Статијата прв пат е објавена во списанието Тангента на ДМ на Србија во 2004/05 година

О СОРТИРАЊУ

***мр Ана Капларевић – Малишић,* Крагујевац**

Листа познатих алгоритама за сортирање је прилично дуга. Један њен део смо у прошлом броју описали и на томе ћемо се за сада задржати. Упознајмо се са још неким појмовима везаним за сортирање који имају везе са условима за имплементацију и током алгоритама, о којима прошли пут нисмо водили рачуна. Прво ћемо говорити о томе шта се дешава када немамо довољно физичке меморије. Други део се односи на мање познату причу о правилима уређивања речи. Наиме, прошли пут смо говорили о алгоритмима за сортирање не узимајући у обзир правила уређивања (нпр, речи уређене према енглеском алфабету и нашој ћирилици неће имати исти редослед) која битно могу утицати на сам ток и сложеност поступка сортирања.

Спољашње сортирање (External sort)

Спољашње сортирање је општи термин за класу сортних алгоритама који обрађују велике количине података. Они су потребни у случајевима када величина улазних података превазилази величину унутрашње меморије која је на располагању.

Један пример оваквог алгоритма је спољашњи merge сорт. Предпоставимо да је потребно сортирати 900 MB података, а да нам је на располагању 100 MB физичке меморије. Поступак њиховог сортирања спољњим спајањем би текао овако:

1. пакети од по 100 MB се учитавају у физичку меморију и сортирани враћају на диск; након овог корака на диску имамо 9 уређених листа; сада треба спојити листе без нарушавања уређења;
2. од оних 100 MB RAM-а, 10 остављамо да „глуми “ buffer, а осталих 90 за 9 контејнера од по 10 MB у које смештамо податке са почетка сортираних листа са диска;
3. поређењем почетних елемената контејнера, одређује се који од елемената први иде у buffer, након чега он бива брисан из контејнера; поступак се понавља док се buffer не напуни, након чега се његов садржај пребацује на диск на место предвиђено за смештање сортираних података, а сам buffer се празни; у случају да се неки од контејнера испразни пуни се новим подацима из одговарјуће листе са диска.

Колико је пакета и контејнера потребно направити зависи од тога коју количину података треба сортирати и колико је физичке меморије на располагању...

Collation – уређивање

Collation је појам који се у рачунарским наукама односи на уређивање речи, пре свега, у одређен поредак. У алгортиму за сортирање, њим се дефинише парцијално уређење $\leqslant$ парова, на основу којег се одређује да ли ће се вршити размена места или не. Најстарији системи уређивања су нумеричко и алфабетско.

Нумеричко уређивање

Најједноставнији начин уређења јесте нумерички, којим се бројеви уређују стандарндно, тј. према њиховој вредности. Ако је у питању поређење знакова, нумеричко сортирање захтева поређење нумеричких кодова. Наиме, сваки рачунар мора користити неки од система кодирања знакова (нпр. ASCII или неки од његових проширења), тако да су, на пример, ASCII вредности знакова $,a,b,C,d редом 36, 97, 98, 67, 100, па би сорирани они имали редослед $,C,a,b,d. Када су у питању речи, као низови знакова, оне се пореде од почетка до позиције на којој им се знакови разликују. Однос тих знакова ће одредити која ће реч испред које бити (као и у речницима). Овај поступак се често користи уз измену великих слова у мала (при самом поређењу), јер, сложићете се, већина корисника нема жељу да им списак буде сортиран тако да им на врх списка „истрче“ речи које почињу великим словом, на пример M, а да им се „касније “ јаве још неки подаци под m (који су, на пример, погрешно откуцани малим почетним словом).

Ово је систем који базично користи сваки рачунар. Свако одступање од оваковог поређења, тражи некакву надоградњу.

Овакав систем, очигледно, није добар у случају поређења бројева записаних у облику стринга.

Алфабетско уређивање – алфабетизација

Сложенији начин уређивања јесте алфабетски који разврстава речи на основу редоследа слова (знакова) у алфабету неког језика, значи да ће „распоред“ две речи зависити од језика, тј. његовог писма. Поређење стрингова се, наравно, изводи на исти начин као и код нумеричког уређења, дакле, знак по знак до разлике. Када је текст на енглеском језику, ова два уређења најчешће дају исти резултат. Разлика постаје очигледна када је у питању језик чије писмо представља проширени латинични алфабет.

На пример, шпански алфабет садржи следеће знакове

a, b, c, **ch**, d, e, f, g, h, i, j, k, l, **ll**, m, n, **ñ**, o, p, q, r, s, t, u, v, w, x, y, z

где је **ñ** основни знак алфабета, што су и знаци **ch** и **ll**, који представљају лигатуре [7] или дирафе, зато је реч lyncar испред речи llaca, а czar је испред chabacanada[8]. Поштујући нумеричко уређење, рачунар би знак **ñ** поставио иза z, а **ch** погрешно третирао као c + h.

Unicode алгоритам – уређивање вишејезичних текстуалних података

Када је потребно обезбедити сортирање речи, а да при томе контекст из кога је уређење захтевано (апликација, оперативни систем итд.) не даје информације о језику, односно коришћеном писму, тада је ослањање на **Unicode** добар почетак.

Шта је Unicode?

Unicode је интернационални стандард којим се дефинише начин кодирања (начин бележења од стране рачунара) текстова писаних било којим познатим писмом (и живих и архаичних језика). Он, такође, „покрива “ и знаке који не припадају ни једном конкретном писму, као што су интерпункцијски, математички итд. 1991. је објављен први Unicode стандард, док је његова последња ревизија, Unicode 4.1, изашла 2005.

[7] два знака који и се третирају као један знак и читају као један глас

[8] Наравно, исти третман имају наша слова (у латиничном писму): č, ć, ž, š, односно lj, nj, dj и dž.

Unicode се, тренутно, сматра најкомплетнијим скупом знакова (character set) и јесте најзаступљенија шема у вишејезичним софтверским окружењима (овде се мисли не само на апликације, већ и на оперативне системе, што сте у сопственом раду, вероватно, одавно приметили).

Битно је разумети да Unicode није ни софтверски пакет, нити је фонт. Дакле, у процесирању текста он нема задатак да дефинише облик, величину или стил исписа неког знака, већ само дефинише јединствен код сваког знака. Дакле, да ли ћете слово Љ исписати у овом или неком другом фонту и да ли ће оно бити величине 12 или 24 тачака, сасвим је свеједно, јер је у основи слово исто, па ће и код бити исти. Како ће тај број (код) бити бележен при процесирању текста, је сасвим други проблем. Зашто проблем? Већина софтвера је подешена да ради са 8-битним или, за ову прилику у бољем случају, двобајтним кодовима за карактере.

Софтвери који раде са 8-битним кодовима карактера немају више од 256 различитих кодова на располагању, а они који раде са двобајтним речима су лимитирани на 65,536. Са друге стране Unicode тренутно има преко 96,000 кодираних знакова. Дакле, у сваком случају је потребно извршити мапирање, тј. повезивање Unicode кодова са њиховим репрезентацијама у различитим програмима. У ту сврху Unicode Cosortium предлаже више метода мапирања који се разврставају у две групе: UTF (Unicode Transformation Format) и UCS (Universal Character Set). Тако постоје UTF-32, UTF-16, UTF-8, UTF-EBCDIC, UTF-7 и UCS-4, UCS-2, где се бројеви односе на битове (код UTF мапа), односно бајтове (код UCS мапа) предвиђене за „бележење" карактера. Код UTF-32 и UCS-4, предвиђена величина кода, односно кодна јединица, је довољна за бележење било ког карактера (Unicode шифре), док код осталих варира величина јединице у зависности од карактера. UTF-32 и UTF-16 се користе за интерна процесирања (унутар система), док је UTF-8 стандард који служи за размену Unicode текстова, нпр. за приказ на Web странама.

Више критеријумско уређивање

Вратимо се сада на уређивање вишејезичних текстова. За обезбеђивање што веће прецизности поступак поређења текстуалних података се виши по више критеријума, при чему су критеријуми распоређени по нивоима, тј. мора им се знати редослед, тако да се, на пример, неће вршити поређење акцената [9] ако се утврди разлика у основним знацима.

Ниво	Критеријум поређења	Пример
L_1	основни карактер	role < roles < rule
L_2	акценти	role < rôle < roles
L_3	велико-мало слово	role < Role < rôle
...	...	
L_n	дискреционе црте и сл.	role < ro-le < „role"

Најважнији, примарни, ниво поређења јесте онај у коме се врши поређење карактера у основи текста, док се на осталим, непримарним, нивоима издвајајем карактеристичних делова врше „финија" подешавања. На колико ће се нивоа вршити поређење зависи од потреба корисника, односно апликације, чиме је дефинисана варијација основног Unicode уређења.

[9] обратите пажњу на то да се наши знаци č, ć, ž не третирају као знаци са акцентима јер припадају алфабету, пример језика у коме се акценти „накнадно" дописују јесте француски, јер у њиховом писму не постоји, нпр, ô, већ само o, али зато постоји у текстовима

UNICODE COLLATION ALGORITHM (UCA) – „укратко “

Већина људи гледајући Unicode кодне стране, очекује да ће слова њиховог језика бити у „правилном“ распореду. Пошто уређење стрингова зависи од језика, а са друге стране се толерише вишејезичност, није могућ распоред кодова карактера такав да се једноставним поређењем њихових бинарних записа добија резултат који би се добио и поређењем карактера. За „исправно“ уређење користе се посебне табеле, а комплетан поступак уређења дефинисан је основним Unicode алгоритмом.

Unicode алгоритмом уређивања (UCA) описан је начин поређења два Unicode стринга прилагођен захтевима Unicode стандарда. Основни (default) поредак за све Unicode карактере, које користи овај алгоритам, дат је у, јавно доступној, DUCET (Default Unicode Collation Element Table) табели. Она је направљена тако да се лако може прекрајати према правилима различитих језика, односно према потребама апликација или корисника.

Укратко, UCA за сваки улазни Unicode стринг, а на основу табеле уређења (DUCET или неке њене варијације), формира за почетак кључеве карактера појединачно, а на основу њих и кључеве улазних стрингова по којима ће се вршити поређење (назовимо га сортни кључ). Поређењем формираних кључева добија се распоред стрингова чији су кључеви поређени.

По default-у овај алгоритам врши поређење по три нивоа. За латинична писма ови нивои се могу грубо описати као:

1. алфабетизација
2. поређење акцената
3. поређење великих и малих слова.

Коначно постоји још један (необавезан) ниво задужен за поређење стрингова који се не разликују ни по чему другом, осим по постајању дискреционих знакова (нпр. црта). У сваком случају, имплементације UCA нису лимитиране на постојање само три нивоа. Додатни нивои су дозвољени. Једини захтев јесте да продукују резултате у складу са основним алгоритмом, тј. непроширеним алгоритмом. На пример, ако постоји потреба да се апликацији „испоруче“ уређени подаци, при чему се мора водити рачуна о специјалним карактерима који су при поређењу на претходним нивоима игнорисани и да при томе њихов очекивани распоред није сагласан са Unicode распоредом, тада очигледно треба дефинисати још један ниво. У колатералну штету спада потреба за већим меморијским простором и за CET табелу и за сортне кључеве.

Такође, дозвољена је и преправка CET табеле за потребе прилагођавања правилима различитих језика.

Collation Element Table (CET)

Collation Element Table представља табелу која садржи везе између кодова Unicode карактера и њихових collation елемената, тј. параметара на основу којих ће се вршити поређења. Елемент уређења представља (уређену) листу три 16-битна кључа, тзв. тежине. Први кључ у листи представља тежину првог нивоа, други тежину другог нивоа итд. Кључ неког нивоа уствари служи поређењу карактера на одговарајућем нивоу унутар алгоритма уређења. На пример елемент уређења у хексадекадном запису знака са кодом 0061 ("a") је 06D9.0020.0002.

Ако су X и Y елементи уређења, означимо њихове тежинске кључеве са $X_1, X_2, X_3, \ldots$, односно $Y_1, Y_2, Y_3, \ldots$. У табелама су дати записи релација којима се дефинише уређење collation елемената.

За описивање распореда два стринга се уводи слична нотација. Тако на пример, $A <_2 B$ значи да је A мање од B, при чему међу њима постоји разлика на првом или другом нивоу поређења. Ако важи да је $A <_2 B$ и $A =_1 B$, тада између њих постоји разлика само на другом нивоу. Ако су два стринга једнака према табели (по свим нивоима) тада се њихов однос записује овако $A \equiv B$. Ако су њихови одговарајући елементи бит-по-бит једнаки онда се користи следећи запис $A = B$.

Запис	Значење
$X =_1 Y$	$X_1 = Y_1$
$X =_2 Y$	$X_1 = Y_1$ и $X =_1 Y$
$X =_3 Y$	$X_3 = Y_3$ и $X =_2 Y$

Запис	Значење
$X <_1 Y$	$X_1 < Y_1$
$X <_2 Y$	$X <_1 Y$ или $(X =_1 Y$ и $X_2 < Y_2)$
$X <_3 Y$	$X <_2 Y$ или $(X =_2 Y$ и $X_3 < Y_3)$

За елемент уређења чији тежински кључ неког нивоа једнак 0000 каже се да је занемарљив на том нивоу, тј. на том нивоу поређења се неће узимати у обзир.

У табели уређења је могуће и „стапање“ каракатера, као и „раздвајање“. Наиме, једном карактеру, тј. његовом кључу, не мора одговарати само један елемент уређења, нити једном елементу уређења мора одговарати само један кључ. На пример, знак æ се у неким језицима не третира као један већ као два знака (нпр. у Енглеском језику, у коме је овај знак до недавно био у употреби). У њима се при уређивању захтева раздвајање знака, тако да се за њега добијају два елемента уређења.

Главни алгоритам

Главни алгоритам се састоји из 4 основна корака

1. нормализација улазног стринга

   Сврха овог корака јесте, тзв. канонска декомпозиција стринга. Наиме, collation низ улазног стринга се не добија само простим надовезивањем collation елемената сваког карактера улазног стринга појединачно. Потребна је прерада стринга у толико што је са једне стране поједине знакове потребно „разбити “, а након тога преуредити га не били се добио добар резултат при поређењу са осталим стринговима. Да ли се и како врше разбијања и размене одређује се на основу матичне Unicode табеле, тј. на основу Unicode Character Database у којој се поред кода сваког знака појединачно, између осталог, налази, информација о разбијању и тзв. класа комбиновања означена неким бројем. Поређењем вредности класа комбиновања за свака два суседна знака утврђује се да ли им је потребно разменити места или не.
2. креирање низа елемената уређења за сваки стринг

   У овом кораку се на основу нормализованог стринга одређује collation низ, који се (наравно, уз потенцијалне корекције о којима нећемо говорити) састоји из одговарајућих collation елемената знакова (или скупа знакова) из стринга редом, при чему се вредности елемената читају из CET табеле.
3. формирање сортних кључева за сваки стринг

   Сортни кључеви се добијају сукцесивним додавањем тежинских кључева (по нивоима) елемената collation низа.
4. поређење сортних кључева

Прича о Unicode стандарду је много дубља и сложенија. Овај текст је имао за циљ да вам представи само суштину. Више информација можете наћи на сајту Unicode конзорцијума, www.unicode.com.