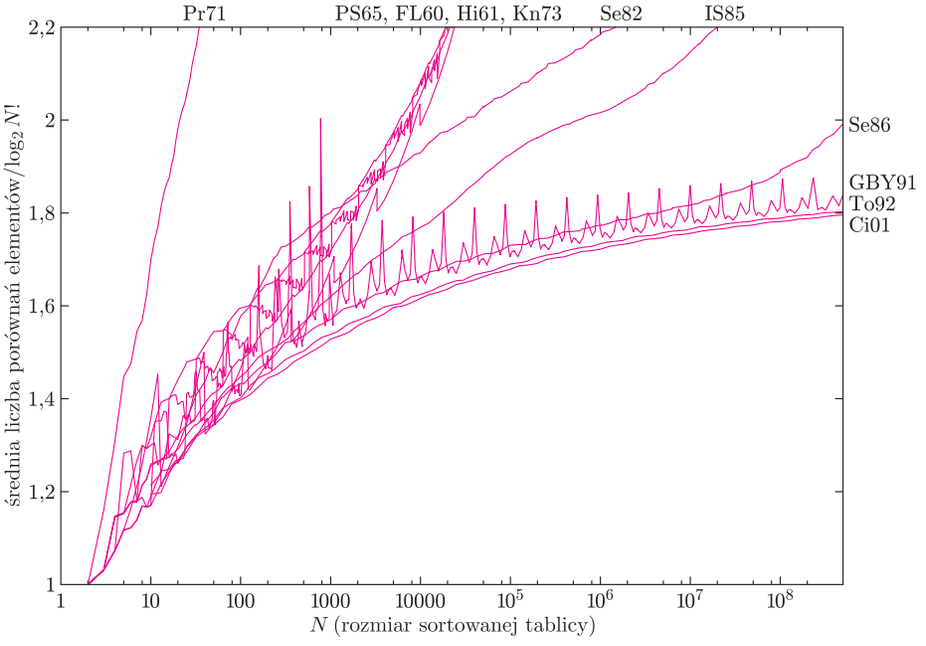
Poniższy wykres przedstawia średnią liczbę porównań elementów w różnych wariantach sortowania Shella, dzieloną przez teoretyczne minimum, czyli $\log_2 N!$.

Aby wykonać ten wykres, do ciągu 1, 4, 10, 23, 57, 132, 301, 701 dodano wyrazy $h_k = \left\lfloor 2\frac{17}{75} h_{k-1} \right\rfloor$.



Jak widać, stosowanie tak zoptymalizowanych ciągów odstępów daje znikomy zysk, ale ciekawą i niewyjaśnioną kwestią pozostaje, dlaczego ich wyrazy przyjmują takie, a nie inne wartości oraz czy można je określić jawnym wzorem.

Łamigłówki bitowe

Jak pamiętamy, komputery operują na bitach (przyjmujących wartości 0, 1), ale zwykle nie przetwarzają ich pojedynczo, lecz większymi grupami, np. słowami 8-, 16-, 32- czy 64-bitowymi. Procesory i języki programowania, poza standardowymi operacjami arytmetycznymi na liczbach całkowitych ($+$, $-$, $\cdot$ itp.), udostępniają też operacje bitowe. Działają one tak, jakby 0 było wartością logiczną *fałsz*, a 1 – *prawda*. Na przykład *AND* (koniunkcja) działa tak:

$$0\ AND\ 0 = 0\ AND\ 1 = 1\ AND\ 0 = 0, \quad 1\ AND\ 1 = 1.$$

Z kolei *OR* (alternatywa) daje wynik 1, gdy choć jeden z argumentów jest jedynką. Negacja *NOT* zamienia zera na jedynki i odwrotnie, zaś *XOR* (od *eXclusive OR* – alternatywa wykluczająca) daje wartość 1, gdy jej argumenty są *różne*:

$$0\ XOR\ 0 = 1\ XOR\ 1 = 0, \quad 0\ XOR\ 1 = 1\ XOR\ 0 = 1.$$

Operacje te stosuje się zwykle nie do pojedynczych bitów, ale do całych słów, bit po bicie. Jeśli, na przykład, pracujemy na liczbach ośmiobitowych, to:

$$\begin{aligned} 28\ XOR\ 11 &= 00011100\ XOR\ 00001011 \\ &= 00010111 = 23. \end{aligned}$$

Te wszystkie wyjaśnienia są po to, aby Czytelnik mógł przystąpić do zadań, w których przydadzą się zarówno operacje bitowe, jak i arytmetyczne.

rozwiązania w numerze

1. n-bitowa liczba całkowita może reprezentować podzbiór zbioru n-elementowego: i-ty bit jest równy 1, gdy i-ty element należy do podzbioru, a 0 w przeciwnym razie. Jak w tej reprezentacji obliczać część wspólną i sumę dwóch podzbiorów? A dopełnienie zbioru?

2. Sprawdź, że $(a\ XOR\ b)\ XOR\ b = a$ dla dowolnych liczb a i b (to niezwykle użyteczna własność!). Kiedy $a\ XOR\ b = 0$?

3. Mamy dwie zmienne a i b, których wartości chcemy zamienić miejscami. Zwykle napisalibyśmy

$$\texttt{tmp} := \texttt{a},\ \texttt{a} := \texttt{b},\ \texttt{b} := \texttt{tmp}.$$

A jak poradzić sobie bez pomocniczej zmiennej?

4. Autor Teorii Wszystkiego zapisał swoją teorię w pliku i chce go przekazać dwóm osobom tak, aby żadna z nich nie mogła poznać ani rąbka tajemnicy, ale aby obie razem mogły odtworzyć cały plik i poznać sekret. Jak to zrobić? Wskazówka: można zacząć od wygenerowania losowego ciągu bitów o długości takiej jak cały plik.

5. Napisz pojedynczą instrukcję, która sprawdzi, czy dana liczba a jest potęgą dwójki.

6. Napisz instrukcję obliczającą maksymalną potęgę dwójki, która dzieli daną liczbę a.

7. Jak wyznaczyć liczbę elementów zbioru, reprezentowanego jak w zadaniu 1, wykonując liczbę operacji proporcjonalną do tej liczby elementów?

Zebrali M.A. i F.W.