mgr inż. LESŁAW KOŁAKOWSKI

Przemysłowy Instytut Automatyki
i Pomiarów MERA–PIAP

Warszawa

O PEWNEJ METODZIE ORGANIZACJI ZBIORÓW DANYCH

W artykule przedstawiono organizację zbiorów danych tzw. metodą tablic rozproszonych. Uwagi teoretyczne zostały poparte opisem rozwiązania konkretnego problemu, w którym wykorzystano prezentowaną metodę. Zamieszczono schematy blokowe ważniejszych programów.

1. Wstęp

Stosowanie maszyn cyfrowych do przetwarzania informacji wiąże się z koniecznością organizowania różnych zbiorów danych. Sposób organizacji zbioru ma istotny, a czasem decydujący wpływ na efektywność rozwiązania całego problemu.

Istnieje kilka podstawowych operacji, które są charakterystyczne dla wielu problemów wyszukiwania informacji. Autorzy pracy [2] wymieniają siedem takich operacji:

MEMBER (a,S) – sprawdź, czy a jest elementem zbioru S

INSERT (a,S) – zastąp zbiór S zbiorem $S + \{a\}$

DELETE (a,S) – zastąp zbiór S zbiorem $S - \{a\}$

UNION (S1,S2,S3) – zastąp zbiory S1 i S2 zbiorem S3 = S1 + S2. Dla uproszczenia: celem uniknięcia konieczności usuwania powtórzeń zakłada się, że zbiory S1 i S2 są rozłączne.

FIND (a) – drukuj nazwę zbioru, do którego należy element a. Jeżeli a należy do kilku zbiorów to efekt działania operacji jest nieokreślony.

SPLIT (a,S) – operacja ta powoduje podzielenie zbioru S na takie dwa zbiory S1 i S2, że

$$S1 = \{b \mid b <= a \text{ i } b \in S\} \text{ i } S2 = \{b \mid b > a \text{ i } b \in S\}$$

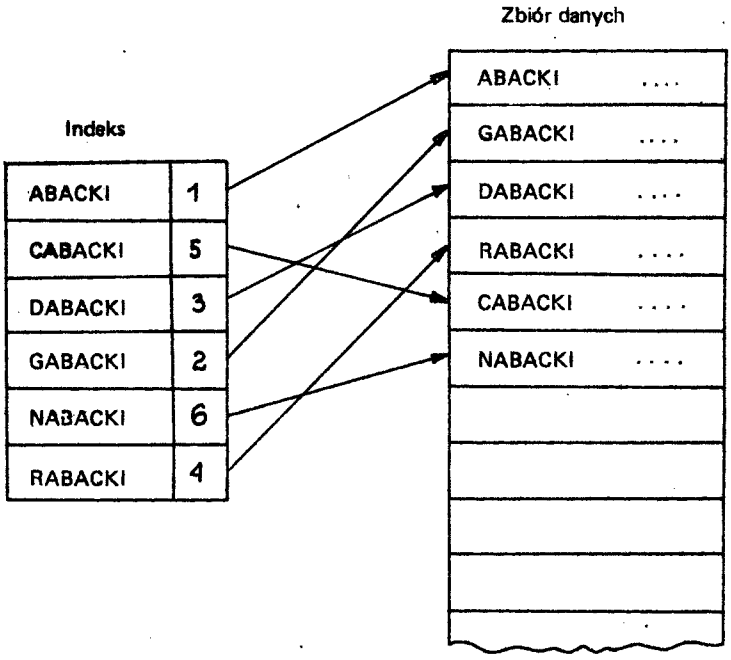
MIN (S) – drukuj najmniejszy (ze względu na relację <=) element zbioru S.

Można rozważać różne struktury danych pod kątem możliwości realizacji ciągów wymienionych wyżej operacji. W niniejszej pracy przedstawiono taką organizację zbioru danych, która pozwala na szybkie wykonywanie trzech pierwszych z nich. Innymi słowy jest to struktura, która pozwala na łatwe wstawianie elementów do zbioru, ich usuwanie oraz wyszukiwanie. Wbrew pozorom istnieje wiele problemów praktycznych, które można sprowadzić do wykonywania trzech wymienionych wyżej operacji. Klasycznym przykładem mogą tu być tablice symboli dla kompilatorów.

2. „Rozpraszanie" jako metoda organizacji zbiorów danych

Rozpatrzmy dla przykładu listę nazwisk pracowników jakiejś fabryki. Z nazwiskiem każdego pracownika łączą się takie dane jak: stanowisko, wykształcenie, wiek, staż pracy itp. Można utworzyć zbiór

danych o wszystkich pracownikach, a dostęp do zbioru zapewnić poprzez utworzenie pomocniczej struktury informacyjnej zwanej indeksem. Indeks zawiera nazwiska wszystkich pracowników, które są identyfikatorami poszczególnych rekordów zbioru danych oraz numery tych rekordów. Schematycznie przedstawia to rys. 1.

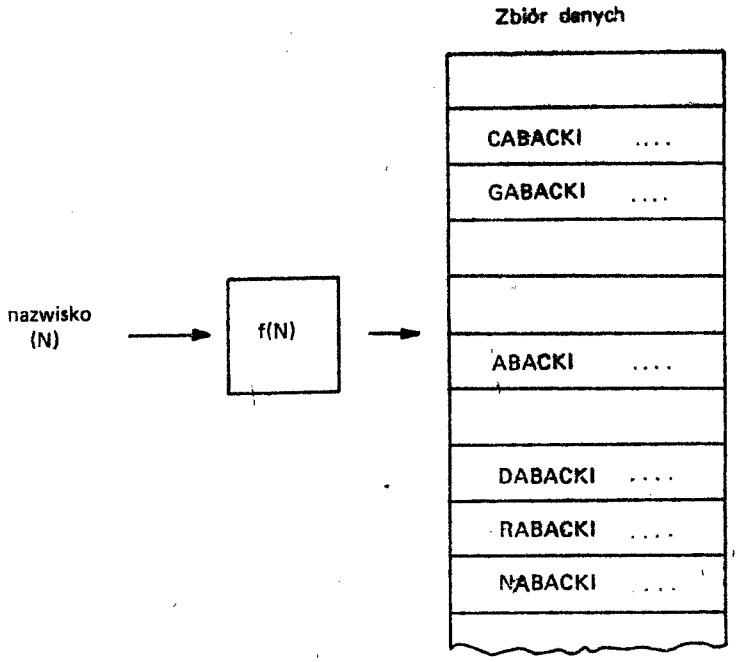


Rys. 1.

Można zauważyć, że rys. 1 obrazuje funkcję przyporządkowującą nazwiskom pracowników adresy rekordów w pamięci. Gdyby udało się funkcję tę określić algorytmicznie to indeks nie byłby potrzebny (ilustruje to rys. 2).

Metoda organizowania zbiorów danych przy pomocy takich funkcji nosi nazwę metody tablic rozproszonych, a same funkcje określa się mianem funkcji mieszających.

Liczba możliwych identyfikatorów rekordów (zwanych także kluczami) jest zwykle bardzo duża, a czasem nawet praktycznie nieograniczona jak w przypadku zbioru nazwisk pracowników przedsiębiorstwa. Funkcja mieszająca nie może być więc różnowartościowa czyli taka, która przyporządkowuje każdej potencjalnej wartości klucza oddzielne miejsce w pamięci. Oznacza to, że będą się zdarzały sytuacje zwane konfliktami, polegające na przyporządkowaniu kilku rekordom o różnych kluczach tego samego adresu w pamięci. Praktyczne stosowanie metody tablic rozproszonych wymaga zatem rozwiązania dwóch problemów: wyboru funkcji mieszającej i usunięcia zaistniałych konfliktów.



Rys. 2

3. Sposoby określania funkcji mieszającej

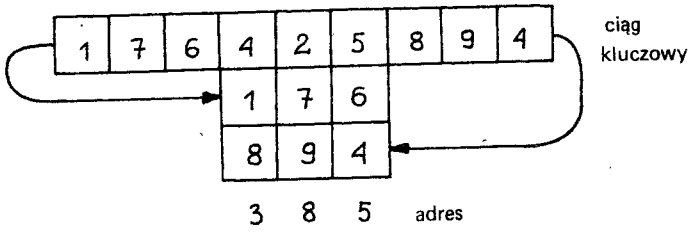
Wprowadźmy na początek kilka oznaczeń

k – klucz (ciąg cyfr)
n – liczba kluczy występujących w danym zagadnieniu
p – rozmiar tablicy (liczba rekordów)
α – współczynnik zapełnienia tablicy $(\frac{n}{p})$
d_i – adres (nr rekordu) w tablicy
h(k) – funkcja mieszająca (rozpraszająca); $h(k) = d_0$
$s(d_0, i)$ – funkcja szukająca
i – wskaźnik kolejnych kroków szukania: $0 \leqslant i \leqslant p-1$.

Jednym z najdawniejszych sposobów rozmieszczania rekordów w tablicy jest tzw. randomizacja. Adres otrzymuje się jako liczbę pseudolosową, wygenerowaną z wartości danego klucza k przy użyciu generatora liczb pseudolosowych. Można to zrobić np. następująco: ze środka klucza traktowanego jako ciąg cyfr wydzielamy pewien fragment. Otrzymaną w ten sposób liczbę podnosimy do kwadratu, z wyniku znowu wybieramy pewien fragment i traktujemy jak szukany adres. Powyższa metoda nie daje zbyt dobrych rezultatów, szczególnie w sytuacji, gdy klucze zawierają zera (występuje wtedy tendencja do generowania adresów zerowych). Średnią liczbę konfliktów można wyrazić wzorem (patrz [1]):

$$E = n - p + p \cdot (1 - \frac{1}{p})^n$$

Innym wariantem randomizacji jest tzw. metoda składania (stosowana zwykle dla długich ciągów kluczowych).
Otrzymywanie adresu ilustruje rysunek 3.



Rys. 3

Bardzo często stosowaną metodą generowania adresów jest tzw. metoda dzielenia. Funkcja mieszająca o postaci:

$$d_0 = h(k) = k \cdot \mathbf{mod}\ p$$

pozwala na równomierne „rozpraszanie" rekordów po całym obszarze zbioru przy małej wrażliwości na rozkład funkcji prawdopodobieństwa poszczególnych kluczy, pod warunkiem odpowiedniego doboru dzielnika „p". Wpływ wartości liczby „p" na skuteczność rozpraszania można zilustrować następującym przykładem:
Dla zbioru kluczy, który w większości składa się z liczb parzystych przyjęto parzysty dzielnik „p". W efekcie otrzymywane adresy są także w większości parzyste, co powoduje gwałtowny wzrost liczby konfliktów.
Autorzy prac [1], [2] proponują, żeby „p" było liczbą pierwszą lub liczbą nieparzystą bez małych dzielników.
Podobnie skuteczna i prosta jest metoda mnożenia. Funkcja mieszająca ma postać:

$$d_0 = h(k) = \text{entier}\ [p \cdot (\tfrac{a}{b} \cdot k)\ \mathbf{mod}\ 1]$$

gdzie a i b są liczbami względnie pierwszymi, x**mod**(1) oznacza część ułamkową liczby x, a entier [x] liczbę całkowitą nie większą od x.
Przedstawione wyżej funkcje mieszające były niezależne od funkcji rozkładu prawdopodobieństwa wystąpienia poszczególnych kluczy. Istnieją metody wykorzystujące znajomość funkcji rozkładu klucz przy tworzeniu funkcji mieszających. Znając np. pewne dane statystyczne o znakach lub ciągach znaków występujących w reprezentacjach kluczy można niektóre z tych znaków pominąć tak, aby pozostałe można było traktować jako adres w tablicy rozproszonej. Obszerną literaturę na temat różnych funkcji mieszających, w tym także zależnych od rozkładu kluczy, można znaleźć w pracy [1].
Omawiając możliwości zastosowania różnych funkcji rozpraszających należy wyodrębnić dwa obszary ich zastosowań. Pierwszy to tablice danych w pamięci operacyjnej, których przykładem są wspomniane już wcześniej tablice symboli kompilatorów, natomiast drugi to różnego rodzaju zbiory danych w pamięciach zewnętrznych. Czas dostępu do pamięci zewnętrznej jest na tyle długi w porównaniu z cza-

sem wykonywania rozkazów w procesorze, że wybór skomplikowanych funkcji mieszających nie ma większego wpływu na czas dostępu do rekordu w zbiorze. W przypadku tablic w pamięci operacyjnej czas obliczania wartości funkcji mieszającej jest istotny w porównaniu z czasem dostępu do pamięci. W tego rodzaju sytuacjach wybiera się zwykle proste funkcje mieszające, które można zrealizowac przy pomocy prostych operacji języka wewnętrznego maszyny jak np. przesuwanie, testowanie z maską itp.

4. Metody usuwania konfliktów

Istnieją dwa, zasadniczo odmienne, sposoby usuwania konfliktów zaistniałych przy obliczaniu adresów rekordów dla różnych wartości kluczy.

W pierwszym przypadku zakłada się istnienie, prócz właściwej tablicy rozproszonej, tzw. obszaru nadmiarowego. W obszarze nadmiarowym umieszczone są synonimy, czyli rekordy, którym przydzielono ten sam adres. Drugie podejści zakłada rozwiązanie problemu konfliktów w granicach właściwej tablicy rozproszonej.

Jako ilustrację rozwiązywania problemu konfliktów za pomocą obszaru nadmiarowego przedstawmy metodę list synonimów w obszarze nadmiarowym. W metodzie tej synonimy łączy się w listy. Do rekordu należy zatem dodać jedno pole na wskaźnik—odsyłacz do rekordu następnego. We właściwej tablicy rozproszonej znajduje się pierwszy (w kolejności wprowadzania do bazy danych) z grupy synonimów, natomiast następne są połączone w listę umieszczoną w obszarze nadmiarowym. Kwestia wyboru adresu w obszarze nadmiarowym pozostaje otwarta. Można np. wprowadzić wskaźnik ostatniego zajętego rekordu.

Przy założeniu, że funkcja mieszająca jest losowa a rekordy w obszarze nadmiarowym przydzielane są zgodnie z sugestią przedstawioną wyżej, można w przybliżeniu określić średnią liczbę prób koniecznych do wyszukania rekordu w zbiorze. Wynosi ona — patrz [1]:

$$E = 1 + \frac{\alpha}{2}$$

Ze względu na istnienie obszaru nadmiarowego można nawet przyjąć, że α zdefiniowana w punkcie 4. jako $\frac{n}{p}$ jest większa od 1, ale przy ocenie efektywności opisanej metody należy pamiętać, że tak mały wskaźnik E został osiągnięty kosztem znacznego zwiększenia zajmowanego obszaru pamięci.

Z metodami rozwiązywania problemu konfliktów bez pomocy obszaru nadmiarowego wiąże się pojęcie tzw. funkcji szukającej. Nie chodzi tu jednak o odszukiwanie informacji w tablicy rozproszonej, lecz o szukanie w niej wolnego miejsca na rekord w momencie zaistnienia konfliktu.

$$s(d_0, i) = (d_0 + a \cdot i) \bmod p$$

a — pewna stała całkowita przyjmowana zwykle jako 1.

Szukanie liniowe (nazwane tak od postaci funkcji szukającej) ma istotną wadę — grupuje zapisy w pamięci. Jeżeli funkcja h(k) przydziela kluczom k1, k2, k12 ten sam adres np. 3, a kluczom k3, k4, k11 adres 6, to próba wprowadzenia rekordu o słowie kluczowym k16, którego $d_0 = h(k16)$ również wynosi 3, wymaga przejrzenia sekwencji siedmiu rekordów. Dostęp do rekordów w tablicy przestaje być bezpośredni, staje się sekwencyjny.

Dla lepszego przedstawienia tego przypadku prześledźmy adresy przydzielane poszczególnym rekordom.

k1 — 3
k2 — 4 (rekord o adresie 3 jest już zajęty)
k3 — 6
k4 — 7 (rekord o adresie 6 jest już zajęty)

k11 — 8 (rekordy o adresach 6 i 7 są już zajęte)
k12 — 5 (rekordy o adresach 3 i 4 są już zajęte)
k16 — 9 rekordy o adresach 3, 4, 5, 6, 7 i 8 są już zajęte)

Tego rodzaju grupowanie rekordów nosi nazwę „grupowania pierwotnego". Jedną z metod uniknięcia grupowania pierwotnego jest zastąpienie liniowej funkcji szukającej funkcją kwadratową o postaci:

$$d_i = (d_0 + i^2) \bmod p$$

Zależność kwadratowa nie eliminuje jednak tzw. „grupowania wtórnego". Grupowanie wtórne w tablicy rozproszonej, w odróżnieniu od grupowania pierwotnego, polega na tym, że przecięcie się dwóch ścieżek szukania powoduje ich złączenie jedynie wtedy, gdy nastąpiło w tych samych krokach. Bardziej precyzyjnie można to wyrazić posługując się wzorem:

$$d_i(k1) = d_j(k2) \Rightarrow d_{i+1}(k1) = d_{j+1}(k2), \text{ jedynie wtedy gdy } i = j$$

Funkcja kwadratowa nie zapewni niestety przejrzenia całej tablicy rozproszonej lecz jedynie jej połowę [1]. Kolejnym krokiem na drodze do znalezienia odpowiedniej funkcji szukającej jest zależność sześcienna.

$$d_i = (d_0 + i^3) \bmod p$$

Na podstawie pewnych twierdzeń z teorii liczb, dotyczących tzw. reszt stopnia n z dzielenia *) można wykazać [1], że dla liczb pierwszych postaci:

$$p = 3j + 2, \quad \text{gdzie } j \text{ jest liczbą naturalną}$$

liczba reszt sześciennych wynosi p. Przedstawiona funkcja szukająca spełnia zatem wymagania — umożliwia generowanie adresów z pełnego zakresu od 1 do p, a także ze względu na swą nieliniowość eliminuje grupowanie pierwotne. Stosując jako funkcję rozpraszającą procedurę dzielenia modulo p można także [1] ograniczyc liczbę grupowań wtórnych.
Na zakończenie można jeszcze wspomnieć o stosowanej czasem losowej funkcji szukającej postaci:

$$d_i = (d_0 + \text{los}(i)) \bmod p$$

gdzie los(i) oznacza generator liczb losowych, dający liczby całkowite z przedziału od 0 do p—1 w rozkładzie równomiernym.

5. Problemy związane z usuwaniem rekordów. Reorganizacja zbiorów

Po określeniu funkcji mieszającej i funkcji szukającej można przystąpić do wprowadzania zapisów do tablicy rozproszonej. Pozostają jednak do rozwiązania pewne problemy związane z usuwaniem rekordów z tablicy i tzw. ich kompresją lub ogólniej reorganizacją. Łatwo zauważyć, że miejsca po usunię-

*) *Resztą stopnia n z dzielenia przez liczbę pierwszą p jest liczba całkowita r, jeżeli istnieje taka liczba całkowita a, że spełniona jest zależność:*

$$r \cdot \bmod p = a^n \cdot \bmod p$$

tym rekordzie nie można traktować jako rekordu wolnego. Rekordy tworzą bowiem (w wyniku istnienia konfliktów) pewne sekwencje bądź to w obszarze nadmiarowym, bądź też w obszarze właściwej tablicy rozproszonej. Wstawienie wewnątrz sekwencji rekordu pustego uniemożliwia jej całkowite przeszukanie. Należy wprowadzić rozróżnienie między rekordem pustym a rekordem opróżnionym i mówić raczej o skreślaniu rekordów w tablicy rozproszonej, a nie o ich usuwaniu. Można uzyskać to rozróżnienie odmiennie oznaczając rekordy puste i rekordy skreślone.

Reorganizacja zbiorów może być dwojakiego rodzaju. Po pierwsze może ona wynikać ze zmiany warunków eksploatacji oprogramowania np. zaistniała konieczność dwukrotnego zwiększenia rozmiaru bazy danych lub ze względu na specjalny zestaw wartości kluczowych należy zmienić funkcję rozpraszającą lub szukającą. Tak postawiony problem reorganizacji najłatwiej rozwiązać tworząc nową tablicę w innym obszarze pamięci i umieszczając w niej wszystkie rekordy ze starej tablicy.

Potrzeba reorganizacji drugiego rodzaju wynika z konieczności poprawy efektywności dostępu do danych. Częste usuwanie przez użytkownika rekordów ze zbioru sprawia, że tworzy się wiele rekordów skreślonych. Wydłużają się zatem niepotrzebnie sekwencje rekordów przeglądanych podczas wyszukiwania informacji. Korzystna jest więc taka reorganizacja zbiorów (zwana zwykle kompresją), która eliminuje z niego skreślone rekordy. O ile reorganizacja pierwszego rodzaju powoduje zwykle (choć niekoniecznie) utworzenie nowego zbioru w innym obszarze, to kompresja dokonywana jest raczej w tym samym obszarze pamięci.

6. Tablica materiałowa — przykład zbioru zorganizowanego metodą tablic rozproszonych

Przedstawioną wyżej metodę tablic rozproszonych zastosowano w opracowanym w MERA—PIAP systemie sterowania i zarządzania magazynem wysokiego składowania (SEZAM). W systemie SEZAM zaistniała bowiem potrzeba utworzenia zbioru, w którym zapisane byłyby informacje o tym z jakich materiałów można wytwarzać różne części maszyn. Zbiór ten nosi nazwę „tablicy materiałowej". Wymagania stawiane tablicy materiałowej ograniczają się do stworzenia możliwości wykonywania operacji wstawiania, usuwania i odszukiwania rekordów. Ciągiem kluczowym jest 15—znakowy, alfanumeryczny identyfikator części. Prócz niego rekord zawiera trzy 15—znakowe identyfikatory materiałów, z których można tę część wykonać.

W prezentowanym rozwiązaniu zastosowano metodę dzielenia dla zdefiniowania funkcji rozpraszającej, natomiast konflikty rozwiązywano przy pomocy sześciennej funkcji szukającej, bez użycia obszaru nadmiarowego.

Użytkownik korzysta z danych zapisanych w tablicy materiałowej dzięki wprowadzonym z monitora ekranowego dyrektywom operatorskim. Tablica obsługiwana jest przez następujące dyrektywy:

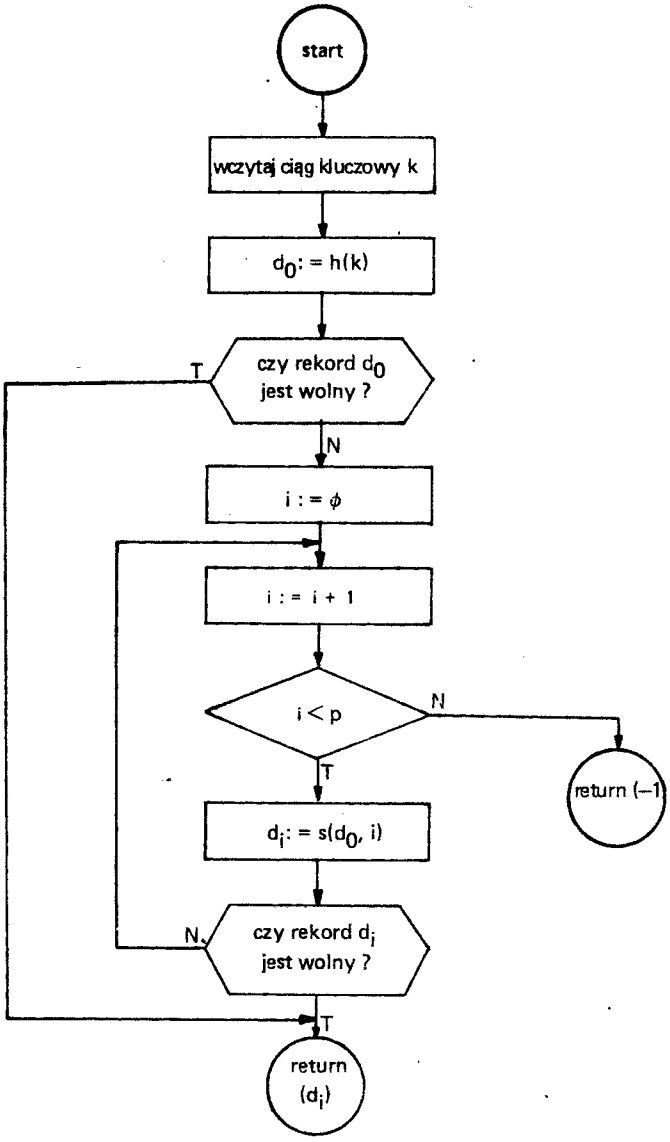
F51 idcz, idmat1, idmat 2, idmat3
F52 idcz
F53 idcz
F54

gdzie „idcz" oznacza identyfikator części, a „idmat" — identyfikator materiału.

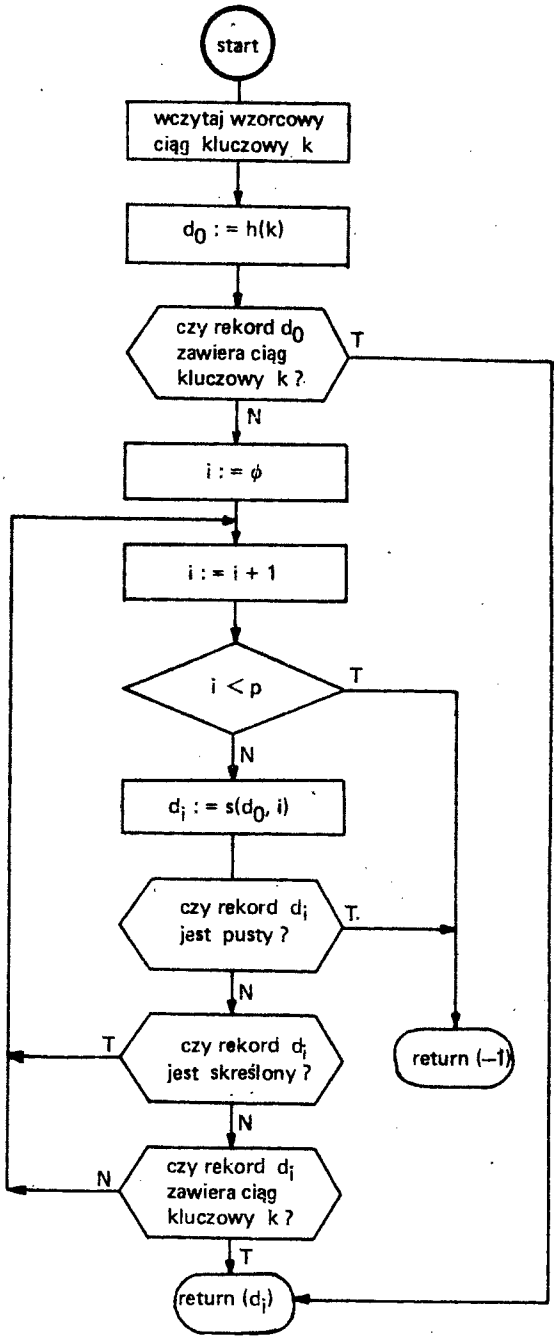
Dyrektywa F51 pozwala założyć rekord części w zbiorze. Dyrektywa F52 służy do usuwania, a ściślej mówiąc skreślania rekordów, natomiast dyrektywa F53 do wyświetlania informacji zapisanych w rekordzie. Użycie dyrektywy F54 powoduje kompresję zbioru.

W końcu artykułu zamieszczono trzy schematy blokowe programów, oznaczone kolejno liczbami 1,2, 3. Schemat 1. przedstawia program znajdujący w zbiorze wolne miejsce na rekord o podanym ciągu kluczowym. Schemat 2. dotyczy podprogramu służącego do odnajdywania w zbiorze rekordu o podanym, wzorcowym ciągu kluczowym, nastomiast schemat 3. obrazuje program kompresji zbioru.

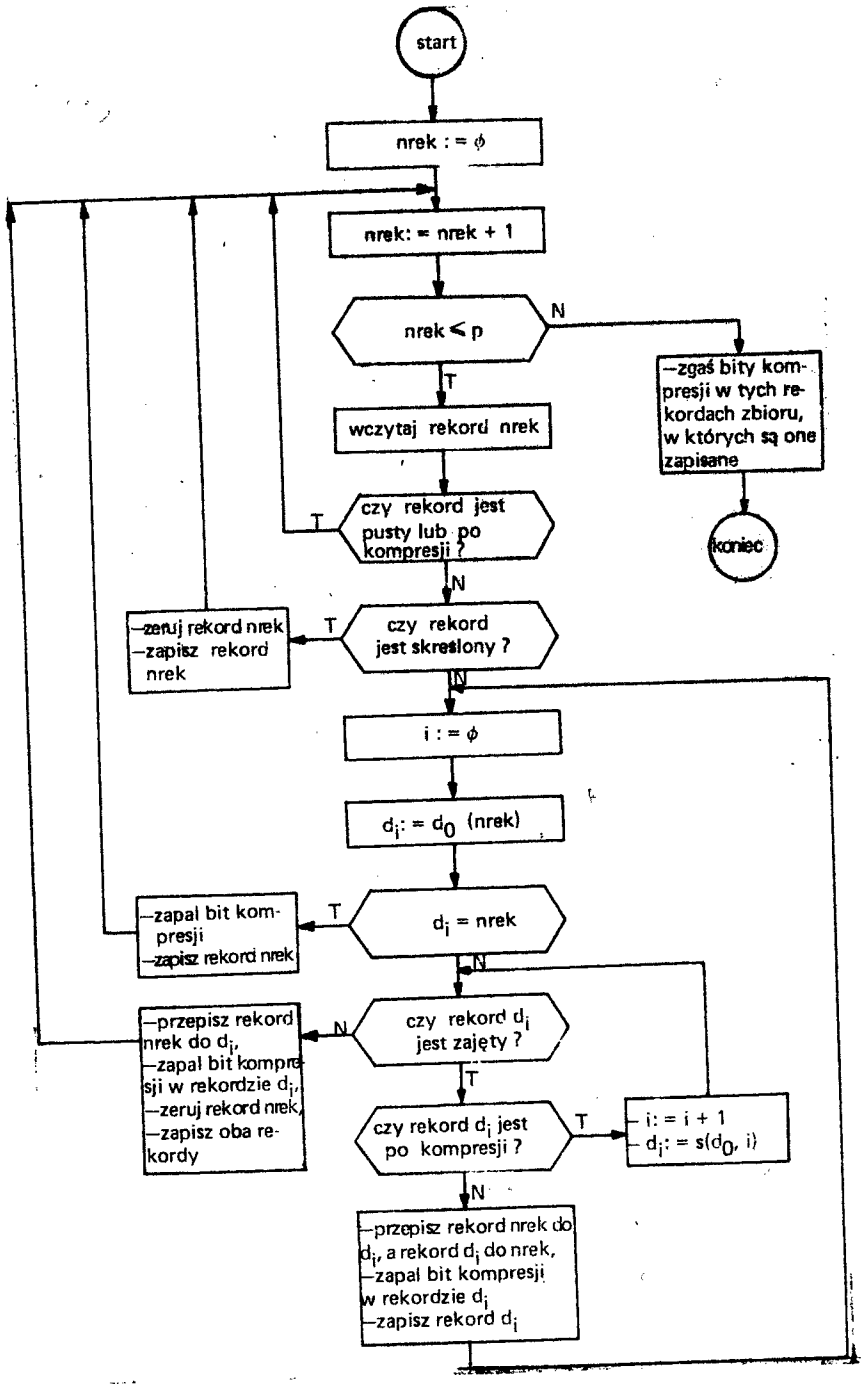
W prezentowanych schematach uwidoczniono tylko te elementy, które są istotne dla zrozumienia za-


start
wczytaj ciąg kluczowy k
d_0: = h(k)
czy rekord d_0 jest wolny ?
T
N
i : = ϕ
i : = i + 1
i < p
N
T
return (–1)
d_i: = s(d_0, i)
czy rekord d_i jest wolny ?
N
T
return (d_i)

Schemat 1


start
wczytaj wzorcowy ciąg kluczowy k
$d_0 := h(k)$
czy rekord d_0 zawiera ciąg kluczowy k ?
T
N
$i := \phi$
$i := i + 1$
$i < p$
T
N
$d_i := s(d_0, i)$
czy rekord d_i jest pusty ?
T
N
czy rekord d_i jest skreślony ?
T
return (–1)
N
czy rekord d_i zawiera ciąg kluczowy k ?
N
T
return (d_i)

Schemat 2


start
nrek : = ϕ
nrek: = nrek + 1
nrek ⩽ p
N
T
–zgaś bity kompresji w tych rekordach zbioru, w których są one zapisane
koniec
wczytaj rekord nrek
czy rekord jest pusty lub po kompresji ?
T
N
–zeruj rekord nrek
–zapisz rekord nrek
T
czy rekord jest skreślony ?
N
i : = ϕ
d_i: = d_0 (nrek)
–zapal bit kompresji
–zapisz rekord nrek
T
d_i = nrek
N
–przepisz rekord nrek do d_i,
–zapal bit kompresji w rekordzie d_i,
–zeruj rekord nrek,
–zapisz oba rekordy
N
czy rekord d_i jest zajęty ?
T
czy rekord d_i jest po kompresji ?
T
i: = i + 1
d_i: = s(d_0, i)
N
–przepisz rekord nrek do d_i, a rekord d_i do nrek,
–zapal bit kompresji w rekordzie d_i
–zapisz rekord d_i

Schemat 3

sady organizacji tablic rozproszonych. Pewnych dodatkowych wyjaśnień wymagają zastosowane oznaczenia. W każdym rekordzie wydzielono dwa pola na dane o charakterze organizacyjnym: tzw. bit kompresji i wartość funkcji h(k) dla klucza zawartego w rekordzie. Bit kompresji pozwala stwierdzić, czy dany rekord jest przed, czy po kompresji. Druga informacja organizacyjna – h(k), oznaczona na schemacie d(nrek) pozwala stwierdzic, czy rekord zajmuje miejsce wyznaczone na podstawie funkcji mieszającej czy szukającej tzn, czy jest w „swoim d_0", czy nie. Oznaczenie return (d_i) wskazuje, że podprogram zwraca do programu, który go wywołał, liczbę d_i.

7. Uwagi końcowe

Trudno formułować jakieś ogólne wnioski dotyczące efektywności opisanego oprogramowania, ponieważ wymagałoby to przeprowadzenia systematycznych badań statystycznych dla różnych zestawów kluczy. Porównanie poszczególnych metod organizacji tablic rozproszonych można znaleźć w pracy [1] i w zamieszczonej w niej bibliografii. W tym miejscu można tylko stwierdzić, że przeprowadzono próbę zapisania do zbioru około 500 rekordów, co odpowiadało w przybliżeniu jego założonej pojemności i w każdym przypadku czas reakcji maszyny był prawie niezauważalny dla użytkownika. Jest to oczywiście miernik bardzo przybliżony, daje jednak pewien pogląd na możliwość stosowania przedstawionej metody tworzenia zbiorów danych.

Opisane oprogramowanie zostało napisane, podobnie jak cały system SEZAM, w języku MACRO–11, który jest assemblerem maszyn serii PDP–11 firmy DEC i kompatybilnych z nimi maszyn SM–4. Pewne trudności wynikały z braku w dostępnej maszynie SM–4 procesora zmiennoprzecinkowego, dostarczającego specjalnych rozkazów znacznie ułatwiających i przyspieszających wszelkie obliczenia. Trudności te pokonano używając w kilku miejscach podprogramów w języku FORTRAN, który ma wbudowane mechanizmy programowe umożliwiające wykorzystywanie operacji zmiennoprzecinkowych.

Literatura

[1] R. Jagielski – Tablice rozproszone, WNT, Warszawa 1982.

[2] A.V.Aho, J,E.Hopcroft, J.D.Ullman – Projektowanie i analiza algorytmów komputerowych, PWN, Warszawa 1983.

[3] J. Martin – Organizacja baz danych, PWN, Warszawa 1983.

[4] Komputerowy system zarządzania i sterowania magazynem wysokiego składowania. Zeszyny Naukowe Politechniki Śląskiej nr 812 (zeszyt 76), Gliwice 1984. [R. Zbiegieni i inni].