Złożoność informacyjna Kołmogorowa

Paweł Parys

Serock 2012

niektóre liczby łatwiej zapamiętać niż inne...
(to zależy nie tylko od wielkości liczby)

$1\underbrace{00...0}_{100}$

$100^{100^{\cdots^{100}}}$ (100 poziomów)

25839496603316858921

31415926535897932384

niektóre liczby łatwiej zapamiętać niż inne...
(to zależy nie tylko od wielkości liczby)

$1\underbrace{00...0}_{100}$

$100^{100^{\cdots^{100}}}$ (100 poziomów)

25839496603316858921 ← 20 „losowych” cyfr

31415926535897932384 ← π

$$\frac{1}{1}-\frac{1}{3}+\frac{1}{5}-\frac{1}{7}+\frac{1}{9}-\ldots=\frac{\pi}{4}$$

wybrana księga „Pana Tadeusza”

10 stron „losowych” symboli

Jakie pojęcia matematyczne kryją się za tym zjawiskiem?

niektóre liczby można opisać znacznie krócej
niż podając ich rozwinięcia dziesiętne

złożoność liczby = długość najkrótszego opisu

Jakie pojęcia matematyczne kryją się za tym zjawiskiem?

niektóre liczby można opisać znacznie krócej
niż podając ich rozwinięcia dziesiętne

złożoność liczby = długość najkrótszego opisu

najmniejsza liczba, której nie da się
opisać w języku polskim co najwyżej
trzydziestoma słowami = ???

(paradoks Berry'ego)

najmniejsza liczba, której nie da się opisać w języku polskim co najwyżej trzydziestoma słowami = ???

(paradoks Berry'ego)

niezbędne jest wprowadzenie „notacji”

Idea Kołmogorowa:

złożoność liczby = długość najkrótszego programu wypisującego tą liczbę

np. w języku Pascal

- bez znaczenia czy to liczba, czy napis, czy program, czy ciąg 0 i 1...
- program + wejście do programu

złożoność liczby = długość najkrótszego programu wypisującego tą liczbę

najmniejsza liczba, której nie da się wygenerować programem w Pascalu długości co najwyżej 3000 = ???

złożoność liczby = długość najkrótszego programu wypisującego tą liczbę

najmniejsza liczba, której nie da się wygenerować programem w Pascalu długości co najwyżej 3000 = ???

dlaczego nie da się jej wygenerować w Pascalu programem długości <=3000?

a taki program?
„zasymuluj wszystkie programy długości <=3000,
one wypiszą jakieś liczby, weź najmniejszą niewypisaną”

złożoność liczby = długość najkrótszego programu wypisującego tą liczbę

najmniejsza liczba, której nie da się wygenerować programem w Pascalu długości co najwyżej 3000 = ???

dlaczego nie da się jej wygenerować w Pascalu programem długości <=3000?

a taki program?
„zasymuluj wszystkie programy długości <=3000,
one wypiszą jakieś liczby, weź najmniejszą niewypisaną”

problem: niektóre programy się nie zatrzymają
(program działa bardzo długo - czy już się zawiesił,
czy obliczenia skończą się po długim czasie?)

Wniosek: Nie istnieje program, który bierze na wejściu program P i stwierdza, czy program P zakończy się.

(problem stopu)

gdyby istniał, dostalibyśmy sprzeczność

Ile najwyżej wynosi złożoność liczby x?

Ile najwyżej wynosi złożoność liczby x?

złożoność(x) ≤ x+stała

Zawsze można użyć programu „print x”

Jaki język programowania wybrać do definicji?

$złoż_{Pascal}$(liczba) = długość najkrótszego programu w Pascalu wypisującego tą liczbę

$złoż_{C++}$(liczba) = długość najkrótszego programu w C++ wypisującego tą liczbę

$złoż_{Java}$(liczba) = długość najkrótszego programu w Java wypisującego tą liczbę

...

Jak te wartości mają się do siebie?

Jaki język programowania wybrać do definicji?

istnieje stała taka, że dla każdego x

$$| \text{złoż}_{\text{Pascal}}(x) - \text{złoż}_{\text{C++}}(x) | \leq \text{stała}$$

Dowód:
W C++ możemy napisać interpreter Pascala
i dołączyć program w Pascalu.
(„stała” to rozmiar tego interpretera)

Zatem: wybór języka programowania (prawie) nie ma znaczenia

definicja:

złożoność liczby = długość najkrótszego programu w Pascalu wypisującego tą liczbę

było:

- złożoność(x) ≤ x+stała
- wartość złożoności niewiele zależy od języka programowania
- nie istnieje program, który bierze na wejściu program P i stwierdza, czy program P zakończy się

definicja:

złożoność liczby = długość najkrótszego programu w Pascalu wypisującego tą liczbę

było:

- złożoność(x) ≤ x+stała
- wartość złożoności niewiele zależy od języka programowania
- nie istnieje program, który bierze na wejściu program P i stwierdza, czy program P zakończy się

Czy istnieje program, który dla danej liczby x wyznacza jej złożoność?

Czy istnieje program, który dla danej liczby x wyznacza jej złożoność?

NIE!

Dowód:
gdyby istniał, to dla danego n moglibyśmy znaleźć najmniejszą (w porządku „wojskowym”) liczbę x, która ma złożoność n (sprawdzając kolejne liczby).
Ale to znaczy, że ma ona złożoność około log(n).

definicja:

złożoność liczby = długość najkrótszego programu w Pascalu wypisującego tą liczbę

było:

- złożoność(x) ≤ x+stała
- wartość złożoności niewiele zależy od języka programowania
- nie istnieje program, który bierze na wejściu program P i stwierdza, czy program P zakończy się
- nie istnieje program, który dla danej liczby x wyznacza jej złożoność

Stała Chaitina

Stała Chaitina

$$\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$$

tzn. losujemy dowolny (nieskończony) ciąg znaków, jakie jest prawdopodobieństwo że na początku jest poprawny składniowo program, który się zatrzyma

założenia techniczne:

- załóżmy, że te programy nic nie wczytują (operacja odczytu nie istnieje)
- w tej sumie uwzględniamy tylko programy, które się poprawnie kompilują
- bezprefiksowość: jeśli v jest poprawnym programem, to vw nie jest

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

Co o niej wiemy?

- liczba ta jest mocno „losowa”
- gdybyśmy ją znali, to moglibyśmy rozwiązać problem stopu

(oczywiście konkretna wartość liczby Ω zależy od języka programowania)

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

gdybyśmy ją znali, to moglibyśmy rozwiązać problem stopu

tzn. istnieje program, który

- wczytuje program P
- wczytuje pierwsze n=|P| cyfr $w_1 w_2 ... w_n$ liczby Ω
- stwierdza, czy P zatrzymuje się.

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

gdybyśmy ją znali, to moglibyśmy rozwiązać problem stopu

tzn. istnieje program, który

- wczytuje program P
- wczytuje pierwsze n=|P| cyfr $w_1 w_2 ... w_n$ liczby Ω
- stwierdza, czy P zatrzymuje się.

Dowód (algorytm):

- uruchamiamy program P (jeśli się zatrzyma wypisz TAK)
- równocześnie uruchamiamy wszystkie inne programy; niech S to zbiór tych które już się zatrzymały; wypisz NIE jeśli $0.w_1 w_2 ... w_n \leqslant \sum_{v \in S} 2^{-|v|}$

Stała Chaitina: Ω

ruchem „zygzakowym”

p_1 - 6 kroków
p_2 - 5 kroków
p_3 - 4 kroków
p_4 - 3 kroków
p_5 - 2 kroków
p_6 - 1 kroków
...

gdybyśmy ją znali, to mo … stopu

tzn. istnieje program, któ…

- wczytuje program P
- wczytuje pierwsze n=|P…
- stwierdza, czy P zatrzy…

Dowód (algorytm):

- uruchamiamy program P (jeśli się zatrzyma wypisz TAK)
- równocześnie uruchamiamy wszystkie inne programy; niech S to zbiór tych które już się zatrzymały; wypisz NIE jeśli $0.w_1 w_2 \ldots w_n \leqslant \sum_{v \in S} 2^{-|v|}$

tzn. istnieje program, który

- wczytuje program P
- wczytuje pierwsze n=|P| cyfr $w_1 w_2 ... w_n$ liczby Ω
- stwierdza, czy P zatrzymuje się.

Dowód (algorytm):

- uruchamiamy program P (jeśli się zatrzyma wypisz TAK)
- równocześnie uruchamiamy wszystkie inne programy;
  niech S to zbiór tych które już się zatrzymały;
  wypisz NIE jeśli $0.w_1 w_2 ... w_n \leqslant \sum_{v \in S} 2^{-|v|}$

Dlaczego odpowiedź NIE jest prawidłowa?
bo jeśli jeśli P się zatrzymuje, to Ω jest większa jeszcze o 2^{-n}

Dlaczego któryś z tych przypadków zajdzie?
bo powyższa nierówność na pewno zachodzi
dla pewnego skończonego zbioru S
(sumę nieskończoną można dowolnie przybliżyć sumami skończonymi)

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

liczba ta jest mocno „losowa”:

Niech $w_1 w_2 ... w_n$ to pierwsze n cyfr liczby Ω.

złożoność($w_1 w_2 ... w_n$) ≥ n-stała

(czyli nie da się jej sprytnie wyliczyć, można ją jedynie zapamiętać)

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

złożoność($w_1 w_2 ... w_n$) ≥ n-stała (Ω jest mocno „losowa”)

Dowód:
Mamy program, rozmiaru (złożoność($w_1 w_2 ... w_n$)+stała) generujący liczbę o złożoności co najmniej n:

1) wygeneruj $\Omega_n = 0.w_1 w_2 ... w_n$
2) uruchamiaj równocześnie wszystkie programy („zygzak”), zapamiętuj (S) programy, które się zakończyły, i ich wyjścia
3) jeśli $\Omega_n \leqslant \sum_{v \in S} 2^{-|v|}$ to wypisz najmniejszą nie wypisaną liczbę X;

widzimy, że złożoność(X)≥n

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

złożoność($w_1 w_2 ... w_n$) ≥ n-stała (Ω jest mocno „losowa”)

Wniosek:

każde skończone słowo ustalonej długości
pojawia się w Ω tak samo często

(wpp moglibyśmy skompresować)

Stała Chaitina: $\Omega = \sum_{P \text{ zatrzymuje się}} 2^{-|P|}$

(prawdopodobieństwo, że losowy program zatrzyma się)

Co o niej wiemy?

- liczba ta jest mocno „losowa”
- gdybyśmy ją znali, to moglibyśmy rozwiązać problem stopu

Wylosujmy program - co on wypisze?

$$p(y)=\frac{1}{\Omega}\sum_{P\ wypisuje\ y} 2^{-|P|}$$

(prawdopodobieństwo, że losowy zatrzymujący się program wypisze y)

Wylosujmy program - co on wypisze?

$$p(y)=\frac{1}{\Omega}\sum_{P\ wypisuje\ y} 2^{-|P|}$$

(prawdopodobieństwo, że losowy zatrzymujący się program wypisze y)

Okazuje się, że

$$p(y)\approx\frac{1}{2^{złożoność(y)}}$$

Dokładniej: istnieje stała c, że dla dowolnego y

złożoność(y)-c<-$\log_2$p(y)<złożoność(y)+c

Uniwersalny test Martina-Löfa

test mierzy jak dziwny jest ciąg - zwraca liczbę „dziwactw”
typowy ciąg to taki, który należy do każdej rozsądnej większości, czyli ma niewiele „dziwactw”

Def. Test to funkcja δ:(0+1)* $\rightarrow$ℕ taka, że

$$\frac{\left|\left\{w \in (0+1)^n : \delta(w) \geqslant m\right\}\right|}{2^n} \leqslant \frac{1}{2^m}$$

oraz że istnieje program wypisujący wszystkie pary (m,x) takie, że δ(x)$\geq$m

Przykładowy test: δ(x) = od ilu jedynek zaczyna się ciąg x
(tylko co 2^m-ty ciąg zaczyna się od $\geq$m jedynek)

Uniwersalny test Martina-Löfa

test mierzy jak dziwny jest ciąg - zwraca liczbę „dziwactw”
typowy ciąg to taki, który należy do każdej rozsądnej większości, czyli ma niewiele „dziwactw”

Def. Test to funkcja δ:(0+1)* $\rightarrow$ℕ taka, że

$$\frac{\left|\left\{w \in (0+1)^n : \delta(w) \geq m\right\}\right|}{2^n} \leqslant \frac{1}{2^m}$$

oraz że istnieje program wypisujący wszystkie pary (m,x) takie, że δ(x)$\geq$m

Moglibyśmy się spodziewać, że
dla różnych własności testy będą różne.

Tymczasem istnieje test uniwersalny!

Uniwersalny test Martina-Löfa

Istnieje test uniwersalny δ^U

czyli taki, że dla każdego testu δ istnieje stała c_δ, że dla każdego ciągu x

$$\delta^U(x) \geq \delta(x) - c_\delta$$

Uniwersalny test Martina-Löfa

Istnieje test uniwersalny δ^U

czyli taki, że dla każdego testu δ istnieje stała c_δ, że dla każdego ciągu x

$$\delta^U(x) \geq \delta(x) - c_\delta$$

Dowód

Uruchom równocześnie wszystkie możliwe programy („zygzak”).

Dla każdego zapamiętaj wypisane dotychczas pary (m,x).

Jeśli pary dotychczas wypisane przez k-ty program spełniają warunek testu, to wypisz parę (m-k,x)

Względna złożoność Kołmogorowa

K(x|y) = jak długi jest najkrótszy program wypisujący x,
jeśli na wejściu dostanie y

Względna złożoność Kołmogorowa

$K(x|y)$ = jak długi jest najkrótszy program wypisujący x, jeśli na wejściu dostanie y

Oczywiście $K(x|y) \leq K(x) \leq K(y)+K(x|y)$

Złożoność Kołmogorowa a test uniwersalny

Okazuje się, że następująca funkcja jest testem uniwersalnym:

$$\delta^U(x)=|x|-K(x||x|)$$

Złożoność Kołmogorowa a test uniwersalny

Okazuje się, że następująca funkcja jest testem uniwersalnym:

$$\delta^U(x)=|x|-K(x||x|)$$

Dowód

Dlaczego da się ją obliczać?

Uruchamiamy (współbieżnie) każdy program P
dając mu na wejście każdą liczbę n.
Jeśli wypisze ciąg x długości n to wiemy, że $K(x||x|)\leq|P|$;
możemy więc wypisać parę $(|x|-|P|, x)$ oznaczającą, że $\delta^U(x)\geq|x|-|P|$.

Złożoność Kołmogorowa a test uniwersalny

Okazuje się, że następująca funkcja jest testem uniwersalnym:

$$\delta^U(x)=|x|-K(x||x|)$$

<u>Dowód</u>

Da się ją obliczać.

Dlaczego mało jest ciągów, dla których wychodzi duża wartość?

Liczba ciągów takich, że $|x|-K(x||x|)\geq m$ (tzn. $K(x||x|)\leq|x|-m$)
nie może być więcej niż programów długości $|x|-m$
tzn. $2^{|x|-m}$

Złożoność Kołmogorowa a test uniwersalny

Okazuje się, że następująca funkcja jest testem uniwersalnym:

$$\delta^U(x)=|x|-K(x||x|)$$

Dowód

Da się ją obliczać.
Mało jest ciągów, dla których wychodzi duża wartość.

Dlaczego dobrze symuluje dowolny test δ?

Wystarczy pokazać, że $K(x||x|)\leq|x|-\delta(x)+$stała (stała może zależeć od δ)
Rozważmy zbiór $A=\{z:\delta(z)\geq\delta(x), |z|=|x|\}$.
Znając $|x|$ oraz $\delta(x)$ możemy wyliczyć elementy A
(za pomocą programu liczącego δ).
Niech s będzie numerem x w tym wyliczeniu.
Elementów A jest mało - najwyżej $2^{|x|-\delta(x)}$.
Wystarczy więc zapamiętać liczbę s (jej długość daje nam też $\delta(x)$).

Dziękuję za uwagę