



Rozwiązanie zadania F 828.

Siła oddziaływania między dwoma ładunkami nie może zależeć od wyboru jednostek. Pisząc zatem ogólnie, mamy

$$F = k \cdot \frac{Q_1 Q_2}{r^2}.$$

Oznaczając primami wielkości pisane w konwencji anglosaskiej (np. układ jednostek CGSE lub Gaussa), mamy

$$\frac{Q^2}{Q'^2} = \sqrt{\frac{k'}{k}}.$$

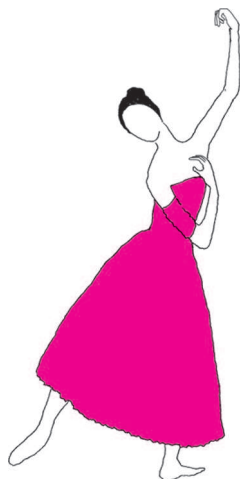
Stąd:

$$Q' = Q / \sqrt{4\pi\epsilon_0} = Q c \sqrt{\frac{\mu_0}{4\pi}},$$

gdzie $\mu_0 = 10^{-7} \text{N/A}^2$ jest przenikalnością magnetyczną próżni. Podstawiając wartości liczbowe dla ładunku elementarnego, otrzymujemy $e' \approx 1,5 \cdot 10^{-15} \text{m}\sqrt{\text{N}}$.

Zwróćmy uwagę, że kolejność przeglądania podzbiorów w pętli jest istotna, tzn. jeśli $S' \subset S$, to S' powinien zostać rozpatrzony przed S . Szczęśliwie się składa, że przy reprezentacji za pomocą masek bitowych przeglądanie w kolejności rosnących liczb (od 1 do $2^n - 1$) ma tę własność.

Element v można wyznaczyć, znajdując najmniejszy zapalony bit w masce bitowej. Przykładowo, w języku C++ (kompilatory GCC) służy do tego operacja `__builtin_ctz` (ang. *count trailing zeros*). Jeśli nie mamy dostępu do takiej operacji, możemy zrobić to tak: $m[S] = 1$, jeśli S jest nieparzyste, oraz $m[S] = 1 + m[S/2]$ w przeciwnym przypadku.



Informatyczny kącik olimpijski (60): Kliki

Dany jest graf nieskierowany $G = (V, E)$. *Kliką* nazwiemy taki podzbiór wierzchołków $S \subseteq V$, że każde dwa wierzchołki w zbiorze S są połączone krawędzią w grafie G . Problem znalezienia w grafie kliki o jak największej liczbie wierzchołków jest NP-zupełny, a jak wiadomo, dla takich problemów nikt jeszcze nie pokazał algorytmu wielomianowego. Podobne trudności są z algorytmem dla problemu *zliczania klik* w grafie, choć tutaj stosowną klasę złożoności stanowią tzw. problemy #P-zupełne.

Nic więc dziwnego, że gdy takie właśnie zadanie pojawiło się na obozie w Petrozawodzku w 2009 r. (zadanie pt. *Work for Robots*), nie oczekiwano od zawodników rozwiązania wielomianowego. Występujące w zadaniu ograniczenie $|V| \leq 50$ wymagało jednak zastosowania nietrywialnego algorytmu wykładniczego.

Niech $V = \{1, \dots, n\}$. Nasz algorytm będzie wykonywał różne operacje na podzbiórach zbioru V . Podzbiory takie będziemy reprezentowali jako maski bitowe o długości n (ponieważ $n \leq 50$, każda taka maska zmieści się w 64-bitowej zmiennej całkowitej). W pseudokodach będziemy zapisywać operacje sumy, iloczynu i różnicy zbiorów przy użyciu standardowej symboliki matematycznej, ale implementując program, można je równie prosto zapisać przy użyciu operacji bitowych (patrz *Delta* 11/2008). Niech $adj[v]$ oznacza zbiór wierzchołków, z którymi połączony jest wierzchołek v . Przez $cnt[S]$ oznaczamy liczbę klik, które są zawarte w podzbiórze S (wynikiem programu będzie $cnt[V]$). Dla $S = \emptyset$ mamy dokładnie jedną taką klikę (pustą). Załóżmy zatem, że $S = \{v\} \cup S'$ dla pewnego wierzchołka v . Każda klika zawarta w S jest albo całkowicie zawarta w S' , albo zawiera wierzchołek v i jest zawarta w $(S' \cap adj[v]) \cup \{v\}$. Poniższy pseudokod pokazuje, jak wypełniać tablicę cnt :

```
cnt[∅] := 1;
foreach niepusty podzbiór S ⊆ V w kolejności zawierania do
  v := dowolny element z S;
  S' := S \ {v};
  cnt[S] := cnt[S'] + cnt[S' ∩ adj[v]];
```

Powyższe rozwiązanie działa w czasie $O(2^n)$.

Szybsze rozwiązanie uzyskamy następująco. Podzielmy zbiór wierzchołków na dwa możliwie równe podzbiory $A = \{1, \dots, \lfloor n/2 \rfloor\}$, $B = \{\lfloor n/2 \rfloor + 1, \dots, n\}$. Na początek wykonajmy powyższy algorytm, ale tylko dla wierzchołków ze zbioru A ; zajmie to czas $O(2^{n/2})$. Zbiór $S \subseteq V$ jest kliką dokładnie wtedy, gdy oba zbiory $S \cap A$, $S \cap B$ są klikami oraz istnieją wszystkie krawędzie między nimi. Rozważmy pewien $S_B \subseteq B$, który jest kliką. Oznaczmy

$$neigh[S_B] = \bigcap_{v \in S_B} (adj[v] \cup \{v\}),$$

czyli zbiór tych wierzchołków, które są połączone ze wszystkimi wierzchołkami z S_B . Wówczas klika S , dla której $S \cap B = S_B$, musi spełniać $S \cap A \subseteq neigh[S_B] \cap A$. Ostatecznie wynikiem jest:

$$\sum_{neigh[S_B] \cap S_B = S_B} cnt[neigh[S_B] \cap A],$$

przy czym warunek sumy oznacza po prostu, że S_B jest kliką.

To, czego nam teraz potrzeba, to obliczenie $neigh[S_B]$ dla wszystkich $S_B \subseteq B$. Robimy to podobnie jak w przypadku tablicy cnt :

```
neigh[∅] := V;
foreach niepusty podzbiór S_B ⊆ B w kolejności zawierania do
  v := dowolny element z S_B;
  neigh[S_B] := neigh[S_B \ {v}] ∩ (adj[v] ∪ {v});
```

Całe rozwiązanie działa w czasie $O(2^{n/2})$. Jako zadanie dla Czytelnika proponujemy przerobić ten algorytm tak, by znajdował najliczniejszą klikę.

Tomasz IDZIASZEK