Zadania, których nie umiemy rozwiązać

Jakub Radoszewski

finał XVII Olimpiady Informatycznej

16 kwietnia 2010

Bardzo ostro

Faktoryzacja liczb:
Dana liczba naturalna n. Znaleźć jakikolwiek dzielnik $d \mid n$ (oczywiście $d \neq 1, d \neq n$) w złożoności czasowej $O((\log n)^c)$, dla jakiegokolwiek $c \in \mathbb{N}$.
A może tak szybko się nie da?

Bardzo ostro

Faktoryzacja liczb:
Dana liczba naturalna n. Znaleźć jakikolwiek dzielnik $d \mid n$ (oczywiście $d \neq 1, d \neq n$) w złożoności czasowej $O((\log n)^c)$, dla jakiegokolwiek $c \in \mathbb{N}$.
A może tak szybko się nie da?

Znane algorytmy:

- sprawdzanie do pierwiastka: $O(\sqrt{n})$

Bardzo ostro

Faktoryzacja liczb:
Dana liczba naturalna n. Znaleźć jakikolwiek dzielnik $d \mid n$ (oczywiście $d \neq 1, d \neq n$) w złożoności czasowej $O((\log n)^c)$, dla jakiegokolwiek $c \in \mathbb{N}$.
A może tak szybko się nie da?

Znane algorytmy:

- sprawdzanie do pierwiastka: $O(\sqrt{n})$
- rho Pollarda: $O(\sqrt[4]{n})$.

Ciekawe, choć może ambitne

Mnożenie macierzy:
Trzeba pomnożyć dwie macierze A oraz B, powiedzmy rozmiaru $n \times n$. Jak szybko umiemy to zrobić?

$$A \cdot B[i][j] := \sum_{k=1}^{n} A[i][k] \cdot B[k][j].$$

Ciekawe, choć może ambitne

Mnożenie macierzy:
Trzeba pomnożyć dwie macierze A oraz B, powiedzmy rozmiaru $n \times n$. Jak szybko umiemy to zrobić?

$$A \cdot B[i][j] := \sum_{k=1}^{n} A[i][k] \cdot B[k][j].$$

Znane algorytmy:

- brutalnie: $O(n^3)$

Ciekawe, choć może ambitne

Mnożenie macierzy:
Trzeba pomnożyć dwie macierze A oraz B, powiedzmy rozmiaru $n \times n$. Jak szybko umiemy to zrobić?

$$A \cdot B[i][j] := \sum_{k=1}^{n} A[i][k] \cdot B[k][j].$$

Znane algorytmy:

- brutalnie: $O(n^3)$
- algorytm Strassena: $O(n^{2.807})$

Ciekawe, choć może ambitne

Mnożenie macierzy:
Trzeba pomnożyć dwie macierze A oraz B, powiedzmy rozmiaru $n \times n$. Jak szybko umiemy to zrobić?

$$A \cdot B[i][j] := \sum_{k=1}^{n} A[i][k] \cdot B[k][j].$$

Znane algorytmy:

- brutalnie: $O(n^3)$
- algorytm Strassena: $O(n^{2.807})$
- najlepszy znany algorytm: $O(n^{2.376})$.

Ciekawe, choć może ambitne

Mnożenie macierzy:
Trzeba pomnożyć dwie macierze A oraz B, powiedzmy rozmiaru $n \times n$. Jak szybko umiemy to zrobić?

$$A \cdot B[i][j] := \sum_{k=1}^{n} A[i][k] \cdot B[k][j].$$

Znane algorytmy:

- brutalnie: $O(n^3)$
- algorytm Strassena: $O(n^{2.807})$
- najlepszy znany algorytm: $O(n^{2.376})$.

Ale właściwie czemu nie miałoby się dać $O(n^2)$?
Czyli np. mnożyć macierze 2000×2000?

Mój ulubiony problem

Najdłuższy wspólny podciąg:
Dane ciągi $a_1, \ldots, a_n$ oraz $b_1, \ldots, b_m$. Chcemy znaleźć najdłuższy wspólny podciąg.
Np. **a***a***bc**c*c*, *c***ab***ba***c**, wynik to **abc**.

- klasyczny algorytm: $O(n \cdot m)$, programowanie dynamiczne

Mój ulubiony problem

Najdłuższy wspólny podciąg:
Dane ciągi $a_1, \ldots, a_n$ oraz $b_1, \ldots, b_m$. Chcemy znaleźć najdłuższy wspólny podciąg.
Np. **a***a***b**c**c***c*, *c***ab***ba***c**, wynik to **abc**.

- klasyczny algorytm: $O(n \cdot m)$, programowanie dynamiczne
- da się jakoś $O\left(\frac{n \cdot m}{\log(n+m)}\right)$.

Mój ulubiony problem

Najdłuższy wspólny podciąg:
Dane ciągi $a_1, \ldots, a_n$ oraz $b_1, \ldots, b_m$. Chcemy znaleźć najdłuższy wspólny podciąg.
Np. **a***a***b***c***c***c*, *c***ab***ba***c**, wynik to **abc**.

- klasyczny algorytm: $O(n \cdot m)$, programowanie dynamiczne
- da się jakoś $O\left(\frac{n \cdot m}{\log(n+m)}\right)$.

Ale właściwie czemu nie miałoby się dać istotnie szybciej, np. $O((n+m)\sqrt{n+m})$ lub $O((n+m)\log(n+m))$?
Np. $n = m = 10^5$.

Zadanko numer 1

Zadanko numer 1

Widoczność prostokątów w 3D:
Mamy n równoległych płaszczyzn. Na każdej chcemy narysować *izotetyczny* prostokąt, tak żeby każde dwa prostokąty *widziały się*. Max $n =$??.

Dwa prostokąty *widzą się*, jeżeli możemy połączyć ich wnętrza odcinkiem prostopadłym do nich i niedotykającym żadnego z pozostałych prostokątów.

Zadanko numer 1

Widoczność prostokątów w 3D:
Mamy n równoległych płaszczyzn. Na każdej chcemy narysować *izotetyczny* prostokąt, tak żeby każde dwa prostokąty *widziały się*. Max $n =??$.

Dwa prostokąty *widzą się*, jeżeli możemy połączyć ich wnętrza odcinkiem prostopadłym do nich i niedotykającym żadnego z pozostałych prostokątów.

- znany jest przykład z $n = 22$

Zadanko numer 1

Widoczność prostokątów w 3D:
Mamy n równoległych płaszczyzn. Na każdej chcemy narysować *izotetyczny* prostokąt, tak żeby każde dwa prostokąty *widziały się*. Max $n =$??.

Dwa prostokąty *widzą się*, jeżeli możemy połączyć ich wnętrza odcinkiem prostopadłym do nich i niedotykającym żadnego z pozostałych prostokątów.

- znany jest przykład z $n = 22$
- wiadomo, że nie da się więcej niż 55

Zadanko numer 1

Widoczność prostokątów w 3D:
Mamy n równoległych płaszczyzn. Na każdej chcemy narysować *izotetyczny* prostokąt, tak żeby każde dwa prostokąty *widziały się*. Max $n =$??.

Dwa prostokąty *widzą się*, jeżeli możemy połączyć ich wnętrza odcinkiem prostopadłym do nich i niedotykającym żadnego z pozostałych prostokątów.

- znany jest przykład z $n = 22$
- wiadomo, że nie da się więcej niż 55
- (dla kwadratów jednostkowych maksymalne n to 7).

Zadanko numer 2

Udwudzielnianie:
Dany nieskierowany graf G, n wierzchołków, m krawędzi. Czy da się usunąć co najwyżej k wierzchołków, aby G stał się dwudzielny?

Zadanko numer 2

Udwudzielnianie:
Dany nieskierowany graf G, n wierzchołków, m krawędzi. Czy da się usunąć co najwyżej k wierzchołków, aby G stał się dwudzielny?

- znany algorytm $O(3^k \cdot poly(n + m))$

Zadanko numer 2

Udwudzielnianie:
Dany nieskierowany graf G, n wierzchołków, m krawędzi. Czy da się usunąć co najwyżej k wierzchołków, aby G stał się dwudzielny?

- znany algorytm $O(3^k \cdot poly(n+m))$
- czy da się szybciej, np. $O(2^k \cdot poly(n+m))$?

Zadanko numer 3

Szerokość grafu:
Dany nieskierowany graf $G = (V, E)$, $|V| = n$. Narysować go na prostej, wpisując wierzchołki w punkty $1..n$, tak aby zminimalizować długość najdłuższej krawędzi.

Zadanko numer 3

Szerokość grafu:
Dany nieskierowany graf $G = (V, E)$, $|V| = n$. Narysować go na prostej, wpisując wierzchołki w punkty $1 \ldots n$, tak aby zminimalizować długość najdłuższej krawędzi.

- zaczęło się od algorytmu $O^*(10^n)$

Zadanko numer 3

Szerokość grafu:
Dany nieskierowany graf $G = (V, E)$, $|V| = n$. Narysować go na prostej, wpisując wierzchołki w punkty $1..n$, tak aby zminimalizować długość najdłuższej krawędzi.

- zaczęło się od algorytmu $O^*(10^n)$
- aktualnie znamy algorytm $O^*(4.383^n)$

Zadanko numer 3

Szerokość grafu:
Dany nieskierowany graf $G = (V, E)$, $|V| = n$. Narysować go na prostej, wpisując wierzchołki w punkty $1..n$, tak aby zminimalizować długość najdłuższej krawędzi.

- zaczęło się od algorytmu $O^*(10^n)$
- aktualnie znamy algorytm $O^*(4.383^n)$
- a może da się szybciej, np. $O^*(3^n)$, $O^*(2^n)$?

Zadanko numer 4

Problem Eulera:
Znaleźć dodatnie liczby całkowite spełniające:

$$a_1^k + \ldots + a_n^k = b_1^k + \ldots + b_m^k$$

dla danych k, n, m.

Zadanko numer 4

Problem Eulera:
Znaleźć dodatnie liczby całkowite spełniające:

$$a_1^k + \ldots + a_n^k = b_1^k + \ldots + b_m^k$$

dla danych k, n, m.

- ludzie analizują rozmaite przypadki: `http://euler.free.fr`

Zadanko numer 4

Problem Eulera:
Znaleźć dodatnie liczby całkowite spełniające:

$$a_1^k + \ldots + a_n^k = b_1^k + \ldots + b_m^k$$

dla danych k, n, m.

- ludzie analizują rozmaite przypadki:
  `http://euler.free.fr`
- np. wciąż nie jest znane żadne rozwiązanie równania:
  $a_1^6 = b_1^6 + b_2^6 + b_3^6 + b_4^6 + b_5^6 + b_6^6$,
  choć dla $k = 6$, $n = 1$, $m = 7$ już owszem.

Zadanko numer 5

Wyszukiwanie antywzorca w tekście:
Czyli na jakich pozycjach tekstu t wzorzec w *nie* pasuje na *żadnej* literce?

Zadanko numer 5

Wyszukiwanie antywzorca w tekście:
Czyli na jakich pozycjach tekstu t wzorzec w *nie* pasuje na *żadnej* literce?

- jak to rozwiązać efektywnie?

Zadanko numer 5

Wyszukiwanie antywzorca w tekście:
Czyli na jakich pozycjach tekstu t wzorzec w *nie* pasuje na *żadnej* literce?

- jak to rozwiązać efektywnie?
- jak to rozwiązać, jeżeli alfabet jest *mały*?

Jeszcze więcej ciekawych zadanek?

Zapraszamy do udziału
w Potyczkach Algorytmicznych 2010!

runda próbna już trwa
runda 1 już od wtorku 20.04

Link: `http://www.konkurs.adb.pl`

Koniec

Pytania i/lub obiad