Rozwiązanie zadania F 754.
Spadek napięcia na drugim oporniku to $U = IR_2 = IR_{02} - aUI$, stąd

$$U = \frac{IR_{02}}{1 + aI}.$$

Z prawa Ohma mamy

$$U_0 = IR_1 + U = IR_1 + \frac{IR_{02}}{1 + aI},$$

otrzymujemy więc

$$I = \frac{1}{2aR_1}\left(aU_0 - R_1 - R_{02} + \right. \left. + \sqrt{(R_1 + R_{02} - aU_0)^2 + 4aR_1U_0}\right).$$

Informatyczny kącik olimpijski (25): Odgadywanie funkcji

Tematem kącika w tym miesiącu będzie zadanie *Wires and Switches* z Międzynarodowej Olimpiady Informatycznej z 1995 roku.

Dana jest liczba naturalna n. Należy odgadnąć funkcję $f : \{0, 1, \ldots, n-1\} \to$ $\to \{0, 1, \ldots, n-1\}$ (tzn. jej wartości dla wszystkich argumentów), która jest reprezentowana przez zewnętrzną bibliotekę komunikującą się z programem stanowiącym rozwiązanie. Biblioteka umożliwia jedynie wykonywanie następujących operacji:

- zapytanie, czy $f(x) \in B$ dla pewnego x i aktualnego zbioru B,
- dodanie do zbioru B jednego elementu,
- usunięcie ze zbioru B jednego elementu.

Zbiór B jest początkowo pusty. Celem jest uzyskanie jak najmniejszej łącznej liczby wykonanych operacji. Konkretniej, na zawodach było wiadomo, że $n \leqslant 90$, i należało zmieścić się w 900 operacjach.

Najprostszym pomysłem jest rozdzielenie problemu na n mniejszych: szukanie oddzielnie wartości $f(x)$ dla kolejnych x. Dla ustalonego x można to zrobić, na przykład, wyszukując binarnie. W tym celu sprawdzamy, czy $f(x) \in \{0, 1, \ldots, y\}$ dla $y = \lfloor \frac{n}{2} \rfloor$, następnie zależnie od wyniku dla $y = \lceil \frac{n}{4} \rceil$ lub $y = \lfloor \frac{3n}{4} \rfloor$ itd. Oczywiście, wykonamy około $\log n$ zapytań, ale równolegle do nich trzeba jeszcze zmieniać zbiór B, co zajmie, dla danego x, około n operacji. Łącznie zapytań do biblioteki wychodzi mniej więcej $n \cdot (\log n + n)$, a konkretnie dla $n = 90$ – prawie 9000, czyli zdecydowanie za dużo.

O wiele lepszym rozwiązaniem jest połączenie wszystkich wyszukiwań binarnych w jedno, żeby nie musieć nieustannie zmieniać zbioru B (co jest głównym kosztem pierwszego algorytmu). Sprawdzenie, dla których wartości x mamy $f(x) \in \{0, 1, \ldots, \frac{n}{2}\}$, zajmuje około $\frac{3n}{2}$ operacji ($\frac{n}{2}$ razy dokładamy element do zbioru B, po czym wykonujemy n zapytań). Pozwala to podzielić dziedzinę na dwa zbiory C i D, z których każdy odpowiada fragmentowi przeciwdziedziny rozmiaru mniej więcej $\frac{n}{2}$. W kolejnym kroku z aktualnego zbioru B usuwamy elementy od $\frac{n}{4}$ do $\frac{n}{2}$, a dokładamy do niego elementy od $\frac{n}{2}$ do $\frac{3n}{4}$ (razem $\frac{n}{2}$ operacji). Wówczas wykonanie zapytań dla każdego elementu z C i D pozwala na rozdrobnienie tych zbiorów na odpowiadające podprzedziałom przeciwdziedziny rozmiaru $\frac{n}{4}$ (dlaczego?). W tej fazie znów wykonaliśmy $\frac{3n}{2}$ operacji. Odpowiednio modyfikując zbiór B, to rozdrabnianie można kontynuować – po $\log n$ takich fazach otrzymany podział dziedziny wyznacza dokładnie wszystkie wartości funkcji. Daje to razem $\frac{3}{2} n \log n$ operacji, przy czym cały czas pomijałem fakt, że $\frac{n}{2}$ i $\log n$ mogą nie być całkowite, co może prowadzić do drobnych błędów w tym oszacowaniu. Dla $n = 90$ otrzymujemy łączną liczbę operacji około 900; z powodu opisanych niedokładności ciężko ją jednak obliczyć dokładnie. Na zawodach takie rozwiązanie wymagało dodatkowych usprawnień, aby uzyskać maksymalną liczbę punktów.

Rozwiązanie proponowane przez autora kącika opiera się na programowaniu dynamicznym i jest pewnym ulepszeniem poprzedniego. Będziemy w nim wyznaczać dla każdej pary (x, y) liczbę $t(x, y)$ operacji potrzebnych nam do odgadnięcia funkcji, jeśli jej dziedzina liczy x elementów, a przeciwdziedzina – y elementów, przy założeniu, że początkowo zbiór B nie zawiera elementów z przeciwdziedziny lub zawiera ją całą (może natomiast zawierać dowolne elementy spoza niej). Algorytm zadawania zapytań do biblioteki możemy teraz oprzeć chociażby na następującym, bezpośrednim pomyśle: aby odgadnąć funkcję, dodajmy kilka (a konkretnie q) elementów do zbioru B (a w przypadku, gdy B zawierał całą przeciwdziedzinę – usuńmy z niego q elementów), a następnie sprawdźmy, dla których argumentów funkcja daje wartości z B (niech będzie ich p). W ten sposób podzielimy zarówno dziedzinę, jak i przeciwdziedzinę, na dwa podzbiory i będziemy mogli rozwiązywać dwa otrzymane podproblemy osobno. Zauważmy, że mamy wpływ na wybór q, ale nie mamy żadnego na p. W takim razie, zakładając, że wystąpi najgorsza ewentualność (pod względem wyboru p), możemy przyjąć, że:

$$t(x, y) = \min_{0<q<y} \max_{0 \leqslant p \leqslant x} (q + x + t(p, q) + t(x - p, y - q)),$$

gdyż q operacji zużywamy na dodawanie elementów do B (lub ich usuwanie), następnie x operacji na dzielenie dziedziny, a potem rozwiązujemy oba pozostałe podproblemy. Zauważmy, że przy takim algorytmie zapytań dla każdego z podproblemów w momencie rozpoczęcia przetwarzania zbiór B, zgodnie z poczynionym założeniem, zawiera wszystkie elementy z danej przeciwdziedziny lub nie zawiera żadnego z nich.

Powyższy wzór pozwala obliczyć wszystkie wartości $t(x, y)$ dla $x, y \leqslant n$. Po zakodowaniu rozwiązania okazuje się, że $t(90, 90) = 877$, co pozwala z zapasem odgadnąć każdą funkcję, mieszcząc się w wyznaczonym limicie. Pytanie kontrolne: jak na podstawie wartości $t(x, y)$ zaprojektować konkretny algorytm zadawania zapytań do biblioteki?

Na koniec zachęcam Czytelnika do zastanowienia się, czy strategia, zgodnie z którą obliczaliśmy wartości $t(x, y)$, jest optymalna, a jeśli nie, to czy potrafi on znaleźć taką, która wymaga mniejszej liczby operacji (w ogólności lub chociaż dla pewnych n).

Tomasz KULCZYŃSKI