

ELŻBIETA TRYBUŚ (Wrocław)

## Metody porządkowania

### 1. Sformułowanie zagadnienia porządkowania

Zakładamy, że dany jest ciąg

$$(1) \quad x_1, x_2, \dots, x_n,$$

złożony z elementów pewnego skończonego zbioru  $X$ , oraz relacja  $R$  liniowo porządkująca zbiór  $X$ .

Mówimy, że relacja  $R$  *porządkuje* zbiór  $X$  jeśli jest (por. [12], str. 112) zwrotna, przechodnia i antysymetryczna, czyli gdy spełnia następujące warunki:

$$(z) \quad (\forall x \in X) (x R x),$$

$$(p) \quad (\forall x, y, z \in X) (x R y \wedge y R z \Rightarrow x R z),$$

$$(a) \quad (\forall x, y \in X) (x R y \wedge y R x \Rightarrow x = y).$$

Relację porządkującą  $R$  spełniającą w zbiorze  $X$  warunek spójności:

$$(s) \quad (\forall x, y \in X) (x \neq y \Rightarrow x R y \vee y R x),$$

nazywamy relacją *liniowo porządkującą* zbiór  $X$ . Parę uporządkowaną  $\langle X, R \rangle$ , gdzie  $R$  jest relacją liniowo porządkującą, nazywamy *zbiorem liniowo uporządkowanym* lub *łańcuchem*.

Często będziemy używać następujących oznaczeń:

(a) jeżeli  $R$  jest relacją porządkującą, to zamiast  $x R y$  piszemy  $x \rightarrow y$ ,

(b) jeżeli  $R$  jest relacją liniowo porządkującą, to zamiast  $x R y$  piszemy  $x \preceq y$ .

Element  $x_m$  nazywamy elementem *pierwszym* (lub *minimalnym* ze względu na relację  $R$ ) zbioru liniowo uporządkowanego  $\langle X, R \rangle$ , jeżeli  $x_m R y$  dla wszystkich  $y \in X$ .

Podobnie, element  $x_M$  nazywamy *ostatnim* (lub *maksymalnym* ze względu na relację  $R$ ) zbioru liniowo uporządkowanego  $\langle X, R \rangle$ , jeżeli  $y R x_M$  dla wszystkich  $y \in X$ .

Często w algorytmach będziemy używać następujących zapisów:

$$x_m = \min_R (x_1, x_2, \dots, x_n), \quad x_M = \max_R (x_1, x_2, \dots, x_n).$$

Oznaczmy przez  $\pi = (p_1, p_2, \dots, p_n)$  dowolną  $n$ -elementową permutację liczb całkowitych  $1, 2, \dots, n$ .

Zagadnienie porządkowania elementów zbioru  $X$  polega na wybraniu takiej permutacji  $\pi$ , aby elementy ciągu (1) utworzyły nowy ciąg, którego każdy wyraz pozostaje w relacji  $R$  do wyrazu następnego. Ciąg ten zapisujemy

$$(2) \quad x_{p_1} \preceq x_{p_2} \preceq \dots \preceq x_{p_n}$$

i nazywamy *ciągami uporządkowanym* przez relację  $R$  lub *posortowanym*. Operację, w wyniku której z ciągu (1) otrzymujemy ciąg (2) nazywamy *operacją porządkowania*. Istnieje wiele różnych metod realizacji operacji porządkowania, które nazywamy *metodami porządkowania*.

Niniejszy artykuł zawiera przegląd algorytmów porządkowania w pamięciach operacyjnych o dostępie bezpośrednim. Metody te podzielono na kilka grup ze względu na sposób realizacji operacji porządkowania, mianowicie:

1. porządkowanie za pomocą dołączania nowych elementów,
2. porządkowanie za pomocą zamiany sąsiednich elementów,
3. porządkowanie za pomocą łączenia,
4. porządkowanie za pomocą podziału,
5. porządkowanie za pomocą podziału z wyborem elementów minimalnych.

W prezentowanych algorytmach zakładamy więc, że ciąg pierwotny  $x_1, x_2, \dots, x_n$  umieszczony jest w pamięci operacyjnej maszyny cyfrowej, czyli zajmuje  $n$  kolejnych pozycji pamięci. Oprócz tej liczby podaje się w każdym algorytmie liczbę pomocniczych, zwanych czasem roboczymi, pozycji pamięci bądź rejestrów potrzebnych do zrealizowania danego algorytmu.

Dla zwiększenia czytelności algorytmów będziemy oznaczać wartości zmiennych tymi samymi symbolami, którymi oznaczaliśmy zmienne. Przy czym czynność obliczenia nowej wartości zmiennej, czyli czynność podstawiania opisujemy (tak jak w języku ALGOL 60) za pomocą symbolu „:=”. Np.  $k := k + 1$  oznacza, że za nową wartość zmiennej  $k$  należy podstawić poprzednią wartość powiększoną o 1. Podobnie operację zamiany dwóch zmiennych oznaczamy symbolem „ $\leftrightarrow$ ”, gdzie zapis „ $x \leftrightarrow y$ ” oznacza, że po wykonaniu zamiany zmienna  $x$  przyjmuje wartość zmiennej  $y$ , a zmienna  $y$  wartość zmiennej  $x$ . W większości maszyn cyfrowych dla zrealizowania operacji zamiany należy użyć zmiennej pomocniczej, którą oznaczmy np. przez  $u$  oraz wykonać w podanej kolejności trzy następujące podstawienia:

$$u := x, \quad x := y, \quad y := u.$$

Niemal we wszystkich metodach porządkowania wykonuje się sprawdzanie warunku  $x_i R y_j$  dla pewnych wartości  $i, j$  ( $i \neq j$ ), którą to operację nazywać będziemy operacją *porównania*, a dokładną liczbę wykonywanych porównań dla ciągu  $n$ -elementowego oznaczać będziemy symbolem  $P(n)$ . W przypadku, gdy obliczamy minimalną, maksymalną, lub średnią liczbę porównań używać będziemy symboli  $P_{\min}(n)$ ,  $P_{\max}(n)$ , lub  $P_{\text{sr}}(n)$ .

W większości metod, gdy warunek  $x_i R x_j$  nie jest spełniony wykonuje się operację *zamiany*. Dokładną liczbę wykonywanych zamian, potrzebnych do uporządkowania ciągu  $n$ -elementowego, oznaczać będziemy symbolem  $Z(n)$ . W metodach, w których nie wykonuje się zamiany elementów, stosuje się operację *przestawienia* (przemieszczenia) elementów z jednej pozycji pamięci maszyny cyfrowej na inną, która nie jest zajęta, bądź została specjalnie zwolniona. Liczbę wykonywanych przestawień oznaczać będziemy symbolem  $Q(n)$ .

O efektywności danej metody porządkowania decyduje przede wszystkim liczba wyko-

nywanych porównań, a w niektórych metodach także liczba wykonywanych zamian lub przestawień. W niniejszym opracowaniu przeprowadzono analizę (maksymalnej i minimalnej bądź średniej) liczby porównań oraz zamian, podając teoretyczne oszacowania tych wielkości. W paragrafie 7 przedstawiono wyniki empiryczne, uzyskane za pomocą testowania algorytmów porządkowania w języku ALGOL 60 na maszynie cyfrowej ODRA 1204.

Zamieszczone tutaj metody porządkowania dotyczą, jak już zaznaczaliśmy, porządkowania w pamięci operacyjnej maszyny cyfrowej. Jedynie metodę łączenia von Neumanna można również zastosować do porządkowania na taśmach magnetycznych. Metody porządkowania w różnych rodzajach pamięci zewnętrznej maszyny cyfrowej wymagają oddzielnych omówień, gdyż związane są najczęściej z określonym rodzajem tej pamięci. Wiele uwagi poświęcono tym zagadnieniom w pracach [1], [3], [17].

## 2. Metody porządkowania za pomocą dołączania nowych elementów

Metody porządkowania, które zaliczyliśmy do tej grupy charakteryzują się tym, że do uporządkowanego  $k$ -elementowego podciągu

$$(3) \quad x_1 \preceq x_2 \preceq \dots \preceq x_k$$

dołącza się element  $x$  nie należący do ciągu (3) tak, aby otrzymany ciąg  $(k+1)$ -elementowy był ciągiem uporządkowanym przez relację  $\mathcal{R}$ , czyli by był postaci:

$$(4) \quad x_{p_1} \preceq x_{p_2} \preceq \dots \preceq x_{p_{k+1}}.$$

Rozróżniamy kilka zasadniczych metod w tej grupie:

1. porządkowanie za pomocą wyboru lub selekcji,
2. porządkowanie za pomocą znajdowania miejsca elementu w podciągu uporządkowanym,
3. porządkowanie za pomocą przestawiania elementów.

Niniejszy paragraf poświęcamy omówieniu niektórych z tych metod.

**2.1. Porządkowanie za pomocą wyboru lub selekcji.** Do tej grupy metod porządkowania należą te, w których korzysta się z operacji wyboru lub selekcji. W wyniku operacji selekcji zastosowanej do zbioru  $X$  otrzymujemy pewien jego podzbiór lub w przypadku operacji wyboru pewien jego element. Operację selekcji oznaczamy symbolem  $S_W$ , gdzie  $W$  oznacza warunek, który spełniają elementy wybrane. Podamy dla przykładu kilka warunków:  $W_1$  — być pierwszym (minimalnym) elementem zbioru,

$W_2$  — być  $k$ -tym wyrazem ciągu uporządkowanego,

$W_3$  — być pracownikiem o stażu powyżej 5 lat, gdy zbiór  $X$  stanowią karty personalne pracowników pewnego przedsiębiorstwa, itp.

Należy podkreślić, że operację wyboru stosujemy do danego ciągu (1), natomiast wybrane elementy dołączane są kolejno do podciągu już uporządkowanego. Przedstawimy dwie metody porządkowania za pomocą operacji wyboru. Jedną z nich jest porządkowanie za pomocą wyboru elementów minimalnych, a drugą porządkowanie za pomocą wyboru elementów maksymalnych, oparte na tej samej idei, a różniące się jedynie sposobem dołączania nowych elementów.

W metodzie porządkowania za pomocą wybierania elementów minimalnych ze zbioru  $X$ , który na ogół jest dany w postaci ciągu (1), wybiera się element minimalny ze względu na relację  $\mathcal{R}$ . Element ten stanowi pierwszy wyraz ciągu uporządkowanego

$$(5) \quad x_{p_1} \preceq x_{p_2} \preceq \dots \preceq x_{p_n}.$$

Wprowadzając oznaczenia

$$Y_1 = \{x_{p_1}\}, \quad \text{gdzie} \quad x_{p_1} = \min_{\mathcal{R}}(x_1, \dots, x_n),$$

następne wyrazy ciągu (5) otrzymamy realizując algorytm:

$$\text{m1. } x_{p_k} = \min_{\mathcal{R}}(X - Y_{k-1}) \text{ dla } k = 2, \dots, n,$$

$$\text{m2. } Y_k = Y_{k-1} \cup \{x_{p_k}\}.$$

Gdy  $k$  osiągnie wartość  $n$  otrzymujemy uporządkowany ciąg (5). Gdy wybieramy elementy największe ciągu, wówczas otrzymujemy metodę porządkowania za pomocą wyboru elementów maksymalnych, której algorytm jest następujący:

Oznaczmy:

$$Y_n = \{x_{p_n}\}, \quad \text{gdzie} \quad x_{p_n} = \max_{\mathcal{R}}(x_1, \dots, x_n).$$

Realizując punkty:

$$\text{M1. } x_{p_k} = \max_{\mathcal{R}}(X - Y_{k+1}),$$

$$\text{M2. } Y_k = Y_{k+1} \cup \{x_{p_k}\} \text{ dla } k = n-1, n-2, \dots, 1$$

otrzymamy ciąg (5) uporządkowany przez relację  $\mathcal{R}$ .

Ponieważ dla wybrania elementu minimalnego (lub maksymalnego) z ciągu  $k$ -elementowego trzeba wykonać  $k-1$  porównań, stąd dla uporządkowania przedstawioną metodą ciągu  $n$ -elementowego należy wykonać

$$P(n) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}$$

porównań.

Jeżeli na wyrazy ciągu uporządkowanego (5) nie możemy zarezerwować  $n$  dodatkowych pozycji pamięci maszyny, a dysponujemy jedynie jedną pozycją roboczą, wówczas po wybraniu elementu minimalnego (maksymalnego) należy dokonać zamiany elementów  $x_1 \leftrightarrow x_{p_1}$  (lub  $x_n \leftrightarrow x_{p_n}$ ). Aby więc uporządkować ciąg tą metodą wykonuje się  $n-1$  zamian.

Algorytm porządkowania metodą wybierania elementów minimalnych znany jest w literaturze pod nazwą selekcji liniowej [17] lub „linear selection” [5].

**2.2. Metody porządkowania za pomocą wyznaczania miejsca elementu w podciągu uporządkowanym.** Cechą wspólną tej grupy metod porządkowania jest również dołączanie nowych elementów do uporządkowanego podciągu  $k$ -elementowego, przy czym istotny jest sposób wyznaczania miejsca dołączonego elementu. Wyznacza się je za pomocą metody liniowej albo za pomocą podziału dychotomicznego. Przedstawimy krótko ideę dołączania nowego elementu do uporządkowanego podciągu  $k$ -elementowego. Aby otrzymać ciąg (4) należy określić wskaźnik  $j$  elementu  $x$  w ciągu (3) za pomocą wzoru:



$$(6) \quad j = \begin{cases} 1 & \text{dla } x \prec x_1, \\ s + 1 & \text{dla } x_s \preceq x \prec x_{s+1}, \text{ gdzie } 1 \leq s < k, \\ k + 1 & \text{dla } x_k \preceq x \end{cases}$$

oraz dokonać następujących podstawień:

$$\begin{aligned} x_{p_1} &:= x_1, \\ x_{p_2} &:= x_2, \\ &\dots\dots\dots \\ x_{p_j} &:= x, \\ x_{p_{j+1}} &:= x_j, \\ &\dots\dots\dots \\ x_{p_k} &:= x_{k-1}, \\ x_{p_{k+1}} &:= x_k. \end{aligned}$$

**2.2.1. Metoda liniowa.** Metoda liniowa polega na porównywaniu elementu  $x$  z kolejnymi wyrazami ciągu (3). Minimalna oraz maksymalna liczba porównań, potrzebnych do uporządkowania ciągu  $n$ -elementowego, wyrażają się wzorami:

$$P_{\min}(n) = n - 1, \quad P_{\max}(n) = \sum_{k=1}^{n-1} k = \frac{n(n-1)}{2}.$$

Zatem średnio (w sensie średniej arytmetycznej z minimalnej oraz maksymalnej liczby porównań) wykonuje się:

$$P_{\text{sr}}(n) = \frac{(n-1)(n+1)}{4}$$

porównań.

**2.2.2. Metoda podziału dychotomicznego.** Wartość wskaźnika  $j$  można określić metodą podziału dychotomicznego, która polega na porównaniu elementu  $x$  z wyrazem środkowym ciągu (3), tj. z wyrazem o numerze  $l$  danym wzorem

$$l = \left\lfloor \frac{1+k}{2} \right\rfloor,$$

gdzie  $\lfloor y \rfloor$  oznacza część całkowitą liczby  $y$ .

Jeżeli  $x \prec x_l$ , to element  $x$  porównujemy z wyrazem środkowym pierwszej części ciągu, tj. ciągu  $x_1, x_2, \dots, x_{l-1}$ , a w przeciwnym razie z wyrazem środkowym drugiej części ciągu.

Proces ten kontynuujemy tak długo, aż otrzymamy w wyniku dzielenia podciąg jednoelementowy. Algorytm wyznaczania miejsca elementu w ciągu uporządkowanym metodą podziału dychotomicznego jest następujący:

1. Niech  $i_1 = 1$ ,  $m_1 = k$ .

2. Obliczamy  $l_t = \left\lfloor \frac{i_t + m_t}{2} \right\rfloor$ ,  $t = 1$ ;

2a. jeżeli  $x_{l_t} \leq x$ , to  $i_{t+1} = l_t + 1$  oraz  $m_{t+1} = m_t$  i przechodzimy do punktu 3;

2b. jeżeli  $\neg(x_{l_t} \leq x)$ , to  $m_{t+1} = l_t$  oraz  $i_{t+1} = i_t$  i przechodzimy do punktu 3.

3. Realizujemy punkt 2 dla kolejnych wartości  $t = 2, 3, \dots$  tak długo, dopóki  $i_{t+1} < m_{t+1}$ .

Gdy  $i_{t+1} \geq m_{t+1}$  miejsce elementu  $x$  jest określone, tzn.

$$j = i_{t+1}.$$

Liczba porównań  $P(k)$  potrzebnych do określenia miejsca elementu  $x$  w ciągu uporządkowanym (3) metodą podziału dychotomicznego spełnia nierówność

$$(7) \quad [\log_2 k] \leq P(k) \leq 1 + [\log_2 k].$$

Można udowodnić (por. [16]), że maksymalna oraz minimalna liczba porównań potrzebnych do uporządkowania ciągu  $n$ -elementowego metodą podziału dychotomicznego dane są wzorami:

$$P_{\max}(n) = 1 + nl - 2^l, \quad \text{gdzie} \quad l = 1 + [\log_2(n - 1)],$$

$$P_{\min}(n) = P_{\max}(n) + [\log_2 n] - n + 1.$$

Są to wzory na minimalną oraz maksymalną liczbę porównań. W praktyce są one dość nie wygodne, stąd też w pracy [8] podano, korzystając z własności logarytmów i wzoru Stirlinga, następujące oszacowanie średniej liczby porównań

$$P_{\text{sr}}(n) = n \log_2 \left( \frac{n}{e} \right).$$

Zarówno w metodzie liniowej jak też w metodzie podziału dychotomicznego po określeniu miejsca dołączonego elementu  $x$  wyrazy  $x_i$  spełniające warunek  $x \prec x_i$  przesuwają się o jedną pozycję w prawo, czyli przenumerowuje się. Liczba przestawień elementów wyraża się wzorami:

$$Q_{\min}(n) = 0 \quad \text{oraz} \quad Q_{\max}(n) = \frac{n(n-1)}{2}.$$

Stąd średnia liczba przestawień wyraża się wzorem

$$Q_{\text{sr}}(n) = \frac{n(n-1)}{4}.$$

W literaturze światowej metoda podziału dychotomicznego znana jest pod nazwą „binary insertion” [5].

**2.3. Porządkowanie metodą przestawiania elementów.** Zakładamy, że dany jest podciąg (3), uporządkowany przez relację  $\mathcal{R}$ , oraz dowolny element  $x \in X$ . Należy utworzyć  $(k+1)$ -elementowy ciąg uporządkowany również przez relację  $\mathcal{R}$ , czyli ciąg (4). Algorytm porządkowania metodą przestawiania elementów jest następujący:

Przyjmijmy, że  $i = k$ .

1. Jeżeli  $x_i \preccurlyeq x$ , to otrzymujemy ciąg (4) uporządkowany przez relację  $\mathcal{R}$ .
2. Jeżeli  $\neg(x_i \preccurlyeq x)$ , to wykonuje się zamianę  $x_i \leftrightarrow x$  i sprawdza się, czy po tej zamianie  $x_i \preccurlyeq x$  dla  $i = k-1$ .
3. Powtarzamy punkt 2 tak długo dla kolejnych  $i = k-2, k-3, \dots, 2, 1$  aż warunek  $x_i \preccurlyeq x$  będzie spełniony.

Realizując te punkty dla  $k = 1, 2, \dots, n-1$ , gdzie  $x$  oznacza aktualnie dołączany element ciągu danego (1), otrzymujemy  $n$ -elementowy ciąg (5) uporządkowany przez relację  $\mathcal{R}$ . Liczba porównań jaką wykonuje się w tej metodzie dana jest wzorami:

$$P_{\min}(n) = n-1, \quad P_{\max}(n) = \frac{n(n-1)}{2}.$$

### 3. Metoda porządkowania za pomocą zamiany sąsiednich elementów

Pewną modyfikację metody przestawiania elementów stanowi tzw. metoda zamiany sąsiednich par. Podobnie jak w metodzie poprzedniej dokonuje się sprawdzenia warunku

$$(8) \quad x_k \preccurlyeq x_{k+1} \quad \text{dla} \quad k = 1, 2, \dots, n-1,$$

przy czym, gdy warunek (8) nie jest spełniony dla pewnego  $k$ , dokonuje się zamiany elementów  $x_k$  i  $x_{k+1}$ , czyli operacji  $x_k \leftrightarrow x_{k+1}$  i kontynuuje się sprawdzanie warunku (8) dla  $k+1, k+2, \dots$  itd. Jednocześnie liczy się ilość wykonanych zamian i po sprawdzeniu warunku (8) dla  $k = n-1$  w przypadku, gdy była wykonana przynajmniej jedna zamiana, powtarza się całe postępowanie od początku, czyli sprawdza się ponownie warunek (8) dla  $k = 1, 2, \dots, n-1$ . Ciąg jest uporządkowany, gdy nie wystąpi ani jedna zamiana wyrazów sąsiednich. Łatwo zauważyć, że liczba porównań w tej metodzie jest większa niż w metodzie liniowej, czy w metodzie przestawiania elementów, a mianowicie:

$$P_{\min}(n) = n-1, \quad P_{\max}(n) = n(n-1).$$

Natomiast liczba zamian spełnia nierówność

$$0 \leq Z(n) \leq \frac{n(n-1)}{2}.$$

### 4. Porządkowanie za pomocą łączenia

Wyróżniamy dwie zasadnicze metody łączenia ciągów. Jedna z nich, której idea pochodzi od J. von Neumanna wymaga  $n$  (gdzie  $n$  jest długością ciągu) dodatkowych pozycji w pamięci maszyny cyfrowej, natomiast druga, której autorem jest D. L. Shell nie wymaga dodatkowych pozycji w pamięci.

**4.1. Metoda von Neumanna.** Algorytm metody von Neumanna, który przedstawiamy, znany jest w literaturze pod nazwą „metoda łączenia dwustrumieniowego” [17]. W algorytmie tym wykorzystuje się pewien sposób łączenia dwóch ciągów uporządkowanych przez relację  $\mathcal{R}$  w trzeci ciąg również uporządkowany przez relację  $\mathcal{R}$ . Ciągi, które chcemy połączyć oznaczamy następująco:

$$\begin{aligned} x_1 \preccurlyeq x_2 \preccurlyeq \dots \preccurlyeq x_r, \\ y_1 \preccurlyeq y_2 \preccurlyeq \dots \preccurlyeq y_s, \quad \text{gdzie } r, s \geq 1. \end{aligned}$$

Natomiast

$$z_1 \preccurlyeq z_2 \preccurlyeq \dots \preccurlyeq z_{r+s}$$

niech oznacza ciąg otrzymany w wyniku połączenia.

Algorytm łączenia jest następujący:

(a) sprawdza się, czy spełniony jest warunek  $x_i \preccurlyeq y_j$  dla aktualnych wskaźników  $i, j$ , gdzie  $1 \leq i \leq r$ ,  $1 \leq j \leq s$ ;

(b) dokonuje się następujących podstawień:

gdy  $x_i \preccurlyeq y_j$ , to  $z_{i+j-1} := x_i$  oraz  $i := i + 1$ ,

gdy  $\neg(x_i \preccurlyeq y_j)$ , to  $z_{i+j-1} := y_j$  oraz  $j := j + 1$ .

Sprawdzanie warunku  $x_i \preccurlyeq y_j$  rozpoczynamy od  $i = 1$  oraz  $j = 1$  i kontynuujemy tak długo, aż wskaźnik  $i$  osiągnie wartość  $r + 1$  lub wskaźnik  $j$  wartość  $s + 1$ . Gdy  $i = r + 1$ , to trzeba wykonać podstawienia

$$z_{r+k} := y_k \quad \text{dla} \quad k = j, j+1, \dots, s.$$

Gdy natomiast  $j = s + 1$ , to dokonuje się podstawień

$$z_{s+k} := x_k \quad \text{dla} \quad k = i, i+1, \dots, r.$$

Algorytm porządkowania metodą von Neumanna jest następujący: Załóżmy dla uproszczenia, że  $n = 2^k$ . Na każdym etapie algorytmu stosujemy opisaną metodę łączenia podciągów uporządkowanych w jeden ciąg uporządkowany.

1. Elementy  $x_i$  oraz  $x_{i+1}$  ( $i = 1, 3, 5, \dots, n-1$ ) łączymy w pary uporządkowane przez relację  $\mathcal{R}$  otrzymując w ten sposób  $n/2$  ciągów dwuelementowych.

2. Z  $n/2$  ciągów dwuelementowych tworzymy za pomocą łączenia  $n/4$  ciągi cztero-elementowe.

$j$ . Z  $n_{j-1} = 2^{k-j+1}$  podciągów  $2^{j-1}$  elementowych tworzymy za pomocą łączenia  $n_j = 2^{k-j}$  ciągów  $2^j$  elementowych ( $j = 1, 2, \dots, k$ ) uporządkowane przez relację  $\mathcal{R}$ .

Łączenie kontynuujemy tak długo, aż otrzymamy ciąg  $n$ -elementowy.

W przypadku, gdy  $n \neq 2^k$  na poszczególnych etapach łączenia możemy otrzymać nieparzystą liczbę ciągów. Łatwo sprawdzić, że minimalna liczba porównań wyraża się wzorem:

$$P_{\min}(2^k) = \frac{2^k}{2} + 2 \cdot \frac{2^k}{4} + 4 \cdot \frac{2^k}{8} + \dots + 2^{k-1} \cdot \frac{2^k}{2^k} = \sum_{j=0}^{k-1} 2^j \cdot 2^{k-1-j} = k \cdot 2^{k-1}.$$

Minimalną liczbę porównań wykonujemy, gdy ciąg jest już uporządkowany przez relację  $\mathcal{R}$  (lub przez relację odwrotną do  $\mathcal{R}$ ).

Natomiast

$$\begin{aligned} P_{\max}(2^k) &= \frac{2^k}{2} + 3 \cdot \frac{2^k}{4} + 7 \cdot \frac{2^k}{8} + \dots + (2^k - 1) \frac{2^k}{2^k} = \\ &= \sum_{j=1}^k (2^j - 1) 2^{k-j} = (k-1) 2^k + 1. \end{aligned}$$

W przypadku, gdy  $n \neq 2^k$  w ostatnim etapie procesu łączenia mamy połączyć dwa ciągi, z których jeden liczy  $n_1 = 2^k$  elementów (gdzie  $k$  jest maksymalną potęgą 2 dla liczby  $n$ ), a drugi liczy  $n_2 = n - 2^k$  elementów. Aby otrzymać te dwa uporządkowane ciągi należy wykonać co najmniej  $P_{\min}(2^k)$  porównań dla pierwszego ciągu oraz  $P_{\min}(n-2^k)$  porównań dla drugiego ciągu. Aby ciągi te połączyć w jeden ciąg uporządkowany należy wykonać co najmniej  $n - 2^k$  porównań (bo tyle jest elementów w drugim ciągu). Stosując podobne rozumowanie dla maksymalnej liczby porównań otrzymujemy następujące wzory:

$$(9) \quad P_{\min}(n) = P_{\min}(2^k) + P_{\min}(n-2^k) + n - 2^k,$$

$$(10) \quad P_{\max}(n) = P_{\max}(2^k) + P_{\max}(n-2^k) + n - 1 \quad \text{gdzie} \quad k = \lfloor \log_2 n \rfloor.$$

Wzory (9) i (10) są rekurencyjne, a tym samym niewygodne w praktyce. Wielkości te można oszacować następująco:

$$(11) \quad \left\lceil \frac{n}{2} \right\rceil \cdot \lfloor \log_2 n \rfloor \leq P_{\min}(n) \leq \left\lfloor \frac{n}{2} \right\rfloor (\lfloor \log_2 n \rfloor + 1),$$

$$(12) \quad P_{\max}(n) \leq n \lfloor \log_2 n \rfloor.$$

W pracach [1] i [5] podano następujący wzór na średnią liczbę porównań w metodzie łączenia po dwa ciągi:

$$P_{\text{sr}}(n) = n \lceil \log_2 n \rceil,$$

gdzie przez  $\lceil x \rceil$  oznaczono najmniejszą liczbę całkowitą nie mniejszą niż  $x$ . Wzór ten może jedynie służyć za bardzo niedokładne oszacowanie (od góry) maksymalnej liczby porównań. Według tego wzoru średnia liczba porównań dla ciągu 1000 elementowego wynosi  $P_{\text{sr}}(1000) = 10000$ , natomiast ze wzoru (10) otrzymujemy

$$P_{\max}(1000) = 8985.$$

W metodzie tej nie wykonuje się zamian, można natomiast liczyć ilość wykonywanych przedstawień elementów z jednego miejsca pamięci maszyny cyfrowej w drugie. W opisanej metodzie porządkowania wykonuje się

$$Q(n) = n \left( \lfloor \log_2(n-1) \rfloor + 1 \right)$$

przestawień.

Dla  $n = 2^k$  maksymalna liczba porównań w metodzie von Neumanna jest taka sama jak maksymalna liczba porównań w metodzie podziału dychotomicznego.

Liczba etapów łączenia w metodzie von Neumanna dana jest wzorem  $\lceil \log_2 n \rceil$  (por. np. [17]).

Zasadniczą wadą przedstawionego algorytmu jest to, że dla uporządkowania ciągu  $n$ -elementowego należy zarezerwować  $2n$  pozycji w pamięci maszyny cyfrowej, co przy dużych ciągach przekreśla przydatność tej metody. Metoda ta jest szczególnie efektywna przy użyciu taśm magnetycznych, a więc przy porządkowaniu ciągów zapisanych w pamięci zewnętrznej maszyny cyfrowej.

**4.2. Porządkowanie za pomocą łączenia ciągów o zmiennej długości.** Metoda ta jest jedynie pewną modyfikacją metody von Neumanna, a mianowicie wykorzystuje naturalne uporządkowanie ciągu podlegającego operacji porządkowania (dlatego też nazywana jest metodą naturalnego łączenia von Neumanna). W związku z tym na każdym etapie łączenia występują ciągi o różnych długościach. Liczba porównań i przestawień, jak podano w pracy [8], jest rzędu

$$P(n) = 2n (\lceil \log_2 n \rceil - 1), \quad Q(n) = n (\lceil \log_2 n \rceil - 1).$$

Zatem liczba porównań jest dwukrotnie większa niż w metodzie łączenia dwustrumieniowego (klasycznej metodzie von Neumanna).

Uogólnieniem dwustrumieniowej metody łączenia jest tzw. łączenie wielostrumieniowe, które polega na tym, że łączy się równocześnie  $s$  ciągów, gdzie  $s > 2$ . Łatwo zauważyć, że ze wzrostem  $s$  maleje liczba etapów łączenia, która wyraża się w tym przypadku wzorem

$$\lceil \log_s n \rceil.$$

Jednocześnie jednak łączenie  $s$ -strumieniowe wymaga zarezerwowania  $sn$  pozycji w pamięci maszyny cyfrowej, co przy dużej liczbie danych jest często praktycznie niemożliwe ze względu na małą pojemność pamięci operacyjnej. Dlatego też metoda ta jest szczególnie użyteczna przy łączeniu zbiorów zapisanych w pamięciach taśmowych.

**4.3. Metoda Shella.** Algorytm metody, którą przedstawiamy, został opracowany przez D.L. Shella [14]. Charakterystyczny dla tej metody jest sposób łączenia dwóch ciągów uporządkowanych przez relację  $\mathcal{R}$  w trzeci ciąg również uporządkowany przez tę relację.

Zakładamy, że dane są dwa ciągi uporządkowane przez relację  $\mathcal{R}$ :

$$(13) \quad x_{k_1} \preccurlyeq x_{k_3} \preccurlyeq \dots \preccurlyeq x_{k_{n-1}},$$

$$(14) \quad x_{k_2} \preccurlyeq x_{k_4} \preccurlyeq \dots \preccurlyeq x_{k_n},$$

gdzie  $k_1 < k_2 < k_3 < \dots < k_{n-1} < k_n$ , oraz, że liczba elementów w obu ciągach jest równa. Ciągi te należy tak połączyć, aby po przenumerowaniu otrzymać ciąg

$$(15) \quad x_{l_1} \preccurlyeq x_{l_2} \preccurlyeq \dots \preccurlyeq x_{l_{n-1}} \preccurlyeq x_{l_n}.$$

Aby otrzymać ciąg (15) stosuje się następujący algorytm:

(a) sprawdza się, czy spełniony jest warunek

$$(16) \quad x_{k_i} \preceq x_{k_{i+1}} \quad \text{dla} \quad i = 1, 2, \dots, n-1,$$

(b) jeżeli warunek (16) nie jest spełniony dla pewnego  $k_j$  ( $j = 1, 2, \dots, n-1$ ), to elementy  $x_{k_j}$  i  $x_{k_{j+1}}$  zamieniamy miejscami i sprawdzamy, czy warunek (16) jest spełniony dla  $i < j$ . Jeżeli tak, to kontynuujemy sprawdzanie (16) dla  $i > j$ . W przeciwnym przypadku elementy, dla których (16) nie zachodzi, zamieniamy miejscami. Kontynuując to postępowanie otrzymujemy ciąg (15).

Ten sposób łączenia ciągów nie wymaga dodatkowych pozycji pamięci maszyny cyfrowej (co najwyżej jedną pomocniczą na wykonanie porównania).

Algorytm porządkowania metodą Shella (dla  $n = 2^k$ ):

1. Tworzy się  $m_1 = n/2$  podciągów dwuelementowych o własności

$$x_j \preceq x_{m_1+j} \quad \text{dla} \quad j = 1, 2, \dots, m_1.$$

2. Z  $m_{i-1}$  podciągów  $n/m_{i-1}$  elementowych otrzymuje się za pomocą przedstawionego algorytmu łączenia  $m_i = m_{i-1}/2$  podciągów  $2^i$  elementowych o własności

$$x_j \preceq x_{j+m_1} \preceq \dots \preceq x_{j+(2^i-1)m_1} \quad \text{gdzie} \quad j = 1, 2, \dots, m_i.$$

3. Realizując punkt 2 dla  $i = 2, 3, \dots, k$  otrzymuje się uporządkowany ciąg

$$(17) \quad x_{p_1} \preceq x_{p_2} \preceq \dots \preceq x_{p_n}.$$

Minimalną liczbę porównań można przedstawić za pomocą wzoru

$$P_{\min}(n) = \sum_{i=1}^{\lfloor \log_2 n \rfloor} (n - m_i)' = n \lfloor \log_2 n \rfloor - \sum_{i=1}^{\lfloor \log_2 n \rfloor} \left\lceil \frac{n}{2^i} \right\rceil.$$

Gdy  $n = 2^k$ , to

$$P_{\min}(n) = n(k-1) + 1.$$

Maksymalna liczba porównań jest trudna do wyliczenia i prawdopodobnie ma miejsce wówczas, gdy mamy uporządkować ciąg liczbowy o długości  $n = 2^k$  postaci

$$2^{k-1} + 1, 1, 2^{k-1} + 2, 2, \dots, 2^k, 2^{k-1}.$$

W tym przypadku porządkowanie tego ciągu sprowadza się, jak zauważono w pracy [6], do oddzielnego porządkowania ciągów

$$x_1, x_3, x_5, \dots \quad \text{oraz} \quad x_2, x_4, x_6, \dots$$

Trudność tę można ominąć dobierając odpowiednio wartości  $m_i$ . Optymalny wybór wartości  $m_i$  jest sprawą otwartą,

W pracy [18] średnią liczbę porównań w metodzie Shella oszacowano wzorem

$$P_{sr}(n) = K \cdot n \cdot \log_2 n,$$

gdzie  $K$  jest współczynnikiem proporcjonalności, a w pracy [10] wzorem

$$P_{sr}(n) = n^{3/2} \log_2 n.$$

Metoda Shella jest pod względem szybkości taka sama, jak metoda łączenia parami von Neumanna ([1], str. 120) i ma tę ważną niekiedy zaletę, że rezerwuje jedynie  $n$  pozycji w pamięci maszyny cyfrowej na zapamiętanie wyrazów ciągu, podczas, gdy metoda von Neumanna rezerwuje  $2n$  pozycji, gdzie  $n$  jest liczbą wyrazów ciągu.

## 5. Metody porządkowania za pomocą podziału (lub segregacji)

Metody porządkowania za pomocą podziału należą do najszybszych, a więc i najefektywniejszych metod porządkowania. Najważniejsza z nich to metoda Scowena, która jest pewną modyfikacją metody Hoare'a, metoda Singletona oraz metoda van Emдена.

**5.1. Opis algorytmu w przypadku ogólnym.** Charakterystyczny dla tej grupy metod jest następujący algorytm podziału.

1. Ustala się liczbę podzbiorów, na które chcemy podzielić zbiór  $X = \{x_1, x_2, \dots, x_n\}$  oraz elementy  $t_1, t_2, \dots, t_k$  (gdzie  $t_1 < t_2 < \dots < t_k$  oraz  $t_i \neq t_j$  dla  $i \neq j$ ,  $1 \leq k < n$ ), które nazywamy punktami podziału.

Otrzymujemy w ten sposób  $k+1$  podzbiorów

$$X_1 = \{x_i \mid x_i < t_1\},$$

$$X_j = \{x_i \mid t_{j-1} \leq x_i < t_j\}, \quad j = 2, 3, \dots, k,$$

$$X_{k+1} = \{x_i \mid t_k \leq x_i\}$$

o własnościach

$$X_1 \cup X_2 \cup \dots \cup X_{k+1} = X, \quad X_i \cap X_j = \emptyset \quad \text{dla} \quad i \neq j$$

oraz

$$(\forall x \in X_i) (\forall y \in X_j) (i < j \Rightarrow x < y).$$

2. Dla każdego ze zbiorów  $X_1, X_2, \dots, X_{k+1}$ , który zawiera co najmniej dwa elementy, ustala się podobnie jak dla zbioru  $X$  odpowiednią wartość  $k'$  oraz elementy  $t'_1, t'_2, \dots, t'_{k'}$ , a następnie dokonuje się podziału.

3. Postępowanie opisane w punkcie 2 kontynuuje się tak długo, aż każdy ze zbiorów otrzymanych w wyniku podziału będzie zawierał jeden element.

**5.2. Porządkowanie za pomocą podziału na dwa podzbiory.** Zbiór  $X$  dzieli się na pewną liczbę podzbiorów przede wszystkim po to, aby zmniejszyć liczbę porównań w procesie porządkowania. Niecelowy jest więc taki podział, w którym zachodziłyby duże dysproporcje między liczbą elementów w każdym ze zbiorów. Należy zatem tak dobrać liczbę  $k$  oraz elementy  $t_1, t_2, \dots, t_k$ , aby zbiory otrzymane w wyniku podziału były równoliczne oraz aby liczba elementów w każdym ze zbiorów była możliwie mała. Praktycznie jest to



niemożliwe, gdyż tak sformułowany algorytm porządkowania wymaga  $n$  dodatkowych pozycji w pamięci maszyny cyfrowej na zapamiętanie miejsca każdego elementu zbioru  $X$  w ciągu  $t_1 < t_2 < \dots < t_k$ , gdy  $k < n$ , co przy określonej pojemności pamięci operacyjnej może być niewykonalne.

Dlatego też prawie we wszystkich algorytmach porządkowania za pomocą podziału przyjmuje się  $k = 1$ . Zasadniczym problemem jest więc zagadnienie wyboru elementu  $t_1$  dla zbioru  $X$ , a następnie dla zbiorów  $X_1, X_2$  otrzymanych w wyniku podziału zbioru  $X$ , itd. Chodzi o to, aby porządkowania elementów zbioru  $X$  dokonać w możliwie małej liczbie kroków, tj. aby liczba iteracji, czyli krotność realizacji punktu 2 była najmniejsza. Ma to miejsce wtedy, gdy element  $t_1$ , który wyznacza się ze zbioru podlegającego podziałowi jest taki, że w wyniku podziału zbioru o liczbie elementów  $n$ , obydwa zbiory zawierają po  $n/2$  elementów, gdy  $n$  jest parzyste lub po  $(n-1)/2$  i  $(n+1)/2$  elementów, gdy  $n$  jest nieparzyste. Za  $t_1$  należy więc przyjąć medianę elementów zbioru, który podlega podziałowi. Termin „mediana” pochodzi ze statystyki. Mediana ma tę własność, że rozdziela zbiór obserwacji statystycznych na dwie równe części takie, że w jednej z nich znajdują się obserwacje o wartościach mniejszych od mediany, a w drugiej o wartościach większych od mediany. Ponieważ mediana nie jest na ogół znana, aproksymuje się ją w następujący sposób.

W metodzie Scowena, której procedura quickersort zamieszczona jest w [13], przyjmuje się

$$t_i = x_{f_i}, \quad \text{gdzie} \quad f_i = \left\lfloor \frac{1+n}{2} \right\rfloor, \quad i = 1, 2, \dots$$

Indeks  $i$  oznacza etapy podziału zbioru  $X$ , a następnie podzbiorów  $X_1, X_2$ , itd.

Wykonując porównania elementów zbioru  $X$  z elementem  $x_{f_1}$  dzielimy zbiór  $X$  na podzbiory  $X_1, X_2$  takie, że

$$X_1 = \{x_i \mid x_i < x_{f_1}\}, \quad X_2 = \{x_i \mid x_{f_1} \leq x_i\}.$$

Przy następnych podziałach  $f_i = \left\lfloor \frac{m_i + n_i}{2} \right\rfloor$ , gdzie  $m_i$  jest numerem pierwszego elementu aktualnie dzielonego podzbioru, a  $n_i$  jest numerem ostatniego elementu dzielonego podzbioru.

Natomiast w metodzie Singletona, której procedura podana jest w [15] za  $t_1$  przyjmuje się medianę elementów  $x_m, x_{\left\lfloor \frac{m+n}{2} \right\rfloor}, x_n$  (w pierwszym etapie dzielenia  $m=1$ ).

Algorytm Singletona różni się ponadto od algorytmu Scowena tym, że zbiór  $X$  lub zbiory otrzymane w wyniku realizacji jednego z punktów 1, 2 jeżeli zawierają co najwyżej 10 elementów nie podlegają dalszemu podziałowi. Ciągi, w których liczba elementów nie przekracza 10, porządkuje się metodą zamiany sąsiednich par, która dla ciągów o tej długości jest tak samo efektywna jak metoda Shella.

Zarówno w metodzie Scowena jak i Singletona używa się w programie dwóch zmiennych dodatkowych z indeksami. W metodzie Scowena zmiennymi tymi są  $p_i, q_i$ , gdzie  $1 \leq i \leq \lfloor \log_2 (n-1) + 1 \rfloor$ .

W metodzie Singletona używa się również zmiennych  $p_i, q_i$ , gdzie  $0 \leq i \leq \lfloor \log_2 n - 0,9 \rfloor$ , na co zwrócono uwagę w [7]. Zmiennych  $p_i$  oraz  $q_i$  używa się dla zapamiętania adresów pierwszego i ostatniego elementu w poszczególnych ciągach otrzymanych w wyniku podziału.

Najefektywniejszym, jeśli chodzi o czas porządkowania, jest algorytm van Emdena, którego procedura w języku ALGOL 60 jest zamieszczona w [2]. Idea algorytmu van Emdena jest analogiczna do idei Scowena i Singletona z tym, że cały zbiór  $X$  dzieli się w zasadzie na trzy podzbiory takie, że

$$X_1 = \{x_i \mid x_i \preccurlyeq t_1\}, \quad X_2 = \{t_1, t_2\}, \quad X_3 = \{x_i \mid t_2 \preccurlyeq x_i\}, \quad \text{gdzie} \quad t_1 \prec t_2.$$

Elementy  $t_1$  i  $t_2$  nie zmieniają swego położenia do końca porządkowania. W metodzie van Emdena zbiór  $X$  zostaje więc podzielony na dwie części parą elementów ekstremalnych jego podzbiorów.

**5.3. Metoda funkcyjna.** Metoda ta jest pewną odmianą metody należącej do grupy metod zwanych „sorting by adress calculation” [5]. Stosuje się ją jedynie do pewnych zbiorów liczbowych. Zakłada się, że elementy zbioru liczbowego  $X = \{x_1, x_2, \dots, x_n\}$  pochodzą z populacji o rozkładzie jednostajnym na odcinku  $[a, b]$ . Aby je uporządkować proponuje się przyjąć za  $t_1, t_2, \dots, t_{k-1}$  takie liczby, aby

$$t_1 - a = t_2 - t_1 = \dots = t_{k-1} - t_{k-2} = b - t_{k-1} = \frac{b-a}{k}.$$

Relację porządkującą  $\mathbb{R}$  stanowi relacja  $\prec$ .

Zbiory  $X_1, X_2, \dots, X_k$  określamy następująco:

$$X_1 = \{x_i \in X \mid a \leq x_i < t_1\},$$

$$X_2 = \{x_i \in X \mid t_1 \leq x_i < t_2\},$$

.....

$$X_k = \{x_i \in X \mid t_{k-1} \leq x_i < b\}.$$

Wskaźnik  $j$  adresu zbioru  $X_j$ , do którego ma być zaliczony element  $x \in X$ , wyznacza się z funkcji

$$j = \left[ k \cdot \frac{x - a}{b - a} \right] + 1.$$

Po zaliczeniu każdego elementu  $x_i \in X$  do jednego ze zbiorów  $X_1, X_2, \dots, X_k$ , zbiory te porządkuje się dowolną metodą.

Aby wykonać jak najmniejszą liczbę porównań należy za  $k$  przyjąć  $n$ . Wówczas oczekiwana liczba elementów w każdym ze zbiorów  $X_i$  ( $i = 1, 2, \dots, k$ ) wynosi 1. Metoda ta jest bardzo szybka, lecz wymaga zarezerwowania  $n + 2k$  pozycji w pamięci maszyny cyfrowej,  $n$  na elementy porządkowanego zbioru,  $k$  na punkty podziału oraz  $k$  na liczbę elementów w każdym ze zbiorów  $X_1, X_2, \dots, X_k$ . Do metod porządkowania za pomocą podziału należy tzw. metoda porządkowania według podstawy, zwana też metodą dystrybucji pozycyjnej [17]. W podobny sposób pracują sortery — urządzenia mechaniczne stosowane do sortowania danych. Metoda ta jest przedstawiona szczegółowo w cytowanej pracy.

## 6. Metoda podziału z wyborem elementów minimalnych

Metoda zwana selekcją kwadratową jest kompozycją dwóch metod porządkowania: metody podziału oraz metody wyboru elementów najmniejszych. Stosując jednokrotnie podział zbioru  $X$  na  $k$  podzbiorów, a następnie wybierając z każdego z podzbiorów elementy minimalne mamy do czynienia z tzw. selekcją kwadratową [17]. Podział dwustopniowy można stosować wielokrotnie na otrzymanych podzbiorach i wówczas otrzymuje się uogólnienie selekcji kwadratowej.

**6.1. Podział dwustopniowy (selekcja kwadratowa).** Ogólny schemat porządkowania tą metodą jest następujący:

1. Zbiór  $X = \{x_1, x_2, \dots, x_n\}$  dzieli się na  $k$  rozłącznych podzbiorów  $X_1, X_2, \dots, X_k$  o liczebnościach odpowiednio równych:  $n_1, n_2, \dots, n_k$ , gdzie  $n_1 + n_2 + \dots + n_k = n$ .
2. Z każdego ze zbiorów:  $X_1, X_2, \dots, X_k$  wybiera się element minimalny. Elementy te tworzą ciąg  $k$ -elementowy, który oznaczamy następująco:

$$y_1, y_2, \dots, y_k \quad \text{gdzie} \quad y_i = \min \{x_i \mid x_i \in X_i\}, i = 1, 2, \dots, k$$

3. Z ciągu  $y_1, y_2, \dots, y_k$  wybiera się element najmniejszy, który jest równocześnie pierwszym elementem ciągu uporządkowanego, czyli

$$x_{p_1} = y_i, \quad \text{gdzie} \quad y_i = \min (y_1, y_2, \dots, y_k).$$

4. Ze zbioru  $X_i - \{y_i\}$  wybiera się następny najmniejszy element ustawiając go na  $i$ -te miejsce w ciągu  $y_1, y_2, \dots, y_k$ .

Punkty 3 i 4 powtarza się tak długo, aż każdy element zbiorów  $X_1, X_2, \dots, X_k$  zostanie włączony do ciągu  $y_1, y_2, \dots, y_k$ . Wówczas wystarczy już tylko uporządkować ten ciąg. Dla wykonania porządkowania na maszynie cyfrowej należy zarezerwować  $n + 3k$  pozycji w pamięci maszyny,  $n$  pozycji na elementy uporządkowanego ciągu,  $k$  pozycji na liczebności zbiorów  $X_1, X_2, \dots, X_k$ ,  $k$  pozycji na elementy ciągu  $y_1, y_2, \dots, y_k$  oraz  $k$  pozycji na zapamiętanie numerów zbiorów, z których pochodzą elementy  $y_i$  ( $i = 1, \dots, k$ ). Obliczmy maksymalną liczbę porównań potrzebnych do uporządkowania tą metodą  $n$ -elementowego ciągu. Aby wybrać element minimalny ze zbioru  $X_i$  należy wykonać  $n_i - 1$  porównań. Zatem, aby utworzyć ciąg  $y_1, y_2, \dots, y_k$  wykonuje się następującą liczbę porównań

$$\sum_{i=1}^k (n_i - 1).$$

Aby znaleźć element minimalny w ciągu  $y_1, \dots, y_k$  trzeba wykonać  $k-1$  porównań. Zatem dla wyznaczenia pierwszego elementu ciągu uporządkowanego wykonuje się

$$\sum_{i=1}^k (n_i - 1) + k - 1$$

porównań. Łatwo zauważyć, że dla uzyskania  $k$  następnych elementów ciągu uporządkowanego wykonuje się co najwyżej

$$\sum_{i=1}^k (n_i - 2) + k - 1$$

porównań itd. Zakładamy, że z każdego ze zbiorów  $X_i$  ( $i = 1, 2, \dots, k$ ) wybranych zostanie dokładnie  $n/k - 1$  elementów. W rezultacie wykonujemy tyle porównań ile potrzeba do uporządkowania każdego ze zbiorów oraz  $(k-1)(n-k)$  porównań dla wybrania elementów minimalnych ciągu  $y_1, y_2, \dots, y_k$  i wreszcie  $k(k-1)/2$  porównań potrzebnych do uporządkowania ciągu  $y_1, y_2, \dots, y_k$ , gdy wszystkie elementy zbiorów  $X_i$  zostaną wybrane. Stąd

$$P_{\max}(n, k) = \sum_{i=1}^k \frac{n_i(n_i-1)}{2} + (k-1)(n-k) + \frac{k(k-1)}{2}.$$

Po przekształceniu otrzymujemy

$$P_{\max}(n, k) = \sum_{i=1}^k \frac{n_i(n_i-1)}{2} + (k-1)\left(n - \frac{k}{2}\right).$$

Powstaje zagadnienie wyznaczenia optymalnej liczby podzbiorów  $k$  oraz liczebności poszczególnych podzbiorów. Zauważmy, że  $P_{\max}(n, k)$  jest dla ustalonego  $n$  funkcją  $k-1$  zmiennych

$$P_{\max}(n, k) = F(n_1, n_2, \dots, n_{k-1})$$

bo  $n_k = n - (n_1 + n_2 + \dots + n_{k-1})$ .

Funkcja ta przyjmuje ekstremum dla

$$n_1 = n_2 = \dots = n_{k-1} = n/k.$$

Pozostaje problem wyznaczenia wielkości  $k$ . Liczba ta powinna być taka, aby maksymalna liczba porównań, która jest zależna od  $k$ , przyjęła wartość najmniejszą. Wstawiając  $n_i = n/k$  do  $P_{\max}(n, k)$  otrzymujemy

$$P_{\max}(n, k) = \frac{n(n-k)}{2k} + (k-1)\left(n - \frac{k}{2}\right)^2.$$

Szukamy

$$\min \left[ \frac{n(n-k)}{2k} + (k-1)\left(n - \frac{k}{2}\right)^2 \right].$$

Łatwo pokazać, że funkcja ta przyjmuje ekstremum dla  $k_1 = n$  i  $k_2 = \frac{1}{4}(1 + \sqrt{1 + 8n})$ . W przypadku  $k_1 = n$  funkcja  $P_{\max}(n, k)$  osiąga maksimum i wówczas jest to zwyczajne porządkowanie metodą wybierania elementów minimalnych, natomiast dla  $k = \frac{1}{4}(1 + \sqrt{1 + 8n})$  maksymalna liczba porównań jest najmniejsza. Tym samym udowodniliśmy następujące

**TWIERDZENIE 1.** *Maksymalna liczba porównań potrzebnych do uporządkowania elementów zbioru  $X$  metodą podziału z wyborem elementów minimalnych jest najmniejsza dla  $k = \frac{1}{4}(1 + \sqrt{1 + 8n})$  oraz  $n_i = n/k$ .*

Jest to oczywiście rozwiązanie przybliżone, gdyż funkcja  $P_{\max}(n, k)$  przyjmuje wartości całkowite, podczas gdy ekstremum szukano dla funkcji ciągłych.

Przy stosowaniu tego algorytmu w praktyce zaokrąglamy  $k$  do liczby całkowitej, tak aby  $n/k$ , było również całkowite. Minimalną liczbę porównań wykonujemy wówczas, gdy będzie się zmniejszała liczba wyrazów ciągu  $y_1, y_2, \dots, y_k$ . Ma to miejsce wtedy, gdy wybrane zostaną kolejno elementy jednego podzbioru, drugiego itd. Stąd

$$P_{\min}(n, k) = \frac{\left(\frac{n}{k} - 1\right) \frac{n}{k}}{2} \cdot k + \frac{n}{k}(k-1) + \frac{n}{k}(k-2) + \dots + \frac{n}{k} = \frac{n}{2} \left( \frac{n}{k} + k - 2 \right).$$

W pracach [1] i [5] algorytm selekcji kwadratowej różni się sposobem wybierania elementów minimalnych. Mianowicie, aby wybrać element minimalny z ciągu  $r_1, r_2, \dots, r_n$  wykonuje się  $n-1$  porównań. Załóżmy, że jest to element  $r_k$ . W miejsce tego elementu wstawia się element większy od wszystkich wyrazów ciągu  $\{r_n\}$ . Aby więc wybrać następny element minimalny wykonuje się znowu  $n-1$  porównań. Aby uporządkować ciąg  $r_1, r_2, \dots, r_n$  tą metodą wykonuje się  $n(n-1)$  porównań. Wówczas maksymalna liczba porównań przy podziale dwustopniowym dana jest wzorem

$$P_{\max}(n, k) = \left(\frac{n}{k} - 1\right) \frac{n}{k} \cdot k + (k-1)n,$$

czyli

$$P_{\max}(n, k) = n \left( \frac{n}{k} + k - 2 \right).$$

Łatwo sprawdzić, że dla  $k = \sqrt{n}$  funkcja  $P_{\max}(n, k)$  osiąga minimum.

**6.2. Podział wielostopniowy.** Metodę podziału dwustopniowego można uogólnić na podział wielostopniowy w następujący sposób.

1. Zbiór  $X$  dzielimy na  $k_1$  rozłącznych podzbiorów, z których każdy liczy  $n_1$  elementów. Z każdego z tych zbiorów wybieramy element minimalny i tworzymy z nich zbiór  $X_1$ .

2. Zbiór  $X_1$  dzielimy na  $k_2$  rozłącznych podzbiorów, z których każdy liczy  $n_2$  elementów. Z każdego ze zbiorów wybieramy element minimalny tworząc z nich zbiór  $X_2$ .

s. Zbiór  $X_{s-1}$  dzielimy na  $k_s$  podzbiorów, z których każdy liczy  $n_s$  elementów. Z każdego z nich wybieramy element minimalny i tworzymy zbiór  $X_s$ .

Punkt s realizujemy dla każdego  $X_s$  ( $s = 1, 2, \dots$ ) liczącego co najmniej trzy elementy. Element najmniejszy zbioru  $X_s$  jest pierwszym wyrazem ciągu uporządkowanego. Uzupełnianie wybranych elementów jest analogiczne jak przy podziale dwustopniowym.

Interesuje nas problem optymalnego wyznaczenia liczb  $k_1, k_2, \dots, k_s$  dla ustalonego  $s$ . Dla uproszczenia rozważań przeprowadzimy obliczenia dla  $s = 3$ , czyli dla podziału trzystopniowego. Algorytm tego podziału jest następujący:

1. Zbiór  $X$  dzielimy na  $k_1$  podzbiorów  $X_{1,1}, X_{1,2}, \dots, X_{1,k_1}$  o tej samej liczebności.

2. Zbiór  $X_1$  utworzony z  $k_1$  elementów, które są najwcześniejszymi elementami podzbiorów  $X_{1,i}$  ( $i = 1, 2, \dots, k_1$ ) dzielimy na  $k_2$  równolicznych podzbiorów  $X_{2,1}, X_{2,2}, \dots, X_{2,k_2}$ . W każdym z nich znajdujemy element minimalny i tworzymy z nich zbiór  $X_2$ .

3. Znajdujemy element minimalny w zbiorze  $X_2$ . Jest to pierwszy wyraz ciągu upo-

ządkowanego przez relację  $R$ . Oznaczmy go przez  $x_{p_1}$ . Zbiór  $X_2$  uzupełniamy kolejnym następnym elementem najwcześniejszym tego zbioru, z którego pochodził element  $x_{p_1}$ .

Punkt 3 realizujemy tak długo, aż wszystkie elementy zbioru  $X$  znajdują się w zbiorze  $X_2$ .

Liczby  $k_1, k_2$  należy tak dobrać, aby ogólna liczba porównań była najmniejsza. Wyznamy maksymalną liczbę porównań. Aby wybrać kolejno wszystkie elementy ze zbiorów  $X_{1,1}, X_{1,2}, \dots, X_{1,k_1}$  metodą podaną dla podziału dwustopniowego, tzn. wstawiając w miejsce elementu najwcześniejszego pewien wyraz maksymalny, należy wykonać

$$\left(\frac{n}{k_1} - 1\right) \frac{n}{k_1} \cdot k_1$$

porównań.

Podobnie aby wybrać wszystkie elementy ze zbiorów  $X_{2,1}, X_{2,2}, \dots, X_{2,k_2}$  (przez każdy z tych zbiorów musi przejść  $n/k_1$  elementów) potrzeba wykonać

$$\left(\frac{k_1}{k_2} - 1\right) \cdot \frac{n}{k_1} \cdot k_1$$

porównań.

Ponieważ wszystkie elementy muszą przejść przez zbiór  $X_2$  liczący  $k_2$  elementów, z którego wybiera się elementy minimalne będące odpowiednimi wyrazami uporządkowanego ciągu, stąd wykonuje się

$$(k_2 - 1)n$$

porównań.

Łącznie wykonuje się co najwyżej

$$P_{\max}(n, k_1, k_2) = \left(\frac{n}{k_1} - 1\right)n + \left(\frac{k_1}{k_2} - 1\right)n + (k_2 - 1)n$$

porównań.

Po redukcji otrzymujemy

$$P_{\max}(n, k_1, k_2) = n \left( \frac{n}{k_1} + \frac{k_1}{k_2} + k_2 - 3 \right).$$

Łatwo wykazać, że funkcja  $F(k_1, k_2) = P_{\max}(n, k_1, k_2)$  osiąga minimum dla  $k_1 = \sqrt[3]{n^2}, k_2 = \sqrt[3]{n}$ . Można zatem wysłowić następujące

**TWIERDZENIE 2.** *Liczba porównań jaką co najwyżej należy wykonać, aby uporządkować  $n$ -elementowy ciąg za pomocą podziału trzystopniowego z wyborem elementów minimalnych jest najmniejsza, jeżeli*

$$k_1 = \sqrt[3]{n^2}, \quad k_2 = \sqrt[3]{n}.$$

Stosując podział  $s$ -stopniowy, w którym zbiór  $X$  dzieli się na  $k_1$  równolicznych podzbiorów, z kolei zbiór  $X_2$  na  $k_2$  równolicznych podzbiorów, itd. aż w końcu zbiór  $X_s$  na  $k_s$  równolicznych podzbiorów maksymalna liczba porównań, jaką co najwyżej należy wykonać wynosi:

$$P_{\max}(n, k_1, k_2, \dots, k_s) = n \left( \frac{n}{k_1} + \frac{k_1}{k_2} + \dots + \frac{k_{s-1}}{k_s} + k_s - s - 1 \right).$$

Łatwo sprawdzić, że przy ustalonym  $s$  funkcja  $P_{\max}(n, k_1, k_2, \dots, k_s)$  osiąga minimum dla

$$k_i = \sqrt[s+1]{n^{s-i+1}}, \quad \text{gdzie} \quad i = 1, 2, \dots, s.$$

Po wstawieniu tych wielkości do wzoru na  $P_{\max}(n, k_1, \dots, k_s)$  otrzymujemy, że

$$P_{\max}(n) = n \left( \underbrace{\sqrt[s+1]{n} + \dots + \sqrt[s+1]{n}}_{(s+1) \text{ razy}} - s - 1 \right),$$

czyli

$$P_{\max}(n) = n(s+1) \left( \sqrt[s+1]{n} - 1 \right).$$

Dla  $n = 2^{s+1}$  otrzymujemy

$$P_{\max}(2^{s+1}) = (s+1)2^{s+1}.$$

Okazuje się, że maksymalna liczba porównań jaką co najwyżej należy wykonać, aby uporządkować ciąg o długości  $n = 2^{s+1}$  stosując podział  $s$ -stopniowy jest nie mniejsza niż liczba porównań w metodzie łączenia von Neumanna. Zatem metoda podziału  $s$ -stopniowego nie jest tak efektywna ze względu na liczbę porównań jak metoda łączenia von Neumanna, czy też metoda podziału dychotomicznego.

## 7. Wyniki testowania procedur porządkowania przeprowadzonego na maszynie ODRA 1204

Testowanie przeprowadzono na ciągach liczb losowych pochodzących z populacji generalnej o rozkładzie jednostajnym na odcinku  $[0, 1]$ . Ciągi te otrzymano za pomocą generatora liczb losowych o rozkładzie jednostajnym. Eksperyment, który przeprowadzono, polegał na generowaniu zbioru o liczebności  $n$ , a następnie porządkowaniu tego zbioru za pomocą poszczególnych metod. Porządkowano zbiory o liczbie elementów

$$n = 50, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 2000, 5000.$$

Dla zbiorów 50-elementowych powtarzano eksperyment 100 razy, natomiast dla zbiorów o liczbie elementów 100, 200, 300, 400, 500 powtarzano 50 razy, a dla pozostałych zbiorów 5 razy. Programy zostały skonstruowane tak, aby dla każdego  $n$  i dla każdej metody obliczać następujące wielkości: rzeczywistą liczbę wykonywanych porównań, a następnie średnią liczbę wykonywanych porównań oraz odchylenia standardowe liczby porównań. Osobno obliczono rzeczywiste czasy porządkowania w sekundach, a następnie średni czas porządkowania w sekundach oraz odchylenie standardowe, dla każdej z metod i każdego  $n$ .

Testowaniem nie została objęta jedynie metoda selekcji kwadratowej. W tabeli 1 podajemy średnią liczbę porównań  $\bar{p}$ , natomiast w tabeli 2 średni czas porządkowania  $\bar{t}$  w sekundach dla poszczególnych metod porządkowania ciągów  $n$ -elementowych.

Dla większości metod porządkowania czas porządkowania jest dla ustalonego  $n$  proporcjonalny do liczby wykonanych porównań. Zatem prawdziwa jest następująca równość:

$$T_m(n) = c_m(n) \cdot P_m(n),$$

gdzie  $T_m(n)$  oznacza czas porządkowania ciągu  $n$ -elementowego metodą  $m$ ,  $P_m(n)$  – liczba porównań (najczęściej średnia) dla danej metody  $m$ .

TABELA 1

Średnia liczba porównań

$n$	Metoda liniowa	Metoda wyboru elementów maksymalnych	Metoda przestawiania elementów	Metoda zamiany sąsiednich par
50	658	1225	657	2109
100	2555	4950	2583	8753
200	10171	19900	10107	35820
300	22409	44850	22599	84676
400	39953	79800	40634	151620
500	63624	124750	62112	—

  

$n$	Metoda podziału dychotomicznego	Metoda von Neumanna	Metoda Shella	Metoda Scowena	Metoda Singletona	Metoda van Emdena
50	216	229	332	269	294	356
100	530	560	871	673	711	880
200	1257	1304	2128	1678	1592	2081
300	2057	2211	3514	2673	2681	3497
400	2906	3040	5433	3787	3796	4969
500	3797	3852	6310	4909	4917	6468
600	4711	5025	8285	5965	6063	8094
700	5637	5810	8274	8288	6525	9534
800	6611	6897	13088	8896	8368	11335
900	7597	7834	14928	10191	9818	13191
1000	8574	8737	15152	11604	10872	14846
2000	19146	19463	34158	27915	23215	32894
5000	54449	—	99720	80262	62186	94289

TABELA 2

Średni czas porządkowania w sekundach

$n$	Metoda liniowa	Metoda wyboru elementów maksymalnych	Metoda przestawiania elementów	Metoda zamiany sąsiednich par
50	3	4	4	9
100	12	16	16	40
200	49	64	64	165
300	110	145	145	395
400	165	257	259	706
500	305	402	396	—



Tabela 2 (c.d.)

$n$	Metoda podziału dycho- tomicznego	Metoda von Neumanna	Metoda Shella	Metoda Scowena	Metoda Singletona	Metoda van Emdena	Metoda funkcyjna
50	2	2	1	1	1	1	1
100	6	5	4	2	2	2	2
200	22	12	11	7	6	5	5
300	47	20	19	11	10	9	7
400	81	27	30	15	14	13	10
500	124	34	34	20	18	18	13
600	176	43	44	24	23	22	16
700	238	50	42	32	24	26	18
800	306	58	73	36	32	30	20
900	384	65	84	41	37	36	22
1000	437	72	83	46	41	40	25
2000	1843	160	188	109	87	89	48
5000	11115	—	542	308	232	258	—

Znając teoretyczną (lub empiryczną) liczbę porównań  $P_m(n)$  oraz empiryczne czasy porządkowania  $T_m(n)$ , można wyznaczyć współczynnik  $c_m(n)$  dla każdej metody porządkowania przy danym  $n$ . Obliczając współczynnik  $c_m(n)$  dla każdego  $n$  otrzymuje się, że prawie we wszystkich metodach dla odpowiednio dużych  $n$  funkcja  $c_m(n)$  dąży do pewnej stałej (różnej dla różnych metod). Zagadnienie to jednak wymaga dodatkowego opracowania. Pewne próby wyznaczenia  $c_m(n)$  przeprowadzono np. w pracach [8] i [10]. Znajomość liczby porównań  $P_m(n)$  oraz współczynnika  $c_m(n)$  dla danej metody porządkowania i określonej maszyny cyfrowej pozwala oszacować czas porządkowania ciągów o dowolnej długości  $n$ .

### 8. Przykłady zastosowań metod porządkowania

Jedną z dziedzin zastosowań metod porządkowania jest statystyka matematyczna. Zagadnienie porządkowania można rozpatrywać jako zagadnienie znajdowania kolejnych statystyk pozycyjnych. Przypomnimy w tym celu definicję statystyki pozycyjnej [4].

Niech

$$(X_1, X_2, \dots, X_n)$$

będzie  $n$ -wymiarowym wektorem losowym. Rozważmy zmienną  $\zeta_k^n$  będącą funkcją tego wektora losowego, określoną następująco: Ustawmy każdy zespół wartości  $(x_1, x_2, \dots, x_n)$  wektora losowego  $(X_1, X_2, \dots, X_n)$  niemalejąco, tj. tak, aby uzyskać ciąg  $x_{r_1}, x_{r_2}, \dots, x_{r_n}$  spełniający nierówność

$$x_{r_1} \leq x_{r_2} \leq \dots \leq x_{r_n}.$$

Statystyką pozycyjną  $\zeta_k^n$  nazywamy zmienną losową będącą funkcją wektora losowego  $(X_1, X_2, \dots, X_n)$ , która przybiera  $k$ -tą co do wielkości wartość, tj. wielkość  $x_{r_k}$  każdego z możliwych zespołów wartości  $(x_1, x_2, \dots, x_n)$ .

Sformułowanie zagadnienia porządkowania jako znajdowania kolejnych statystyk pozycyjnych implikuje automatycznie przykłady zastosowań metod porządkowania. Są to na przykład takie testy nieparametryczne jak: test Smirnowa – Kołmogorowa, test Wilcozona oraz test porządkowy Galtona. Ponadto bardzo często stosuje się metody wyznaczania miejsca elementu w ciągu do budowy szeregów rozdzielczych, do budowy tablic wielodzielczych a także do wyznaczania współczynnika zależności Z. Hellwiga [9]. W pracy [16] zostały sporządzone procedury w języku ALGOL 60 na zweryfikowanie hipotezy nieparametrycznej za pomocą testu  $\lambda$  Kołmogorowa – Smirnowa, na budowę szeregu rozdzielczego oraz na współczynnik zależności Z. Hellwiga.

Statystyka matematyczna nie stanowi jednakże głównej dziedziny zastosowań metod porządkowania. Metody te stosuje się np. do porządkowania węzłów w zagadnieniach sieciowych, a więc w metodzie Pert, do porządkowania wyników uzyskanych metodą Monte Carlo, ze szczególnym uwzględnieniem metod programowania matematycznego oraz do niektórych zagadnień z teorii prognozowania.

Z porządkowaniem informacji spotykamy się we wszystkich instytucjach, w których prowadzona jest ewidencja i sprawozdawczość. Szczególne znaczenie mają zagadnienia szybkiego wyszukiwania i dostarczania informacji. Wyszukiwanie w zbiorze danych potrzebnych informacji bądź też pewnego podzbioru informacji, a następnie uporządkowanie go według zadanej relacji jest niezmiernie ważne przy podejmowaniu odpowiednich decyzji przez kierownictwo placówki (bądź jednostki nadrzędnej), której te dane dotyczą.

Z zagadnieniem porządkowania i podziału informacji spotykamy się często przy opracowywaniu badań ankietowych. Na przykład może to być badanie struktury pracowników zatrudnionych w przemyśle, czy w innych gałęziach gospodarki narodowej według wieku, zawodu, stażu pracy, wykształcenia, itp. Przyporządkowując odpowiednie kody cyfrowe zawodom oraz poziomom wykształcenia, możemy dany zbiór pracowników uporządkować raz ze względu na wiek, drugi raz ze względu na wykształcenie itp. Po dokonaniu porządkowania ze względu na jedno kryterium, np. ze względu na wykształcenie, można uporządkować pracowników w danej grupie wykształcenia ze względu na wiek, staż pracy itp.

Na ogół wszelkie zbiory informacji niezbędne przy zarządzaniu przedsiębiorstwem powinny być uporządkowane np. po to, aby szybko wyszukiwać potrzebne informacje oraz aby uaktualniać stare zbiory przez dołączanie nowych informacji.

Podobnie opracowywane są katalogi biblioteczne. Uporządkowanie leksykograficzne pozwala na szybkie wyszukanie danej pozycji, jak również na łatwe i szybkie dołączenie nowych pozycji do katalogu.

W związku z tym, że mamy do czynienia ze zbiorami informacyjnymi rzędu milionów danych dąży się do tego, aby zarówno wyszukiwanie informacji, jak też porządkowanie realizować za pomocą nowoczesnych maszyn cyfrowych. Ale nawet w automatycznym przetwarzaniu informacji porządkowanie (sortowanie) zajmuje około 70% czasu przeznaczonego na całość obliczeń. Fakt ten jest tak wymowny, że nie trzeba uzasadniać dlaczego poszukuje się coraz szybszych i efektywniejszych metod porządkowania. Zagadnienie to jest niezwykle aktualne w chwili obecnej, gdy tworzy się w Polsce Krajowy System Informacji (bank danych). Wiąże się to z wieloma bardzo

ważnymi problemami takimi jak tworzenie zbiorów na różnych nośnikach informacji, porządkowanie tych zbiorów w różnych rodzajach pamięci maszyn cyfrowych oraz odpowiednie przetwarzanie i dostarczanie potrzebnej informacji.

### Bibliografia

- [1] З.В. Алферова, М.А. Волович, *Сортировка информации с помощью электронных вычислительных машин*, Москва 1965.
  - [2] M. H. van Emden, *Increasing the efficiency of quicksort*, CACM, 13, (11), 1970.
  - [3] F. P. Fisher i G. F. Swindle, *Systemy programowania maszyn cyfrowych*, Warszawa 1971.
  - [4] M. Fis z, *Rachunek prawdopodobieństwa i statystyka matematyczna*, Warszawa 1967.
  - [5] J. Flores, *Analysis of internal computer sorting*, JACM, 1, 1961.
  - [6] R. M. Frank and R. B. Lazarus, *A high speed sorting procedure*, CACM, 3, 1960.
  - [7] R. Griffin and K. A. Redish, *Remark on algorithm 347, an efficient algorithm for sorting with minimal storage*, CACM, 13, (1), 1970.
  - [8] M. H. Hall, *Method of comparing the time requirement of sorting methods*, CACM, 6, (5), 1963.
  - [9] Z. Helwig, *On the measurement of stochastic dependence*, Zastosowania Matematyki, 10 (1969).
  - [10] T. N. Hibbard, *An empirical study of minimal storage sorting*, CACM, 6, (5), 1963.
  - [11] С.С. Лавров, Л.И. Гончарова, *Автоматическая обработка данных. хранение информации в памяти ЭВМ*, Москва 1971.
  - [12] H. Rasiowa, *Wstęp do matematyki współczesnej*, Warszawa 1971.
  - [13] R. S. Scowen, *Algorithm 271, Quicksort M1*, CACM, 8, (11), 1965.
  - [14] D. L. Shell, *A high speed sorting procedure*, CACM, 3, 1959.
  - [15] R. C. Singleton, *An efficient algorithm for sorting with minimal storage*, CACM, 12, (3), 1969.
  - [16] E. Trybuś, *Metody porządkowania zbiorów skończonych* (rozprawa doktorska), WSE, Wrocław, 1971.
  - [17] W. M. Turski, *Struktury danych*, Warszawa 1971.
  - [18] В.С. Зубов, *К вопросу о классификации методов внутренней сортировки*, Цифровая вычислительная техника и программирование 4, 1968.
-