motywy Logo

Okno programowania

Andrzej WALAT

Programowanie jest czasem postrzegane jako czynność czysto techniczna. Jeśli rozwiązanie jakiegoś zadania wymaga żmudnych obliczeń, to warto napisać stosowny program, żeby komputer wykonał je za nas. Ale, zdaniem autorów ważnego raportu ACM (patrz [1]), programowanie jest „oknem na świat” – patrząc przez nie, można lepiej poznać i zrozumieć rzeczywistość i różne ważne problemy nie tylko programistyczne; jest ważną aktywnością poznawczą, dzięki której stajemy się mądrzejsi i zdolniejsi. Tezę tę zilustruję klasycznym przykładem zadania programistycznego o wyszukiwaniu maksymalnego rosnącego (albo malejącego) podciągu w danym ciągu liczb. Jest to zadanie interesujące i ważne z dwóch powodów. Po pierwsze, występuje w wielu sytuacjach praktycznych. Po drugie, jest to stosunkowo elementarny przykład, na którym można się nauczyć ważnej metody algorytmicznej – *programowania dynamicznego*.

Dla jasności: podciąg to ciąg uzyskany z danego ciągu przez skreślenie pewnej liczby (niekoniecznie kolejnych) wyrazów.

Gdy rozwiązujemy ten problem w środowisku Logo, to sprowadza się on do zdefiniowania funkcji, nazwijmy ją `mrpc`, która dla dowolnego ciągu liczb, danego w postaci listy, znajduje jego rosnący podciąg o maksymalnej długości. Na przykład dla danego ciągu `a = [4 6 8 0 4 5 2 6 2]` poprawnym wynikiem `mrpc` może być `[0 4 5 6]`. Istotnym elementem dynamicznego rozwiązania tego problemu jest zbudowanie (od końca) listy maksymalnych długości rosnących podciągów zaczynających się odpowiednio od pierwszego, drugiego, i-tego wyrazu danego ciągu `a`. Dla naszego przykładowego ciągu `a` odpowiednią listą maksymalnych długości rosnących podciągów `a` jest `mdrp = [3 2 1 4 3 2 2 1 1]`. Zdefiniowanie funkcji `mdrp :a`, która dla dowolnego a znajduje odpowiednią sekwencję maksymalnych długości rosnących podciągów, pozostawiam ambitnym Czytelnikom (rozwiązanie w aneksie na stronie WWW *Delty*). W tym miejscu zdefiniuję i objaśnię funkcję `mrp`, która znajduje maksymalny rosnący podciąg danego ciągu `a`, jeśli ma dany również odpowiedni ciąg `mdrp`.

```
oto mrp :a :mdrp
wynik mpr (lista pierw :a) pierw :mdrp bp :a bp :mdrp
już
```

Treść definicji tej funkcji składa się tylko z jednego, ale trzeba przyznać, wyglądającego trochę kryptograficznie polecenia. Wywołujemy w nim kolejną pomocniczą funkcję `mpr` (maksymalny podciąg rosnący), którą można zdefiniować następująco:

```
oto mpr :pmp :maxdrp :oa :omdrp
jeśli puste? :oa [wynik :pmp]
jeśli pierw :omdrp > :maxdrp
    [wynik mpr (lista pierw :oa) pierw : omdrp bp :oa bp :omdrp]
jeśli i (pierw :oa > ost :pmp)
        ((pierw :omdrp) + (długość :pmp) = :maxdrp)
    [wynik mpr nak pierw :oa :pmp :maxdrp bp :oa bp :omdrp]
wynik mpr :pmp :maxdrp bp :oa bp :omdrp
już
```

Treść tej definicji jest już trochę bardziej złożona. Zacznę od objaśnienia użytych skrótów.

`pmp` – oznacza początek maksymalnego (rosnącego) podciągu `a`,

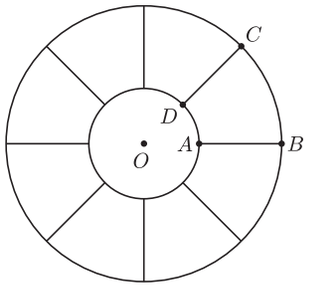
`maxdrp` – oznacza maksymalną długość rosnącego podciągu `a`,

`oa` – oznacza „ogon” ciągu `a`,

`omdrp` – oznacza „ogon” ciągu `mdrp`,

Załóżmy, że (w poszukiwaniu maksymalnego rosnącego podciągu) przejrzeliśmy już początkowe trzywyrazowe odcinki danego ciągu `a = [4 6 8 0 4 5 2 6 2]` oraz odpowiedniego `mdrp = [3 2 1 4 3 2 2 1 1]`. W tym momencie możemy sądzić, że

Rozwiązanie zadania M 1211.
Dane koło o środku O dzielimy na dziewięć obszarów, jak pokazano na rysunku (mniejszy okrąg ma środek O i średnicę 4).



Średnica każdego z uzyskanych obszarów nie przekracza 4. Istotnie: średnicą obszaru $ABCD$ jest odcinek AC lub BC. Wtedy jednak

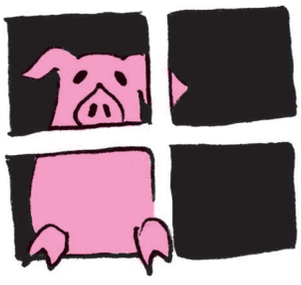
$$BC^2 = 5^2 + 5^2 - 50\cos 45^\circ = \\ = 50 - 25\sqrt{2} < 16$$

oraz

$$AC^2 = 2^2 + 5^2 - 20\cos 45^\circ = \\ = 29 - 10\sqrt{2} < 16\,.$$

Jeśli wszystkie dane punkty leżą wewnątrz mniejszego koła, to teza zadania jest spełniona. Wykonując obrót wokół punktu O, możemy więc przyjąć, że pewien z danych punktów leży na odcinku AB. Jeśli któryś z pozostałych punktów leży w obszarze, którego brzeg zawiera odcinek AB, to teza zadania jest spełniona. W przeciwnym razie któreś dwa spośród ośmiu punktów leżą w jednym z pozostałych siedmiu obszarów. To zaś oznacza, że ich odległość nie przekracza 4.

maksymalna długość rosnącego podciągu `a` `maxdrp = 3`, oraz że początkiem odpowiedniego maksymalnego rosnącego podciągu jest `pmp = [4 6 8]`, ale pozostał jeszcze do przejrzenia niepusty ogon `a`, tj. `[0 4 5 2 6 2]` oraz ogon `mdrp`, tj. `[4 3 2 2 1 1]`. Po obejrzeniu pierwszych wyrazów obu ogonów musimy zmienić wartości: `pmp`, `maxdrp`, `oa` oraz `omdrp`. Teraz powinno być: `pmp = [0]`, `maxdrp = 4`, `oa = [4 5 2 6 2]` oraz `omdrp= [3 2 2 1 1]`. Kiedy ogon `a` skurczy się do zera, to wartość `pmp` będzie ostatecznym wynikiem, tj. poszukiwanym maksymalnym rosnącym podciągiem danego ciągu. Zjadając ogony, musimy rozważać (w każdym kroku) cztery różne przypadki, ale nie będę ich szczegółowo objaśniał, pozostawiając przyjemność analizy treści procedury `mpr` dociekliwym Czytelnikom. Główną funkcję rozwiązania można zdefiniować następująco.



```
oto mrpc :a
wynik mrp :a mdrp :a
już
```

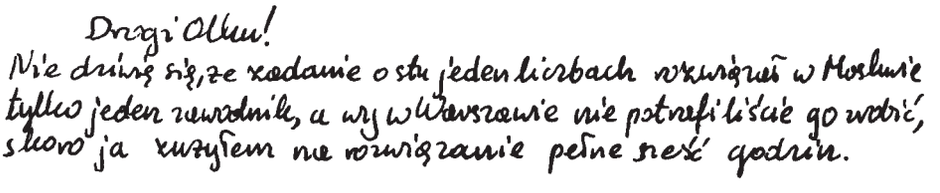
Ale pamiętajmy, że aby jej użyć, trzeba odrobić pracę domową i zdefiniować `mdrp`. Po napisaniu polecenia `pokaż mrpc [4 6 8 0 4 5 2 6 2]` komputer wypisze: `[0 4 5 6]`.

Co widać przez to okno?

Teraz powrócę do tezy ze wstępu. Czy doświadczenie programowania może nam rzeczywiście pomóc w rozwiązywaniu jakichś innych problemów, nie tylko informatycznych? Wiele lat temu w finale XIII Moskiewskiej Olimpiady Matematycznej pojawiło się następujące zadanie (patrz [3] str. 28):

„Liczby od 1 do 101 wypisano w dowolnym porządku. Udowodnić, że można z tych 101 liczb wykreślić 90 tak, aby pozostałych 11 tworzyło ciąg monotoniczny, tzn. albo ciąg malejący, albo rosnący.

Zyskało ono pewną sławę; Aleksander Pełczyński poświęcił mu interesujący artykuł [2] w historycznym już 5. numerze *Delty*. Artykuł jest barwną opowieścią o tym, jak autor, wówczas student pierwszego roku Wydziału Matematyki UW, wraz z grupą kolegów bezskutecznie poszukiwali pomysłu na rozwiązanie zadania o stu jeden liczbach. W końcu poszukali pomocy u wybitnych autorytetów: profesora Stefana Kulczyckiego oraz „zdolnego kolegi z Poznania". Bardzo zabawna odpowiedź nadeszła po kilku dniach z Poznania:


Drogi Olku!
Nie dziwię się, że zadanie o stu jeden liczbach rozwiązań w Moskwie tylko jeden zawodnik, a my w Warszawie nie potrafiliśmy go wtedy, skoro ja zużyłem na rozwiązanie pełne sześć godzin.

Ten „dowód skromności" autora był poparty pięknym, ale bardzo skomplikowanym rozwiązaniem. Podobne rozwiązanie podał profesor Kulczycki. A teraz pomyślmy, jak mógłby rozwiązać zadanie o stu jeden liczbach zdolny gimnazjalista, który ma trochę doświadczenia w rozwiązywaniu problemów algorytmicznych.

Załóżmy, że dana jest dowolna permutacja `a` liczb od 1 do 101. Budujemy dla niej odpowiednią listę `mdrp`. Jeśli na tej liście występuje liczba 11, to z danej permutacji `a` można wybrać 11-wyrazowy podciąg rosnący. W przeciwnym przypadku, jeśli wszystkie wyrazy na liście `mdrp` są mniejsze niż 11, jakaś liczba `i < 11` musi wystąpić na liście `mdrp` przynajmniej 11 razy. Załóżmy, że występuje ona na pozycjach: `p1, p2, p3 ...`, wtedy wyrazy ciągu `a` na pozycjach `p1, p2, p3 ...` tworzą co najmniej 11-wyrazowy podciąg malejący.

Wymyślenie takiego prostego rozwiązania zajęłoby zdolnemu gimnazjaliście zapewne nie więcej niż sześć minut. Proponuję porównać je z innymi rozwiązaniami przedstawionymi w artykule Aleksandra Pełczyńskiego i w zbiorze problemów z Moskiewskich Olimpiad Matematycznych [3].

Literatura

[1] ACM (2003). *A Model Curriculum for K-12 Computer Science*. Final Report of the ACM K-12 Task Force Curriculum Committee. CSTA. http://www.acm.org/education/curris_vols/k12final1022.pdf.

[2] Aleksander Pełczyński (1974). *Trzy rozwiązania zadania o 101 liczbach*, Delta Nr 5, 1974.

[3] Shklyarsky D. O., Chentsov N. N., Yaglom I.M. (1979). *Selected Problems and Theorems in Elementary Mathematics*. Mir Publishers Moscow.