Rozwiązanie zadania M 1255.
Niech n będzie najmniejszą liczbą, dla której teza zadania nie jest spełniona. Przyjmijmy, że wówczas $a < 2^{n-1}$. Ponieważ liczby a, b, c, d są dodatnie, więc $n > 1$. Wobec tego prawa, a zatem i lewa strona danej równości jest podzielna przez 8. Ponieważ kwadrat liczby całkowitej daje z dzielenia przez 8 resztę 0, 1 lub 4, więc liczby a, b, c, d, spełniające daną zależność, muszą być parzyste, tzn. $a = 2a_1$, $b = 2b_1$, $c = 2c_1$, $d = 2d_1$. Stąd wniosek, że $a_1 < 2^{n-2}$ oraz

$$a_1^2 + b_1^2 + c_1^2 + d_1^2 = 7 \cdot 4^{n-1},$$

co oznacza, że teza zadania nie jest spełniona również dla liczby $n - 1$. Otrzymaliśmy sprzeczność.

Rozwiązanie zadania M 1256.
Przypuśćmy, że $a_1, a_2, \ldots, a_n < 2$ oraz niech $b_i = 2 - a_i$ $(i = 1, 2, \ldots, n)$. Wówczas $b_i > 0$ oraz

$$b_1 + b_2 + \ldots + b_n \leqslant n.$$

Ponadto

$$n^2 \leqslant \sum_{i=1}^{n} (2 - b_i)^2 =$$

$$= 4n - 4\sum_{i=1}^{n} b_i + \sum_{i=1}^{n} b_i^2 <$$

$$< 4n - 4\sum_{i=1}^{n} b_i + \Big(\sum_{i=1}^{n} b_i\Big)^2.$$

Niech $B = b_1 + b_2 + \ldots + b_n$. Wówczas $n(n-4) < B(B-4)$. Wiemy, że $n \geqslant 4$, więc liczba $B(B-4)$ jest dodatnia. Ponadto $B > 0$, więc $B - 4 > 0$.

Wiemy też, że $B \leqslant n$, więc $B - 4 > n - 4$, czyli $B > n$. Otrzymaliśmy sprzeczność.

Informatyczny kącik olimpijski (24): Barykady

W finale Potyczek Algorytmicznych 2007 zawodnicy mieli do rozwiązania, między innymi, zadanie *Barykady*. Mając dane drzewo (czyli spójny graf nieskierowany o n wierzchołkach i $n-1$ krawędziach), dla każdej z podanych liczb dodatnich k_i należało wyznaczyć minimalną liczbę krawędzi, które należy z tego drzewa usunąć, aby jedna z pozostałych po tym zabiegu spójnych części zawierała dokładnie k_i wierzchołków.

Rozwiązanie opiera się na programowaniu dynamicznym. Wybierzmy najpierw dowolny wierzchołek i uznajmy go za korzeń drzewa. Będziemy chcieli dla każdego poddrzewa wyznaczonego przez wierzchołek (czyli fragmentu złożonego z tego wierzchołka i wszystkich jego potomków) i każdej liczby całkowitej dodatniej x wyznaczyć minimalną liczbę krawędzi, taką że po usunięciu z tego poddrzewa tych krawędzi w poddrzewie zostaje spójny fragment zawierający dokładnie x wierzchołków, z których jednym jest korzeń tego poddrzewa.

Dla liści jest to bardzo łatwe: można to zrobić tylko dla $x = 1$, nie usuwając z poddrzewa (złożonego przecież tylko z samego liścia) żadnych krawędzi. Aby obliczyć wyniki dla pewnego węzła wewnętrznego v i wszystkich możliwych wartości x, należy najpierw uczynić to dla wszystkich jego synów v_1, v_2, …, v_m. Niech w_{v_i} oznacza tablicę wyników dla poddrzewa wierzchołka v_i, a s_i – tablicę dla wierzchołka v wraz z poddrzewami v_1, v_2, …, v_i. Wynikiem dla v jest wówczas tablica s_m. Łatwo zauważyć, że $s_1[x+1] = w_{v_1}[x]$ (gdyż przecinając te same krawędzie, mamy poddrzewo z jednym wierzchołkiem więcej) oraz $s_1[1] = 1$ (przecinamy krawędź łączącą v z v_1). Ponadto, dla $i \geqslant 2$:

$$(*) \qquad s_i[x] = \min\Big(\min_{j=1,2,\ldots,x-1}(s_{i-1}[j] + w_{v_i}[x-j]),\, s_{i-1}[x] + 1\Big).$$

Powyższa równość wynika z tego, że możemy z poddrzewa wierzchołka v_i wziąć pewną liczbę wierzchołków (konkretnie: $x - j$) i dołączyć je do stworzonego wcześniej poddrzewa (o j wierzchołkach) albo nie wziąć żadnych wierzchołków z tego poddrzewa, tj. przeciąć krawędź łączącą v z v_i.

Obliczywszy tablice wyników w_v dla wszystkich wierzchołków, otrzymujemy wynik dla k_i równy (r jest korzeniem drzewa):

$$\min\Big(w_r[k_i],\, \min_{v \in V \setminus \{r\}}(w_v[k_i] + 1)\Big).$$

W powyższym napisie dodatkowa jedynka w przypadku $v \neq r$ wzięła się stąd, że musimy przeciąć krawędź łączącą v z jego ojcem, czego nie uwzględnialiśmy przy liczeniu w_v.

Pozostaje jeszcze zastanowić się, jak szybko działa algorytm oparty na opisanych pomysłach. Wzór $(*)$ na elementy s_i daje bezpośrednią metodę liczenia tablic w_v. Widać, że algorytm ten ma złożoność $O(n^3)$, gdyż każde poddrzewo dołączamy raz, w czasie $O(n^2)$, do jakiegoś obliczonego wcześniej s_i. Udowodnimy teraz, że algorytm ten działa istotnie szybciej. Wielu finalistów Potyczek nie dostrzegło tego faktu i w związku z tym nawet nie próbowało implementować omawianego rozwiązania, myśląc, że będzie zbyt wolne. Rzeczoną lepiej oszacowaną złożonością algorytmu jest $O(n^2)$.

Oznaczmy przez $a(v)$ rozmiar poddrzewa o korzeniu w v. Udowodnię indukcyjnie po $a(v)$, że łączna liczba operacji wykonywanych w poddrzewie o korzeniu w v jest $O(a(v)^2)$. Dla liści ($a(v) = 1$) jest to niewątpliwie zdanie prawdziwe. Weźmy teraz węzeł v i jego synów v_1, v_2, …, v_m. W algorytmie najpierw obliczamy tablice wynikowe dla synów, co daje nam $O(a(v_1)^2 + a(v_2)^2 + \ldots + a(v_m)^2)$ operacji, zgodnie z założeniem indukcyjnym. Zauważmy, że w tablicy s_{i-1} jest $1 + a(v_1) + a(v_2) + \ldots + a(v_{i-1})$ elementów, a w tablicy w_i jest $a(v_i)$ elementów. Stąd utworzenie tablicy s_i z tablic s_{i-1} i w_i zgodnie ze wzorem $(*)$ zajmuje czas

$$O(\text{rozmiar}(s_{i-1}) \cdot \text{rozmiar}(w_i)) = O((1 + a(v_1) + a(v_2) + \ldots + a(v_{i-1})) \cdot a(v_i))$$

(dlaczego?). W takim razie wszystkie obliczenia w poddrzewie wierzchołka v zajmują czas

$$O\Big(a(v_1)^2 + \ldots + a(v_m)^2 + \sum_{1 \leqslant i < j \leqslant m} a(v_i)a(v_j) + \sum_{i=1}^{m} a(v_i)\Big) = O\Big(\Big(\sum_{i=1}^{m} a(v_i) + 1\Big)^2\Big)$$
$$= O(a(v)^2),$$

co dowodzi, że w całym drzewie wykonamy zaledwie $O(n^2)$ operacji.

Tomasz KULCZYŃSKI