Trudność obliczeniowa

Problem obliczeniowy to odpowiedź na pytanie za pomocą odpowiednich obliczeń, na przykład „Czy 2017 jest liczbą pierwszą?" lub „Ile liter *i* znajduje się w słowie *niezrozumienie*?". Trudność obliczeniowa to własność problemów obliczeniowych, dla których nie ma algorytmu, który działałby w rozsądnym czasie. Takie problemy określa się też jako *problemy trudne obliczeniowo* (*infractable problems*), gdyż często są praktycznie niemożliwe do rozwiązania.

Zadziwiające jest, że trudność obliczeniowa jest niezależna od rodzaju urządzenia używanego do obliczeń – czy jest to procesor ogólnego zastosowania, układ scalony, czy też mechaniczna maszyna Turinga. W istocie jednym z pierwszych odkryć teorii złożoności obliczeniowej jest fakt, że wszystkie modele obliczeniowe są równoważne. Jeśli problem może zostać skutecznie rozwiązany za pomocą jednego urządzenia obliczeniowego, to może też zostać skutecznie rozwiązany za pomocą każdego innego urządzenia przez przeniesienie algorytmu na język innego urządzenia – wyjątek stanowią komputery kwantowe, ale one nie istnieją (jeszcze). Wynika stąd, że nie musimy określać wykorzystywanego urządzenia lub sprzętu, gdy omawiamy trudność obliczeniową; musimy tylko rozmawiać o algorytmach.

Aby wyznaczyć trudność, musimy najpierw znaleźć sposób zmierzenia złożoności algorytmu lub czasu jego wykonania. Podzielimy czasy wykonania na trudne i łatwe.

Pomiar czasu wykonania

Większość programistów zna *złożoność obliczeniową* (*computational complexity*), czyli przybliżoną liczbę działań wykonywaną przez algorytm w funkcji wielkości danych na wejściu. Wielkość jest liczona w bitach lub w liczbie elementów pobieranych na wejściu. Jako przykład weźmy algorytm pokazany na listingu 9.1, napisany w pseudokodzie.

Szuka on wartości *x*, w macierzy złożonej z *n* elementów i zwraca indeks jego położenia.

```
search(x, array, n):
    for i from 1 to n {
        if (array[i] == x) {
            return i;
         }
    }
    return 0;
}
```

Listing 9.1. Prosty algorytm wyszukiwania napisany w pseudokodzie o złożoności liniowej względem długości macierzy n. Algorytm zwraca indeks miejsca, w którym w macierzy [1,n] znajduje się x, lub 0, jeśli nie ma go w macierzy

W tym algorytmie używamy pętli do znalezienia określonej wartości *x*, iterując po macierzy. Przy każdej iteracji przypisujemy zmiennej *i* kolejną