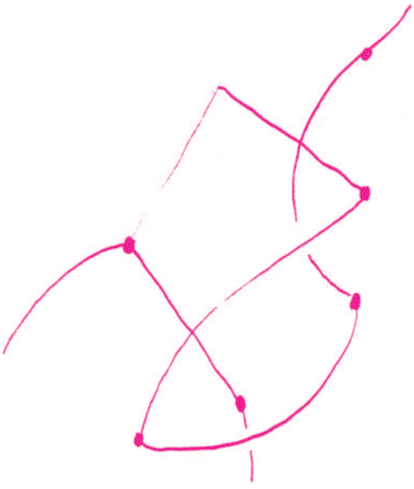
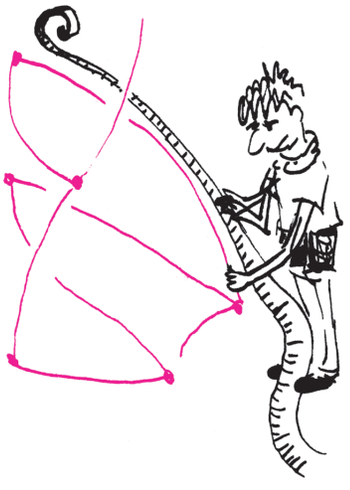


Jak osiągnąć poziom bezpieczeństwa 3
Początkowo traktujemy każdą grupę jako pojedynczą osobę i rozwiązujemy problem w starym modelu dla k osób i dla wymagania, że do odzyskania sekretu potrzebnych jest $\frac{k}{2}$ osób. Następnie uzyskane punkty traktujemy jako kolejne sekrety, które w obrębie grupy ponownie szyfrujemy za pomocą tej samej techniki, ale tym razem dla pięciu osób łącznie i trzech potrzebnych do odszyfrowania.



Informatyczny kącik olimpijski (38): Oszczędny kod

Tematem kolejnej edycji Kącika jest zadanie pochodzące z zeszłorocznej Międzynarodowej Olimpiady Informatycznej o nazwie *Oszczędny kod* (ang. *Saveit*).

Naszym celem jest zakodowanie, a później odkodowanie pewnych informacji. Musimy zaimplementować dwie części programu, z których pierwsza otrzyma pewien nieskierowany spójny graf oraz liczbę H. Jej zadaniem jest wygenerowanie ciągu bitów. Na podstawie tegoż ciągu, liczby N – liczby wierzchołków grafu – oraz liczby H, druga część programu powinna odkodować interesujące nas informacje, tzn. najkrótsze odległości pomiędzy pewnymi parami wierzchołków grafu. Wyjaśnijmy teraz, że wierzchołki grafu są ponumerowane od 0 do $N-1$, a interesują nas odległości wierzchołków o numerach od 0 do $H-1$ do wszystkich wierzchołków grafu. Kluczowe w zadaniu jest wygenerowanie możliwie krótkiego ciągu bitów, który pozwoli na odkodowanie potrzebnych informacji. Przyjmujemy, że numery wierzchołków (oraz inne liczby mniejsze niż N) możemy zapisać, używając B bitów (czyli w istocie $B = \lceil \log N \rceil$).

Najprostszym sposobem jest po prostu przesłanie całego grafu. Niewątpliwie jest to wystarczające do uzyskania wszelkich informacji na jego temat. Graf można przesłać jako liczbę krawędzi, a następnie ich listę. W przypadku, gdy graf jest pełny, daje to $(N^2 - N + 2)B$ bitów. Można też przesłać macierz sąsiedztwa, która, naturalnie, zawiera $N(N-1)/2$ istotnych bitów (mimo że jest ich łącznie $N(N-1)$, ale połowę z nich można ustalić, korzystając z symetryczności tej macierzy).

Kolejnym pomysłem jest najpierw obliczyć wszystkie interesujące nas odległości, a następnie zakodować je i przesłać do drugiej części programu. Interesujących nas liczb jest NH i każda zajmie B bitów, co da łącznie NHB bitów. Jest to dosyć satysfakcjonujące rozwiązanie i było na Olimpiadzie doceniane połową liczby punktów za to zadanie.

Istnieją jednak lepsze rozwiązania, opierające się na następujących spostrzeżeniach. Przede wszystkim zwróćmy uwagę na to, jaki jest związek odległości dwóch sąsiednich wierzchołków a i b od ustalonego wierzchołka c. Otóż, nie mogą one różnić się o więcej niż 1. Weźmy bowiem najkrótsze ścieżki z c do tych dwóch wierzchołków, długości $d(c,a)$ i $d(c,b)$, odpowiednio. Oczywiście, moglibyśmy z c pójść do a, a następnie przejść krawędzią z a do b, co dałoby odległość $d(c,a)+1$, a skoro długość najkrótszej ścieżki z c do b to $d(c,b)$, więc mamy $d(c,b) \leqslant d(c,a)+1$. Analogicznie w drugą stronę. Różnica między $d(c,a)$ i $d(c,b)$ należy więc do zbioru $\{-1,0,1\}$.

Aby skorzystać z powyższego faktu, musimy najpierw wiedzieć o jakichś parach wierzchołków, że są sąsiadami. W związku z tym spójrzmy na wierzchołek o numerze 0 i dla każdego innego wierzchołka x znajdźmy f_x – najbliższy od x wierzchołek na najkrótszej ścieżce z x do 0. Zauważmy, że pary (x, f_x) to krawędzie naszego grafu, które tworzą drzewo rozpinające. Na ich podstawie możemy łatwo wyznaczyć odległości wszystkich wierzchołków od 0 ze wzoru $d(0,x) = d(0,f_x)+1$. Zastanówmy się teraz, jak wyznaczyć odległości od wszystkich wierzchołków z wierzchołka $z \in \{1,2,\ldots,H-1\}$. Przede wszystkim znamy $d(0,z) = d(z,0)$. Ponadto, dla każdego $x \neq 0$ mamy: $d(z,x) = d(z,f_x) + r(z,x)$, przy czym wiemy już, że $r(z,x) \in \{-1,0,1\}$, gdyż x i f_x są sąsiadami. W takim razie wystarczy przesłać wszystkie liczby f_x oraz $r(z,x)$, które zajmują $B(N-1) + 2(H-1)(N-1)$ bitów. Dwójka oznacza tu dwa bity potrzebne na przesłanie liczby ze zbioru $\{-1,0,1\}$.

Ostatnie usprawnienie, wymagane na zawodach do uzyskania maksymalnej noty, polega na tym, aby zmniejszyć tę właśnie dwójkę do jakiejś mniejszej stałej. Widzimy bowiem, że nie wykorzystujemy w pełni tych dwóch bitów, skoro kodujemy na nich tylko trzy różne możliwości, zamiast dostępnych czterech. Dobrym pomysłem jest spojrzenie na pełen bajt (tzn. 8 bitów) i spostrzeżenie, że można w nim zakodować 256 różnych możliwości, a więc w szczególności $243 = 3^5$ różnych możliwości, czyli pięć wartości $r(z,x)$. Stąd dwójkę możemy mniej więcej zastąpić liczbą $\frac{8}{5}$ – liczba wykorzystanych bitów to wówczas $\lceil \log N \rceil (N-1) + 8\left\lceil \frac{(H-1)(N-1)}{5} \right\rceil$.

Zachęcamy Czytelnika do zastanowienia się, czy do rozwiązania zadania wystarczy użyć $\lceil (N-1)\log N \rceil + \lceil (H-1)(N-1)\log 3 \rceil$ bitów (zauważmy, że nie jest to wielka poprawka, gdyż $\log 3 \simeq 1{,}58496$). Warto także zastanowić się, jak szybko działają wszystkie opisane wyżej rozwiązania.

Tomasz KULCZYŃSKI