Algorytmy sortujące

sortowanie kubełkowe, sortowanie grzebieniowe

Sortowanie kubełkowe (bucket sort)

- Jest to jeden z najbardziej popularnych algorytmów sortowania. Został wynaleziony w 1956 r. przez **E.J. Issaca i R.C. Singletona**.
- Algorytm działa w czasie liniowym **O(n)**.
- Algorytm najlepiej się sprawdza dla zbiorów zawierających dużą liczbę elementów, liczb całkowitych jednak o małym zakresie ich wartości.
- Zazwyczaj przyjmuje się, że sortowane liczby należą do przedziału od 0 do 1. Jeśli tak nie jest, to można podzielić każdą z nich, przez największą możliwą (jeśli znany jest przedział) lub wyznaczoną.
- Dla osiągnięcia optymalnej złożoności **liczba kubełków powinna być rzędu liczby elementów.**

Zasada działania algorytmu

1. Na początku określamy **zakres wartości** jakie mogą przyjmować elementy sortowanego zbioru.
2. Dla każdej liczby występującej w sortowanym zbiorze tworzymy **kubełek (licznik)**, do którego będziemy przyporządkowywać poszczególne elementy zbioru. **Kubełek zlicza nam ilość wystąpień konkretnego elementu w zbiorze sortowanym**. Na początku, każdy kubełek – licznik ma wartość zero.
3. Następnie rozpoczynamy przeglądanie sortowanego zbioru **od początku do końca**. Poszczególne elementy „wrzucamy” do kubełków oznaczonych daną cyfrą. W rezultacie kubełki będą nas informować o ilości wystąpień każdej z możliwych wartości w sortowanym zbiorze.
4. Ostatnim etapem jest spisanie do zbioru wynikowego liczb przypisanych do poszczególnych kubełków, tyle razy ile wynosi wartość licznika. W ten sposób zbiór wyjściowy będzie posortowany.

Przykład

Posortujemy algorytmem kubełkowym zbiór

[3 6 5 2 3 8 2 2 5 7 9 8 5 2 6].

W pierwszej kolejności określamy zakres wartości jakie mogą przyjmować elementy sortowanego zbioru. W naszym przypadku sortowane liczby nie należą do przedziału od 0 do 1, dlatego wyszukujemy w zbiorze element najmniejszy i największy – będzie to w_{min}=2 i w_{max}=9, . Teraz obliczmy liczbę potrzebnych kubełków.

$w_{max} - w_{min}+1 = 9 – 2 + 1 = 8$

Będziemy potrzebowali **8** kubełków (liczników). Dla każdego z nich przypiszemy wartość, którą będą zliczały i ustawimy wartość licznika na zero:

[2:0] [3:0] [4:0] [5:0] [6:0] [7:0] [8:0] [9:0]

Przykład

W następnym kroku przeglądamy kolejne elementy zbioru od początku do końca, zliczając ilość ich wystąpień i zapisując to w odpowiednich kubełkach:

[3 6 5 2 3 8 2 2 5 7 9 8 5 2 6]

[2:4] [3:2] [4:0] [5:3] [6:2] [7:1] [8:2] [9:1]

Podobnie jak w sortowaniu przez zliczanie, zapis [2:4] oznacza że, kubełek (licznik) liczby 2 zawiera 4 elementy – czyli liczba 2 występuje 4 razy w zbiorze. Jeżeli zbiór ma być posortowany rosnąco, rozpoczynamy przeglądanie kolejnych kubełków od tego o najmniejszym numerze, w przeciwnym wypadku posortujemy zbiór malejąco. W zbiorze wynikowym zapisujemy poszczególne liczby tyle razy ile wystąpiły w liczniku.
Zbiór jest posortowany:

[2 2 2 2 3 3 5 5 5 6 6 7 8 8 9]

Sortowanie grzebieniowe (combsort)

Sortowanie grzebieniowe należy do **algorytmów niestabilnych** - czyli kolejność wynikowa elementów równych jest nieokreślona (zwykle nie zostaje zachowana).

Algorytm został wynaleziony w 1980 roku przez **Włodzimierza Dobosiewicza**, wybitnego specjalistę techniki komputerowej.

W **1991** roku ponownie odkryta i opisana przez Stephena **Lacey'a i Richarda Boxa** metoda sortowania tablicowego.

Sortowanie grzebieniowe należy do metod o **złożoności liniowo-logarytmicznej.** Złożoności obliczeniowej algorytmu dotychczas nie udało się dowieść formalnie.

Sortowanie grzebieniowe – zasada działania

Ogólna zasada działania algorytmu opiera się, jak sama nazwa wskazuje na analogii czesania. Ze zbioru wyczesujemy najpierw duże elementy "z grubsza", podobnie jak najpierw czeszemy się grzebieniem o rzadszym rozmieszczeniu ząbków a dopiero później grzebieniem o ząbkach umieszczonych gęściej.

Sortowanie grzebieniowe dla **wariantu podstawowego:**

- za rozpiętość przyjmuje się długość tablicy, dzieli się rozpiętość przez 1.3, odrzuca część ułamkową
- bada się kolejno wszystkie pary obiektów odległych o rozpiętość (jeśli są ułożone niemonotonicznie - zamienia się je miejscami)
- wykonuje się powyższe w pętli dzieląc rozpiętość przez 1.3 do czasu, gdy rozpiętość osiągnie wartość 1

Gdy rozpiętość spadnie do 1 metoda zachowuje się tak jak **sortowanie bąbelkowe**. Tylko wtedy można określić, czy dane są już posortowane czy nie. W tym celu można użyć zmiennej typu **bool**, która jest ustawiana po zamianie elementów tablicy miejscami. Przerywane jest wykonywanie algorytmu, gdy podczas przejścia przez całą tablicę nie nastąpiła zamiana.

Przykład

Posortować tablicę 10 elementową algorytmem sortowania grzebieniowego:

[88 24 97 47 41 31 99 97 33 65]

Wielkość tablicy jest równa 10 - jest dzielona całkowitoliczbowo (bez reszty) przez 1.3 co daje (10/1.3)=7. W pierwszym przebiegu pętli porównywane są elementy oddalone od siebie o 7. Najpierw porównujemy element na pozycji 1 z elementem na pozycji 8. Podobnie jak w metodzie bąbelkowej zamieniamy je jeśli element na pozycji 8 jest mniejszy od elementu na pozycji 1 – w naszym przykładzie nie przestawiamy elementów.

88	24	97	47	41	31	99	97	33	65
1	2	3	4	5	6	7	8	9	10

Postępujemy tak aż porównamy element na pozycji 3 z elementem 10-tym - następuje zamiana miejsc 3 z 10.

88	24	65	47	41	31	99	97	33	97
1	2	3	4	5	6	7	8	9	10

Przykład – c.d

Następnie ponownie dzielimy aktualną odległość (7 dla 10-elementowej tablicy) przez 1.3. Otrzymujemy odległość równą **5**. Porównujemy elementy 1 z 6 i przestawiamy jeśli 1-ka jest większa od 6-ki, itd. aż do porównania elementu 5-go z 10-tym.

88	24	65	47	41	31	99	97	33	97
1	2	3	4	5	6	7	8	9	10

Element z pozycji 1 jest większy od elementu z pozycji 6 – zamieniamy je miejscami, podobnie postępujemy z elementami 4 i 9.

31	24	65	33	41	88	99	97	47	97
1	2	3	4	5	6	7	8	9	10

Przykład – c.d

Ponownie ustalamy aktualną odległość pomiędzy porównywanymi elementami - ustalamy go na 3. Element z pozycji 6 jest większy od elementu z pozycji 9 więc je zamieniamy miejscami

31	24	65	33	41	88	99	97	47	97
1	2	3	4	5	6	7	8	9	10

Element z pozycji 7 jest większy od elementu z pozycji 10 więc zamieniamy je miejscami

31	24	65	33	41	47	97	97	88	99
1	2	3	4	5	6	7	8	9	10

Przykład – c.d

Odległość pomiędzy porównywanymi elementami ustalamy na 2

31	24	65	33	41	47	97	97	88	99
1	2	3	4	5	6	7	8	9	10

Ponownie porównujemy elementy zbioru. Element z pozycji 3 jest większy od elementu na pozycji 5 dlatego zamieniamy je miejscami.
Podobnie z elementami nr 7 i 9 – je także zamieniamy miejscami. Zbiór po przestawieniach będzie przedstawiał się następująco:

31	24	41	33	65	47	88	97	97	99
1	2	3	4	5	6	7	8	9	10

Przykład – c.d

Odległość pomiędzy porównywanymi elementami ustalamy na 1. Algorytm przekształca się w ten sposób w algorytm bąbelkowy, lecz istnieje dodatkowy warunek zatrzymania go (brak inwersji, przestawień elementów), który statystycznie pozwala na zakończenie algorytmu szybciej niż algorytm bąbelkowy. Globalnym warunkiem stop jest koniunkcja warunku braku inwersji i odległości równej 1, co zapewnia właściwy porządek w tablicy wynikowej.

31	**24**	**41**	**33**	**65**	**47**	**88**	**97**	**97**	**99**
1	2	3	4	5	6	7	8	9	10

Ponownie porównujemy elementy zbioru. Element z pozycji 1 jest większy od elementu na pozycji 2 dlatego zamieniamy je miejscami.
Podobnie z elementami nr 3 i 4 oraz 5 i 6 – je także zamieniamy miejscami. Zbiór po przestawieniach będzie przedstawiał się następująco:

24	**31**	**33**	**41**	**47**	**65**	**88**	**97**	**97**	**99**
1	2	3	4	5	6	7	8	9	10

Odległość pomiędzy porównywanymi elementami ustalona na 1. Odległość między poszczególnymi elementami wynosi 1 i żaden nie został przestawiony co oznacza, że tablica jest posortowana

Bibliografia

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, *"Wprowadzenie do algorytmów"*, WNT 2001
- W. Dobosiewicz, An efficient variation of bubble sort , Information processing letters 11(1):5-6, 1980
- Sysło M.: *Algorytmy,* WSiP, 1997
- Wróblewski P.: Algorytmy, struktury danych i techniki programowania, Wyd. Helion, 2003
- www.pl.wikipedia.org
- www.algorytm.org
- www.encyklopedia.pwn.pl
- www.edu.i-lo.tarnow.pl