Czy wszystko można policzyć na komputerze

Maciej M. Sysło

Uniwersytet Wrocławski, UMK w Toruniu

syslo@ii.uni.wroc.pl, syslo@mat.uni.torun.pl

http://mmsyslo.pl/

Streszczenie

Przedmiotem wykładu jest łagodne wprowadzenie do złożoności problemów i algorytmów. Przewodnim pytaniem jest, jak dobrze sprawują się algorytmy i komputery i czy komputery już mogą wszystko obliczyć. Z jednej strony dla niektórych problemów (jak znajdowanie najmniejszego elementu) znane są algorytmy, które nie mają konkurencji, gdyż są bezwzględnie najlepsze, a z drugiej – istnieją takie, o których przypuszcza się, że komputery nigdy nie będą w stanie ich rozwiązywać dostatecznie szybko. Przedstawione zostaną problemy, których nie potrafimy rozwiązywać szybko, nawet z użyciem najszybszych komputerów. Problemy z tej drugiej grupy znajdują zastosowanie na przykład w kryptografii. Podejmowane kwestie będą ilustrowane praktycznymi zastosowaniami omawianych problemów i ich metod obliczeniowych.

Spis treści

1 WPROWADZENIE

Można odnieść wrażenie, że komputery są obecnie w stanie wykonać wszelkie obliczenia i dostarczyć oczekiwany wynik w krótkim czasie. Budowane są jednak coraz szybsze komputery, co może świadczyć o tym, że moc istniejących komputerów nie jest jednak wystarczająca do wykonania wszystkich obliczeń, jakie nas interesują, potrzebne są więc jeszcze szybsze maszyny.

Zgodnie z nieformalnym prawem Moore'a, szybkość działania procesorów stale rośnie i podwaja się co 24 miesiące. Jednak istnieje wiele trudno rozwiązywalnych problemów, których obecnie nie jesteśmy w stanie rozwikłać za pomocą żadnego komputera i zwiększanie ich szybkości niewiele zmienia tę sytuację. Kluczowe staje się więc opracowywanie coraz szybszych algorytmów.

2 SUPERKOMPUTERY I ALGORYTMY

Superkomputery

Komputery, które w danej chwili lub w jakimś okresie mają największą moc obliczeniową przyjęło się nazywać **superkomputerami**. Prowadzony jest ranking najszybszych komputerów świata (patrz strona http://www.top500.org/), a faktycznie odbywa się wyścig wśród producentów superkomputerów i dwa razy w roku publikowane są listy rankingowe. Szybkość działania komputerów jest oceniana na specjalnych zestawach problemów, pochodzących z algebry liniowej – są to na ogół układy równań liniowych, złożone z setek tysięcy, a nawet milionów równań i niewiadomych. Szybkość komputerów podaje się w jednostkach zwanych **FLOPS** (**flops** lub **flop/s**) – jest to akronim od ***FL**oating point **O**perations **P**er **S**econd*, czyli zmiennopozycyjnych operacji na sekundę. Dla uproszczenia można przyjąć, że FLOPS to są działania arytmetyczne (dodawanie, odejmowanie, mnożenie i dzielenie) wykonywane na liczbach dziesiętnych z kropką. W użyciu są jednostki: YFlops (yotta flops) – 10^{24} operacji na sekundeę, ZFlops (zetta flops) – 10^{21}, EFlops (exa flops) – 10^{18}, PFlops (peta flops) – 10^{15}, TFlops (tera flops) – 10^{12}, GFlops (giga flops) – 10^{9}, MFlops (mega flops) – 10^{6}, KFlops (kilo flops) – 10^{3}.

W rankingu z listopada 2009 roku, najszybszym komputerem świata był **Cray XT Jaguar** firmy Cray, którego moc wynosi 1,75 PFlops, czyli wykonuje on ponad 10^{15} operacji na sekundę. Komputer ten pracuje w Department of Energy's Oak Ridge National Laboratory w Stanach Zjednoczonych. Komputer Jaguar jest zbudowany z 224 162 procesorów Opteron firmy AMD. Drugie miejsce należy do Roadrunnera firmy IBM o mocy 1,04 PFlops. Dwa dalsze miejsca zajmują również komputery Cray i IBM, a w pierwszej dziesiątce są jeszcze dwa inne komputery IBM.

Na początku 2010 roku najszybszym procesorem PC był Intel Core i7 980 XE o mocy 107,6 GFlops. Większą moc mają procesory graficzne, np. Tesla C1060 GPU (nVidia) posiada moc 933 GFlops (w pojedynczej dokładności), a HemlockXT 5970 (AMD) osiąga 4640 GFlops (w podwójnej dokładności).

Bardzo dużą moc uzyskują obliczenia wykonywane na rozproszonych komputerach osobistych połączonych ze sobą za pomocą Internetu. Na przykład, ma początku 2010 roku Folding@Home osiągnął moc 3,8 PFlops, a komputery osobiste pracujące w projekcie sieciowym GIMPS nad znalezieniem coraz większej liczby pierwszej uzyskały moc 44 TFlops.

Przy obecnym tempie wzrostu możliwości superkomputerów przewiduje się, że moc 1 EFlops (10^{18} operacji na sekundę) zostanie osiągnięta w 2019 roku. Naukowcy oceniają, że dla pełnego modelowania zmian pogody na Ziemi w ciągu dwóch tygodni jest potrzebny komputer o mocy 1 ZFlops (10^{21} operacji na sekundę). Przewiduje się, że taki komputer powstanie przed 2030 rokiem.

W dalszych rozważaniach będziemy przyjmować, że dysponujemy najszybszym komputerem, który ma moc 1 PFlops, czyli wykonuje 10^{15} = 1 000 000 000 000 000 operacji na sekundę.

Superkomputery a algorytmy

Ciekawi nas teraz, jak duże problemy możemy rozwiązywać za pomocą komputera o mocy 1 PFlops. W tabeli 1 zamieszczano czasy obliczeń wykonanych na tym superkomputerze, posługując się algorytmami o różnej złożoności obliczeniowej (pracochłonności) i chcąc rozwiązywać problemy o różnych rozmiarach.

Tabela 1.
Czasy obliczeń za pomocą algorytmów o podanej złożoności, wykonywanych na superkomputerze o mocy 1 PFlps (« 1 sek. oznacza w tabeli, że czas obliczeń stanowił niewielki ułamek sekundy; parametr n określa rozmiar danych.)

Złożoność algorytmu	$n = 100$	$n = 500$	$n = 1000$	$n = 10000$
$\log n$	« 1 sek.	« 1 sek.	« 1 sek.	« 1 sek.
n	« 1 sek.	« 1 sek.	« 1 sek.	« 1 sek.
$n \log n$	« 1 sek.	« 1 sek.	« 1 sek.	« 1 sek.
n^2	« 1 sek.	« 1 sek.	0,000000001 sek.	0,0000001 sek.
n^5	0,00001 sek.	0,03125 sek.	1 sek.	1,15 dni
2^n	$4*10^7$ lat	$1{,}038*10^{128}$ lat	$3{,}3977*10^{278}$ lat	
$n!$	$2{,}959*10^{135}$ lat			

Wyraźnie widać, że dwa ostatnie algorytmy w tabeli są całkowicie niepraktyczne nawet dla niewielkiej liczby danych ($n = 100$). Istnieją jednak problemy, dla których wszystkich możliwych rozwiązań może być 2^n lub $n!$, zatem nawet superkomputery nie są w stanie przejrzeć wszystkich możliwych rozwiązań w poszukiwaniu najlepszego, a zdecydowanie szybszymi metodami dla tych problemów nie dysponujemy.

Można łatwo przeliczyć, że gdyby nasz superkomputer był szybszy, na przykład miał moc 1 ZFlops (10^{21} operacji na sekundę), to dwa ostatnie wiersze w tabeli 1 uległyby tylko niewielkim zmianom, nie na tyle jednak, by móc stosować dwa ostatnie algorytmy w celach praktycznych.

W następnym rozdziale przedstawimy kilka przykładowych problemów, dla których moc obliczeniowa superkomputerów może nie być wystarczająca, by rozwiązywać je dla praktycznych rozmiarów danych. Ma to złe i dobre strony. Złe – bo nie jesteśmy w stanie rozwiązywać wielu ważnych i istotnych dla człowieka problemów, takich np. jak prognozowanie pogody, przewidywanie trzęsień Ziemi i wybuchów wulkanów, czy też planowanie optymalnych podróży lub komunikacji. Dobre zaś – bo można wykorzystywać trudne obliczeniowo problemy w metodach szyfrowania, dzięki czemu nasz przeciwnik nie jest w stanie, nawet posługując się najszybszymi komputerami, deszyfrować naszych wiadomości, chociaż my jesteśmy w stanie szybko je kodować i wysyłać.

3 PRZYKŁADY ZŁOŻONYCH PROBLEMÓW

W tym rozdziale, jak i w dalszych, problemy będą definiowane w postaci **specyfikacji**, która zawiera ścisły opis *Danych* i oczekiwanych *Wyników*. W specyfikacji są też zawarte zależności między danymi i wynikami. Specyfikacja problemu jest również specyfikacją algorytmu, który ten problem rozwiązuje, co ma na celu dokładne określenie przeznaczenia algorytmu, stanowi zatem powiązanie algorytmu z rozwiązywanym problemem.

3.1 NAJKRÓTSZA TRASA ZAMKNIĘTA

Jednym z najbardziej znanych problemów związanych z wyznaczaniem tras przejazdu, jest **problem komiwojażera**, oznaczany zwykle jako **TSP** (ang. *Travelling Salesman Problem*). Podany jest dany zbiór miejscowości oraz odległości między nimi. Należy znaleźć drogę zamkniętą najkrótszą, przechodzącą przez każdą miejscowość dokładnie jeden raz. Problem ten został omówiony w punkcie 5.2.1 w rozdziale Informatyka – klucz do zrozumienia, kariery, dobrobytu.

3.2 ROZKŁAD LICZBY NA CZYNNIKI PIERWSZE

Liczby pierwsze stanowią w pewnym sensie „pierwiastki" wszystkich liczb, każdą bowiem liczbę całkowitą można jednoznacznie przedstawić, z dokładnością do kolejności, w postaci iloczynu liczb pierwszych. Na przykład, 4 = 2*2; 10 = 2*5; 20 = 2*2*5; 23 = 23; dla liczb pierwszych te iloczyny składają się z jednej liczby.

Matematycy interesowali się liczbami pierwszymi od dawna. Pierwsze spisane rozważania i twierdzenia dotyczące tych liczb znajdujemy w działach Euklidesa. Obecnie liczby pierwsze znajdują ważne zastosowania w kryptografii, m.in. w algorytmach szyfrujących. Do najważniejszych problemów związanych z liczbami pierwszymi należy badanie, czy dana dodatnia liczba całkowita n jest liczbą pierwszą i ewentualny jej rozkład na czynniki, jeśli jest liczbą złożoną.

3.3 PODNOSZENIE DO POTĘGI

Podnoszenie do potęgi jest bardzo prostym, szkolnym zadaniem. Na przykład, aby obliczyć 3^4, wykonujemy trzy mnożenia 4*4*4*4. A zatem w ogólności, aby obliczyć szkolną metodą wartość x^n, należy wykonać $n - 1$ mnożeń, o jedno mniej niż wynosi wykładnik potęgi. Czy ten algorytm jest na tyle szybki, by obliczyć na przykład wartość:

$$x^{12345678912345678912345678912345}$$

która może pojawić przy szyfrowaniu informacji przesyłanych w Internecie? Odpowiedź na to pytanie w punkcie 4.2.

3.4 PORZĄDKOWANIE

Problem **porządkowania**, często nazywany **sortowaniem**, jest jednym z najważniejszych problemów w informatyce i w wielu innych dziedzinach. Jego znaczenie jest ściśle związane z zarządzaniem danymi (informacjami), w szczególności z wykonywaniem takich operacji na danych, jak wyszukiwanie konkretnych danych lub umieszczanie danych w zbiorze.

Warto zauważyć, że mając n elementów, które chcemy uporządkować, istnieje $n!$ możliwych uporządkowań, wśród których chcemy znaleźć właściwe uporządkowanie. A zatem, przestrzeń możliwych rozwiązań w problemie porządkowania n liczb ma moc $n!$, czyli jest taka sama jak w przypadku problemu komiwojażera. Jednak dzięki odpowiednim algorytmom, n elementów można uporządkować w czasie proporcjonalnym do $n \log n$ lub n^2.

Dysponując uporządkowanym zbiorem danych, znalezienie w nim elementu lub umieszczenie nowego z zachowaniem porządku jest znacznie łatwiejsze i szybsze, niż gdyby zbiór nie był uporządkowany.

3.5 OBLICZANIE WARTOŚCI WIELOMIANU – SCHEMAT HORNERA

Obliczanie wartości wielomianu o zadanych współczynnikach jest jedną z najczęściej wykonywanych operacji w komputerze. Wynika to z ważnego faktu matematycznego, zgodnie z którym każdą funkcję (np. funkcje trygonometryczne) można zastąpić wielomianem, którego postać zależy od funkcji i od tego, jaką chcemy uzyskać dokładność obliczeń. Najefektywniejszym sposobem obliczania wartości wielomianu jest **schemat Hornera**, wynikający z następującego przekształcenia postaci wielomianu:

$$\begin{aligned} w_n(x) &= a_0x^n + a_1x^{n-1} + \ldots + a_{n-1}x + a_n = (a_0x^{n-1} + a_1x^{n-2} + \ldots + a_{n-1})x + a_n \\ &= ((a_0x^{n-2} + a_1x^{n-3} + \ldots + a_{n-2})x + a_{n-1})x + a_n = \\ &= (\ldots((a_0x + a_1)x + a_2)x + \ldots + a_{n-2})x + a_{n-1})x + a_n \end{aligned}$$

Ten ostatni wzór można zapisać w następującej postaci algorytmicznej:

$y := a_0$
$y := yz + a_i$ dla $i = 1, 2, \ldots, n$.

Stąd otrzymujemy algorytm:

Algorytm: schemat Hornera
Dane: n – nieujemna liczba całkowita (stopień wielomianu);
$a_0, a_1, \ldots, a_n$ – n+1 współczynników wielomianu;
z – wartość argumentu.
Wynik: $y = w_n(z)$ – wartość wielomianu stopnia n dla wartości argumentu $x = z$.
Krok 1. $y := a_0$ – początkową wartością jest współczynnik przy najwyższej potędze.
Krok 2. Dla $i = 1, 2, \ldots, n$ oblicz wartość dwumianu $y := yz + a_i$.

Z postaci algorytmu wynika, że obliczenie wartości wielomianu stopnia n wymaga wykonania n mnożeń i n dodawań. Udowodniono, że jest to najszybszy sposób obliczania wartości wielomianu.

Schemat Hornera ma wiele zastosowań, m.in. do:
- obliczania dziesiętnej wartości liczby, danej w systemie o podstawie p – współczynniki wielomianu są w tym przypadku cyframi $\{0, 1, \ldots, p-1\}$, a argumentem p podstawa systemu,
- szybkiego obliczania wartości potęgi (patrz punkt 4.2).

4 DWA TRUDNE PROBLEMY

Warto wrócić tutaj do dwóch problemów przedstawionych w rozdziale 3. Jednym z nich jest sprawdzanie, czy dana liczba jest liczbą pierwszą, a drugim – szybkie podnoszenie do potęgi. Dla pierwszego z tych problemów nie mamy dobrej wiadomości, nie jest bowiem znana żadna metoda szybkiego testowania pierwszości liczb. Korzysta się z tego na przykład w szyfrze RSA, częścią kluczy w tej metodzie jest liczba będąca iloczynem dwóch dużych liczb pierwszych i brak znajomości tego rozkładu jest gwarancją bezpieczeństwa tego szyfru.

Drugim problemem jest podnoszenie do dużych potęg i tutaj mamy bardzo dobrą wiadomość – podnoszenie nawet do bardzo dużych potęg, zawierających setki cyfr, trwa ułamek sekundy.

4.1 BADANIE ZŁOŻONOŚCI LICZB

Dla problemu badania, czy dana liczba n jest liczbą pierwszą czy złożoną, dysponujemy prostym algorytmem, który polega na dzieleniu n przez kolejne liczby naturalne i wystarczy dzielić tylko przez liczby nie większe niż $\sqrt{n}$, gdyż liczby n nie można rozłożyć na iloczyn dwóch liczb większych od $\sqrt{n}$. Algorytm ten ma bardzo prostą postać:

```
var i,n:integer;
i:=2;
while i*i <= n do begin
   if n mod i = 0 then return 1;  {n jest podzielne przez i}
   i=i+1
end;
return 0    {n jest liczba pierwszą}
```

Ten fragment programu zwraca 0, jeśli n jest liczbą pierwszą, i 1, gdy n jest liczbą złożoną. W tym drugim przypadku znamy także podzielnik liczby n. Ten fragment programu można rozszerzyć do programu generującego wszystkie dzielniki liczby n.

Liczba operacji wykonywanych przez powyższy program jest w najgorszym przypadku (gdy n jest liczbą pierwszą) proporcjonalna do $\sqrt{n}$, a więc jeśli $n = 10^m$, to wykonywanych jest $10^{m/2}$ operacji. Zatem są niewielkie szanse, by tym algorytmem rozłożyć na czynniki pierwsze liczbę, która ma kilkaset cyfr lub stwierdzić, że się jej nie da rozłożyć.

Rozkładem liczby złożonej na czynniki pierwsze mogą być zainteresowani ci, którzy starają się złamać szyfr RSA. Wiadomo, że jedna z liczb tworzących klucz publiczny i prywatny jest iloczynem dwóch liczb pierwszych. Znajomość tych dwóch czynników umożliwia utworzenie klucza prywatnego (tajnego). Jednak ich wielkość – są to liczby pierwsze o kilkuset cyfrach (200-300) – stoi na przeszkodzie w rozkładzie n.

Istnieje wiele algorytmów, które służą do testowania pierwszości liczb oraz do rozkładania liczb na czynniki pierwsze. Niektóre z tych algorytmów mają charakter probabilistyczny – wynik ich działania jest poprawny z prawdopodobieństwem bliskim 1. Żaden jednak algorytm, przy obecnej mocy komputerów, nie umożliwia rozkładania na czynniki pierwsze liczb, które mają kilkaset cyfr. Szyfr RSA pozostaje więc nadal bezpiecznym sposobem utajniania wiadomości, w tym również przesyłanych w sieciach komputerowych.

4.2 SZYBKIE PODNOSZENIE DO POTĘGI

Wracamy tutaj do obliczania wartości potęgi x^m, dla dużych wartości wykładnika m[18]. W punkcie 3.3 podaliśmy przykład dla m = 12345678912345678912345678912345. Stosując „szkolny” algorytm, czyli kolejne mnożenia, i korzystając z naszego superkomputera, obliczenie potęgi dla tego wykładnika zajęłoby $3*10^8$ lat. Ten wykładnik jest faktycznie niewielką liczbą, w porównaniu z wykładnikami, jakie pojawiają się przy stosowaniu szyfru RSA, potrzebny jest więc znacznie efektywniejszy algorytm.

Do wyprowadzenia algorytmu szybkiego potęgowania posłużymy się rekurencyjnym zapisem operacji potęgowania:

$$x^m = \begin{cases} 1 & \text{jeśli } m = 0 \\ (x^{m/2})^2, & \text{jeśli } m \text{ jest liczbą parzystą} \\ (x^{(m-1)/2})^2 x & \text{jeśli } m \text{ jest liczbą nieparzystą} \end{cases}$$

Na przykład, jeśli chcemy obliczyć x^{22}, to powyższa zależność rekurencyjna prowadzi przez następujące odwołania rekurencyjne: $x^{22} = (x^{11})^2 = ((x^5)^2x)^2 = (((x^2)^2x)^2x)^2$.

Warto zauważyć, jaki jest związek kolejności wykonywania mnożeń, wynikającej z powyższej zależności rekurencyjnej, z postacią binarną wykładnika, zapisaną z użyciem schematu Hornera. Dla naszego przykładu mamy $m = (22)_{10} = (10110)_2$. Porównując kolejność mnożeń z postacią binarną widać, że podnoszenie do kwadratu odpowiada kolejnym pozycjom w rozwinięciu binarnym, a mnożenie przez x odpowiada cyfrze 1 w rozwinięciu binarnym. A zatem, liczba mnożeń jest nie większa niż dwa razy długość binarnego rozwinięcia wykładnika. Wiemy skądinąd, że długość rozwinięcia binarnego liczby m wynosi ok. $\log_2 m$. Mamy w przybliżeniu $\log_2 12345678912345678912345678912345 = 104$, w związku z tym podniesienie do tej potęgi wymaga wykonania nie więcej niż ok. 200 mnożeń.

W tabeli 2 zamieściliśmy wartości logarytmu przy podstawie 2 z rosnących wartości liczby m.

[18] Więcej na ten temat – punkt 4.1.1 w rozdziale Informatyka – klucz do zrozumienia, kariery, dobrobytu.