Ing.Vilém ČIMBURA
VVUÚ Ostrava - Radvanice

MOŽNOSTI VYUŽITÍ TECHNIK INDEXOVÝCH SEKVENCÍ S ŘETĚZENÍ VE VĚDECKOTECHNICKÝCH VÝPOČTECH

1. Úvod

Počítače 3. generace používají již jako samozřejmost vnější paměti s přímým přístupem jako jsou např. mg. disky, které nesporně obohacují možnosti programátora.

Ve vědeckotechnických výpočtech (dále VTV) je prakticky nejpoužívanějším jazykem FORTRAN. Ve verzích, které jsou implementovány na počítačích 3. generace (např. FORTRAN IV G a H na počítačích řady IBM 360 a 370, případně FORTRAN pro řadu JSEP) už jsou zahrnuty i výroky pro práci s magnetickými disky. Příslušné výroky umožňují přímý přístup k jednotlivým záznamům souboru. Programátor deklaruje počet záznamů (oblastí) v souboru, jejich maximální velikost, případně zda jde o záznam s konverzí nebo přímo ve vnitřním zobrazení a jméno tzv. sdružené proměnné. Jednotlivé oblasti (v maximální velikosti) jsou systémem jakoby očíslovány a touto adresou je možno se na ně přímo odvolávat a to jak při zápisu, tak při čtení. Adresa může být zadána prostřednictvím libovolné proměnné nebo hodnoty aritmetického výrazu. Sdružená proměnná udává po provedené operaci adresu záznamu následujícího po právě zpracovaném. Metoda odpovídá prakticky metodě REGIONAL 1 v PL/1.

Použití souborů s přímou organisací je výhodné u VTV např. v těchto případech:

- uchovávání velkých tabulek na vnějších pamětech,
- uchovávání velkých matic u nichž se požaduje možnost náhodného výběru prvků,
- tabulky skupin parametrů,
- parametry zpracování pro programové systémy pro různé druhy opakujícího se zpracování,
- soubory, jejichž věty mají proměnlivou strukturu i délku,
- jedno z nejvýhodnějších použití je souborů, jejichž věty jsou často měněny,
- při opravách vstupních dat,
- atd.

Při adresování záznamů jsou v podstatě tyto možnosti:

- přímé adresování, kdy adresa je dána přímo pořádacím kritériem souboru,
- nepřímé, kdy se k adrese dospívá oklikou, buď přepočtem nebo různými randomizačními technikami, případně pomocí tabulek nebo pomocí jiných záznamů.

U nepřímého adresování může dojít k tomu, že pro tutéž hodnotu pořádacího kritéria se vyskytuje záznamů více. To jsou tzv. synonyma, v takovém případě je nutno zajistit uložení do tzv. oblasti přeplnění a zařídit propojení synonym. Při přímém adresování je přístup rychlejší, ale prostor vyhrazený na disku pro soubor bývá využit značně nehospodárně. Naproti tomu nepřímé adresování šetří prostor, ale ze svých důvodů prodlužuje dobu přístupu k záznamu. V praxi se většinou setkáváme s nejrůznějšími technikami nepřímého adresování.

2. Technika řetězení a technika indexových tabulek.

Obě patří k technikám nepřímého adresování, všimneme si jich podrobněji a to zejména s ohledem na praktické použití.

Řetězením lze spojit záznamy, které k sobě logicky pří-

sluší, i když jsou fysicky uloženy na různých místech souboru. Spojení je provedeno pomocí tzv. směrníků, které jsou připojeny ke každému záznamu. Udávají vždy adresu následujícího nebo předchozího záznamu - článku řetězu. Definovaný obsah směrníku (např. 0) udává, že jde o konec řetězu. Celý řetěz je pak tvořen záznamy s touž vlastností. Řetězů může být v souboru několik, pak musí každý záznam obsahovat příslušný počet směrníků. Adresa prvního článku řetězu musí být někde uchována. Př. řetěz mohou tvořit záznamy parametrů všech vzorků křemene v souboru vzorků všech typů hornin. Příklady použití této techniky:

- u zmíněného už přeplnění u nepřímého adresování,
- není-li předem znám počet parametrů příslušející např. určitému vzorku, lze dodatečně pomocí řetězení připojit prakticky libovolný počet dalších parametrů.
- setřídění záznamů podle určitého kritéria bez změny jejich uložení
- atd.

Řetězů je více typů, podrobný rozbor však přesahuje rámec tohoto příspěvku (jinak viz /2-4/).

Indexové tabulky umožňují zajistit přístup k záznamům dodatečně, aniž by bylo nutno brát přitom ohled na způsob a techniky ukládání, případně na jeho časovou návaznost. Tato technika nevyžaduje další vyhrazení místa v záznamu. Každé indexové tabulce přísluší pořádací kritérium, jehož hodnoty nebo intervaly hodnot jsou argumentem tabulky. Tabulkovými hodnotami jsou pak adresy záznamů, které se vyznačují příslušnou hodnotou pořádacího kritéria. Pro hledání v tabulce existují různé, často velmi složité techniky . Někdy se tabulky nevejdou do paměti, která je k dispozici, takže se setkáváme s vícestupňovými indexy, řetězením množiny adres záznamů příslušejících danému argumentu tabulky atd. Podobně jako u řetězení lze vytvořit více indexových tabulek pro různá pořádací kritéria - mnohonásobné indexování. Problémem se pak ale stává údržba souboru, poněvadž kromě změny záznamů je třeba aktualizovat i indexové tabulky.

Obě techniky lze kombinovat, takže např. v indexové tabulce jsou uloženy počátky řetězů souboru nebo přímo indexová tabulka je z nejrůznějších důvodů řetězena apod. Jejich pomocí lze zařídit i propojení mezi soubory.

Tyto techniky úzce souvisí s principy ukládání a struktурami dat v páscích dat. V této literatuře jsou také podrobně popsány /1-5/.

3. Příklad použití diskutovaných technik.

3.1. Formulace zadání.

Při zeměměřických měřeních a pak vyhodnoceních nejrůznějšího typu jsou používána měřická pásma lišící se např. délkou, vlastnostmi tepelnými, deformačními atd., celkem asi 10 typů parametrů. K tomu ještě přistupují tabulky komparací pro každé pásmo (desítky až stovky hodnot). Pásma jsou čas od času vyřazována, nahrazována novými nebo po poškození může dojít ke zkrácení a tedy změně parametrů. Údaje o pásmech mají sloužit jako jeden ze vstupů pro různé programy, které zpracovávají měření. Vzhledem k počtu údajů a frekvenci zpracování nepřipadá v úvahu zadávání z děrných štítků. Požaduje se, aby byly parametry pásma zpřístupněny vyhodnocujícímu programu uvedením identifikačního kódu pásma. Počet pásem je cca 30.

3.2. Analýza úlohy, návrh řešení.

Nejvýhodnější se této situace jeví přímá organizace souboru pásem vzhledem k častým změnám v souboru a k rychlosti výběru hledaného pásma ze souboru.

Soubor bude obsahovat indexovou tabulku pásem, kde každému existujícímu identifikačnímu kódu pásma bude příslušet adresa počátku parametrů pásma. V označování pásem není žádná zákonitost, nemá smysl je tedy třídit. Parametry i jiné logické celky v souboru mohou být děleny do segmentů, které jsou navzájem propojeny řetězením. Vhodným stanovením veli-

kosti segmentu, tj. vlastně i maximální velikosti záznamu souboru, lze optimálně řešit kompromis mezi úsporou místa na disku a rychlostí výběru. Parametry mohou být identifikovány buď pořadím ve zřetězení segmentů nebo jsou indikovány typy segmentů a pak pořadím v nich, případně může být u každého údaje uveden typ a v případě potřeby i počet údajů náležejících témuž typu. Poslední způsob umožňuje přidávat kdykoli další parametry zcela libovolně - je zcela obecný.Vyžaduje však více paměti. Stejně lze zajistit i "roztažitelnost" indexové tabulky. Předpokládáme-li u ní segmentovou strukturu, lze uchovávat prakticky libovolný počet pásem. Přítomnost dohodnutých zvláštních znaků indikuje konec řetězu nebo prázdnou část záznamu. Při změnách nebo vypouštění pásem může dojít ke zrušení některých záznamů, a tedy k jejich uvolnění pro další zápis. Registrace těchto volných záznamů může být prováděna různě např. těmito způsoby:

- ve zvláštním řetězu uchováme volné adresy souboru vzniklé zrušením,přičemž pro rychlejší orientaci máme někde poznamenáno, kolik těchto adres je,
- vyčkáme, až je celý soubor "popeán" (včetně zrušených záznamů) a pak provedeme reorganizaci.

Ukazuje se nutnost vytvoření dalšího význačného záznamu (dále úvodní), kde by byla uložena adresa posledního obsazeného záznamu souboru (myšleno nejvyšší číslo, kterého dosud adresa nabyla). Další záznam se pak provede na adresu vyšší o 1 nebo se obsadí poslední adresa řetězu volných adres. Zmíněný úvodní záznam by tedy musel obsahovat i adresu posledního segmentu řetězu volných adres. Směrníky řetězu volných adres tento propojí od posledního segmentu k prvnímu (výběr z tohoto řetězu se děje podle pravidla LIFO). Jde o tzv. zpětný řetěz. Podobně je organizován i řetěz segmentů indexové tabulky, úvodní záznam musí tedy obsahovat adresu posledního segmentu indexové tabulky. Při vypouštění pásem z indexové tabulky je pak možno postupovat např. těmito způsoby:

- vymaže se příslušná položka v indexové tabulce
- pak je možno buď z posledního segmentu indexové tabulky přepsat poslední položku na uprázdněné místo
- nebo je možno kontrolovat při prohledávání segmentů údaj, který by udával, zda jsou v segmentu ještě nějaké nezrušené položky (to by bylo pro tento údaj další místo v segmentu navíc).

Výhodněji se jeví první způsob, nemůže tak dojít k velkým"dírám" v tabulce a hledání bude tedy úspornější a rychlejší. Řetěz segmentů indexové tabulky musí být ovšem zpětný.

Nad tímto souborem musí pracovat program nebo programy, které zajistí funkce uložení, zrušení a opravu pásma a reorganizaci souboru, kterou je vhodné čas od času provést. Indexová tabulka by měla být bez "děr" a stejně tak by měly na sebe navazovat segmenty parametrů jednotlivých pásem.

Vyhledávání je v takovém souboru jednoduché. V úvodním záznamu zjistíme adresu koncového segmentu indexové tabulky a začneme tabulku prohledávat. Vzhledem k počtu pásem nemá smysl používat speciálních hledacích technik. Po nalezení daného identifikačního kódu pásma může začít na příslušné adrese čtení parametrů pásma.

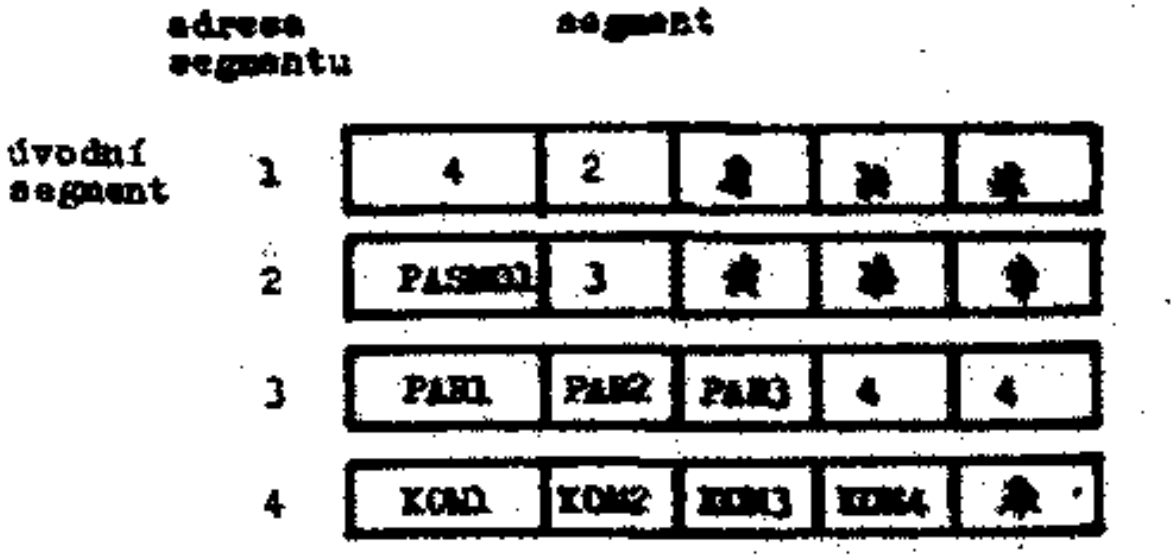
Výhodou tohoto uspořádání je, že veškeré údaje potřebné pro operace nad souborem jsou v souboru samém přímo obsaženy.

Na následujícím obrázku jsou uvedeny první záznamy souboru v situacích a) po uložení prvního pásma,
b) po uložení druhého pásma,
c) po uložení třetího pásma.

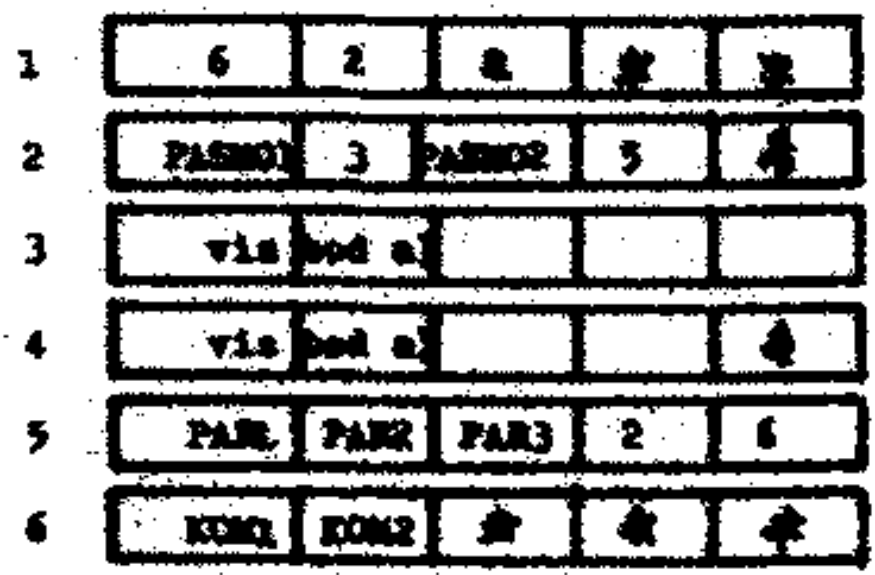
Je přitom použito těchto označení a dán tento rozsah úlohy: Délka segmentu je 5 slov, počet stálých parametrů pásma 3 (jsou označeny PAR1, PAR2, PAR3) - mohou být tedy identifikovány pořadím v prvním segmentu parametrů; PASMO1, PASMO2 a PASMO3 jsou identifikační kódy pásem - počet pásem je 3. Nepoužívá se řetězu volných adres. KOM1, KOM2, KOM3 atd označuje komparace pro jednotlivá pásma. Počet komparací je

dán ve 4.slově prvního segmentu parametrů. V každém segmentu je 5.slovo ukazatel na odpovídající následující segment.
V úvodním segmentu je v prvním slově poslední obsazená adresa souboru, ve druhém slově adresa posledního segmentu indexové tabulky. Prázdný znak je ␣ .

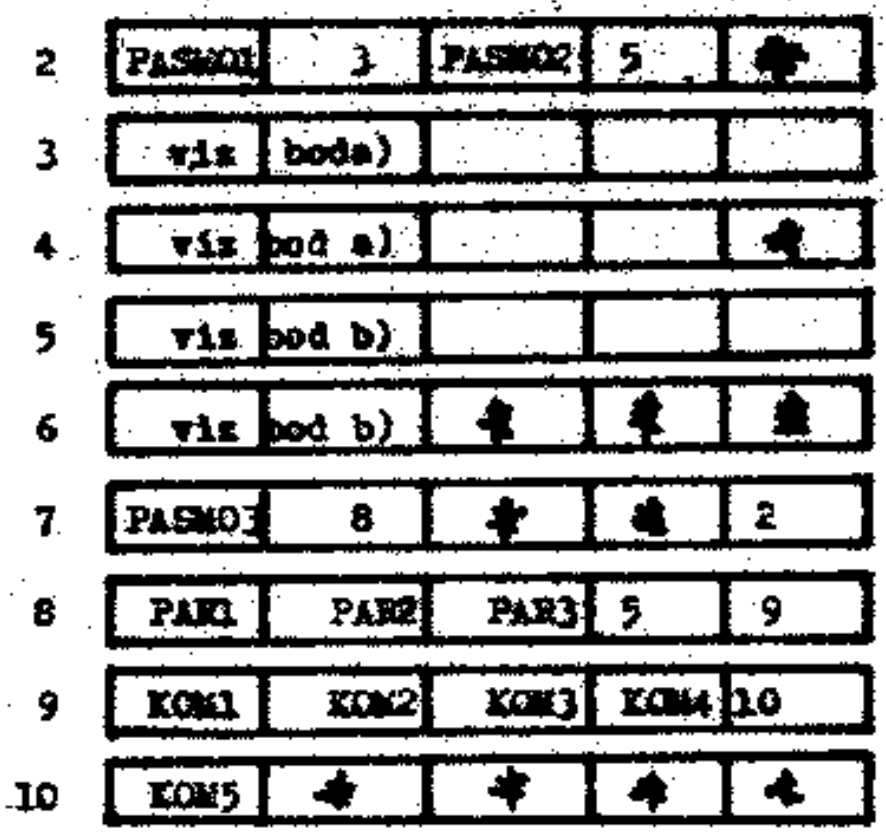
ad a) první pásmo má 4 komparace



ad b) druhé pásmo má 2 komparace



ad c) třetí pásmo má 5 komparací



Segment 1 je úvodní segment, indexová tabulka pásem je v segmentech 2 a 7, parametry 1.pásma v segmentech 3 a 4, parametry 2.pásma v segmentech 5 a 6, parametry 3. pásma v segmentech 8, 9 a 10.

Tento příklad neřeší samozřejmě zadání do detailu, cílem bylo spíše ukázat možnosti, které se v takovém případě naskytají. S výhodou tu bylo použito jak indexové tabulky, tak řetězení záznamů.

4. Metoda třídění jako příklad použití dynamických technik

4.1. Úvodní úvahy.

Vymezení úlohy je takové:
- jako programovací jazyk FORTRAN
- jde o vnitřní třídění

- klíče jsou poměrně krátké (do 20 znaků)
- tříděné soubory mají maximálně 5000 vět

Jak známo, FORTRAN není vybaven výroky pro detailní operace se znaky, znakovými řetězci apod. Nelze rovněž pracovat přímo s byty. Metoda musela tyto skutečnosti respektovat a bylo proto nutné najít cesty, jak tato omezení obejít.

Hrubé schéma vypadá asi takto:

1. Na magnetickou diskovou paměť se zaznamená tříděný soubor a to v přímé organizaci.

2. Z vět souboru jsou čteny klíče vět do hlavní paměti počítače. Je přitom vytvářen řetěz klíčů v pořadí, jak jsou čteny, tzn. v paměti musí být také směrníky příslušející každému klíči.

3. Je provedeno třídění klíčů do požadované posloupnosti, přičemž se nepřepisují a nezaměňují klíče, ale směrníky. Ty pak po ukončení třídění vytvářejí s klíči řetěz, který odpovídá požadovanému předpisu o pořadí klíčů.

4. Ze souboru na disku (viz bod 1) jsou čteny věty souboru podle směrníků utříděného řetězu klíčů a zapisovány na jiné médium buď v sekvenční nebo přímé organizaci. V posledním případě je možná tato varianta: původní soubor obsahuje směrníky už ve svých větách, takže po třídění mohou být přepsány směrníky utříděného řetězu klíčů. Soubor na disku lze pak procházet v pořadí setřídění.

Třídění je prováděno v paměti, celý soubor najednou. Vzhledem k požadavkům sdělení je to postačující i při maximálním rozsahu. V případě větších úloh je nutné provést další zobecnění, která však na vlastní metodu třídění nemají podstatný vliv. Tyto možnosti budou diskutovány dále.

4.2. Vlastní metoda třídění.

Tato metoda se týká vlastně bodu 3 uvedeného postupu.

Předpokládejme zatím, že klíč vět je tvořen jen jedním znakem, má jen jednu pozici. Pak po projití všech klíčů je

možno vytvořit dílčí řetězy, které spojí klíče se stejným znakem. Vytvoříme si tabulku, ve které bude zaznamenán pro každý možný znak počátek a konec dílčího řetězu (pomocí adresy klíče - tj. jejího pořadí od počátku souboru), případně kolik článků každý dílčí řetěz obsahuje. Tabulku nazvěme dále tabulka dílčích řetězů (TDŘ). Pořadí znaků v tabulce je dáno pořadím podle požadavků na třídění (vzestupné, sestupné, obecně určené pořadí). Propojením dílčích řetězů v pořadí, jak za sebou následují v tabulce lze získat řetěz přes celý soubor, který pak bude odpovídat požadovanému pořadí.

Vzhledem k principu metody je možné pracovat s libovolnou množinou znaků (které jsou ovšem pro daný počítač definovány) a volit i libovolné pořadí pro třídění. Prostředkem, který to umožní prakticky, bude konverzní tabulka, která provede jednoznačné přiřazení mezi znaky zvolené množiny a jejich pořadím v TDŘ. Konverzní tabulku lze vytvořit pro každé zadání přesně podle jeho požadavků.

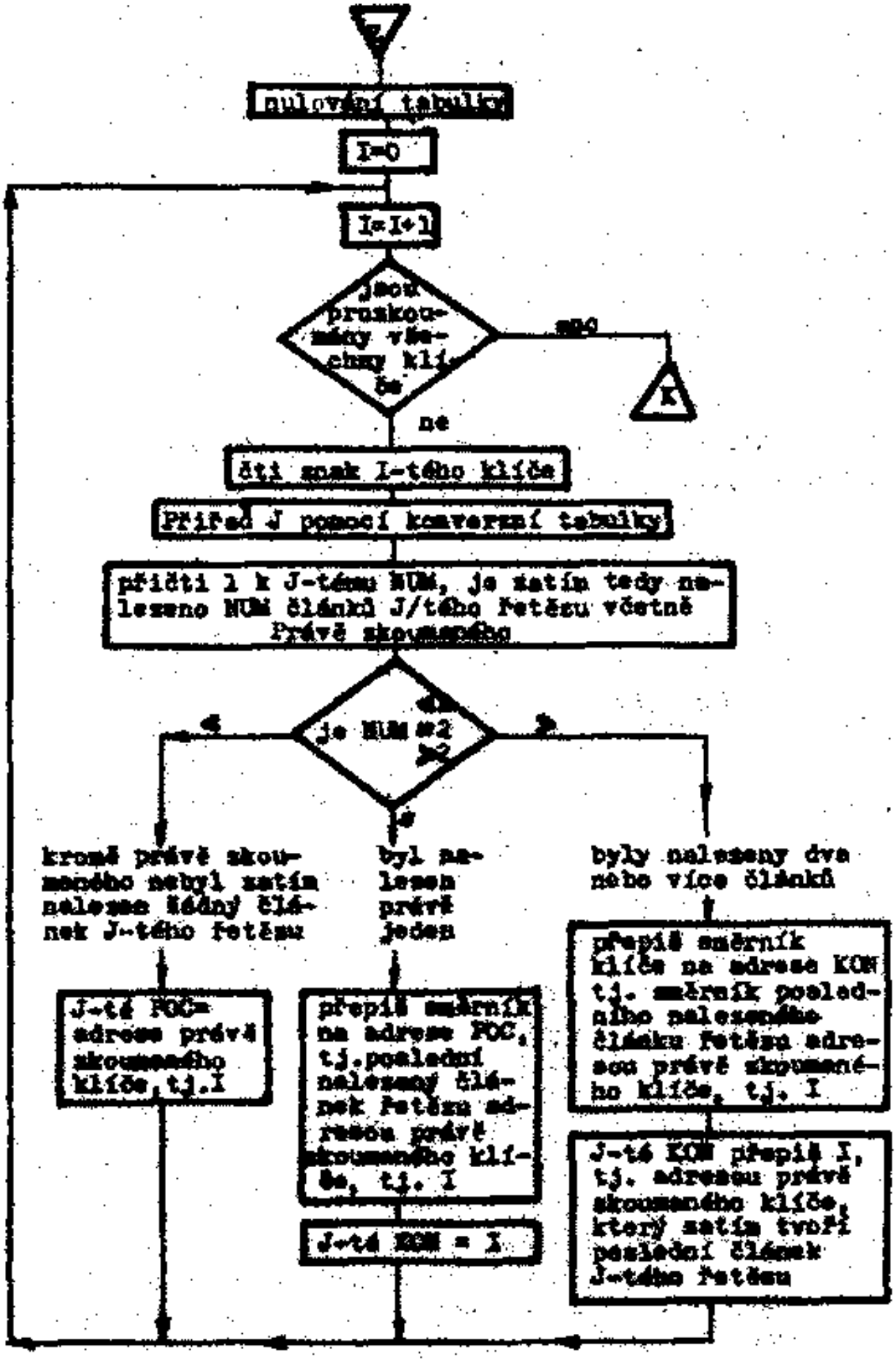
Tabulka dílčích řetězů bude mít tuto formu:např.:

odpovídá	znaku1	znaku2	znaku3	...	znaku N
počátky dílčích řetězů	2	0	18	...	11
konce dílčích řetězů	9	0	0	...	35
počet článků	17	0	1	...	2

Znamená to, že řetěz spojující klíče se"znakem1" začíná na adrese 2 od počátku souboru, končí na adrese 9 a má 17 článků, které jsou propojeny směrníky. Pro "znak2" neexistuje žádný klíč, pro"znak3" jeden klíč na adrese 18 a pro "znakN" 2 klíče na adresách 11 a 35.

4.3. Algoritmus pro vytvoření tabulky dílčích řetězů.

Uvedeme blokové schéma (přitom budou používány zkratky POC pro adresu počátku dílčího řetězu, KON pro jeho konec,


nulování tabulky
I=0
I=I+1
jsou
prozkou-
mány vše-
chny klí-
če
ano
K
ne
čti znak I-tého klíče
Přiřaď J pomocí konverzní tabulky
přičti 1 k J-tému NUM, je zatím tedy nalezeno NUM článků J/tého řetězu včetně
Právě zkoumaného
je NUM =2
<2
=2
>2
kromě právě zkoumaného nebyl zatím nalezen žádný článek J-tého řetězu
byl nalezen právě jeden
byly nalezeny dva nebo více článků
J-tý POC=
adrese právě
zkoumaného
klíče, tj.I
přepiš směrník
na adrese POC,
tj.poslední
nalezený člá-
nek řetězu ad-
resou právě
zkoumaného klí-
če, tj. I
J-tý KON = I
přepiš směrník
klíče na adrese KON
tj. směrník posled-
ního nalezeného
článku řetězu adre-
sou právě zkouma-
ného klíče, tj. I
J-tý KON přepiš I,
tj. adresou prá-
vě zkoumaného klíče,
který zatím tvoří
poslední článek
J-tého řetězu

J pro pořadí znaku v tabulce, pro N znaků zvolené množiny nabývá tedy J hodnot 1 až N; NUM označme počet článků J-tého řetězu).

4.4. Účcha s možnosti konverzní tabulky.

Je zřejmé, že jednou z nejdůležitějších činností v předchozím algoritmu je stanovení pořadí v tabulce tj. určení J. Jak už bylo řečeno, jako prostředek zde slouží konverzní tabulka. Rozeberme si jednu z možností realizace takové tabulky.

Vycházíme přitom z takové představy tabulky:

znak1	znak2	znak3	...	znakN
J_1	J_2	J_3	...	J_N

kde znak1, znak2,..., znakN znamená hodnotu reprezentující tento znak a J_1, J_2, ... , J_N jsou hodnoty udávající pořadí znaku v tabulce dílčích řetězů. Každé J_i přitom může nabývat hodnot 1,2,..,N. Pokud je $J_i \neq J_j$ pro $i \neq j$ znamená to, že každému odpovídajícímu znaku přísluší jiné pořadí v tabulce dílčích řetězů, je-li $J_i = J_j$ pro $i \neq j$ znamená to, že odpovídající znaky mají totéž pořadí v tabulce řetězů a jsou tedy z hlediska třídění považovány za stejné. Tuto úvahu lze rozšířit i na více znaků než na dva.

Tak např. tabulka

A	B	C	M	K	R
2	1	5	3	6	4

udává pro třídění pořadí B, A, M, R, C, K.

Naskýtá se ovšem otázka určení argumentu tabulky pro daný znak, tj. prakticky vyhledání znaku v tabulce. Nejvýhodnější by bylo pracovat přímo s byty, do nichž se u počítačů 3.generace s bytovou strukturou ukládá právě jeden znak. Podle hodnoty uložené v bytu by pak bylo možno určit přímo

argument tabulky a to ve formě čísla udávajícího pořadí znaku v tabulce. Tabulka by pak měla při plném rozsahu bez omezení 2^8=256 argumentů. FORTRAN však není schopen přímo s byty pracovat, proto je nutno volit jinou cestu. Nejkratší proměnná ve FORTRANU IV má délku půlslova tj. 2 byty. Při ukládání klíčů do proměnných se pak nabízejí tyto možnosti:

1. způsob - do obou bytů proměnné jsou formátem A2 uloženy po sobě jdoucí znaky klíče. Počet možných kombinací přitom je 256^2, což by znamenalo, že konverzní tabulka by musela mít tento počet argumentů (podobně i tabulka dílčích řetězů). To je ovšem z praktického hlediska k nepotřebě. I při omezení jen na alfabetické znaky je to nejméně 27^2 argumentů (ve skutečnosti vzhledem k vnitřnímu zobrazení 42^2), což je stále ještě hodně, nehledě ke komplikacím při lichém počtu pozic klíče.

2. způsob - znak je do proměnné ukládán formátem A1, je tedy v prvním bytu vyššího řádu, zatímco v bytu nižšího řádu je mezera (hexadecimálně XX 40, kde X je číslice 1 až F). I když jsme při ukládání do proměnné použili znakové konverze (A), můžeme nyní proměnnou považovat za číslo typu INTEGER. Bude mít hodnoty obecně jak kladné, tak i záporné, což by se dalo snadno odstranit posunutím počátku, přičemž současně vyloučíme mezeru v bytu nižšího řádu (odečtením čísla 40 hexadecimálně tj. 64). Pokud bychom však takto získané číslo chtěli využít k určení argumentu, nevyhneme se operaci dělení tohoto čísla 256-ti. Výsledný rozsah by byl v mezích 0-255 (1-256). Operace jsou však poměrně časově náročné (dělení).

3. způsob - klíč je v tomto případě zmenšován v původním souboru tak, že je prokládán mezerami (prvním znakem rozšířeného klíče je mezera, pak první znak původního klíče, mezera, druhý znak ....). Tento klíč je pak ukládán do proměnné formátem A2. V prvním bytu proměnné je tedy mezera, ve druhém nižšího řádu znak klíče. Hexadecimálně je to 40 XX. Stačí tedy odečíst mezeru z 1. bytu tj. dekadické číslo 16384 (4.16^3). Získáme tak přímo čísla v rozsahu 0-255, která určí

argument konverzní tabulky. Jedinou nevýhodou je větší spotřeba paměti při ukládání klíčů, což ovšem u moderních počítačů již není tak kritická záležitost. Zavedeme-li navíc zkrácení tabulky posunutím počátku, což se provede současně s eliminováním mezery v prvním bytu, lze např. pro alfabetické třídění použít tabulky pouze o 42 argumentech (odpovídá 26 znakům a mezeře). V konverzní tabulce je v tomto případě 15 "děr", které nemá smysl nijak eliminovat, poněvadž na průběh výpočtu nemají žádný vliv.

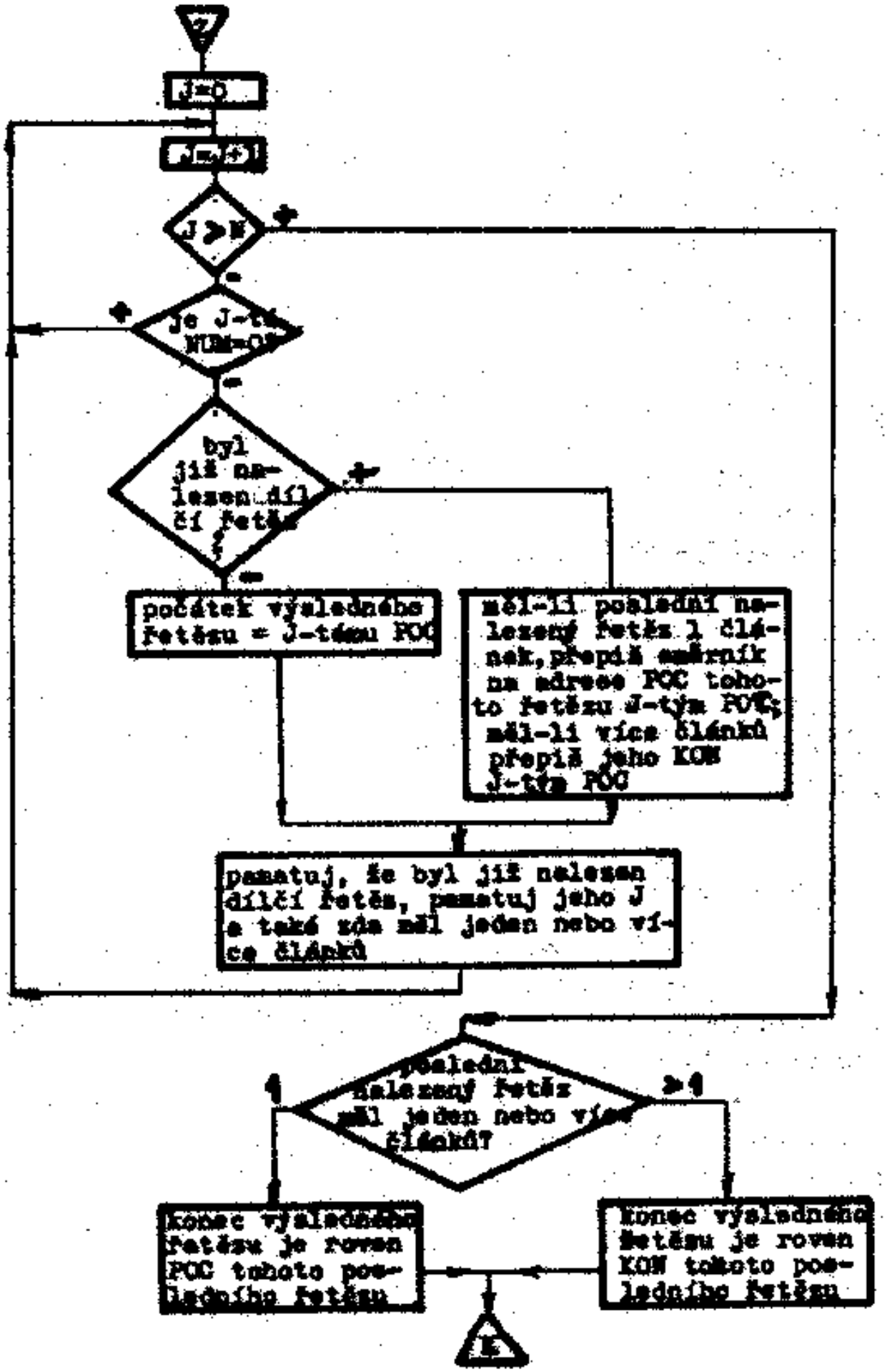
Konverzní tabulka tedy může být vytvořena jako jednorozměrné pole. Ze znaku klíče lze jednoduchou operací (odečtením) určit přímo index příslušející určitému znaku. Prvek pole odpovídající tomuto indexu pak udává pořadí v tabulce dílčích řetězů.

Určit pořadí v tabulce řetězů by bylo samozřejmě možné i jinými způsoby. V naší metodě jsme použili konverzní tabulky a jejích principů tak, jak byla popsána spolu s 3. způsobem určení jejího argumentu a to zejména z důvodu značné jednoduchosti a tím i rychlosti a z důvodu možností, které se naskýtají při zadávání množiny tříděných znaků a jejich pořadí.

4.5. Algoritmus propojení dílčích řetězů

Další částí metody je algoritmus, který provádí propojení dílčích řetězů podle tabulky dílčích řetězů. Při označování proměnných použijeme téhož označení jako u předchozího algoritmu. Blokové schéma je na následující straně.

Po ukončení tohoto algoritmu jsou tedy známy adresa prvního článku výsledného řetězu a adresa posledního článku řetězu. Směrníky u jednotlivých klíčů propojují pak klíče podle zadaných požedavků na pořadí znaků.


J=0
J=J+1
J > N
+
-
je J-tý NUM=0?
+
-
byl již nalezen dílčí řetěz
+
-
počátek výsledného řetězu = J-tému POC
měl-li poslední nalezený řetěz 1 článek, přepiš ukazatel na adrese POC tohoto řetězu J-tým POC; měl-li více článků přepiš jeho KON J-tým POC
pamatuj, že byl již nalezen dílčí řetěz, pamatuj jeho J a také zda měl jeden nebo více článků
poslední nalezený řetěz měl jeden nebo více článků?
=1
>1
konec výsledného řetězu je roven POC tohoto posledního řetězu
konec výsledného řetězu je roven KON tohoto posledního řetězu
E

4.6. Rozšíření metody pro klíč libovolné délky

Tato dosud popsaná metoda pracovala jen s klíči o jedné pozici (jeden znak). Lze ji však snadno rozšířit i na klíče o více pozicemi. Vyložíme ideu takové rozšíření.

Zavedeme označení úrovně, které má spojitost s pozicí znaku v klíči. Pro 1. pozici je určena úroveň 1, pro 2. pozici úroveň 2 atd., až pro pozici N úroveň N. Přitom úroveň 1 pokládáme za nejvyšší, úroveň N za nejnižší. Úroveň budeme označovat U_i pro i-tou pozici klíče.

Pro každou úroveň pak budeme vytvářet tabulku dílčích řetězů, pokud již vyšší úroveň jednoznačně neurčí pořadí klíčů souboru. Postup je přitom takový:

Po vytvoření tabulky dílčích řetězů úrovně U_i je testováno, zda tyto řetězy určují jednoznačně pořadí klíčů v souboru. Tabulka je postupně procházena pro J=1,2,...,N -tý dílčí řetěz a je zkoumán počet prvků tohoto řetězu. Mohou nastat tyto význačné případy:

a) počet prvků je roven nule - pak tento řetěz "nulové" délky nemá vliv na další tabulku, poněvadž nezahrnuje žádný klíč; provedeme tedy J=J+1 a postoupíme s testováním k dalšímu dílčímu řetězu tabulky úrovně U_i,

b) počet prvků je roven 1. Pořadí je jednoznačné, lze tedy J=J+1 a postoupit s testováním k dalšímu dílčímu řetězu tabulky úrovně U_i.

c) počet prvků je roven 2. Pak je možno jednoduchým testem ověřit, jaké by mělo být pořadí těchto článků řetězu, dotážeme-li se na hodnoty pozic klíče nejbližše nižší úrovně, pokud ovšem tato existuje. Pokud ne, nezáleží na pořadí těchto článků řetězu. Po zajištění správného pořadí opět J=J+1 a pokračujeme v testování tabulky úrovně U_i.

d) počet prvků je větší než 2. Pak nelze obecně jednoduše určit, jaké má být pořadí článků řetězu a je třeba postoupit do další nižší úrovně klíče, kde můžeme získat další informace mající vliv na pořadí klíčů v tomto dílčím řetězu. Je tedy třeba znovu vytvořit tabulku dílčích řetězů úrovně

U_{i+1} (tj. nižší) a právě testovaného dílčího řetězu tabulky úrovně U_i, ověřit testováním, zda je už pořadí jednoznačně určeno a pokud tomu tak je nebo pokud jsou už vyčerpány všechny pozice klíče, tabulku úrovně U_{i+1} propojit a vrátit do původní úrovně U_i správný počátek a konec dílčího řetězu (současně je provedeno i správné propojení jednotlivých článků tohoto řetězu). Pokud by znovu ještě nebylo určení pořadí jednoznačné, je nutno pokračovat dále postupováním do dalších úrovní klíče U_{i+2}, U_{i+3} .... U_j kam až bude potřeba pro jednoznačné určení pořadí. Pak zpětným postupem propojováním tabulek od poslední zkoumané úrovně a návratem počátečních a koncových adres dílčího řetězu vyšší úrovně do této úrovně je možno celý proces postupně uzavírat. Celý proces končí propojením tabulky nejvyšší úrovně tj. úrovně 1, která odpovídá první pozici klíče. Výsledkem je pak výsledný řetěz, který je určen směrníky klíčů a údaji o jeho počátční a koncové adrese.

Je zřejmé, že jde o proces svou podstatou rekursivní. Poněvadž FORTRAN nepovoluje rekurzi - nelze volat tentýž modul, pokud není jeho provádění z předchozího výpočtu ukončeno, bylo potřeba toto omezení obejít. Možným řešením je rezervovat pro každou úroveň klíče jednu tabulku dílčích řetězů současně s údajem, který udává, kam až dosud je tabulka už uvedeným postupem otestována. Poněvadž tyto tabulky nezabírají příliš místa, mohou být trvale v paměti. Tabulky jsou pak dynamicky "otevírány" a "uzavírány" podle potřeby. Tohoto postupu bylo použito i při praktické realizaci metody. Je možná i tato varianta: poněvadž pro každou úroveň je "otevřena" vždy jen jedna tabulka, musí být v paměti přítomna jen právě ona. Lze tedy tabulky vyšších úrovní, které jsou již otevřené a ještě zcela netestované uchovávat na vnější paměti, nejlépe v souboru s přímou organizací tedy na mg. disku s ohledem na doby výběru, a odsud je podle potřeby přepisovat zpět do paměti. Tedy při postupu do nižší úrovně by byla právě testovaná tabulka "uklizena" na vnější paměť, při návratu do této úrovně by byla "natažena" zpět.

4.7. Diskuse metody

Při třídění dochází k testování klíčů, pokud pak není třídění prováděno v paměti k jejich čtení ze souboru. Celý soubor se tak prochází, přičemž počet průchodů u obvyklých metod je většinou silně závislý na rozložení klíčů vzhledem k jejich požadovanému pořadí a může dosahovat značných čísel. Navíc dochází k přepisování vět nebo záznamů, což se projeví značným zvýšením spotřeby strojového času, ať už jde o třídění v paměti (vyžaduje to pak přesuny dlouhých řetězců znaků) nebo o třídění mimo paměť (operace čtení a zápisu patří k vůbec nejdelším.

Uváděná metoda se tomu vyhýbá a odsud plynou také její výhody:

- nezávisí na rozložení klíčů, počet průchodů souborem je prakticky dán počtem pozic klíče; tato vlastnost vynikne zvláště u rozsáhlých souborů a souborů, jejichž klíče jsou zcela náhodně rozloženy
- díky testování tabulek dílčích řetězů je třídění prováděno jen po pozici klíče, která poslední se nějakým způsobem podílí na určení pořadí klíče v souboru; metoda se tedy jakoby dynamicky přizpůsobuje konkrétnímu zadání úlohy
- nedochází k přepisování vět, tedy nejdelší ztrátové časy jsou eliminovány
- rozsah metody lze snadno "ušít na míru" pro dané zadání
- stejně tak lze zadat libovolnou množinu znaků pro třídění a jejich poř... díky konverzní tabulce
- různé varianty a způsoby jednotlivých dílčích algoritmů umožňují do jisté míry vyhovět i požadavkům na maximální rychlost nebo naopak na maximální úsporu místa v paměti

Je třeba ovšem se zmínit i o nevýhodách metody

- hlavní nevýhodou je poměrně velká spotřeba paměti - jde o třídění v paměti a tím i omezení velikosti sou-

bord zhruba na velikosti uvedené v zadání (při počítači s více než 256K paměti nebo s virtuální pamětí). Pokud ovšem zavedeme třídění souborů počástech , které jsou pak slučovány, jak je tomu u většiny metod, je i toto omezení prakticky eliminováno

- další nevýhodou je nutnost používání proloženého klíče, což sice lze zařídit přímo při vytváření tříděného souboru, ale znamená to nárok na média. Využitím 2. způsobu určení argumentu konverzní tabulky by se dalo i toto odstranit, je však třeba ověřit, jak se to projeví na spotřebě str. času

Pohlédneme-li na metodu z hlediska tématu příspěvku, lze konstatovat, že to byly právě techniky indexových tabulek (tabulky dílčích řetězů, konverzní tabulka) a zejména technika řetězení (řetězy směrníků klíčů), které umožnily realizaci této myšlenky.

Metoda je v současné době (leden 1975) ve stavu ověřování a zadání zpracování malých souborů opravňují k závěru, že je vysoce účinná.

5. Závěr

Domníváme se, že uvedené příklady ukázaly některé z možností obou diskutovaných technik, i když nejtypičtějšími případy jejich použití jsou spíše úlohy tzv. hromadných dat a zejména oblast databázových systémů. V dnešní době, kdy i ve VTV se zřetelně projevují tendence ke zvětšování rozsahu úloh, které v některých směrech mají přímo i vlastnosti charakteristické pro HD,by bylo nerozumné nepoučit se u "konkurence z hromadných dat" a doufáme, že tento příspěvek poskytne i skalním programátorům VTV aspoň podněty k zamyšlení.

LITERATURA:

/1/ FORTRAN IV Language, GC-28-6515-9, manuál fy IBM

FORTRAN Programmer's Guide,GC-28-6817-3, manuál fy IBM

/2/ Petroušková : Síťová struktura v bance dat, sešity INORGy, INORGA Praha

/3/ Databanky v informačních systémech, sborník ČKVŠ, 1973

/4/ T.Lutz, H.Klimesch : Die Datenbank im Informationssystem, R.Oldenbourg Verlag München, Wien 1971

/5/ F.R.A.Hopgood: Metódy kompilovania, ALFA 1973

/6/ J.M.Foster: Spracovanie zoznamov, ALFA 1973

/7/ W.M.Turski : Struktury danych, Wydawnictwa Naukowo-techniczne, Warszawa 1971