Jak przepłynąć labirynt?

Najpierw należy napełnić labirynt wodą, a potem wskoczyć do niego. Po wodzie rozejdą się kręgi. Pierwszy, który dotrze do punktu docelowego, nie tylko pokaże, że można tam dopłynąć, ale na dodatek przebędzie najkrótszą z możliwych dróg. Ten prosty eksperyment, bardziej intuicyjny niż fizyczny (brak przecież w jego opisie wielu założeń co do fizycznych własności labiryntu) łatwo przekłada się na precyzyjny język liczb i algorytmów.

Narysujmy labirynt na kartce kratkowanego papieru. Obierzmy w nim punkt startowy i docelowy. Ponumerujmy kolejnymi liczbami naturalnymi kręgi, rozchodzące się wokół startu. Pamiętajmy, że ścianki korytarzy labiryntu całkowicie pochłaniają dochodzące do nich kręgi, a w każdym korytarzu mogą się one rozchodzić tylko jeden raz.

Jeżeli zapełnimy w ten sposób labirynt i nie dojdziemy do celu, to poszukiwanej drogi nie ma. W przeciwnym przypadku — gdy dotrzemy wreszcie do miejsca docelowego — droga istnieje, a znaleźć ją łatwo cofając się tak, by przeskakiwać z jednego kręgu na najbliższy, o numerze o jeden niższym.

Opisany tu algorytm nie na wiele zdałby się Tezeuszowi — nie miał on przecież planu labiryntu kreteńskiego. Wykorzystywany jest za to przez twórców programów komputerowych przeznaczonych do projektowania połączeń między elementami układów elektronicznych.

Na koniec pytanie dla dociekliwych:
W jaki inny sposób można numerować kręgi w labiryncie, by odnaleźć drogę do celu? Jak użyć do tego najmniejszej liczby różnych liczb?

K. B.

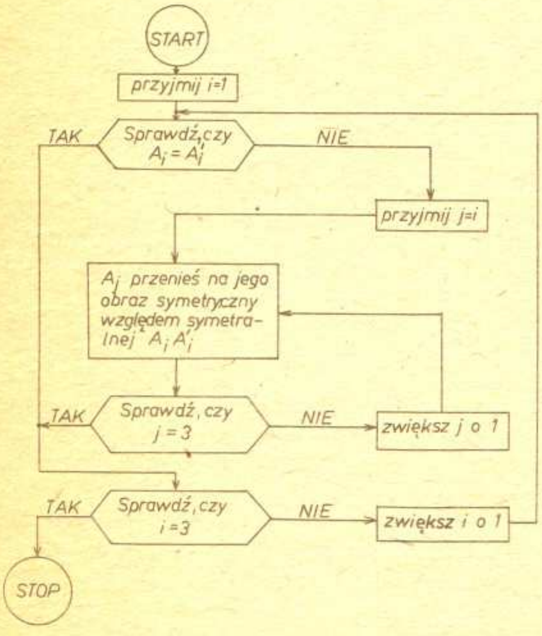
Symetrie

Poniższy algorytm stanowi konstruktywny dowód twierdzenia:

Dowolne dwa przystające trójkąty na płaszczyźnie można nałożyć za pomocą nie więcej niż trzech symetrii osiowych.

Jeśli znajdziemy się w punkcie START z dwoma przystającymi trójkątami $A_1 A_2 A_3$ i $A'_1 A'_2 A'_3$, to w punkcie STOP będziemy mieli $A_i = A'_i$ dla $i = 1, 2, 3$.

Czytelnik zechce sprawdzić, że tak jest istotnie, a także uzasadnić, że jest to dowód przytoczonego twierdzenia.



Wieże Hanoi

Gdy w pępku świata, indyjskim mieście Benares, stawiał Brahma na brązowej tabliczce trzy diamentowe pałeczki o wysokości jednego łokcia i o grubości tułowia pszczoły, wiedział już zapewne w swej niezmierzonej mądrości, że nim mnisi przełożą 64 złote krążki i nastąpi koniec świata, ludzie będą podawać grę w „Wieże Hanoi" jako koronny przykład algorytmu rekursywnego.

Algorytm — to składający się ze skończonego ciągu elementarnych poleceń przepis na wykonanie pewnej bardziej złożonej czynności. Jedną z własności większości algorytmów jest to, że do działania niezbędny jest im zestaw produktów wejściowych, które my dalej będziemy nazywali parametrami. Jak z opisu tego widać, cały algorytm jest też pewnym poleceniem. Nic zatem nie stoi na przeszkodzie, aby go użyć jako elementu innego, bardziej złożonego polecenia. Stąd już tylko jeden krok do algorytmu rekursywnego, czyli takiego, który jako jednego z elementów używa samego siebie.

Przejdźmy do algorytmu gry w „Wieże Hanoi". Przypomnijmy w skrócie zasady gry:
1. rekwizytami są trzy pałeczki (nazwijmy je A, B i C), na które nanizano krążki o parami różnych średnicach,
2. w stanie początkowym na pałeczce A znajduje się n krążków ułożonych tak, że każdy (oprócz najniższego) leży na krążku o większej średnicy,
3. zadanie polega na przeniesieniu wszystkich krążków, jednego po drugim, z pałeczki A na pałeczkę C z wykorzystaniem pałeczki B. Należy przy tym przestrzegać zasady, że krążek można przełożyć albo na pałeczkę pustą, albo na inny krążek, byle o większej średnicy.

Przystąpmy zatem do budowy algorytmu. Nadajmy mu nazwę i określmy jego parametry. Będziemy zapisywali:

Hanoi (*n*, *skąd*, *poprzez*, *dokąd*)

i odczytywali:

przenieś, według zasad gry, *n* krążków z pałeczki o nazwie *skąd* na pałeczkę o nazwie *dokąd*, wykorzystując pałeczkę *poprzez*.

W naszym algorytmie użyjemy tylko jednego elementarnego polecenia. Będziemy zapisywali:

przenieś (*skąd*, *dokąd*)

i odczytywali:

przenieś jeden krążek z pałeczki o nazwie *skąd* na pałeczkę o nazwie *dokąd*.

Algorytm *Hanoi* (*n*, *skąd*, *poprzez*, *dokąd*) ma działać dopóty, dopóki na pałeczce początkowej jest jeszcze choć jeden krążek. Aby grę zakończyć, wystarczy wykonać kolejno tylko trzy polecenia. Zapiszmy je od razu w postaci gotowego algorytmu używając wprowadzonych poprzednio oznaczeń.

Algorytm *Hanoi*
parametry: *n* liczba przenoszonych krążków
skąd nazwa pałeczki początkowej
poprzez nazwa pałeczki pośredniej
dokąd nazwa pałeczki końcowej

1. jeżeli $n > 0$:
   a. *Hanoi* ($n-1$, *skąd*, *dokąd*, *poprzez*)
   b. *przenieś* (*skąd*, *dokąd*)
   c. *Hanoi* ($n-1$, *poprzez*, *skąd*, *dokąd*)
   d. koniec
2. jeżeli $n = 0$: koniec.

Zauważcie, jak prosty jest ten algorytm. Jako jedynego rzeczywistego działania używa on polecenia *przenieś*. Pozostałe kroki służą jedynie do organizowania rekursji i modyfikacji parametrów.

Na zakończenie słów kilka o końcu świata: kiedy on nastąpi? Jeżeli przez $P(n)$ oznaczymy liczbę elementarnych przeniesień dla gry z n krążkami, to ze struktury algorytmu widać, że:

$$P(n) = \begin{cases} 0 & \text{dla } n = 0, \\ 2P(n-1)+1 & \text{dla } n > 0. \end{cases}$$

Radzimy dociekliwemu Czytelnikowi wykazać, że $P(n) = 2^n - 1$. My zaś pokażemy, że koniec świata nie może nastąpić wcześniej niż po $2^n - 1$ (gdzie $n = 64$) przełożeniach krążka. Oznaczmy przez $p(n)$ minimalną możliwą liczbę przełożeń dla n krążków. W momencie gry, w którym należy przenieść największy krążek z pałeczki początkowej na końcową, na pałeczce pośredniej musi być ułożona piramida z $n-1$ krążków. By stan ten uzyskać, trzeba wykonać zatem co najmniej $p(n-1)$ przełożeń. Teraz należy przełożyć krążek największy i znowu wykonać $p(n-1)$ ruchów. Stąd:

$$P(n) \geqslant 2p(n-1)+1 \geqslant 2^n - 1 \text{ (gdyż } p(0) = 0).$$

Przy 64 krążkach mamy zatem jeszcze sporo czasu, nim mnisi wykonają ostatnie z prawie 10^{20} przemieszczeń krążków.

K. B.