Opracowanie: ARS
Ars Longa

HISTORIA:
• v. 1.0: 2005.xx.xx, [SA] xxx

dokument systemu SINOL 1.5

1 Wstęp

Na finałach ACM czasem pojawiają się zadania, które mają coś wspólnego z fizyką. Nie wymagają one na ogół zbyt głębokiej wiedzy na ten temat, do ich rozwiązania wystarczają pojęcia fizyczne z poznane w starych dobrych czasach liceum. Tak jest i tym razem.

Zazwyczaj zadania z fizyki sprowadzały się do policzenia jednej lub kilku wielkości. Tutaj zadanie z pozoru wydaje się inne – każą nam tylko stwierdzić, czy konstrukcja stoi, czy stoi solidnie. W jaki sposób zabrać się za coś takiego?

Pewnie warto przyjrzeć się siłom działającym w danym układzie. Dla ustalenia uwagi, oznaczmy kolejne kulki symbolami $K_1, K_2, \ldots, K_n$. Oto trzy rodzaje sił, które będą nas interesować:

- siły ciężkości działające na wszystkie kulki

Rysunek 1: Na każdą kulkę działa siła grawitacji.

- siły sprężystości prętów; można je również nazwać siłami *przenoszonymi przez pręty*

Rysunek 2: Rozciągany pręt działa na kulki zgodnie z III zasadą dynamiki Newtona.

  Kierunki tych sił są zgodne z kierunkami prętów. Każdy pręt działa na kulki, do których jest przyczepiony, siłami o tej samej wartości, ale o przeciwnych zwrotach. Wprowadźmy w tym miejscu oznaczenie f_i, mówiące jaka jest wartość w Niutonach siły rozciągającej i-ty pręt. Oczywiście w naszych rozważaniach utożsamiamy długość wektora siły z jej wartością w niutonach.

- siły sprężystości stołu, działające tylko na kulki leżące na stole, równoważące wszystkie inne siły działające na te kulki

Rysunek 3: Siła sprężystości stołu. Zauważmy, że tutaj pręty 2 i 4 są ściskane, więc mamy $f_2, f_4 < 0$.

2 Podstawowe rozwiązanie

To co? Wiadomo o co chodzi? Wystarczy policzyć siły, sprawdzić, czy wypadkowe siły działające na kulki są zerowe i wiemy, czy stoi, czy nie? (Zostawmy póki co pytanie, jak stwierdzić, czy konstrukcja stoi stabilnie, czy nie, a skupmy sie na pytaniu, czy w ogóle stoi, czy nie).

Mniej więcej tak to będziemy robić, choć trochę sprytniej. Po pierwsze zauważmy, że układ sił przenoszonych przez pręty może nie być wyznaczony jednoznacznie. Na pierwszy rzut oka może się to wydać zaskakujące, w końcu albo na pręt działa taka a taka siła, albo nie. W fizyce klasycznej zazwyczaj nie ma miejsca na niejednoznaczności. Jednak w tym zadaniu mamy trochę wyidealizowany model, w którym pręty są nieściśliwe. Proponuję więc spojrzeć na poniższy rysunek i zastanowić sie, jakie tam siły mogą działać.

Rysunek 4: W tym układzie siły przenoszone przez pręty nie są wyznaczone jednoznacznie.

Drugim problemem jest to, że policzenie sił w ogólnym przypadku nie jest proste. Ale nie jest też konieczne. Zróbmy więc tak: *załóżmy, że układ stoi* i spróbujmy teraz policzyć siły przenoszone przez wszystkie pręty. Cały trik polega na tym, że owe założenie można przedstawić w postaci, którą matematycy lubią najbardziej – w postaci układu równań.

Ale zanim to zrobimy, dołóżmy parę nowych oznaczeń. Niech $R(K_i)$ oznacza zbiór numerów prętów połączonych z K_i. Ponadto niech $B(i, K_x)$ będzie numerem kulki, do której przeczepiony jest koniec i-tego pręta, ale nie ten, który jest przyczepiony do K_x.

Teraz dla każdej kulki zapiszmy fakt, że wypadkowa siła działająca na nią jest zerowa:

$$\begin{cases} \sum_{i \in R(K_1)} f_i \cdot \dfrac{\overrightarrow{K_1 K_{B(i,K_1)}}}{\left|\overrightarrow{K_1 K_{B(i,K_1)}}\right|} = -\vec{g} \\ \sum_{i \in R(K_2)} f_i \cdot \dfrac{\overrightarrow{K_2 K_{B(i,K_2)}}}{\left|\overrightarrow{K_2 K_{B(i,K_2)}}\right|} = -\vec{g} \\ \vdots \\ \sum_{i \in R(K_n)} f_i \cdot \dfrac{\overrightarrow{K_n K_{B(i,K_n)}}}{\left|\overrightarrow{K_n K_{B(i,K_n)}}\right|} = -\vec{g} \end{cases}$$

Zauważmy, że jest to układ równań wektorowych, tak więc jeśli rozpatrujemy go w przestrzeni trójwymiarowej, to każde równanie możemy rozpisać na trzy zwykłe równania skalarne – po trzech współrzędnych. Traktując wszystkie f_i jako niewiadome, otrzymujemy układ $3 \cdot n$ równań liniowych z m niewiadomymi.

Teraz korzystając standardowo z algorytmu eliminacji Gaussa, sprowadzamy rozszerzoną macierz układu do postaci schodkowej. Jeśli okaże się, że nasz układ jest sprzeczny, oznacza to, że założenie o tym, że układ stoi nie jest słuszne (ponieważ nie istnieje układ sił przenoszonych przez pręty opisujący stan równowagi). Zatem umiemy już stwierdić, czy konstrukcja stoi, czy nie. Pozostaje druga część – wiedząc, że układ stoi, trzeba stwierdzić, czy przyadkiem nie jest niestabilny.

Przeformułujmy więc to pytanie w terminach naszego układu równań. Pytamy, czy istnieją takie

małe wektorki $\vec{\alpha_1}, \ldots, \vec{\alpha_n}$, że układ

$$\begin{cases} \sum_{i \in R(K_1)} f_i \cdot \frac{\overrightarrow{K_1 K_{B(i,K_1)}}}{\left|\overrightarrow{K_1 K_{B(i,K_1)}}\right|} = -\vec{g} + \vec{\alpha_1} \\ \sum_{i \in R(K_2)} f_i \cdot \frac{\overrightarrow{K_2 K_{B(i,K_2)}}}{\left|\overrightarrow{K_2 K_{B(i,K_2)}}\right|} = -\vec{g} + \vec{\alpha_2} \\ \vdots \\ \sum_{i \in R(K_n)} f_i \cdot \frac{\overrightarrow{K_n K_{B(i,K_n)}}}{\left|\overrightarrow{K_n K_{B(i,K_n)}}\right|} = -\vec{g} + \vec{\alpha_3} \end{cases}$$

jest sprzeczny. Problem w tym, że nie wiemy dokładnie, jakie są wartości α_i. Oto dwa pomysły, jakie mogą nam tutaj przyjść z pomocą:

1. Można spróbować w sposób losowy wygenerować małe wektorki α_i, następnie za pomocą eliminacji Gaussa stwerdzić, czy otrzymany układ jest sprzeczny[1]. Jeśli jest, to mamy pewność, że konstrukcja stoi niestabilnie. Jeśli sprzeczności nie ma, to możemy dla pewności kilkukrotnie powtórzyć losowanie. Można zauważyć, że prawdopodobieństwo, iż w ten sposób nie dostaniemy poprawnej odpowiedzi jest bardzo, bardzo małe.

2. Ale może nie trzeba się uciekać do takich niedeterministycznych metod. Lepiej przyjrzeć się temu, co otrzymaliśmy w pierwszej faziw algorytmu. Tam sprowadzaliśmy macierz układu do postaci schodkowej. Jeśli nam się to udało, to w efekcie otrzymaliśmy macierz takiej postaci:

$$\left[\begin{array}{ccc|c} a_{1,1} & \cdots & a_{1,m} & b_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{k,m} & b_k \\ 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 \end{array}\right]$$

   Po prostu najpierw jest k niezerowych wierszy, a na końcu *być może* są wiersze zerowe. Jeśli teraz do wyjściowego układu równań dopisaliśmy z prawej współczynniki α_i, to jeśli teraz byśmy powtórnie sprowadzili macierz do postaci schodkowej, to wyglądałaby ona tak:

$$\left[\begin{array}{ccc|c} a_{1,1} & \cdots & a_{1,m} & b_1 + \Delta_1 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & a_{k,m} & b_k + \Delta_k \\ 0 & \cdots & 0 & \Delta_{k+1} \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \cdots & 0 & \Delta_{3n} \end{array}\right]$$

   gdzie, uwaga, Δ_i są niezerowymi kombinacjami liniowymi współrzędnych wektorów α_i. A zatem kiedy można tak dobrać α_i, żeby układ zepsuć? Na pewno nam się to nie uda, jeśli

[1] Tak naprawdę nie jest istotne, czy wygenerowane wektorki są „małe" czy nie. Mając jakiś układ dodatkowych sił destabilizujących układ, to jeśli wartość każdego wektorka pomnożymy przez stałą, to otrzymany układ też destabilizuje konstrukcję.

w postaci schodkowej naszej macierzy nie będzie zerowych wierszy. Natomiast jeśli jest choć jeden wiersz zerowy, to zawsze możemy tak dobrać wektory α_i, żeby wartość np. Δ_{k+1} była niezerowa, co już implikuje sprzeczność układu.

3 Dodatkowe ulepszenia

Przedstawiony powyżej algorytm jest poprawnym rozwiązaniem zadania. Jednak czasem rozwiązywanie problemu na zawodach nie sprowadza się tylko do opracowania poprawnego, czy nawet dodatkowo efektywnego algorytmu. Jeszcze trzeba go móc zaimplementować w taki sposób, żeby być pewnym, że jeśli nie ma błędów, to zostanie zaakceptowany.

Ja osobiście bałbym się implementować ten podstawowy algorytm głównie ze względu na możliwą dużą niestabilność numeryczną eliminacji Gaussa. Co więcej, wyobraźmy sobie, że nie miałnbym w gotowcach wersji lepszej numerycznie, z wybieraniem odpowiednich elementów dzielących. Co by było, gdyby się okazało, że dokładność jest niewystarczająca? Otrzymałbym pewnie od systemu odpowiedź `Wrong answer` i nie wiedziałbym, czy chodzi o dokładność, czy też o zwykły błąd w programie.

Dlatego chciałbym opracować takie rozwiązanie, w którym nie wykorzystuję wartości zmiennoprzecinkowych. Mogę np. założyć, że wartości na wejściu nie będą podane z dokładnością większą niż 20 miejsc po przecinku. To pozwala mi przechowywać wartości z wejścia jako liczby całkowite (tak intuicyjnie, po prostu mnożę to, co jest na wejściu przez 10^{20} i tyle). Wtedy współrzędne wszystkich wektorów występujących w pierwotnym układzie równań również będą całkowite. Pozostają póki co wartości w mianownikach, będące długościami wektorów, które mogą być w ogóle niewymierne. Ale na to też jest sposób. Równanie

$$\sum_{i \in R(K_1)} f_i \cdot \frac{\overrightarrow{K_1 K_{B(i,K_1)}}}{\left|\overrightarrow{K_1 K_{B(i,K_1)}}\right|} = -\vec{g}$$

możemy zastąpić innym

$$\sum_{i \in R(K_1)} g_i \cdot \overrightarrow{K_1 K_{B(i,K_1)}} = -\vec{g}$$

gdzie teraz $g_i = \frac{f_i}{\left|K_1 K_{B(i,K_1)}\right|}$ traktujemy jako niewiadomą.

Niestety to nadal nie koniec, gdyż sprowadzanie macierzy do postaci schodkowej nie jest wykonywalne w $\mathbf{Z}$, gdyż $\mathbf{Z}$ nie jest ciałem — nie zawsze da się wykonywać dzielenie. Lekarstwem na to jest prowadzenie wszystkich operacji modulo pewna duża liczba pierwsza p, np. $2^{31} - 1$. Jest to też pewnego rodzaju „oszustwo”, gdyż rozwiązywalność układu w $\mathbf{Z}_p$ nie implikuje rozwiązywalności w $\mathbf{Q}$, lecz na potrzeby zawodów, implikacja działa z dostatecznie dużym prawdopodobieństwem.

Przedstawiony algorytm, wraz z ulepszeniami, został zaimplementowany i można go znaleźć w materiałach dołączonych do tego opracowania.

4 Testy

TODO