Journal of the Korea Academia-Industrial cooperation Society
Vol. 14, No. 3 pp. 1451-1458, 2013

http://dx.doi.org/10.5762/KAIS.2013.14.3.1451

전수데이터를 생성하는 빠른 콤비나토리얼 프로그램

장재수[1], 신재원[1], 전홍식[2], 서창진[1*]
[1]숭실대학교 컴퓨터학과, [2]숭실대학교 경영학부

Fast Combinatorial Programs Generating Total Data

Jae-Soo Jang[1], Shin-Jae Won[1], Hong-Sik Cheon[2] and Chang-Jin Suh[1*]
[1]**School of Comp. Sci. and Eng., Soongsil University**
[2]**College of Business Administration, Soongsil University**

요 약 이 논문은 콤비나토리얼 문제인 조합(combination)과 순열(permutation), r-순열(r-permutation) 규칙에 의거하여 전수데이터를 생성하는 빠른 프로그램과 알고리즘을 다룬다. 이들 프로그램은 전수데이터 검사나 시뮬레이션의 입력값 선정과 같은 응용에서 사용된다.
본 연구에서는 조합, 순열, r-순열 문제의 규칙을 만족하는 프로그램들을 수집하여 부문별로 가장 빠른 프로그램을 선정하고 추가연구를 통하여 수집된 프로그램보다 수행시간을 단축한 프로그램을 완성하였다.
본 연구를 위해서 다음과 같은 선행조사가 이루어졌다. 첫째 인터넷에 공지된 백 개 이상의 프로그램을 수집하고 완성하였다. 둘째, 확보된 프로그램을 구동하여 수행시간을 측정하였고, 그 결과 가장 빠르게 수행하는 프로그램을 부분별로 발췌하였다. 셋째, 선별된 가장 빠른 프로그램에 대해서 알고리즘을 설명하고 의사코드로 정리하였다.
본 논문에서는 이러한 기초작업을 토대로 수행시간이 단축된 프로그램을 완성할 수 있었다. 첫째로 조합 문제에서는 재귀형식에서 비재귀형식으로 변형시켰고, 둘째로 r-순열 문제에서는 조합 프로그램과 순열 프로그램을 결합하는 방법으로 수행시간을 단축하였다. 분석결과에 따르면 전자와 후자는 수집한 가장 빠른 프로그램에 비해서 수행속도를 각각 22%에서 34%, 및 62%에서 226%의 범위로 개선하였다.
본 논문에서 제공한 의사코드를 바탕으로 응용에 쉽게 적용시킬 수 있으며, 전수조사 방법에 소요되는 수행시간을 예측하여 전수조사의 타당성 여부를 결정할 수 있다. 또한, 제공한 코드를 바탕으로 최소의 시간으로 전수데이터를 생성할 수 있다.

Abstract This paper deals with the programs and algorithms that generate the full data set that satisfy the basic combinatorial requirement of combination, permutation, partial permutation or shortly r-permutation, which are used in the application of the total data testing or the simulation input.
We search the programs able to meet the rules which is permutations and combinations, r-permutations, select the fastest program by field. With further study, we developed a new program reducing the time required to processing.
Our research performs the following pre-study. Firstly, hundreds of algorithms and programs in the internet are collected and corrected to be executable. Secondly, we measure running time for all completed programs and select a few fast ones. Thirdly, the fast programs are analyzed in depth and its pseudo-code programs are provided.
We succeeded in developing two programs that run faster. Firstly, the combination program can save the running time by removing recursive function and the r-permutation program become faster by combining the best combination program and the best permutation program. According to our performance test, the former and later program enhance the running speed by 22% to 34% and 62% to 226% respectively compared with the fastest collected program.
The programs suggested in this study could apply to a particular cases easily based on Pseudo-code., Predicts the execution time spent on data processing, determine the validity of the processing, and also generates total data with minimum access programming.

Key Words : Combinatorial, Combination, Permutation, R-permutation, Recursive function

본 연구는 숭실대학교 교내연구비 지원으로 이루어졌음.
*Corresponding Author : Chang-Jin Suh(Soongsil Univ.)
Tel: +82-2-820-0686 email: cjsuh@ssu.ac.kr
Received February 21, 2013 Revised (1st February 28, 2013, 2nd March 6, 2013) Accepted March 7, 2013

1. 서론

전수데이터는 모든 발생 가능한 경우를 목록화한 데이터로 성능평가와 테스트를 수행할 때 사용된다. 전수데이터를 이용한 성능평가는 전수데이터를 입력 데이터로 선정하여 발생할 수 있는 모든 경우를 측정한다. 전수데이터를 이용한 입력방법은 무작위 숫자에 의한 입력방법에 비해서 더 공정한 입력을 제공한다. 또한 전수데이터 조사는 설계상의 오류를 검출하거나 성능을 공정하게 측정할 때에 사용한다[1,2]. 1994년에 인텔 펜티엄 CPU의 소수점 계산 모듈인 FPU(Float ing Point Unit)에서 약 10^{-10}의 확률로 발생하는 나눗셈 오류를 사후에 감지하여 이를 교체하기 위해 약 5억 달러를 지불한 사례는 전수데이터 검사의 중요성을 알려준다[3,4].

이 논문은 전산학의 기초적인 콤비나토리얼 문제인 조합(combination), 순열(permutation), r-순열(부분순열, r-permutation)의 조건을 만족하는 전수데이터를 생성하는 빠른 알고리즘과 프로그램을 다룬다.

조합 전수데이터 생성문제 COMBI_Q(M,N)은 M(0, 1, ..., M-1)개의 숫자 중에서 N(M≥N)개를 순서를 고려하지 않고 선택한 M!/((M-N)!N!)개의 단위데이터로 구성된 전수데이터 생성 문제이고, 순열 전수데이터 생성 문제 PERMU_Q(N)은 N(0, 1, ..., N-1)개의 숫자를 순서를 고려하여 배열한 N!개의 단위데이터로 구성된 전수데이터 생성 문제이며, r-순열 전수데이터 생성 문제 R-PERMU_Q(M,N)은 M(0, 1,.., M-1)개의 숫자 중 N개를 순서를 고려하여 배열한 M!/(M-N)!개의 단위데이터로 구성된 전수데이터를 각각 생성하는 문제이다.

이 논문에서 이들을 각각 조합문제, 순열문제, r-순열문제로 약칭하고, 또한 세 문제를 통칭하여 순열조합 문제(combinatorial problem)[5-7]라고 부른다. 순열조합 문제는 단위데이터의 선언순서 지정여부로 문제를 세분할 수 있는데, 이 논문은 선언순서를 고려하지 않는 경우만을 다룬다.

본 연구는 다음과 같은 절차로 수행되었다. 우선 조합, 순열, r-순열 문제의 규칙을 만족하는 프로그램들을 수집하여 완성하였고, 확보된 프로그램의 수행시간을 측정하여 부문별로 가장 빠른 프로그램을 선정하였다. 선정된 프로그램에 대해서는 알고리즘을 소개하거나 프로그램의 의사코드(pseudo-code)를 제시하였다. 또한 추가연구를 통하여 수집된 프로그램보다 수행시간을 단축한 프로그램을 완성하였다.

논문의 구성은 다음과 같다. 2장은 수행할 순열조합 문제를 정의한 후에 가장 빠른 프로그램으로 선정된 알고리즘을 소개하고 있으며, 3장은 본 연구를 통해서 완성한 수행시간이 단축된 두 프로그램과 알고리즘을 제시하였다. 4장에서는 언급된 주요 프로그램들의 수행시간을 정리하였고, 5장에서 결론을 맺었다.

2. 수집된 프로그램

2장에서는 2.1절에서 프로그램의 수집과 논문에서 사용되는 공통적인 규칙에 대해서 다룬 후에 COMBI_Q(M,N)을 2.2절에서, PERMU _Q(N)을 2.3절에서, R-PERMU_Q(M,N)을 2.4절에서 각각 정의한 후 빠른 알고리즘을 설명한다.

이 논문에 소개된 모든 프로그램은 공히 다음과 같은 규칙을 따른다. 프로그램의 함수 ftn은 ftn( )로, 행렬 a는 a[ ]로 각각 표시된다.

전수데이터를 구성하는 기본단위를 단일데이터로 부른다. 함수 declare( )를 1회 호출할 때마다 1차 행렬 a[ ]에 보관된 단일데이터 한 개를 선언하며, 전수데이터에 속하는 단일데이터는 모두 정확하게 한 번만 선언해야 한다.

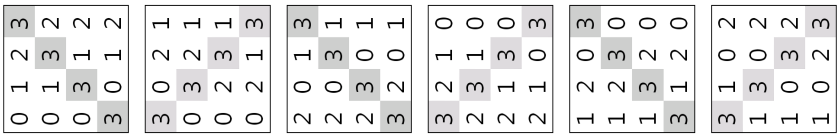
2.1 프로그램의 수집

본 연구진은 theory.cs.uvic.ca, geeksforgeeks. org, mathworld.wolfram.com를 포함한 다양한 인터넷 사이트에서 백 개 이상의 프로그램을 수집하였다. 수집된 프로그램들을 동작할 수 있도록 완성한 후에 세 개의 순열조합문제로 구분하여 수행시간 순으로 정리하였다. 또한 사전적인 순서로 전수데이터를 생성하는 프로그램은 수행시간에 관계없이 최소 한 개를 포함시켰다. 단, 조합 프로그램은 모두 사전적 순서로 출력하기 때문에 비사전적 순서의 프로그램은 제외하였다.

Table 1은 이러한 과정을 거쳐서 최종적으로 선발한 5개 프로그램과 자체적으로 완성한 3개 프로그램(회색 배경)의 목록이며, 각 프로그램별로 세 가지 속성에 대한 만족여부를 표시하고 있다. 연구에서 개발된 프로그램들은 별도로 3장에서 상세히 소개된다.

[Table 1] The fast program list and their properties.

Problems	programs /algorithms	attributes		
		lexico	G_0_G_{M-1}	recursive
COMBI_Q (M,N)	combi( )	○	×	×
	c1( )	○	×	○
PERMU_Q (N)	sj( )	×	×	○
	Gsj( )	×	○	○
	p1( )	×	○	○
	p2( )	○	○	○
R-PERMU_Q (M,N)	comGsj( )	×	×	○
	rp1( )	○	×	○



[Fig. 1] The permutation sequence in the total data by swapping two adjacent numbers (from top to bottom)

첫째 속성인 'lexico'는 단위데이터가 사전적인 순서로 출력되는지를 표시한다. lexico가 '○'이면 N개의 수로 구성된 단일데이터를 N-문자 단어로 간주하여 사전에 우선 등재된 순으로 단일데이터를 선언한다. 둘째 속성인 'G_0_G_{M-1}'이 '×'이면 주어진 M개의 데이터가 {0, 1, 2, ..., M-1}일 때에만 동작한다. 셋째 속성 'recursive'는 자신이 자신의 함수를 호출하는 재귀함수를 포함하는지를 표시한다. 일반적으로 재귀형식의 프로그램은 간단하지만 비재귀형식에 비해 수행시간이 증가하고, 응용에 적합하도록 변형하기가 어렵다.

[Table 2] Internet sources of the fast programs

program	source domain in the internet
c1( )	ftp://osinside.net/perm/perm.c
sj( )	http://theory.cs.uvic.ca/dis/distribute.pl.cgi?package=SJT.c
p1( ), p2( )	http://www.bearcave.com/random_hacks/permute.html
rp1( )	http://www.seba.nos30.ro/start.php?lang=en&pag=5_programare&sub=1_c_plusplus&location=1_combinari

Table 1의 수집된 프로그램에 대한 인터넷 출처는 Table 2에 정리하였다.

2.2 조합 전수데이터 프로그램 c1( )

조합문제에서는 선정된 순서가 고려되지 않기 때문에 두 개의 조합 단일데이터가 일치하는지를 확인하려면 N개의 숫자로 구성된 단일데이터를 일정한 규칙으로 배열해야 한다. 여기서는 순방향 정렬을 배열규칙으로 사용한다. c1( )[8]은 단일데이터를 읽고 평균적으로 1을 약간 상회하는 횟수의 읽기와 쓰기 동작만을 수행하여 새로운 단일데이터를 생성한다.

조합 전수데이터 생성 프로그램 c1( )과 본 연구에서 제안한 combi( )는 선택할 N개의 수를 a[0:N-1]에 순정렬로 배열한다. 재귀함수를 사용하는 프로그램 c1( )과 사용하지 않는 combi( )는 동일한 알고리즘을 다른 구조로 완성하였다. 재귀함수를 이용하면 동작이 중복되어 수행시간이 약간 증가한다. 비재귀형식의 조합 프로그램 combi( )는 3.1절에서 설명한다.

2.3 순열 전수데이터 프로그램 sj( )

순열 전수데이터 생성문제는 1970년대에 이미 거의 완벽한 수준까지 이론이 정립되었다. 여기서는 유용한 주요 이론을 소개한다.

우선 사전적 순서를 따르지 않는 N자리 단일데이터의 출력순서를 알아보자. 단일데이터의 초기값은 0 1 2 ... N-1 이며, 이후에 인접한 두 수를 맞바꾸어 새로운 단일데이터를 형성한다. N자리 순열 전수데이터는 (N-1)자리 전수데이터를 N 회씩 반복적으로 사용한다.

Fig. 1에서는 {0, 1, 2, 3}으로 구성된 4! 순열 전수데이터의 출력순서를 보여준다. 24개의 단일데이터로 구성된 전수데이터는 공간관계상 90도 회전하여 배치하였다. 한 개의 정사면체에서 최고의 숫자(이 경우에는 3)는 우에서 좌로 한 번을 스캔한 후에 다시 좌에서 우로 복귀한다. 또한 한 개의 정사면체에서 숫자 3을 제거하고 남은 세 자리 데이터는 모두 동일하다. 각 정사면체에는 012 021 201 210 120 102의 순서로 배정되는데 이들 세 자리 데이터는 네 자리 데이터에서 요구되는 규칙과 동일하게 생성된다. John son[9,10]은 Fig. 1의 순서로 전수데이터를 출력하는 재귀 알고리즘을 제안하였다.

Johnson의 알고리즘에서 좌우방향의 스캔동작을 위해서는 좌방향 프로그램과 우방향 프로그램으로 구성된 한 쌍의 유사한 루틴(routine)이 필요하다. Even은 보조메모리를 사용하여 두 루틴을 하나로 줄이는데 성공했다[11].

Fig. 2는 sj 알고리즘의 의사코드이다. sj 알고리즘은 단일데이터를 보관하는 행렬 a[ ] 외에도 좌는 -1 우는 1의 값을 갖는 방향행렬 dir[ ]과 Even의 아이디어를 구현하기 위해서 보조 메모리 ai[ ]를 사용한다. sj_ftn(x)는 x자리 전수데이터가 완성된 상태에서 (x+1)자리 전수데이터를 재귀적으로 완성한다. sj_ftn(x)가 호출하는 dmove(x,dir)에서는 a[x]의 내용을 a[x+dir]에 보관하고, a[x]에는 x를 입력한다. sj_ftn(x)는 Even의 아이디어를 포함하여 한 개의 for 문장만으로 동작하지만 함수 dmove( )는 ai[ ]를 추가적으로 포함하여 더 복잡해졌다.

재귀방식의 사전적 배열 기능이 없는 p1( )과 사전적 배열 기능이 있는 p2( )도 우수한 알고리즘이지만, Johnson 계열의 알고리즘의 성능이 더욱 우수하여 일반적으로 널리 알려져 있다.

```
1   int N;
2   int a[N], ai[N], dir[N];

3   sj( )
      // Line 4 to 6 are iset_sj.
4     a[0]=0, a[1]=1,..., a[N-1]=N-1;
5     ai[0]=0, ai[1]=1,..., ai[N-1]=N-1;
6     dir[0]=-1, dir[1]=-1,..., dir[N-1]=-1;
7     sj_ftn(0);
8   end sj;

9   sj_ftn(x)
10    if (x > N-1) then
11    else
12      sj_ftn(x+1);
13      for i = 0 to x-1 do
14        move(x, dir[x]);
15        sj_ftn(x+1);
16      end for;
17      dir[x] = -dir[x];
18    end if;
19  end sj_ftn;

20  move(x, dir)
21    int z = a[ai[x]+dir];
22    a[ai[x]] = z;
23    a[ai[x]+dir] = x;
24    ai[z] = ai[x];
25    ai[x] = ai[x]+dir;
26  end move;
```

[Fig. 2] The pseudo-code of sj( ) : our best solution of PERMU_Q(N)

2.4 r-순열 전수데이터 프로그램 rp1( )

r-순열문제는 활발히 논의되지 않는 주제이기 때문에 해법이 많지 않다. 이 중에서 가장 우수한 프로그램을 rp1( )로 명명하였으나 rp1( )보다 더 빠른 프로그램을 발견하여 rp1( )에 대해서는 의사코드는 포함하지 않고 알고리즘만을 설명한다. rp1( ) 알고리즘에서는 보조행렬 b[i]에 숫자 i의 기사용여부를 기록하여 숫자의 재사용을 방지한다.

rp1_ftn(k,M,N)는 주행렬 a[0:k-1]값이 배정된 상태에서 호출되며 동작 중에 a[k]의 값을 지정한다. a[0:k-1] 구간에 숫자 i가 이미 배정되었다면 b[i]에 1이, i가 아직 배정되지 않았다면 b[i]에 0이 기록된다. a[k]에는 i를 보관하는데, 이 i는 미배정되어 b[i]=0을 만족해야 한다.

rp1_ftn(k,M,N)은 for 문장을 이용하여 b[i]가 0인 모든 i에 대해서 다음을 수행한다. i) a[k]=i를 배정하고, ii) b[i]=1로 바꾸고, iii) rp1_ftn(k+1,M,N)을 호출하며, iv) 재귀함수가 종료된 직후에 b[i]=0로 환원시킨다.

rp1( )은 대소의 비교 없이 행렬의 앞에서부터 사용여부만으로 배정할 숫자를 선정하기 때문에 단일데이터를 사전적 순서로 선언한다. rp1( )은 Table 1에서 'G_0_G_{M-1}'의 속성을 만족하지 않지만 고정행렬 c에 임의의 데이터를 순정렬하여 할당한 후에, a[k]=i의 명령어를 a[k]=c[i]로 교체하면 'lexico'와 'G_0_G_{M-1}'를 모두 만족시킬 수 있다.

3. 제안하는 프로그램

3장에서는 연구를 통해 속도가 개선된 두 개의 프로그램을 소개한다.

3.1 조합 전수데이터 프로그램 combi( )

3.1절에서는 비재귀형식의 combi(M,N)을 소개한다. 알고리즘 combi( )의 핵심은 a[N-k-1]≠M-k를 만족하는 k의 최대값인 k^+의 계산이다. k^+가 존재하면 a[N-k^+-1]의 값을 j라 할 때 $0 \le k \le k^+$의 모든 k에서 a[N-k-1]에 j+1+(k^+-k)를 할당한다. 만일 k^+가 존재하지 않은 경우, 즉 $0 \le k \le (N-1)$ 전 범위에서 a[N-k-1]==M-k가 성립하면 프로그램은 종료된다. Fig. 3은 combi(M,N)의 의사코드를 보여준다.

```
1   int M, N;
2   int a[N];
3   combi( )
4     a[0]=0,a[1]=1,...,a[N-1]=N-1; // iset_combi
5     while (combi_ftn( ) == true) do
6     end while;
7   end combi;

8   combi_ftn( )
9     int x = N-1;
10    while (a[x] == (M-N+x)) do
11      if (x-- == 0) then
12        return(false);
13    end while;
14    a[x]++;
15    int xi = x;
16    while (xi < N-1) do
17      a[++xi] = a[x] + (xi-x);
18    end while;
19    return(true);
20  end combi_ftn;
```

[Fig. 3] The pseudo-code of combi( ) : our best solution of COMBI_Q(M,N)

3.2 r-순열 전수데이터 프로그램 comGsj( )

3.2절에서는 R-PERMU_Q(M,N) 문제를 3.1절의 combi(M,N)와 2.3절의 sj(N)을 변형한 Gsj (N)를 복합적

으로 사용하는 comGsj(M,N)을 소개한다.

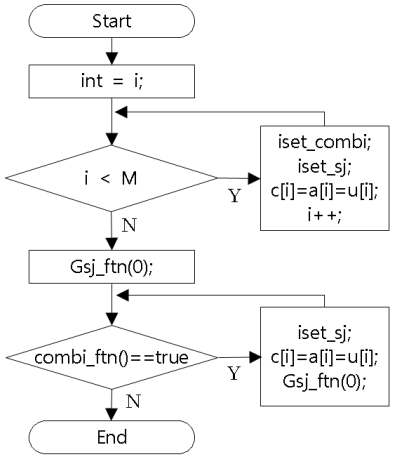
R-PERMU_Q(M,N)에서 생성되는 전수데이터 의 수 M!/(N!(M-N)!)는 COMBI_Q(M,N)의 전수데이터 수 M!/(M-N)!와 PERMU_Q(M,N)의 전수데이터 수 N!의 곱과 일치한다. 이러한 특성을 이용해서 comGsj(M,N)는 combi(M,N)를 통해 M개의 데이터 중에서 N개를 선택하고 다시 Gsj(N)을 활용하여 모든 순열을 나열할 수 있다. combi로 선택한 N개의 수는 {0, 1, ..., (N-1)}로 고정되지 않기 때문에, comGsj( )에서는 일반적인 입력값에도 동작이 가능하도록 하기 위해서 sj(N)를 일반화한 Gsj(N)을 사용한다.

```
 1     int M,N,r,ri;
 2     int a[N], b[N], bi[N], dir[N], c[N], u[N];

 3     comGsj( )
 4       u[0]=0,u[1]=1,...,u[M-1]=M-1; // iset_combi
         // Lines 5 to 7 are iset_sj.
 5       b[0]=0, b[1]=1,..., b[M-1]=M-1;
 6       bi[0]=0, bi[1]=1,..., bi[M-1]=M-1;
 7       dir[0]=-1, dir[1]=-1,..., b[M-1]=-1;
         // Lines 5 to 7 are iset_G(in Gsj)
 8       c[0]=u[0], c[1]=u[1],..., c[M-1]=u[M-1];
 9       a[0]=u[0], a[1]=u[1],..., a[M-1]=u[M-1];
 10      Gsj_ftn(0);
 11      while (combi_ftn( ) == true) do
12-16      Put the commands in the lines 5 to 9.
 17        Gsj_ftn(0);
 18      end while;
 19    end comGsj;

 20    combi_ftn( )
 21      r = N-1;
 22      while (u[r] == (M-N+r)) do
 23        if (r-- == 0) then
 24          return(false);
 25      end while;
 26      u[r]++;
 27      int ri = r;
 28      while (ri < N-1) do
 29        u[++ri] = u[r] + (ri-r);
 30      end while;
 31      return(true);
 32    end combi_ftn;

 33    Gsj_ftn(x)
 34      if (x > N-1) then
 35      else
 36        sj_ftn(x+1);
 37        for i = 0 to x-1 do
 38          Gmove(x, dir[x]);
 39          Gsj_ftn(x+1);
 40        end for;
 41        dir[x] = -dir[x];
 42      end if;
 43    end Gsj_ftn;

 44    Gmove(x,dir)
 45    int bix = bi[x],
 46        bixd = bix + dir;

47, 48  int z = b[bixd];          int zz = a[bixd];
49, 50      b[bix] = z;              a[bix] = zz;
51, 52      b[bixd]=x;               a[bixd]=c[x];

 53         bi[z]=bix;
 54         bi[x]=bixd;
 55    end Gmove;
```

[Fig. 4] The pseudo-code of comGsj( ) : our best solution of R-PERMU_Q(M,N)

comGsj( )에서 사용되는 combi( )와 sj( ) 두 함수는 자체 행렬을 확보하기 위해서 독립적으로 동작하는 경우와 다른 별도 이름의 행렬을 사용한다. Fig. 4의 comGsj( )의 의사코드에서 comGsj( )는 combi( )가 행렬 u[0:N-1]에 사용할 N개의 수를 결정하는 초기작업 완료 후에 Gsj( )가 작동된다. Gsj( )는 sj( )의 기본 기능을 구현하기 위해서 b, bi, dir 행렬을 사용하며, 입력데이터 일반화기능 G를 구현하기 위해서 고정행렬 c와 단일데이터 보관과 출력용으로 행렬 a를 사용한다.

Fig. 4를 순서도로 나타내면 그림 5와 같다. iset_combi는 Fig. 4에서 4번째 줄을 나타내고, iset_sj는 Fig. 4에서 5~7번째 줄을 나타낸 것이다.



[Fig. 5] The flowchart of comGsj( )

Gsj(N)의 차이점은 sj(N)에서 함수 move( ) 대신에 Gmove( )를 호출하는 것이다. move( ) 함수에서 Gmove( )로 바뀌는 과정에서 실제로 추가되는 명령어는 48, 50,

52줄이다. 이들은 좌측의 47, 49, 51의 명령어에서 사용한 포인터는 고정한 채로 변수명만 바뀐다. Gmove( )에서는 포인터의 중복적인 계산을 방지하기 위해서 추가적으로 (라인 45에서 48까지에서) 변수를 정의한다.

4. 프로그램 성능평가

4장에서는 2.1절에서 제시한 표 1의 여러 프로그램들을 실제로 구동하여 동작에 소요되는 시간을 측정하고 비교한다. 4.1절에서는 프로그램을 공정하게 평가하는데 필요한 원칙 및 측정 환경을 소개하고, 4.2절에서는 측정결과를 정리하고 분석한다.

4.1 성능평가 환경

여러 프로그램을 동일한 조건에서 테스트하기 위해서 다음과 같은 요구사항을 만족시키는 프로그램만을 사용하였다. 우선 단위데이터를 출력하는 declare( ) 함수는 한 번 호출에 단 한 개의 단위데이터를 출력하도록 제안하였으며, 실제 동작시간을 측정할 때에는 I/O 동작을 수행하지 않는다. 측정되는 시간은 OS에서 제공되는 time( )과 같은 함수를 사용하여 초기화 직전부터 전수데이터를 선언하기까지의 시간을 동작시간으로 정한다. 공정한 평가를 위해서 측정할 프로그램을 PC에서 단독으로 수행시키며, 동작시간은 동일한 10 번의 실험을 통해 얻은 10개의 평균 동작시간으로 정한다. 실험은 3.40㎓ 쿼드코어 CPU, 4㎇ DRAM이 설치되고, MS Windows 7이 동작하는 PC에서 이루어졌다.

4.2 성능평가

성능평가는 조합, 순열, r-순열의 세 분야로 구분되어 진행된다. OS가 제공하는 clock의 최소단위인 1㎳이상의 수행시간을 요하는 문제를 선정하여 Table 3에 표시하였다. Table 3은 또한 각 실험에서 출력할 단위데이터의 개수를 10^6 단위로 적고 있다. 가령 Table 3.(b)의 N이 10일 경우에서 3.6이란 값은 10!개의 단위데이터가 약 $3.6 \cdot 10^6$ 개임을 의미한다.

[Table 3] Parameters used in the problems and the size of total data under test (1 means 10^6 unit data.)

(a) COMBI_Q(M,N) problem

(M,N)	(30,10)	(30,15)	(40,10)	(45,10)	(50,10)
number of unit data	30.0	155.1	847.7	3190.2	10,272

(b) PERMU_Q(N) problem

N	10	11	12	13	14
number of unit data	3.6	39.9	479	6,227	87,178

(c) R-PERMU_Q(M,N) problem

(M,N)	(12,8)	(12,10)	(16,8)	(15,10)	(16,10)
number of unit data	20	239.5	518.9	10,897	29,059

Table 4, 5, 6은 Table 3에서 각각 조합, 순열, r-순열 문제로 주어진 수치를 적용했을 경우에 동작하는 시간을 ㎳ 단위로 정리하였으며, 동작시간이 빠른 프로그램을 위쪽에 표시하여 정렬하였다. 또한 Fig. 6, 7, 8에서는 각 프로그램의 상대적인 빠르기를 가장 빠른 프로그램이 기준인 막대그래프로 표시하였다. 즉 Fig. 6, 7, 8은 각각 Table 4, 5, 6에서 언급한 수행시간을 A_4, A_5, A_6로 표시된 Table의 맨 윗칸에 기록된 수행시간으로 나눈 값을 그린다. Table에서는 작은 수가, 막대그래프에서는 짧은 막대가 빠른 프로그램임을 나타낸다.

[Table 4] The execution time of combination programs (unit = second)

program \ (M,N)	(30,10)	(30,15)	(40,10)	(45,10)	(50,10)
A_4 : combi( )	0.07	0.47	1.84	7.07	21.60
B_4 : c( )	0.09	0.63	2.31	8.76	26.29
enhancement : B_4/A_4	1.31	1.34	1.26	1.24	1.22

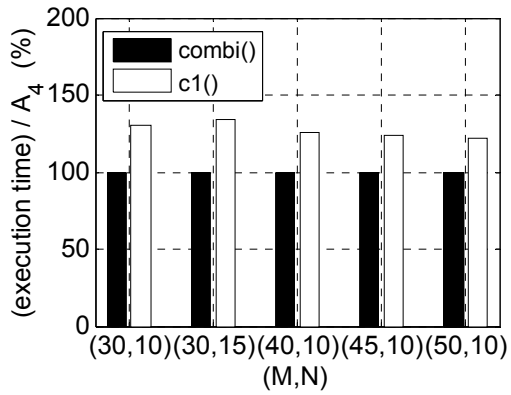
[Table 5] The execution time of permutation programs (unit = second)

program \ N	10	11	12	13	14
A_5 : sj( )	0.02	0.17	2.06	26.20	370.04
Gsj( )	0.02	0.20	2.38	31.22	396.45
p1( )	0.03	0.34	4.03	52.70	751.06
p2( )	0.04	0.48	5.77	75.02	1082.82

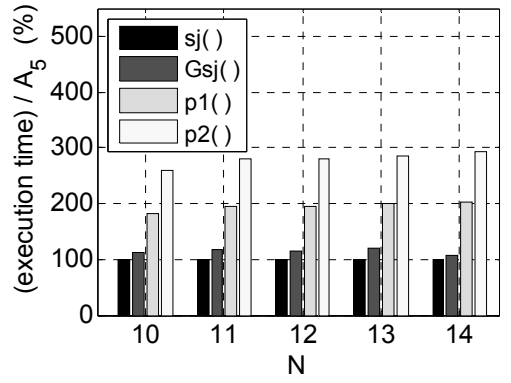
[Table 6] The execution time of r-permutation programs (unit = second)

program \ (M,N)	(12,8)	(12,10)	(16,8)	(15,10)	(16,10)
A_6 : comGsj( )	0.09	1.01	2.28	46.09	122.71
B_6 : rp1( )	0.21	3.29	3.71	110.59	264.72
enhancement : B_6/A_6	2.39	3.26	1.62	2.40	2.16

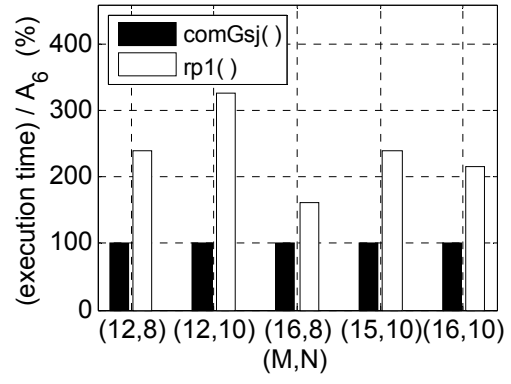
Table 4와 Table 6은 본 연구에서 개발한 프로그램이 인터넷에서 얻은 가장 빠른 프로그램에 비해 감소한 동작시간의 정도를 개선도(enhancement)로 표시하고 있다.



[Fig. 6] Relative execution time of combination programs based on combi( )



[Fig. 7] Relative execution time of permutation programs based on sj( )



[Fig. 8] Relative execution time of r-permutation programs based on comGsj( )

이러한 결과를 바탕으로 본 연구는 다음과 같은 사실을 유추하였다.

- 본 연구에서 수집한 프로그램 중에서 조합 전수데이터를 가장 빠르게 생성하는 프로그램은 combi( )이다. 또한 combi( )는 두 번째 빠른 프로그램 c1( )에 비해서 22%에서 34% 범위로 개선도가 향상되었다.
- 본 연구에서 수집한 프로그램 중에서 순열 전수데이터를 가장 빠르게 생성하는 프로그램은 sj( )이다.
- 본 연구에서 수집한 프로그램 중에서 r-순열 전수데이터를 가장 빠르게 생성하는 프로그램은 comGsj( )이다. 또한 comGsj( )는 두 번째 빠른 프로그램 rp1( )에 비해서 62%에서 226%만큼 수행속도가 향상되었다.
- 단일데이터 하나를 생성하는데 소요되는 시간의 관점에서 조합, 순열, r-순열 문제를 비교하면 (조합 프로그램 동작시간 < 순열 프로그램 동작시간 < r-순열 프로그램 동작시간) 이다.

5. 결론

이 논문은 조합, 순열, r-순열 세 가지 규칙 하에 전수데이터를 생성하는 프로그램과 알고리즘을 수집하고 이들 중에서 가장 빠른 프로그램을 선정하여 이에 대한 의사코드와 동작방법을 설명하고 있다. 이러한 절차를 통해서 조합, r-순열 전수데이터 생성 문제에서는 수집한 프로그램 중 가장 빠른 프로그램보다 수행시간을 더욱 단축한 프로그램을 개발하였다.

이 논문은 다음과 같은 방법으로 전수데이터를 이용한 응용에서 활용할 수 있다. 첫째, 본 논문에서 제공한 의사코드를 바탕으로 응용에 맞게 전수데이터를 쉽게 변형시킬 수 있다. 둘째, 결과에 제시된 동작시간을 참고하여 전수조사 방법에 소요되는 수행시간을 예측하여 전수조사의 타당성 여부를 결정할 수 있다. 셋째, 전수조사를 실제로 실행할 경우 본 논문에서 제공한 코드를 바탕으로 최소의 시간으로 전수데이터를 생성할 수 있다.

이 논문에서 제시한 알고리즘들은 산업적인 제품의 전수검사 뿐만 아니라 망에서의 전체 라우팅 경로를 찾기 위한 방법, 무선망에서의 비경쟁구간을 할당하기 위한 알고리즘으로 활용이 가능할 것으로 기대된다.

References

[1] Michael R. Fellows, et al., “Local Search: Is brute-force avoidable?,” *J. Comput. Syst. Sci. (JCSS)*, vol.78, no.3, p.707-719, 2012.
DOI: http://dx.doi.org/10.1016/j.jcss.2011.10.003

[2] A. B. Morton, I. M. Y. Mareels, “An Efficient Brute-

Force Solution to the Network Reconf. Problem," *IEEE Trans. on Power Delivery*, vol.15, p.996-1000, Jul. 2000. DOI: http://dx.doi.org/10.1109/61.871365

[3] L. Barton, B. Sharangpani, "Statistical Analysis of Floating Point Flaw in the Pentium Processor," *Intel Corp.*, Nov. 1994.

[4] Cleve Moler, "Pentium Division Bug Documents," *MATLAB Central*, May 2002.

[5] Selim G. Akl, Henk Meijer, Ivan Stojmenovic, "An Optimal Systolic Algorithm for Generating Permutations in Lexicographic Order," *J. Parallel Distr. Comput.*, vol.20 no.1, pp.84-91, 1994. DOI: http://dx.doi.org/10.1006/jpdc.1994.1008

[6] Vince Vatter, "Enumeration Schemes for Restricted Permutations," *Combinatorics, Probability and Comp.*, vol.17, p.137-159, 2008. DOI: http://dx.doi.org/10.1017/S0963548307008516

[7] Hesterberg, Tim C., "Perf. Evaluation using Fast Permutation Tests," *Proc. of 10'th Int. Conf. on Telecomm. Systems*, p.465-474, 2002.

[8] Skiena S., "Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica," *Addison-Wesley*, p.40-46, 1990.

[9] Selmer M. Johnson, "Generation of Permutations by Adjacent Transposition," *Mathematics of Computation*, vol.17, no.83, p.282-285, Jul. 1963. DOI: http://dx.doi.org/10.1090/S0025-5718-1963-0159764-2

[10] Youssef Bassil, "A Comparative Study on the Performance of Permutation Algorithms," *Journal of Computer Science & Research(JCSCR)*, vol.1, no.1, p.7-19, 2012.

[11] H. F. Trotter, "Algorithm 115: Perm," *Communications of the ACM,* vol.5, issue.8, p.434-435, Aug. 1962. DOI: http://dx.doi.org/10.1145/368637.368660

장 재 수(Jae-Soo Jang) **[정회원]**

- 2004년 2월 : 숭실대학교 일반대학원 컴퓨터학과 (공학석사)
- 2004년 3월 ~ 현재 : 숭실대학교 일반대학원 컴퓨터학과 박사과정
- 2005년 8월 ~ 2007년 11월 : (주)젠솔소프트 선임연구원
- 2007년 11월 ~ 2011년 6월 : LH 책임연구원
- 2011년 12월 ~ 현재 : 한국보건복지정보개발원 책임

<관심분야>
정보보호, 정보통신, 센서네트워크, U-City

신 재 원(Jae-Won Shin) **[준회원]**

- 2013년 2월 : 호서대학교 컴퓨터공학과 (공학사)
- 2013년 3월 ~ 현재 : 숭실대학교 일반대학원 컴퓨터학과 석사과정

<관심분야>
정보경영, 정보통신

전 홍 식(Hongsik J. Cheon) **[정회원]**

- 1998년 8월 : State University of New York at Buffalo(MS, Bio-Statistics)
- 2004년 8월 : University of Florida(Ph.D. of Advertising)
- 2004년 8월 ~ 2008년 8월 : Professor of Marketing, State University of Maryland at Frostburg
- 2008년 9월 ~ 현재 : 숭실대학교 경영학부 교수

<관심분야>
소비자 행동, 소비자 심리

서 창 진(Chang-Jin Suh) **[정회원]**

- 1984년 2월 : 서울대학교 제어계측학과 (공학석사)
- 1996년 2월 : Univ. of Massachusetts at Amherst, ECE (공학박사)
- 1985년 10월 ~ 1990년 7월 : 한국전자통신연구원 선임연구원
- 1996년 4월 ~ 1997년 2월 : 삼성전자 네트워사업부 부장
- 1997년 3월 ~ 현재 : 숭실대학교 컴퓨터학부 교수

<관심분야>
센서네트워크, 캐리어 이더넷, 스위칭/라우팅 이론