Najprostszy najszybszy

W niektórych zadaniach rozpoznawania mowy i rozpoznawania obrazów występuje następujący problem:

Dany jest ciąg liczb rzeczywistych $x_1, \ldots, x_n$. Szukamy takiego podciągu $x_j, x_{j+1}, \ldots, x_{k-1}, x_k$, aby suma $\sum_{i=j}^{k} x_i$ była jak największa. Podciąg taki będziemy dalej nazywać *najlepszym*. Potrzebny jest algorytm obliczania sumy najlepszego ciągu.

Przypuśćmy, że ciąg wygląda następująco:

i	1	2	3	4	5	6
x_i	−1	1	−2	3	2	−4

Najlepszym podciągiem jest podciąg 3 2 ($j = 4$, $k = 5$), jego sumą jest 5.

Jeśli wszystkie wyrazy ciągu są ujemne, to za najlepszy uznamy podciąg pusty ($j = 0$, $k = 0$), o sumie zerowej.

Na pierwszy rzut oka wydaje się, że najprościej byłoby obliczyć sumy wszystkich możliwych podciągów i wybrać największą z nich:

```
  Największa Suma: = 0
  dla j od 1 do n wykonaj
  ┌ dla k od j do n wykonaj
  │ ┌ Suma: = 0
* │ │ dla i od j do k wykonaj
  │ │ Suma : = Suma+x_i
  │ │ jeśli Suma > Największa Suma
  └ └ to Największa Suma: = Suma
```

Obliczmy, ile dodawań wykona ten algorytm. Dla pary (j, k) trzeba ich wykonać $k-j+1$, a więc liczba wszystkich dodawań wyraża się przez

$$\sum_{j=1}^{n}\sum_{k=j}^{n}(k-j+1) = \sum_{j=1}^{n}\sum_{i=1}^{n-j+1} i = \sum_{j=1}^{n}\frac{(n-j+2)(n-j+1)}{2} =$$

$$= \frac{1}{2}\sum_{l=1}^{n} l(l+1) = \frac{1}{2}\left(\frac{1}{6}\cdot n\cdot(n+1)\cdot(2n+1)+\frac{1}{2}\cdot n\cdot(n+1)\right) =$$

$$= \frac{1}{6}\cdot n\cdot(n+1)(n+2) = \frac{1}{6}n^3+\frac{1}{2}n^2+\frac{1}{3}n,$$

a zatem liczba operacji jest rzędu n^3.

Zastanówmy się, jak można poprawić ten algorytm. Zwróćmy uwagę, że pętla oznaczona gwiazdką oblicza kolejno sumy

$S_{j,j} = x_j$

$S_{j,j+1} = x_j + x_{j+1}$

$S_{j,n} = x_j + x_{j+1} + \ldots + x_n$

spełniające zależność $S_{j,i+1} = S_{j,i} + x_{i+1}$. Można więc zapisać tę pętlę następująco

```
┌ Suma: = 0
│ dla k od j do n wykonaj
│ ┌ Suma: = Suma+x_k
│ │ jeśli Suma > Największa Suma
└ └ to Największa Suma: = Suma
```

Po zastosowaniu tej modyfikacji algorytm wykonuje

$$\sum_{j=1}^{n}(n-j+1) = \sum_{l=1}^{n} l = \frac{1}{2}n^2+\frac{1}{2}n,$$

czyli rzędu n^2 dodawań.

Jeszcze lepszy algorytm można otrzymać stosując metodę „dziel i rządź". Przypuśćmy, że interesuje nas na razie podciąg $x_a, x_{a+1}, \ldots, x_b$. Podzielmy go na dwie w miarę równe części $x_a, x_{a+1}, \ldots, x_c$ oraz $x_{c+1}, x_{c+2}, \ldots, x_b$. Możliwe są trzy różne położenia najlepszego podciągu:

1) $a \leqslant j \leqslant k \leqslant c$

2) $a \leqslant j \leqslant c < k \leqslant b$

3) $c < j \leqslant k \leqslant b$

Można więc znaleźć maksymalne sumy dla ciągów należących do tych trzech grup, a następnie wybrać największą z nich.

Najlepszy ciąg należący do grupy drugiej możemy znaleźć stosując pętlę analogiczną do pętli *, kosztem $b-a+1$ sumowań. Najlepsze ciągi z grupy pierwszej i trzeciej znajdziemy stosując ten sam algorytm do podciągów $x_a, \ldots, x_c$ oraz $x_{c+1}, \ldots, x_b$. Jeśli oznaczyć koszt znalezienia najlepszego ciągu wśród n elementów przez T_n, to zachodzi równość

$$T_n = n + 2\cdot T_{n/2}$$

mająca rozwiązanie $T_n = n\log_2 n$. Pełny zapis algorytmu pozostawiamy Czytelnikowi.

Okazuje się jednak, że istnieje algorytm jeszcze lepszy i — co dziwne — jeszcze prostszy. Korzysta on z faktu, że najlepszy ciąg musi się zaczynać i kończyć liczbą dodatnią (w przeciwnym razie usunięcie skrajnego elementu zwiększyłoby sumę ciągu). Skoro tak, to zacznijmy analizę od pierwszego dodatniego elementu, x_a. Rozważmy najdłuższy podciąg $x_a, x_{a+1}, \ldots, x_b$ taki, że dla każdego c $(a \leqslant c \leqslant b)$ $\sum_{i=a}^{c} x_i > 0$. Zastanówmy się, jak ma się ten podciąg do najlepszego ciągu $x_j, \ldots, x_k$. Otóż albo $a = j \leqslant k \leqslant b$, albo $b < j \leqslant k \leqslant n$. Gdyby bowiem $a < j \leqslant b$, to dołączenie elementów $x_a, \ldots, x_{j-1}$ polepszyłoby ów ciąg. Z drugiej strony, gdyby $a = j < b < k$, to opuszczenie elementów $x_a, \ldots, x_{b+1}$ (o sumie nie większej od zera) co najmniej nie pogorszy ciągu.

Poszukajmy więc sum najlepszych ciągów, dla których $a = j \leqslant k \leqslant b$ oraz $b < j \leqslant k \leqslant n$ i porównajmy je. W pierwszym przypadku wymaga to $b-a+1$ dodawań, natomiast w drugim — powtórzenia opisanego rozumowania dla reszty wyjściowego ciągu, poczynając od elementu $b+1$. Pełny algorytm wygląda następująco

```
Największa Suma: = 0
Suma Częściowa: = 0
dla i od 1 do n wykonaj
┌ Suma Częściowa: = Suma Częściowa+x_i
│ jeśli Suma Częściowa < 0 to Suma Częściowa: = 0
│ jeśli Suma Częściowa > Największa Suma
└ to Największa Suma : = Suma Częściowa
```

Algorytm ten wykonuje dokładnie n dodawań!

Oszacowano czas pracy czterech podanych algorytmów na komputerze VAX 11/750 dla różnych długości ciągu:

n	*Algorytm* n^3	*Algorytm* n^2	*Algorytm* $n \log n$	*Algorytm* n
10^2	3,4 s	130 ms	30 ms	3,3 ms
10^4	39 dni	22 minuty	6,1 s	0,33 s
10^6	108 tys. lat	5 miesięcy	15 minut	33 s

(ms = milisekunda = 1/1000 s).

Jeszcze wymowniejsze jest porównanie czasu pracy algorytmu pierwszego, zaprogramowanego na superkomputerze CRAY-1, i algorytmu czwartego, zaprogramowanego na mikrokomputerze TRS-80 (Meritum 1).

n	CRAY-1	TRS-80
10	0,003 ms	200 ms
1000	3,0 s	20 s
2500	50 s	50 s
10 000	49 minut	3,2 minuty
1 000 000	95 lat	5,4 godziny

Na zakończenie dodajmy, że nie jest znany tani algorytm dla analogicznego problemu dwuwymiarowego (znalezienie najlepszego prostokąta w tablicy $n \times n$). Prymitywny algorytm ma złożoność rzędu n^6, a więc zupełnie nie do zaakceptowania. A może komuś z Was coś przyjdzie do głowy?

J.D.

Rozwiązanie zadania M 416. Wśród 10^n+1 kolejnych potęg trójki: $3, 3^2, 3^3, \ldots, 3^{10^n+1}$ istnieją dwie 3^k, 3^m ($m > k$) dające tę samą resztę przy dzieleniu przez 10^n. Liczba $3^m - 3^k = 3^k(3^{m-k}-1)$ dzieli się przez 10^n, a ponieważ 3^k i 10^n są względnie pierwsze, to $3^{m-k}-1$ dzieli się przez 10^n, czyli $3^{m-k} = l \cdot 10^n + 1$ dla pewnego naturalnego l. Ostatnie n cyfr 3^{m-k} to $n-1$ zer i jedynka. W rozwinięciu dziesiętnym 0,392781243729... występują więc dowolnie długie ciągi złożone z samych zer, nie może więc wystąpić okres, czyli omawiana liczba jest niewymierna.

Czytelnicy proponują

Nasz Czytelnik, pan Tadeusz Foszcz z Ilmenau w NRD, przysłał list z następującym zadaniem:

Żarówki Z_1 i Z_2 połączono szeregowo i włączono do źródła prądu. Obie żarówki pracują pod napięciem znamionowym. Jakie jest prawdopodobieństwo awarii układu przed upływem tysiąca godzin pracy? Znane są p_1 i p_2 — prawdopodobieństwo, że pierwsza (odpowiednio druga) żarówka spali się przed upływem tysiąca godzin pracy.

Pan Foszcz przysłał również rozwiązanie z propozycją znalezienia w nim błędu:

Niech S_1, S_2 będą zdarzeniami polegającymi na tym, że Z_1 (odpowiednio Z_2) spali się przed upływem tysiąca godzin. Układ wytrzymuje co najmniej tysiąc godzin wtedy i tylko wtedy, gdy zachodzi zdarzenie $S_1' \cap S_2'$ (S_1' oznacza zdarzenie przeciwne do S_1, tj. Z_1 wytrzymuje co najmniej tysiąc godzin). Mamy teraz

$$P((S_1' \cap S_2')') = P(S_1 \cup S_2) = P(S_1) + P(S_2) - P(S_1 \cap S_2).$$

Ale $P(S_1 \cap S_2) = 0$, bo zawsze najpierw spali się jedna z żarówek — przez drugą przestanie płynąć prąd i nie będzie miała okazji się spalić. W takim razie szukane prawdopodobieństwo wynosi $p_1 + p_2$ i może być większe od 1.

Panu Foszczowi rozgryzienie tego problemu zajęło wiele czasu i sprawiło, że rachunek prawdopodobieństwa należy do tych działów matematyki, które najbardziej mu się podobają. Nie podaje jednak właściwego rozwiązania — rozumiemy to jako wyzwanie dla Redakcji. Oto nasza odpowiedź.

Błąd polega na tym, że S_1 jest zdarzeniem „pierwsza żarówka spali się *nie później niż druga* przed upływem tysiąca godzin", zatem $P(S_1)$ jest zapewne różne od p_1. To samo odnosi się do zdarzenia S_2. Można to łatwo zobaczyć na rysunku. Rozpatrzmy, zamiast pierwotnego układu, obwód złożony z dwóch żarówek połączonych *równolegle*. Zdarzenie „Z_1 spali się w chwili t_1 i Z_2 spaliła się w chwili t_2" reprezentuje na rysunku punkt (t_1, t_2). Zdarzenia — to teraz „porządne" podzbiory pierwszej ćwiartki płaszczyzny, a prawdopodobieństwo — to unormowana miara na tejże ćwiartce. Widać teraz, że

$$P(S_1 \cup S_2) = 1 - P(S_1' \cap S_2') = 1 - P(A_1 \cap A_2).$$

Przypominamy, że rozpatrujemy teraz żarówki włączone równolegle i gdy przepali się jedna z nich, doświadczenie się kończy. Łatwo uwierzyć, że A_1 i A_2 są niezależne. Zatem

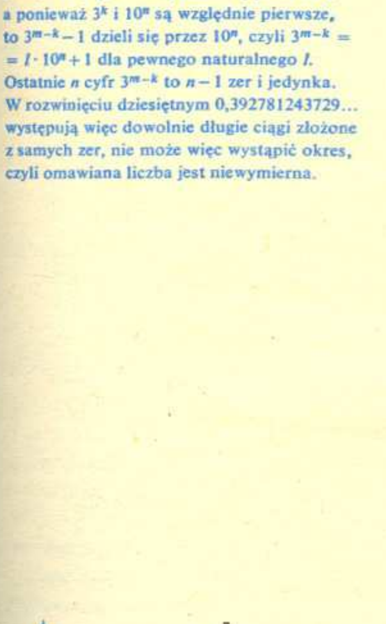
$$P(A_1) = P(A_1) \cdot P(A_2) = (1-p_1) \cdot (1-p_2).$$

Stąd szukane prawdopodobieństwo wynosi

$$1 - (1-p_1) \cdot (1-p_2) = p_1 + p_2 - p_1 p_2.$$

Zauważmy jeszcze, że $S_1 \cap S_2 \neq \emptyset$. Niemniej jednak jest to zbiór „chudy" — przekątna kwadratu. Przy każdym rozsądnym określeniu prawdopodobieństwa będziemy mieli $P(S_1 \cap S_2) = 0$.

R.S.



$A_1 = \{(t_1, t_2): t_1, t_2 \geqslant 0, t_1 > 1000\}$
$A_2 = \{(t_1, t_2): t_1, t_2 \geqslant 0, t_2 > 1000\}$
$S_1 = \{(t_1, t_2): 0 \leqslant t_1 \leqslant t_2, t_1 \leqslant 1000\}$
$S_2 = \{(t_1, t_2): 0 \leqslant t_2 \leqslant t_1, t_2 \leqslant 1000\}$