling and related techniques for geometry problems", STOC '84: Proc. 16th ACM Symp. Theory of Computing, New York, NY, USA: ACM, pp. 135–143, `http://dx.doi.org/10.1145/800057.808675`, ISBN 0-89791-133-4.