AiSD, Lab 06, Zadanie Park Bitowy (mini omówienie)

Tomasz Waleń

Wydział Matematyki, Informatyki i Mechaniki, UW

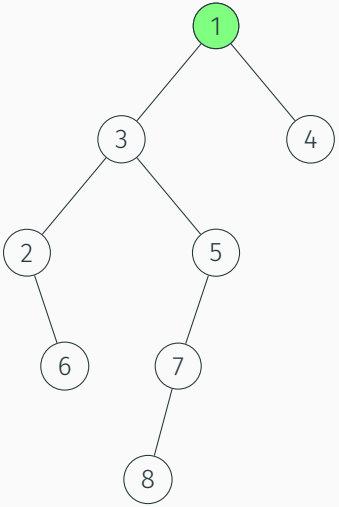
Definicja problemu

Dla ustalonego drzewa binarnego T, należy odpowiedzieć m zapytań postaci QUERY(v_i, d_i):

- wyznacz wierzchołek u_i, taki, że $dist_T(v_i, u_i) = d$,
- jeśli taki wierzchołek nie istnieje odpowiedzią jest -1.

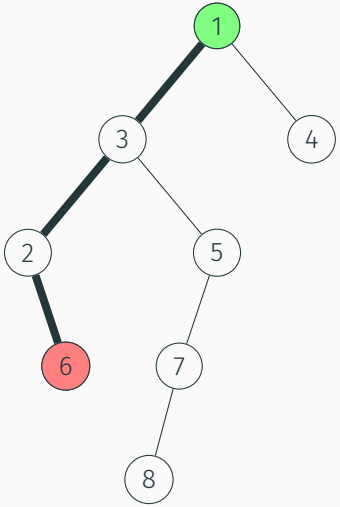
Przykład

QUERY$(v = 1,\ d = 3) : ?$



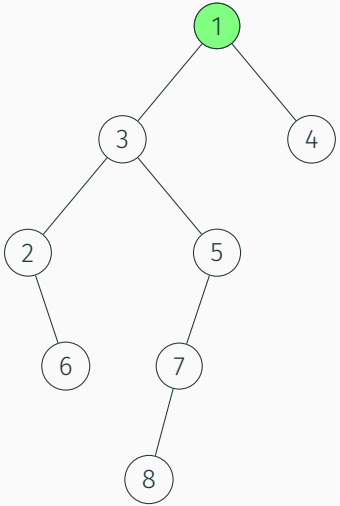
Przykład

QUERY$(v = 1,\ d = 3)$: 6



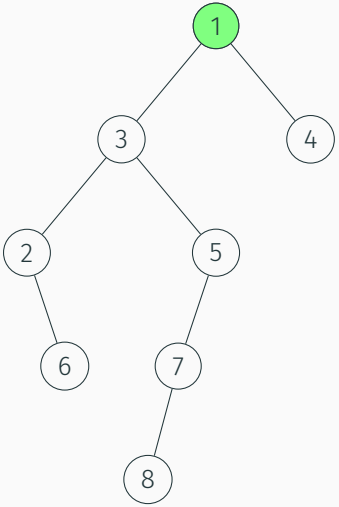
Przykład

QUERY$(v = 1,\ d = 3) : 6$
QUERY$(v = 1,\ d = 5) : ?$



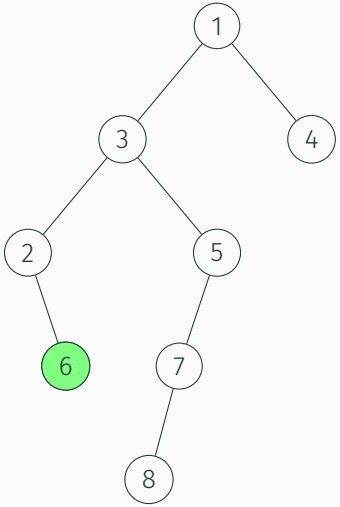
Przykład

QUERY$(v = 1,\ d = 3) : 6$
QUERY$(v = 1,\ d = 5) :$ $brak$



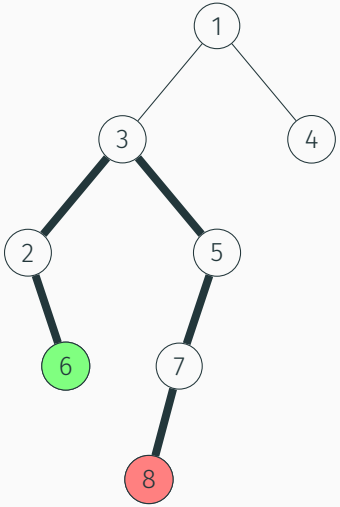
Przykład

QUERY$(v = 1,\ d = 3) : 6$
QUERY$(v = 1,\ d = 5) : brak$
QUERY$(v = 6,\ d = 5) : ?$



Przykład

QUERY$(v = 1,\ d = 3) : 6$
QUERY$(v = 1,\ d = 5) : brak$
QUERY$(v = 6,\ d = 5) : 8$



Pierwsze rozwiązanie

- dla każdego wierzchołka v oblicz maksymalną odległość:

$$\textsc{MaxDist}(v) = \max\{dist(v,u) : u \in T\}$$

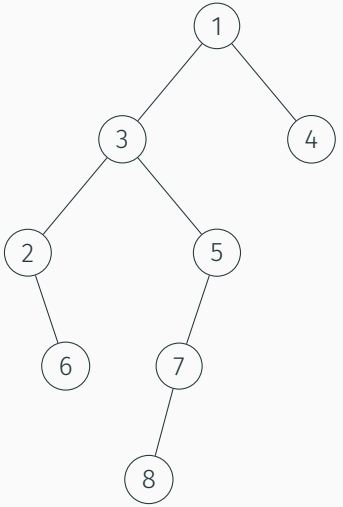
oraz

$$\textsc{ArgMaxDist}(v) = u \text{ taki, że } dist(v,u) = \textsc{MaxDist}(v)$$

- dla zapytania (v, d):
  - jeśli $d > \textsc{MaxDist}(v)$ to brak jest odpowiedzi,
  - wpp. na ścieżce $v \to \textsc{ArgMaxDist}(v)$ znajdujemy wierzchołek u w odległości d od v.

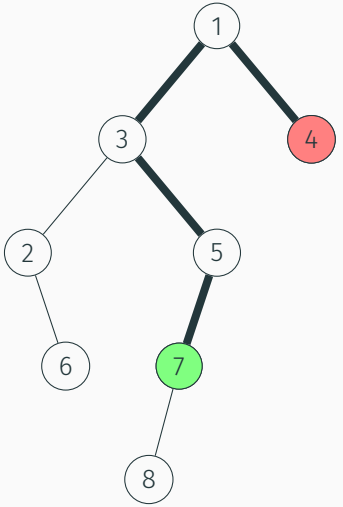
Przykład

v	MaxDist(v)	ArgMaxDist(v)
1	4	8
2	4	8
3	3	8
4	5	8
5	3	6
6	5	8
7	4	4
8	5	4



Przykład

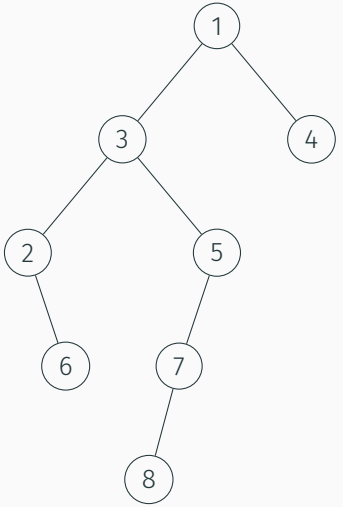
v	MaxDist(v)	ArgMaxDist(v)
1	4	8
2	4	8
3	3	8
4	5	8
5	3	6
6	5	8
7	4	4
8	5	4



Przykład

v	MaxDist(v)	ArgMaxDist(v)
1	4	8
2	4	8
3	3	8
4	5	8
5	3	6
6	5	8
7	4	4
8	5	4

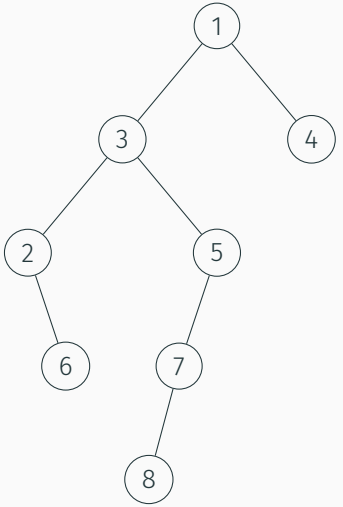
Czy dla tego drzewa to jest jedyna możliwa tabelka?



Przykład

v	MaxDist(v)	ArgMaxDist(v)
1	4	8
2	4	8
3	3	8
4	5	8
5	3	8
6	5	8
7	4	4
8	5	4

Czy dla tego drzewa to jest jedyna możliwa tabelka?



MaxDist i ArgMaxDist

- MaxDist(v) jest oczywiście jednoznacznie wyznaczony,
- natomiast ArgMaxDist może mieć wiele prawidłowych wartościowań,
- jednak dla każdego drzewa możemy znaleźć dwa wierzchołki v_a, v_b takie, że ArgMaxDist$(v) = v_a$ lub ArgMaxDist$(v) = v_b$
- dzieje się tak gdy ścieżka $v_a \to v_b$ to średnica drzewa,
- ta obserwacja znacznie upraszcza nam obliczanie tabelki MaxDist.

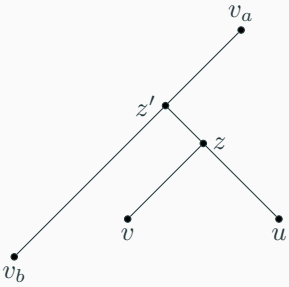
Dlaczego średnica drzewa?

Niech $v_a \to v_b$ to średnica drzewa T, dla dowolnych wierzchołków $v, u \in T$:

$$dist(v,u) \leq \max(dist(v,v_a), dist(v,v_b))$$

Szkic dowodu.
Załóżmy przez sprzeczność, że $dist(u,v) > dist(v,v_a) \geq dist(v,v_b)$ i rozważmy drzewo ukorzenione w v_a. Niech $z = LCA(v,u)$ (najniższy wspólny przodek v i u), oraz $z' = LCA(v_b, z)$.

$dist(v,u) > dist(v,v_a)$



□

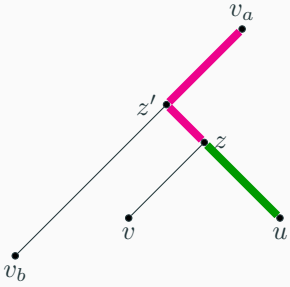
Dlaczego średnica drzewa?

Niech $v_a \to v_b$ to średnica drzewa T, dla dowolnych wierzchołków $v, u \in T$:

$$dist(v,u) \leq \max(dist(v,v_a), dist(v,v_b))$$

Szkic dowodu.
Załóżmy przez sprzeczność, że $dist(u,v) > dist(v,v_a) \geq dist(v,v_b)$ i rozważmy drzewo ukorzenione w v_a. Niech $z = LCA(v,u)$ (najniższy wspólny przodek v i u), oraz $z' = LCA(v_b, z)$.

$dist(v,u) > dist(v,v_a)$
$dist(z,u) > dist(z,z') + dist(z',v_a)$



□

Dlaczego średnica drzewa?

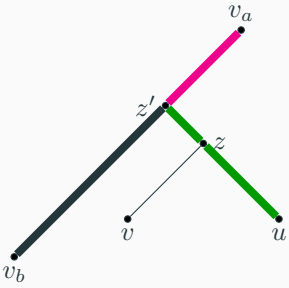
Niech $v_a \to v_b$ to średnica drzewa T, dla dowolnych wierzchołków $v, u \in T$:

$$dist(v,u) \leq \max(dist(v,v_a), dist(v,v_b))$$

Szkic dowodu.
Załóżmy przez sprzeczność, że $dist(u,v) > dist(v,v_a) \geq dist(v,v_b)$ i rozważmy drzewo ukorzenione w v_a. Niech $z = LCA(v,u)$ (najniższy wspólny przodek v i u), oraz $z' = LCA(v_b, z)$.

$dist(v,u) > dist(v,v_a)$
$dist(z,u) > dist(z,z') + dist(z',v_a)$

ale to oznacza, że:
$dist(v_b,z') + dist(z,z') + dist(z,u) > dist(v_b,z') + dist(z',v_a)$
SPRZECZNOŚĆ – bo $v_a \to v_b$ to średnica



□

Obliczanie MaxDist i ArgMaxDist

Algorytm 1: Calc

Input: drzewo T

znajdź średnicę drzewa $v_a \to v_b$

za pomocą DFS znajdź wszystkie $dist(v_a, v)$ i $dist(v_b, v)$

foreach $v \in T$ **do**

MaxDist$(v) = \max(dist(v_a, v), dist(v_b, v))$

ArgMaxDist$(v) = \arg\max(dist(v_a, v), dist(v_b, v))$

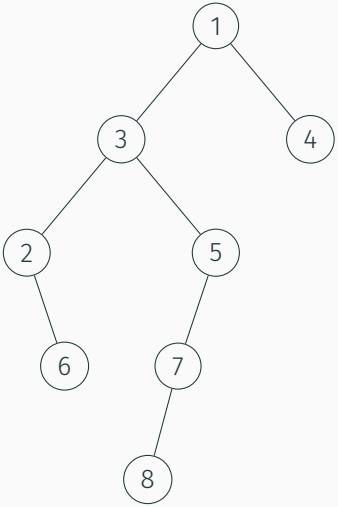
end

Ale jak poruszać się po drzewie?

Na początek nauczymy się poruszać w kierunku korzenia:

$\textsc{up}(v, d) = u$ który jest przodkiem v w odległości d

Przykład:

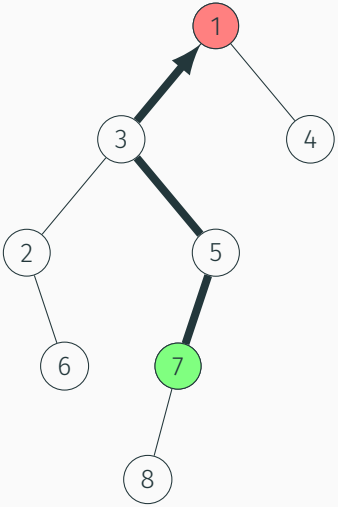


Ale jak poruszać się po drzewie?

Na początek nauczymy się poruszać w kierunku korzenia:

$\textsc{up}(v, d) = u$ który jest przodkiem v w odległości d

Przykład:
$\textsc{up}(7, 3) = 1$



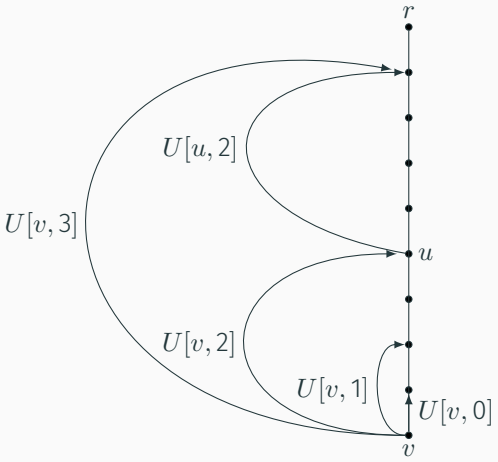
Zapytania UP

Zauważmy, że możemy efektywnie obliczyć wszystkie wartości UP, gdzie d jest potęgą dwójki:

$U[v,k] = \text{UP}(v, 2^k)$ dla $0 \leq k \leq \lfloor \log n \rfloor$

ponieważ:

$$U[v,0] = parent(v)$$
$$U[v,k+1] = U[U[v,k],k]$$



Zapytania UP

Algorytm 2: UP(v, d)

```
foreach i = ⌊log n⌋ downto 0 do
    if d ≥ 2^i then
        v = U[v, i]
        d = d − 2^i
    end
end
return v
```

Rozwiązanie $O(n \log n)$

Algorytm 3: Solve

Input: drzewo T oraz lista zapytań $Q = (v_1, d_1), \ldots, (v_m, d_m)$

znajdź średnicę drzewa $v_a \to v_b$

Niech T_a (odpowiednio T_b) oznacza drzewo o korzeniu v_a (odpowiednio v_b)

Przygotuj drzewa T_a o T_b do zapytań UP

Oblicz głębokości wszystkich wierzchołków w T_a i T_b

foreach $(v, d) \in Q$ **do**

    if $depth_a(v) \geq d$ **then**

        wypisz $\text{UP}_a(v, d)$

    else if $depth_b(v) \geq d$ **then**

        wypisz $\text{UP}_b(v, d)$

    else

        wypisz “brak odpowiedzi”

end

Rozwiązanie $O(n)$ – wstęp

Jeśli dopuścimy rozwiązania off-line, to możemy rozwiązać zadania efektywniej (w czasie $O(n)$) i prościej (bez użycia zapytań UP).

Zauważmy, że w czasie obchodzenia drzewa DFSem na stosie rekurencji dysponujemy pełną ścieżką od bieżącego wierzchołka do korzenia. Stąd zapytania typu UP są bardzo proste.

Rozwiązanie $O(n)$

Algorytm 4: Solve

Input: drzewo T oraz lista zapytań $Q = (v_1, d_1), \ldots, (v_m, d_m)$

znajdź średnicę drzewa $v_a \to v_b$

niech $L(v) = \{d : \text{ istnieje zapytanie } (v, d)\}$

foreach $r \in \{v_a, v_b\}$ **do**

  ukorzeń T drzewo w r

  foreach $u \in$ *w porządku DFS* **do**

    niech $(u_0, r, u_1, u_2, \ldots, u_k = u)$ to ścieżka od r do u

    (w DFS możemy utrzymywać takie ścieżki bez dodatkowego narzutu czasowego)

    foreach $d \in L(u)$ **do**

      if $d \leq k$ **then**

        odpowiedzią na zapytanie $query(u, d)$ jest u_{k-d}

      end

    end

  end

end

Dla zapytań na które nie zostały udzielona odpowiedź wypisz -1.

Bibliografia

Problem można rozwiązać w czasie $O(n)$ nawet w przypadku online.

- Johannes Fischer, Volker Heun: Theoretical and Practical Improvements on the RMQ-Problem, with Applications to LCA and LCE. CPM 2006: 36-48
- Michael A. Bender, Martin Farach-Colton: The Level Ancestor Problem simplified. Theor. Comput. Sci. 321(1): 5-12 (2004)

Dziękuję za uwagę!