김일성종합대학학보 JOURNAL OF KIM IL SUNG UNIVERSITY
(자연과학) (NATURAL SCIENCE)
주체104(2015)년 제61권 제8호 Vol. 61 No. 8 JUCHE104(2015).

자료배포형전자사전구축에서 기억공간할당방법

한승주, 김은성

일반적으로 자료배포형전자사전에서 리용되는 자료기지는 한번만 구축되지 않으며 여러번 반복구축에 의하여 완성된다.[1] 여기서 중요하게 제기되는 문제는 자료의 구축과정에 필요되는 기억공간크기의 효률적인 조종이다.

론문에서는 자료의 구축과정에 생기는 무효기억공간을 재리용하여 기억공간크기의 효률적인 조종을 위한 기억공간할당방법을 제안한다.

1. 자료배포형전자사전구축에서 기억효률문제

정의 1 자료가 보관될 기억구역을 기억공간이라고 하며 $M=(S,\ E)$로 표시한다. 여기서 S는 기억공간의 시작주소이며 E는 기억공간의 끝주소이다.

이때 기억공간 M의 크기는 $|M|=E-S$로 정의한다.

정의 2 기억공간 M의 자료가 유효할 때 기억공간 M을 유효기억공간, 무효할 때 무효기억공간이라고 한다.

그러면 자료의 총체적인 기억공간의 크기는 다음과 같이 계산된다.

$$|M|=\sum|M_{유}|-\sum|M_{무}|$$

이로부터 기억효률은 다음과 같이 평가할수 있다.

$$\lambda=\frac{\sum|M_{유}|}{|M|}$$

즉 기억효률문제는 $\lambda\to 1$ 혹은 $\sum|M_{무}|\to 0$이 되게 하는 문제이다.

2. 자료기억효률문제를 해결하기 위한 자료기억공간할당방법

일반적으로 무효기억공간은 이미 있던 자료를 갱신할 때 생기게 된다. 즉 이미 있던 자료를 버리고 그대신 새로운 자료를 넣을 때 원래 있던 자료의 기억구역은 무효기억공간으로 된다.

무효기억공간의 크기를 최소로 하기 위한 방도는 무효기억공간을 재리용하는것이다. 일반적으로 선행한 방법[2]에서는 새로운 자료의 크기가 원래의 자료의 크기보다 작은 경우에 이미 리용하였던 공간을 재리용하였다. 그러나 자료를 구축할 때 항상 새로운 자료의 크기가 원래의 자료보다 작다는것을 담보하지 못한다. 따라서 우리는 무효기억공간을 재리용하는 한가지 방법으로서 다음과 같은 세가지 알고리듬을 제안한다.

알고리듬 1 기억구역할당알고리듬

① $M_w \leftarrow S(w)$, $ML_{무} \leftarrow ML_{무} \cup \{M_w\}$

② $ML_{무} \leftarrow Se(ML_{무},\ |C_w|)$

③ $ML_{무} \neq \phi$ 이면 $ML_{무} \leftarrow ML_{무} \setminus \{M_{무}\}$, $M_w \leftarrow M_{무}$, 알고리듬을 완료한다.

④ $ML_{무} = \phi$ 이면 $M_{무} \leftarrow \underset{M \in ML_{무}}{\arg\max} |\mathrm{M}|$.

⑤ $|M_{무}| < |C_w|$ 이면 $M_w \leftarrow New(|C_w|)$.

⑥ $|M_{무}| > |C_w|$ 이면

$$S_w \leftarrow S_{무},\ E_w \leftarrow S_{무} + |C_w|,\ ML_{무} \leftarrow ML_{무} \setminus \{M_{무}\},\ S_{무} \leftarrow E_w + 1.$$

⑦ 알고리듬을 완료한다.

우의 알고리듬에서 S_w 는 올림말 w 에 대한 이전의 사전정보가 보관된 기억구역탐색함수, $ML_{무}$ 는 무효기억공간목록, $Se(ML_{무},\ s)$ 는 무효기억공간목록 $ML_{무}$ 에서 크기가 s 인 무효기억공간을 탐색하는 함수, $New(s)$ 는 크기가 s 인 기억공간을 할당하는 함수이다.

우의 알고리듬을 적용하면 기억공간을 줄일수는 있지만 반대로 기억공간할당시간이 늘어날수 있다. 이 문제를 해결하기 위해 우선 다음과 같은 기호들을 도입하자.

$t_s(w)$: 올림말 w 에 대한 사전정보의 탐색시간

$t_i(p)$: 크기가 p 인 무효기억공간의 탐색시간

$t_{\max}$: 최대무효기억공간의 탐색시간

t_τ: 기타 기억기할당소비시간

이때 $|M_{무}| = |C_w|$ 인 무효기억공간 $M_{무}$ 가 존재하는 경우 기억공간할당시간 $t(w)$ 는 다음과 같이 표시될수 있다.

$$t(w) = t_s(w) + t_i(|C_w|) + t_\tau \tag{1}$$

한편 $|M_{무}| = |C_w|$ 인 무효기억공간 $M_{무}$ 가 존재하지 않는 경우 기억공간할당시간 $t(w)$ 는 다음과 같이 표시될수 있다.

$$t(w) = t_s(w) + t_i(|C_w|) + t_{\max} + t_\tau \tag{2}$$

우의 두 식으로부터 알고리듬의 최대시간은 $|C_w|$ 인 무효기억공간이 존재하지 않는 경우의 기억공간할당시간으로 된다는것을 알수 있다.

결론적으로 $t_s(w)$, $t_i(p)$, $t_{\max}$ 를 줄이는것이 구축시간문제를 해결하는 방도라는것을 알수 있다. 그런데 $t_s(w)$ 와 $t_{\max}$ 를 줄이는 문제는 쉽게 해결할수 있으므로 $t_i(p)$ 를 줄이는 문제에 귀착된다. 이 문제를 해결하기 위하여 다음과 같은 n진나무 T_n 을 도입하자.

$$T_n = \{root,\ \ Node\},\ \ root \in Node$$

한편 $\forall N \in Node,\ \ N = (Data,\ \ Child)$ 이다.

여기서 $Data$ 는 무효기억공간의 시작주소의 목록이고

$$Child = (N_0,\ N_1,\ \cdots,\ N_{n-1}),\quad N_i \in Node,\ \ 0 \le \forall i < n$$

이며 이때 N_i 는 ϕ 일수 있다.

일반적으로 N 의 자식목록을 $Child(N)$ 으로 표시한다. 그러면

$$\forall N \in Node,\ \ |Child(N)| = n$$

을 항상 만족시키게 된다.

또한 N_i 가 N 의 자식일 때 $N_i \in Child(N)$ 으로 표시하며 N 이 N_i 의 부모일 때 $N = Parent(N_i)$, N_i 가 마디 N 의 $i+1$ 번째 자식일 때 $Index(N_i) = i$ 로 표시된다.

그리고 $N \in Node$ 에 대하여 $\forall N_i \in Child(N)$, $N_i = \phi$ 일 때 N 을 잎마디라고 한다.

한편 n진나무 T_n 의 마디 $N \in Node$ 의 깊이 $Depth(N)$ 을 다음과 같이 재귀적으로 정의한다.

$Depth(N) = Depth(Parent(N)) + 1$

$Depth(root) = 1$

그리고 n진나무 T_n 의 마디 $N \in Node$ 의 값 $Value(N)$ 을 다음과 같이 재귀적으로 정의한다.

$Value(N) = Value(Parent(N)) + Index(N) * n^{Depth(N)}$

$Value(root) = 0$

여기로부터 n진나무 T_n 을 리용하여 무효기억공간목록 $ML_{무}$ 를 구성할수 있다.

n진나무 T_n 을 리용하여 무효기억공간목록 $ML_{무}$ 를 구축하고 탐색하는 알고리듬은 다음과 같다.

알고리듬 2 n진나무 T_n 을 리용한 무효기억공간목록 $ML_{무}$ 의 구축알고리듬

① $v \leftarrow |M_{무}|$

② $N \leftarrow root$

③ $i \leftarrow v \% n$

④ $N \leftarrow N_i \in Child(N)$

⑤ $N = \phi$ 이면 N 을 창조한다.

⑥ $v \leftarrow v / n$

⑦ $v = 0$ 이면 $Data \leftarrow Data \cup \{S_{무}\}$, 알고리듬을 끝낸다.

⑧ $v \neq 0$ 이면 ③으로 이행한다.

알고리듬 3 n진나무 T_n 을 리용한 무효기억공간목록 $ML_{무}$ 에서의 $M_{무}$ 탐색알고리듬

① $v \leftarrow m$

② $N \leftarrow root$

③ $i \leftarrow v \% n$

④ $N \leftarrow N_i \in Child(N)$

⑤ $N = \phi$ 이면 $M_{무} \leftarrow \phi$, 알고리듬을 끝낸다.

⑥ $N \neq \phi$ 이면, $v \leftarrow v / n$.

⑦ $v = 0$ 이면 $\exists S_{무} \in Data$ 를 선택한다.

⑧ $Data = \phi$ 이면 $M_{무} \leftarrow \phi$ 로 하고 알고리듬을 끝낸다.

⑨ $Data \neq \phi$ 이면 $E_{무} \leftarrow S_{무} + Value(N)$, $M_{무} = (S_{무}, E_{무})$, 알고리듬을 끝낸다.

⑩ $v \neq 0$ 이면 ③으로 이행한다.

3. 실험 및 결과분석

제안된 방법의 효률을 평가하기 위하여 Pentium 4 2.6GHz성능의 콤퓨터로 모의실험을 진행하였다.

모의실험을 위하여 10만단어규모의 올림말을 가진 3개의 사전을 미리 준비하였다.

이 사전들에서 자동적으로 콤퓨터가 매개 올림말에 대하여 1～15까지의 우연수를 발생시켜 그만한 회수로 보관조작을 반복 진행하도록 하였으며 이에 기초하여 총기억공간과 유효기억공간의 평균구축시간을 선행한 방법[2]과 비교하였다.(표)

표. 제안된 방법과 선행한 방법을 비교한 기억효률과 평균구축시간

편집한 올림말수	보관회수	총기억공간/KB		유효기억 공간/KB	구축시간/s		효률성	
		제안된 방법	선행한 방법		제안된 방법	선행한 방법	제안된 방법	선행한 방법
103 092	1 034 532	26 976	53 262	25 795	0.027	0.016	0.956	0.484
103 043	1 053 413	26 980	54 098	25 432	0.026	0.017	0.943	0.470
129 742	1 201 009	30 977	68 923	28 904	0.023	0.015	0.933	0.419

표로부터 론문에서 제안한 방법에 의하여 기억공간의 효률성을 2배이상 개선할수 있다는것을 알수 있다.

참 고 문 헌

[1] Yorick Wilks; Machine Translation, Springer, 162～164, 2009.

[2] B. J. Dorr; Machine Translation, 12, 3, 271, 2010.

주체104(2015)년 4월 5일 원고접수

Memory Space Allocating Method in Data-Distributing Electronic Dictionary Construction

Han Sung Ju, *Kim Un Song*

We solved the memory space allocating problem in data-distributing electronic dictionary construction by the invalid memory space list using *n*-ary tree.

Key words: data-distributing electronic dictionary, memory allocating