Rozwiązanie wzorcowe zadania F/2006

Autor: Tomasz Kazana

OPRACOWANIE ZADANIA

Rozwiązanie polega na zbadaniu pewnej liczby przypadków, wśród których na pewno znajduje się rozwiązanie optymalne.

Z treści zadania wiemy, że rozwiązanie optymalne składa się z dwóch ciągów o pewnym wspólnym prefiksie, tzn. wiemy, że rozwiązanie to ciągi AX i AY, gdzie A, X i Y są pewnymi rozłącznymi ciągami o szczególnych własnościach - element ciągu to pewna zębatka bądź para zębatek. W pierwszej fazie algorytmu generowane są wszystkie możliwe ciągi liczb 1 i 2 o sumie N (przypomnijmy: N to liczba zębatek). Teraz: dla ustalonego wzorca (przez wzorzec rozumiemy tu rozbicie liczby N na składniki 1 i 2, np. $6 = 2 + 2 + 1 + 1 = 1 + 2 + 1 + 2$), zakładamy że jest on postaci A'X'Y'Z' (Z oznacza nie wykorzystane zębatki), gdzie A',X',Y' i Z' stanowią wzorce odpowiadających im ciągów A,X,Y i Z. Podobnie: dla ustalonej permutacji zębatek π przyjmujemy, że konkatenacja ciągów A, X, Y i Z daje π. Ponieważ analizujemy wszystkie możliwe rozbicia wzorca na czwórki i wszystkie możliwe permutacje zębatek, więc optymalne rozwiązanie musi być rozważone.

Dla każdego badanego przypadku najpierw sprawdzamy, czy rzeczywiście ciąg AX generuje obroty odpowiednie dla wskazówki minutowej, a AY - dla godzinowej. Jeśli nie, rozważamy kolejny przypadek. Jeśli tak - porównujemy (wg podanych kryteriów) z najlepszym do tej pory wynikiem i zapamiętujemy lepsze z tych rozwiązań.

Na koniec programy wypisują wynik według podanego schematu. W szczególności w rozwiązaniu optymalnym na ostatniej wspólnej osi zębatek mogą być trzy zębatki - nie ma sprzeczności z opisaną wcześniej procedurą, ponieważ zakładamy, że ostatni element A, pierwszy element X, a także pierwszy element Y, mogą znajdować się na wspólnej osi.

ANALIZA ZŁOŻONOŚCI

Ograniczenia w zadaniu są bardzo małe ($N \leqslant 6$), więc ilość rozważanych przypadków jest nieduża, pomimo generowania wszystkich możliwych permutacji i wszystkich rozbić wszystkich wzorców. Zgrubnie szacując: co najwyżej 6! permutacji, co najwyżej 13 wzorców, co najwyżej $\binom{6}{3} = 20$ rozbić ustalonego wzorca, co daje co najwyżej 187200 przypadków. Każdy przypadek jest sprawdzany liniowo ze względu na N (wymaga przeliczenia obrotów kolejnych osi wg podanego wzoru). Dodatkowo, heurystyka opisana w uwagach implementacyjnych, pozwala w znakomitej większości sytuacji analizę danej permutacji i wzorca zakończyć po rozpatrzeniu tylko niewielkiej części rozbić.

UWAGI IMPLEMENTACYJNE

W implementacji korzystamy z trzech różnych funkcji pozwalających na znalezienie optymalnego rozwiązania (o ile ono istnieje) dla dowolnych danych wejściowych.
Wszystkie sytuacje gdy $Y = \epsilon$ rozwiązuje funkcja `probuj_liniowo(vector<string> &)`. Tutaj wykorzystujemy ważną heurystykę: jeśli nie znajdziemy (przy aktualnej permutacji i zadanym wzorcu) osi poruszającej się jak wskazówka godzinowa lub minutowa, przerywamy dalsze obliczenia związane z tą kolejnością zębatek i tym wzorcem (w takim przypadku oczywiście nie ma rozwiązań dla $Y \neq \epsilon$). Pozwala to wielokrotnie przyspieszyć działanie programu.

Następnie, o ile opisana wyżej heurystyka nie mówi inaczej, dla każdego innego rozbicia wywołujemy:
`probuj_luzno(vector<string> &, int, int, int)` oraz
`probuj_wcisnac(vector<string> &, int, int, int)`.
Pierwsza z tych funkcji sprawdza możliwość znalezienia rozwiązania, gdy każda zębatka napędza co najwyżej jedną zębatkę na innej osi. Druga próbuje zbudować zegar w taki sposób, aby pojedyncza zębatka napędzała dwie inne znajdujące się na różnych osiach.
Widać więc, że te trzy przypadki pokrywają całą przestrzeń poszukiwań, a odpowiednia kolejnoś sprawdzania daje możliwość pominięcia wielu zbędnych obliczeń.

Do generowania wszystkich permutacji w kodzie C++ korzystamy z funkcji bibliotecznej next_permutation, a w kodzie Java - z autorskiej implementacji tej funkcji.