Sortowanie rozrzutowe

Distribution Sort

Algorytm

Wszystkie opisane dotychczas algorytmy sortujące opierają się na sprawdzaniu uporządkowania zbioru, które polega na porównywaniu elementów. Udowodniono, iż barierą efektywności takich algorytmów w przypadku ogólnym (sortowanie zbioru o losowym rozkładzie elementów) jest klasa złożoności obliczeniowej $O(n \log n)$. Zachodzi zatem naturalne pytanie: czy istnieją inne sposoby sortowania o niższej klasie złożoności obliczeniowej?

Na samym początku III części "Sztuki Programowania Komputerów" prof. Donald Knuth opisuje znany od dawna sposób porządkowania talii kart.

Ustalmy kolejność kolorów kart wg ich starszeństwa (pierwszy pik, ostatni trefl):

♠	-	pik
♥	-	kier
♦	-	karo
♣	-	trefl

Teraz ustalmy kolejność figur (dla uproszczenia przyjmujemy talię z 24 kart, chociaż algorytm jest również poprawny dla pełnej talii 52 kart):

A	-	as
K	-	król
D	-	dama
W	-	walet
T	-	dziesiątka
9	-	dziewiątka

Przykład:

Załóżmy, iż mamy potasowaną losowo talię 24 kart:

♥	♦	♣	♥	♠	♣	♠	♥	♣	♦	♦	♠	♥	♠	♥	♣	♦	♥	♠	♣	♦	♦	♠	♣
D	K	K	9	D	W	A	K	9	T	A	K	T	W	A	D	D	W	9	A	9	W	T	T

Chcemy je posortować szybko tak, aby najpierw występowały piki, później kiery, kara a na końcu trefle. W obrębie każdego koloru karty powinny być uporządkowane wg podanej powyżej kolejności. Postępujemy tak:

Najpierw kolejne karty układamy na 6 stosów wg figur (kolorem się chwilowo nie przejmujemy):

A	K	D	W	T	9
♠A	♦K	♥D	♣W	♦T	♥9
♦A	♣K	♠D	♠W	♥T	♣9
♥A	♥K	♣D	♥W	♠T	♠9
♣A	♠K	♦D	♦W	♣T	♦9

Poszczególne stosy łączymy ze sobą w talie kart - karty pobieramy z kupek w tej samej kolejności, w której były na nie wstawiane (u nas wstawianie rozpoczęliśmy od góry, zatem również od góry rozbieramy każdą z kupek):

♠	♦	♥	♣	♦	♣	♥	♠	♥	♠	♣	♦	♣	♠	♥	♦	♦	♥	♠	♣	♥	♣	♠	♦
A	A	A	A	K	K	K	K	D	D	D	D	W	W	W	W	T	T	T	T	9	9	9	9

W drugim kroku otrzymaną talie rozkładamy na 4 stosy wg kolorów:

♠	♥	♦	♣
♠A	♥A	♦A	♣A
♠K	♥K	♦K	♣K
♠D	♥D	♦D	♣D
♠W	♥W	♦W	♣W
♠T	♥T	♦T	♣T
♠9	♥9	♦9	♣9

Teraz wystarczy połączyć ze sobą otrzymane stosy w jedną talię 24 kart:

♠	♠	♠	♠	♠	♠	♥	♥	♥	♥	♥	♥	♦	♦	♦	♦	♦	♦	♣	♣	♣	♣	♣	♣
A	K	D	W	T	9	A	K	D	W	T	9	A	K	D	W	T	9	A	K	D	W	T	9

Talia kart jest posortowana.

Lista kroków

K01: Przygotuj miejsce na tyle stosów, ile figur mogą mieć karty. Pierwszy stos będzie dla najstarszej karty, a ostatni dla najmłodszej.

K02: Rozdziel poszczególne karty na przygotowane wcześniej stosy wg figur - np. wszystkie asy idą do stosu asów, króle idą do stosu króli, itd.

K03: Złóż ze sobą karty w kolejnych stosach poczynając od stosu zawierającego najstarsze figury, a kończąc na stosie z najmłodszymi figurami.

K04: Przygotuj miejsce dla czterech stosów kolorów. Stosy powinny być ułożone w kolejności starszeństwa kolorów, np. piki, kiery, karo, trefle.

K05: Rozdziel karty na poszczególne stosy wg ich kolorów.

K06: Złóż ze sobą karty z kolejnych stosów poczynając od stosu zawierającego karty w najstarszym kolorze a kończąc na stosie z kartami w najmłodszym kolorze. Talia zostanie uporządkowana

K07: Zakończ algorytm

Zwróć uwagę, iż przedstawiona metoda w ogóle nie porównuje elementów ze sobą. Elementy trafiają do odpowiednich stosów (są rozrzucane - stąd nazwa sortowanie rozrzutowe) na podstawie swojej wartości, a nie na podstawie ich relacji z innymi elementami zbioru. Dzięki takiemu podejściu zmieniona zostaje klasa czasowej złożoności obliczeniowej:

Pierwsza operacja - rozłożenie n elementów na m kupek ma klasę złożoności $O(n)$.

Druga operacja - złączenie m kupek w n elementów ma klasę złożoności $O(m)$.

Sumarycznie cały algorytm będzie posiadał klasę złożoności $O(n + m)$. Jest to klasa liniowa, zatem sortowanie będzie bardzo szybkie. Co więcej, algorytm tego typu nie posiada przypadku pesymistycznego - czas sortowania każdego zbioru danych jest porównywalny. Mankament tego rozwiązania stanowi dodatkowe zapotrzebowanie na pamięć $O(n)$ - musimy zarezerwować komórki na elementy przechowywane w stosach.