Spacery po drzewie binarnym (rozwiązanie)

XIV OIJ, próbne zawody II stopnia
29 lutego 2020

Kod zadania: **spa**
Limit czasu: **15 s**
Limit pamięci: **256 MB**

Olimpiada Informatyczna Juniorów

Zadanie pochodzi z Wrocławskich Sparingów Informatycznych (`https://solve.edu.pl/~sparingi/`). Omówienie video znajduje się pod adresem: `https://www.youtube.com/watch?v=jkyKOubQzlA`. Gorąco zachęcamy do udziału w sparingach, które stanowią świetne przygotowanie do Olimpiady Informatycznej Juniorów.

Zadanie polega na odpowiadaniu na zapytania o długość ścieżki między parą węzłów w pełnym drzewie binarnym.

Obserwacje

Numeracja węzłów jest taka, że rodzicem dowolnego węzła, powiedzmy o numerze x, jest węzeł numer $\lfloor \frac{x}{2} \rfloor$. Analogicznie, lewym jego synem jest $2x$, a prawym synem jest $2x+1$. Obserwacja ta pozwala na łatwe nawigowanie po drzewie.

Ponieważ wartości A i B są ograniczone przez 10^{18}, a kolejne poziomy składają się z jednego, dwóch, czterech, ośmiu, 16, 32, $\ldots$ węzłów, łatwo zauważyć, że interesujących nas poziomów w drzewie będzie co najwyżej $\log_2 10^{18} \approx 60$. To oznacza, że wynik dowolnego zapytania ograniczony jest przez 120, a zatem rozwiązanie, które będzie dla każdego zapytania nawigowało po tej ścieżce będzie dostatecznie szybkie.

Rozwiązanie za 25% punktów

Pierwsze podzadanie zawiera jedynie testy, w których A jest przodkiem B w drzewie. Wystarczy więc przesuwać się węzłem B w górę po ścieżce do korzenia drzewa, do momentu, w którym $A = B$. Liczba wykonanych kroków determinuje długość ścieżki.

spa1.py

```python
def dlugosc_sciezki(a, b):
  wynik = 0
  while a != b:
    wynik += 1
    b //= 2
  return wynik


q = int(input())
for _ in range(q):
  a, b = map(int, input().split())
  print(dlugosc_sciezki(a, b))
```

spa1.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int dlugosc_sciezki(long long a, long long b) {
  int wynik = 0;
  while (a != b) {
    wynik++;
    b /= 2;
  }
  return wynik;
}

int main() {
  ios_base::sync_with_stdio(false);
```


© Olimpiada Informatyczna Juniorów, 2020
oij.edu.pl
Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```
  cin.tie(0);

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    long long a, b;
    cin >> a >> b;
    cout << dlugosc_sciezki(a, b) << "\n";
  }

  return 0;
}
```

Rozwiązanie za 50% punktów

W podzadaniu drugim są tylko testy, w których numery węzłów są co najwyżej do miliona. Co więcej, w każdym teście jest tylko jedno zapytanie. Można więc rozważyć algorytm, który będzie działał w czasie proporcjonalnym do liczby rozważanych węzłów w drzewie (a nie długości poszukiwanej ścieżki).

W takim przypadku można uruchomić dowolny ogólny algorytm przeszukiwania grafu, na przykład przeszukiwanie wszerz (BFS). Algorytm ten znajduje ścieżkę między dwoma wierzchołkami dowolnego grafu, przy założeniu, że wszystkie krawędzie mają jednakowy koszt.

Więcej o algorytmie BFS, jego implementacji oraz zastosowaniach można przeczytać na przykład pod adresem: `https://pl.khanacademy.org/computing/computer-science/algorithms#breadth-first-search`

spa2.py

```
from queue import Queue

NIESKONCZONOSC = 1000000001

def bfs(a, b):
  global NIESKONCZONOSC
  odleglosc = [NIESKONCZONOSC] * (max(a,b)+1)

  odleglosc[a] = 0
  kolejka = Queue()
  kolejka.put(a)

  while not kolejka.empty():
    u = kolejka.get()
    for v in [u//2, 2*u, 2*u+1]:
      if (v >= 1) and (v <= max(a, b)) and (odleglosc[v] == NIESKONCZONOSC):
        odleglosc[v] = odleglosc[u] + 1
        kolejka.put(v)

  return odleglosc[b]


q = int(input())
for _ in range(q):
  a, b = map(int, input().split())
  print(bfs(a, b))
```

spa2.cpp

```
#include <bits/stdc++.h>
using namespace std;

```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```cpp
const int MAX_N = 1000005;  // godzimy sie z tym, ze program nie zadziala
                            // dla wiekszych danych
const int NIESKONCZONOSC = 1000000001;

int odleglosc[MAX_N];
queue<int> kolejka;

int bfs(int a, int b) {  // dla max(a,b) do 1 000 000 inty wystarcza
  for (int i = 1; i <= max(a, b); i++)
    odleglosc[i] = NIESKONCZONOSC;

  odleglosc[a] = 0;
  kolejka.push(a);

  while (!kolejka.empty()) {
    int u = kolejka.front();
    kolejka.pop();
    for (int v : vector<int>({u/2, 2*u, 2*u+1}))  // potencjalni sasiedzi wierzcholka u
      if ((v >= 1) && (v <= max(a, b)) && (odleglosc[v] == NIESKONCZONOSC)) {
        odleglosc[v] = odleglosc[u] + 1;
        kolejka.push(v);
      }
  }

  return odleglosc[b];
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    int a, b;
    cin >> a >> b;
    cout << bfs(a, b) << "\n";
  }

  return 0;
}
```

Rozwiązanie wzorcowe

Zauważmy, że każda ścieżka między dwoma węzłami w drzewie może być przedstawiona jako suma fragmentu ścieżki z węzła pierwszego do korzenia oraz fragmentu ścieżki z węzła drugiego do korzenia. Pozostaje jednak ustalić najniżej położony wspólny węzeł tych ścieżek.

Pomysł na rozwiązanie jest następujący: ustalamy, który z węzłów A lub B jest położony niżej i to tym węzłem przesuwamy się w górę drzewa, do jego rodzica. Jeśli A i B są na tym samym poziomie, jednak dalej są różne, możemy poruszyć się dowolnym z nich.

Jak ustalić, który węzeł jest wyżej? Można policzyć jego odległość od korzenia (systematycznie dzieląc numer węzła przez 2 i licząc wykonane kroki), ale nie jest to konieczne: wystarczy przesuwać się w górę zawsze od węzła o numerze większym, węzły drzewa były przecież numerowane poziomami.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

spa3.py

```python
def dlugosc_sciezki(a, b):
  wynik = 0
  while a != b:
    wynik += 1
    if a > b:
      a //= 2
    else:
      b //= 2
  return wynik


q = int(input())
for _ in range(q):
  a, b = map(int, input().split())
  print(dlugosc_sciezki(a, b))
```

spa3.cpp

```cpp
#include <bits/stdc++.h>
using namespace std;

int dlugosc_sciezki(long long a, long long b) {
  int wynik = 0;
  while (a != b) {
    wynik++;
    if (a > b)
      a /= 2;
    else
      b /= 2;
  }
  return wynik;
}

int main() {
  ios_base::sync_with_stdio(false);
  cin.tie(0);

  int q;
  cin >> q;
  for (int i = 0; i < q; i++) {
    long long a, b;
    cin >> a >> b;
    cout << dlugosc_sciezki(a, b) << "\n";
  }

  return 0;
}
```

Dalsze rozważania

W zadaniu mieliśmy do czynienia z tak zwanymi pełnymi drzewami binarnymi, tzn. takimi, że każdy węzeł wewnętrzny ma dwoje dzieci. Dzięki temu można było ograniczyć wysokość drzewa przez logarytm z liczby węzłów. Równie sensownym jednak pytaniem byłoby rozwiązanie zadania dla dowolnych drzew, niekoniecznie tak zbalansowanych. Jedynym problemem w rozwiązaniu zadania w ogólnej wersji jest konieczność szybkiego (szybszego niż wysokość drzewa, która w najgorszym przypadku może być nawet taka jak liczba węzłów) znajdowania najniższego przodka.

Taki algorytm istnieje i był przedmiotem wykładu na dwudziestym Wrocławskim Sparingu Informatycznym: `https://www.youtube.com/watch?v=W_0CQOepEZQ`

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".