The Journal of The Institute of Internet, Broadcasting and Communication (IIBC)
Vol. 22, No. 3, pp.185-191, Jun. 30, 2022. pISSN 2289-0238, eISSN 2289-0246

https://doi.org/10.7236/JIIBC.2022.22.3.185
JIIBC 2022-3-27

거스름돈 만들기 문제의 정확한 나눗셈 알고리즘

An Exact Division Algorithm for Change-Making Problem

이상운*

Sang-Un, Lee*

요 약 본 논문은 NP-난제로 다항시간 알고리즘이 알려져 있지 않은 거스름돈 만들기 문제(CMP)에 대해 $O(\frac{n(n+1)}{2})$ 수행 복잡도의 나눗셈 알고리즘을 제안하였다. CMP는 주어진 돈 C를 $c_j, j=1,2,\cdots,n$의 동전으로 교환할 경우 교환되는 동전 개수 x_j의 합을 최소화 시키는 문제이다. CMP에 대해 알려진 다항시간 알고리즘으로는 욕심쟁이 알고리즘(GA), 분할정복(DC)과 동적 계획법(DP)이 있으나 최적 해는 $O(nC)$의 DP로 구할 수 있으며, 일반적으로 $C>2^n$으로 주어진 경우 수행 복잡도는 지수적으로 증가하는 경향이 있어 다항시간 알고리즘이라고 할 수 없다. 본 논문에서는 $c_j \le C$에 한해, j 열에 n개의 c_j 내림차순으로 배치하고, i행에는 c_j의 약수를 모두 제외시킨 k개의 동전을 배치한 $k \times n$ 행렬에 대해 상삼각행렬과 주대각선 셀에 대해 나눗셈을 하여 몫(quotient)을 구하는 단순한 알고리즘을 제안한다. 제안된 알고리즘을 다양한 유형의 39개 벤치마킹 실험 데이터에 적용한 결과 최적 해를 단순히 계산기만을 사용하여도 빠르고 정확하게 구할 수 있음을 보였다.

Abstract This paper proposed a division algorithm of performance complexity $O(\frac{n(n+1)}{2})$ for a change-making problem(CMP) in which polynomial time algorithms are not known as NP-hard problem. CMP seeks to minimize the sum of the x_j number of coins exchanged when a given amount of money C is exchanged for $c_j, j=1,2,\cdots,n$ coins. Known polynomial algorithms for CMPs are greedy algorithms(GA), divide-and-conquer (DC), and dynamic programming(DP). The optimal solution can be obtained by DP of $O(nC)$, and in general, when given $C>2^n$, the performance complexity tends to increase exponentially, so it cannot be called a polynomial algorithm. This paper proposes a simple algorithm that calculates quotient by dividing upper triangular matrices and main diagonal for $k \times n$ matrices in which only j columns are placed in descending order of c_j of n for $c_j \le C$ and i rows are placed k excluding all the dividers in c_j. The application of the proposed algorithm to 39 benchmarking experimental data of various types showed that the optimal solution could be obtained quickly and accurately with only a calculator.

Key Words : Change making problem, Dynamic programming, Divider, Multiply, Division algorithm

*정회원, 강릉원주대학교 과학기술대학 멀티미디어공학과
접수일자 2022년 2월 5일, 수정완료 2022년 5월 12일
게재확정일자 2022년 6월 10일

Received: 5 February, 2022 / Revised: 12 May, 2022 / Accepted: 10 June, 2022
*Corresponding Author: sulee@gwnu.ac.kr
Dept. of Multimedia Eng., Gangneung-Wonju National University, Korea

Ⅰ. 서 론

주어진 교환 대상 거스름돈 총액 C를 n개의 서로 다른 양의 정수 값을 갖는 동전(coin, c) $c_j, j=1,2,\cdots,n$으로 교환하는 경우 교환되는 거스름 돈(동전)의 개수(number, n) x_i의 합이 최소가 되는 z_n를 찾는 문제를 거스름돈 만들기 문제(change-making problem, CMP)라 한다.[1]

Pearson[2]은 CMP가 다항시간 알고리즘이 알려져 있지 않은 NP-난제(NP-hard)로, $O(n^3)$의 다항시간 알고리즘을 제안하기도 하였다.

CMP에 대해서는 욕심쟁이 알고리즘(greedy algorithm, GA)[3,4], 분할정복(divide-and-conquer, DC)[5]과 동적계획법(dynamic programming, DP)[1,6-9]가 있다. GA는 $C-\max c_j$의 재귀뺄셈법을 반복수행하는 단순한 측면이 있지만 최적 해를 찾지 못하는 경우도 발생한다. DC는 근 노드를 C로 자식 노드를 뺄셈 가능한 c_j개로 하는 가능한 경우의 레벨을 재귀적으로 모두 형성하는 트리뺄셈법(tree subtraction method)으로 C가 증가하면 수행시간은 지수적으로 증가하는 경향이 있어 적용에 어려움이 있다. DP는 n행, C열의 행렬에 대해 재귀적으로 나눗셈의 몫을 계산하는 방법으로 수행 복잡도는 $O(nC)$로 준다항시간으로 알려져 있다. 그러나 C가 2^n보다 큰 경우 지수시간 복잡도를 갖는다. 따라서 지금까지 알려진 CMP의 최적 해를 찾는 알고리즘으로는 다항시간으로 해를 구하는 방법이 없는 실정이다.

본 논문은 CMP에 대해 최악(worst case)의 경우 $O(\frac{n(n+1)}{2})$ 복잡도, 최적(best case)의 경우 $O(n)$ 복잡도를 갖는 나눗셈 알고리즘을 제안한다. 2장에서는 CPM의 해를 얻는 GA, DC와 DP 문제점을 고찰해 본다. 3장에서는 $O(\frac{n(n+1)}{2})$ 수행 복잡도로 CMP의 정확한(최적) 해를 얻는 나눗셈 알고리즘(division algorithm, DA)을 제안한다. 4장에서는 다양한 유형의 벤치마킹 실험 데이터를 대상으로 제안된 알고리즘의 적합성을 검증한다.

Ⅱ. 관련연구와 문제점

주어진 교환 대상 거스름돈의 총액인 C를 n개의 서로 다른 양의 정수 값(distinct positive integer values)을 갖는 동전(coin, c) $c_j, j=1,2,\cdots,n$이 오름차순인 $c_1 < c_2 < \cdots, c_n$으로 주어졌을 때 CMP는 교환되는 거스름돈(동전)의 무게(weight, w_j)는 고려하지 않으며 단지 개수 x_i의 합이 최소가 되는 z를 찾는 문제로 식 (1)과 같이 수학적 모형으로 정의될 수 있다.[1] 결국 CMP는 동전의 무게는 고려하지 않고 단지 개수의 최소화를 추구하는 문제라 할 수 있다.

$$z = \underset{minimize}{} \sum_{j=1}^{n} x_j \quad (1)$$

$$\text{subject to } \sum_{j=1}^{n} c_j x_j = C.$$

CMP를 풀 수 있는 알고리즘으로 지금까지 알려진 방법은 욕심쟁이 알고리즘(GA)[3,4,10], 분할정복(DC)[5,11]와 동적계획법(DP)[1,6-9]이 있다.

GA는 $C-\max c_j$를 반복 수행하는 재귀 뺄셈법(recursive subtraction method, RSM)으로 계속해서 감산을 수행하여 수행횟수가 클 수 있는 단점과 더불어 원, 달러 등 일반적인 표준 동전 체계(canonical coin system)인 c_j ={1c,5c,10c,25c,50c,100c} 또는 {10원,50원,100원,500원} 등과 같은 통화(currency)[13]에는 최적 해를 얻을 수 있지만 다른 단위로 주어지면 최적 해를 찾지 못하는 단점도 갖고 있다.

DC는 근 노드를 C로 자식 노드를 뺄셈 가능한 c_j개로 하는 가능한 경우의 레벨을 재귀적으로 모두 형성하는 트리뺄셈법(tree subtraction method)[14]으로 C가 증가하면 수행시간은 지수적으로 증가하는 경향이 있어 적용에 어려움이 있다. StaksOverflow[5]에 따르면 C= 50,55,60,65,70에 대해 c_j ={1,5,10,12,25,50}인 경우 컴퓨터 수행시간(초)은 1,5,20,126,613초로 지수적으로 증가하여 실제적으로 활용이 불가함을 보였다.

DP는 n행, C열의 행렬에 대해 재귀적으로 나눗셈의 몫을 계산하는 방법으로 수행 복잡도는 $O(nC)$로 준다항시간으로 알려져 있다. 그러나 C가 2^n보다 큰 경우 지수시간 복잡도를 갖는다.

은행이나 시내버스, 자판기 등 거스름돈을 지불하는 수많은 분야에서는 동전 개수 최소화 또는 무게 최소화 목적을 달성하기 위해 CMP를 지속적으로 활용함에도 불구하고, 지금까지 알려진 CMP의 해를 찾는 알고리즘으로는 다항시간으로 해를 구하는 방법이 없는 실정으로 실무에 적용이 불가한 실정이다.

따라서 3장에서는 단순 계산기(또는 핸드폰)만 소지하고 있더라도 CMP의 최적 해를 빠르고 정확하게 찾을 수 있는 알고리즘을 제안한다.

Ⅲ. 다항시간 정확한 알고리즘

본 장에서는 나눗셈법을 적용한 알고리즘을 제안한다. 주어진 문제에서 $c_j > C$ 이면 해당 c_j 동전을 교환할 수 없으므로 교환 대상 동전 후보에서 삭제한다. 개수(또는 무게) 최소화 목표를 달성하기 위해 나머지 후보들을 c_j 내림차순 정렬한다. 또한 $c_k = ac_i (a=2,3,\cdots)$으로 c_k가 c_i의 배수(역으로 c_i는 c_k의 약수)가 된다면 당연히 c_k 개수가 c_i 개수보다 적다는 속성을 적용하여 $c_j, j=1,2,\cdots,n$에서 약수(divisor)를 제외시킨 k개를 행 i에 배정한다. 열에는 $c_j, j=1,2,\cdots,n$를 배정한다. 이와 같이 $k \times n$ 행렬의 각 행 i에 대해 $c_i = c_j$부터 나머지 $r_{j-1} \leftarrow C$에 대해 $r_{j-1} \geq c_j$인 경우 $q_{ii} = quotient(r_{j-1}, c_j)$로 나눗셈의 몫(quotient)을 계산하고 나머지(modulo) $r_{j+1} = \mathrm{mod}(r_j, c_j)$는 다음 j열로 이동시키는 과정을 n열까지 반복 수행한다. 제안된 나눗셈 알고리즘(division algorithm, DA)의 기본 개념은 그림 1과 같다.

	c_n	c_{n-1}	$\cdots$	c_1	$z(\Sigma q_j)$
c_k	q_{ij}	q_{ij}	q_{ij}	q_{ij}	
c_{k-1}	-	q_{ij}	q_{ij}	q_{ij}	
$\cdots$	-	-	q_{ij}	q_{ij}	
c_1	-	-	-	q_{ij}	
최적 해					$\min z$

그림 1. 나눗셈 알고리즘 개념
Fig. 1. Concept of division algorithm

제안된 알고리즘은 $c_j, j=1,2,\cdots,n$의 n개 동전 액면가(coin denomination)에서 약수가 전혀 없는 n개가 교환 대상 후보인 최악의 경우라 할지라도 $k \times n = n \times n$ 행렬의 $i=j$인 주 대각선(main diagonal) n개와 $i<j$인 상 삼각행렬(upper triangle matrix)의 $\frac{n(n-1)}{2}$에 대해서만 나눗셈을 수행하므로 알고리즘 수행 복잡도는 $O(\frac{n(n+1)}{2})$이다. 제안된 나눗셈 알고리즘(DA)은 그림 2와 같이 수행된다.

제안된 알고리즘은 일반적인 표준 동전 체계인 c_j = {1c,5c,10c,25c,50c,100c} 또는 {10원,50원,100원,500원} 등과 같은 통화인 경우 약수를 제외하면 100c, 500원 등 유일한 $k=1$이 존재하여 알고리즘 수행 복잡도는 $O(kn) = O(n)$의 선형시간 복잡도로 최적 해를 얻을 수 있는 장점이 있다.

- $c_j > C$ 삭제.
- c_j 내림차순 정렬.
- $c_j, j=1,2,\cdots,n$를 열 j에 배정.(c_j)
- c_j에서 약수 삭제. 약수가 아닌 c_j의 k개를 행 i에 배정.

$r_1 \leftarrow C$.
for $i=1$ to k /* 행 */
  for $j=i$ to n /* 열 */
    if $r_j \geq c_j$ then
      $q_{ij} = quotient(r_j, c_j)$, $z_n = z_n + q_{ij}$,
      $z_w = z_w + q_{ij} \times w_j$
      $r_{j+1} = \mathrm{mod}(r_j, c_j)$
    else if $r_j < c_j$ then continue
    else if $r_j = 0$ then exit.
  end for
  if $r_k > 0$ then $z_n = \infty$
  $i = i+1$, $j = i$, $r_j \leftarrow C$.
end for
$z_n^* = {}_{min} z_n$, $z_w^* = {}_{min} z_w$

그림 2. 나눗셈 알고리즘
Fig. 2. Division algorithm

다음의 CMP_1 예제 문제에 대해 DA를 수행한 결과는 그림 3과 같다.

CMP_1 : c_j ={1,5,10,20}, C=36

C=36	20	10	5	1	$z(\Sigma q_j)$
20	36/20 q=1, r=16	16/10 q=1, r=6	6/5 q=1 r=1	1/1 q=1 r=0	1+1+1+1 =4
10	-	36/10 q=3 r=6	6/5 q=1 r=1	1/1 q=1 r=0	3+1+1 =5
5	-	-	36/5 q=7 r=1	1/1 q=1 r=0	7+1 =8
1	-	-	-	36/1 q=36 r=0	36

최적 해 : 20x1+10x1+5x1+1x1
(a) Worst case

C=36	20	10	5	1	$z(\Sigma q_j)$
20	36/20 q=1, r=16	16/10 q=1, r=6	6/5 q=1 r=1	1/1 q=1 r=0	1+1+1+1 =4

최적 해 : 20x1+10x1+5x1+1x1
(b) Best case

그림 3. CMP_1에 관한 DA
Fig. 3. DA for CMP_1

그림 3에서는 c_j={20,15,10,1}에서 약수를 제외하면 {20}만 존재하여 c_i={20}이 되어야 하지만 이러한 약수

제거의 타당성을 검증하기 위해 (a)에서는 약수 미 제거인 경우, (b)에서는 약수 제거인 경우를 모두 제시하였다. c_j={20,15,10 ,1}으로 약수를 모두 제거한 $_{\max}c_j = 20$에서 최적 해를 얻을 수 있음을 알 수 있다.

Ⅳ. 적용 및 결과 분석

본 장에서는 표 1의 38개 벤치마킹 실험 데이터를 대상으로 제안된 DA를 적용해 본다.

표 1. 벤치마킹 실험 데이터
Table 1. Benchmarking experimental data

CMP_2 :	c_j ={10,50,100,500}, C=8,600
CMP_3 :	c_j ={10,50,100,500}, C=3,600
CMP_4 :	c_j ={10,50,100,500}, C=8,980
CMP_5 :	c_j ={10,50,100,500}, C=3,960
CMP_6 :	c_j ={10,50,100,500}, C=9,300
CMP_7 :	c_j ={10,50,100,500}, C=4,300
CMP_8 :	c_j ={1,5,10,25,50}, C=91,500
CMP_9 :	c_j ={10,50,100,500}, C=1,300,000
CMP_{10} :	c_j ={10,50,100,500}, C=548,770
CMP_{11} :	c_j ={1,5,10,20}, C=36[14,15]
CMP_{12} :	c_j ={1,5,10,25,50}, C=29[10,16]
CMP_{13} :	c_j ={1,5,10,12,25,50}, C=29[5,10]
CMP_{14} :	c_j ={1,5,12}, C=15[17,18]
CMP_{15} :	c_j ={1,2,3}, C=5[17]
CMP_{16} :	c_j ={1,2,3,4,10}, C=13[17]
CMP_{17} :	c_j ={1,2,5,10,20,50,100,500}, C=93
CMP_{18} :	c_j ={1,4,5}, C=8
CMP_{19} :	c_j ={1,4,6}, C=8
CMP_{20} :	c_j ={1,5,10,25}, C=67
CMP_{21} :	c_j ={1,4,5,6,7}, C=352[1]
CMP_{22} :	c_j ={1,2,3,4,5,6,7}, C=99[19]
CMP_{23} :	c_j ={1,5,10,25}, C=48[20]
CMP_{24} :	c_j ={1,5,10,25}, C=59[15]
CMP_{25} :	c_j ={1,5,12,25}, C=16
CMP_{26} :	c_j ={1,5,10,25}, C=31
CMP_{27} :	c_j ={1,5,20,25}, C=59
CMP_{28} :	c_j ={1,2,3}, C=7
CMP_{29} :	c_j ={1,5,10,25,50,100}, C=283[21]
CMP_{30} :	c_j ={5,10,25,50}, C=80[22]
CMP_{31} :	c_j ={1,5,10,25}, C=100[23]
CMP_{32} :	c_j ={5,9,13}, C=19[7]
CMP_{33} :	c_j ={2,3,5}, C=7[7]
CMP_{34} :	c_j ={1,5,7,20}, C=15[24]
CMP_{35} :	c_j ={1,3,4}, C=6[25]
CMP_{36} :	c_j ={1,2,5,10,20,50}, C=19[26]
CMP_{37} :	c_j ={1,2,6,12,24,48,60,120,480}, C=96[27]
CMP_{38} :	c_j ={1,2,10}, C=15[28]
CMP_{39} :	c_j ={1,6,10}, C=13[28]

CMP_2 ~ CMP_7은 다음과 같이 시내버스 탑승 승객 중에서 교통카드를 지참하지 않아 고액권인 10,000원 또는 5,000원의 현금을 지불하는 경우이다.

시내버스요금 (현금)	성인	중·고등학생	초등학생
	1,400	1,020	700
10,000원	8,600원 CMP_2	8,980원 CMP_4	9,300원 CMP_6
5,000원	3,600원 CMP_3	3,960원 CMP_5	4,300원 CMP_7

CMP_8 ~ CMP_{10}은 직원에게 월급을 지불하지 않다가 노동청에 고발당해 동전으로 지급한 갑질 사례이다. CMP_{11} 부터는 원문의 출처를 문제에 인용하였다. 표 1의 실험 데이터에 대해 DA를 수행한 결과는 그림 4와 같다.

c_j ={10,50,100,500}, C=8,600

8,600	500	100	50	10	z
500	17	1			**18**

(a) CMP_2

c_j ={10,50,100,500}, C=3,600

3,600	500	100	50	10	z
500	7	1			**8**

(b) CMP_3

c_j ={10,50,100,500}, C=8,980

8,980	500	100	50	10	z_n
500	17	4	1	3	**25**

(c) CMP_4

c_j ={10,50,100,500}, C=3,960

3,960	500	100	50	10	z
500	7	4	1	3	**15**

(d) CMP_5

c_j ={10,50,100,500}, C=9,300

9,300	500	100	50	10	z
500	18	3			**21**

(e) CMP_6

c_j ={10,50,100,500}, C=4,300

4,300	500	100	50	10	z
500	8	3			**11**

(f) CMP_7

c_j ={1,5,10,25,50}, C=91,500

91,500	50	25	10	5	1	z
50	1,830					**1,830**

(g) CMP_8

c_j ={10,50,100,500}, C=1,300,000

1,300,000	500	100	50	10	z
500	2,600				**2,600**

(h) CMP_9

c_j ={10,50,100,500}, C=548,770

548,770	500	100	50	10	z
500	1,097	2	1	2	**1,102**

(i) CMP_{10}

c_j ={1,5,10,20}, C=36

36	20	10	5	1	z
20	1	1	1	1	**4**

(j) CMP_{11}

c_j ={1,5,10,25,50}, C=29

29	25	10	5	1	z
25	1			4	**5**

(k) CMP_{12}

c_j ={1,5,10,12,25,50}, C=29

29	25	12	10	5	1	z
25	1				4	5
12	-	2		1		**3**

(l) CMP_{13}

c_j ={1,5,12}, C=15

15	12	5	1	z
12	1		3	4
5	-	3		**3**

(m) CMP_{14}

c_j ={1,2,3}, C=5

5	3	2	1	z
3	1	1		2
2	-	2	1	**3**

(n) CMP_{15}

c_j ={1,2,3,4,10}, C=13

13	10	4	3	2	1	z
10	1		1			2
4	-	3			1	4
3	-	-	4		1	**5**

(o) CMP_{16}

c_j ={1,2,5,10,20,50,100,500}, C=93

93	50	20	10	5	1	z
50	1	2			3	**6**

(p) CMP_{17}

c_j ={1,4,5}, C=8

8	5	4	1	z
5	1		3	4
4	-	2		**2**

(q) CMP_{18}

c_j ={1,4,6}, C=8

8	6	4	1	z
6	1		2	3
4	-	2		**2**

(r) CMP_{19}

c_j ={1,5,10,25}, C=67

67	25	10	5	1	z
25	1	1	1	2	**5**

(s) CMP_{20}

c_j ={1,4,5,6,7}, C=352

352	7	6	5	4	1	z
7	50				2	52
6	-	58		1		59
5	-	-	70		2	72
4	-	-	-	88		**88**

(t) CMP_{21}

c_j ={1,2,3,4,5,6,7}, C=99

99	7	6	5	4	3	2	1	z
7	14						1	15
6	-	16			1			17
5	-	-	19	1				20
4	-	-	-	24	1			**25**

(u) CMP_{22}

c_j ={1,5,10,25}, C=48

48	25	10	5	1	z
25	1	2		3	**6**

(v) CMP_{23}

c_j ={1,5,10,25}, C=59

59	25	10	5	1	z
25	2		1	4	**7**

(w) CMP_{24}

c_j ={1,5,12,25}, C=16

16	12	5	1	z
12	1		4	5
5	-	3	1	**4**

(x) CMP_{25}

c_j ={1,5,10,25}, C=31

31	25	10	5	1	z
25	1		1	1	**3**

(y) CMP_{26}

c_j ={1,5,20,25}, C=59

59	25	20	5	1	z
25	2		1	4	**7**

(z) CMP_{27}

c_j ={1,2,3}, C=7

7	3	2	1	z
3	2		1	3
2	-	3	1	**4**

(aa) CMP_{28}

c_j ={1,5,10,25,50,100}, C=283

283	100	50	25	10	5	1	z
100	2	1	1		1	3	**8**

(ab) CMP_{29}

c_j ={5,10,25,50}, C=80

80	50	25	10	5	z
50	1	1		1	**3**

(ac) CMP_{30}

c_j ={1,5,10,25}, C=100

100	25	10	5	1	z
25	4				**4**

(ad) CMP_{31}

c_j ={5,9,13}, C=19

19	13	9	5	z
13	1		1	∞
9	-	1	2	3
5	-	-	3	**∞**

(ae) CMP_{32}

c_j ={2,3,5}, C=7

7	5	3	2	z
5	1		1	2
3	-	2		∞
2	-	-	3	**∞**

(af) CMP_{33}

c_j ={1,5,7,20}, C=15

15	7	5	1	z
7	2		1	**3**
5	-	3		**3**

(ag) CMP_{34}

c_j ={1,3,4}, C=6

6	4	3	1	z
4	1		2	3
3	-	2		**2**

(ah) CMP_{35}

c_j ={1,3,4}, C=6

6	4	3	1	z
4	1		2	3
3	-	2		**2**

(ai) CMP_{35}

c_j ={1,2,5,10,20,50}, C=19

19	10	5	2	1	z
10	1	1	2		**4**

(aj) CMP_{36}

c_j ={1,2,6,12,24,48,60,120,480}, C=96

96	48	24	12	6	2	1	z
48	2						**2**

(ak) CMP_{37}

c_j ={1,2,10}, C=15

15	10	2	1	z
10	1	2	1	**4**

(al) CMP_{38}

c_j ={1,6,10}, C=13

13	10	6	1	z
10	1		3	4
	-	2	1	**3**

(am) CMP_{39}

그림 4. 실험 데이터에 관한 DA
Fig. 4. DA for experimental data

제안된 DA는 최악의 경우라 할지라도 $O(\frac{n(n+1)}{2})$ 복잡도의 단순한 나눗셈의 몫과 나머지 계산으로 CMP의 최적 해를 찾을 수 있음을 알 수 있어 단순 계산기를 활용하여도 쉽게 해를 구해 실무에 쉽게 적용 가능한 단순함과 정확성의 장점을 모두 갖고 있다고 할 수 있다.

V. 결 론

본 논문은 C를 $c_j, j=1,2,\cdots,n$의 동전으로 교환하는 경우 각 동전의 개수 x_j에 대해 동전 개수를 최소화시키는 $z=\min \sum_{j=1}^{n} x_i$, $\sum_{j=1}^{n} c_i x_i = C$의 거스름돈 만들기 문제에 대해 최악의 경우 $O(\frac{n(n+1)}{2})$ 수행 복잡도의 다항시간 알고리즘을 제안하였다. 제안된 알고리즘은 $c_j \le C$에 대해 j열에 $c_j, j=n,n-1,\cdots,2,1$로 내림차순 배치하고, i행에는 '1'을 포함한 약수를 모두 제외시킨 k개만을 배치한 $k \times n$ 행렬에 대해 $c_i \le c_j$의 셀에 $q_{ij}=quotient(r_{j-1}, c_j)$인 몫만을, 나머지는 $j+1$열의 나눗셈에 활용하는 나눗셈 알고리즘을 제안하였다.

제안된 알고리즘은 매우 간단함에도 불구하고 39개 벤치마킹 실험 데이터에 적용한 결과 최적 해를 빠르고 정확하게 구할 수 있음을 보였다.

References

[1] J. W. Wright, "The Change-Making Problem," Journal of the Association for Computing Machinery, Vol. 22, No. 1, pp. 125-128, Jan. 1975, https:// doi.org/10.1145/321864.321874

[2] D. Pearson, "A Polynomial-time Algorithm for the Change-Making Problem," Operations Research Letters, Vol. 33, No. 3, pp. 231-234, May 2005, https://doi.org/10.1016/j.orl.2004.06.001

[3] P. E. Black, "Greedy Algorithm," Dictionary of Algorithms and Data Structures, U.S. National Institute of Standards and Technology, Feb. 2005.

[4] D. Kozen and S. Zaks, "Optimal Bounds for the Change-Making Problem," Theoretical Computer Science, Vol. 123, No. 2, pp. 377-388, Jan. 1994, https://doi.org/10.1007/3-540-56939-1_69

[5] StackOverflow, "Divide and conquer recursive solution for making change," https://stackoverflow.com/questions/56063638/divide-and-conquer-recursi

ve-solution-for-making-change, Retrieved Feb. 2022.

[6] T. M. Chen and Q. He, "More on Change- Making and Related Problems," Journal of Computer and System Sciences, Vol. 124, pp. 159-169, Mar. 2021, https://doi.org/10.1016/j.jcss.2021.09.005

[7] M. R. Salavatipour, "Tutorial Notes for the Change Making Problem," http://webdocs.cs.ualberta. ca/~mreza /courses/CSC364/tut-notes/tut2, Retrieved Feb. 2022.

[8] J. A. Aslam, "CS7800 Advanced Algorithms: Dynamic Programming Solution to the Coin Changing Problem," Khoury College of Computer and Information Science, Northeastern University, Retrieved Feb. 2022.

[9] Q. Vu, "The Famous Coin Change Problem and its Possible New Applications," Undergraduate Journal of Mathematical Modeling: One+Two, Vol. 11, No. 1, Article 5, pp. 1-9, Jan. 2020, https://doi.org/10.5038/2326-3652.11.1.4924

[10] M. Fienup, "Greedy Coin-Change Algorithm," http://www.cs.uni.edu/~fienup/cs270s04/lectures/lec6_1-29-04_coin_change_web.htm, Retrieved Feb. 2022.

[11] Pweave, "Divide and Conquer Recursive Solution for Making Change," https://stackoverflow.com/questions/56063638/divide-and-conquer-recursive-solution-for-making-change, Retrieved Feb. 2022.

[12] X. Cai, "Canonical Coin Systems for Change- Making Problems," 9th International Conference on Hybrid Intelligent Systems, pp. 499-504, Aug. 2009, https://doi.org/10.1109/HIS.2009.103

[13] T. M. Chen and Q. He, "On the Change- Making Problem," Symposium on Simplicity in Algorithms, pp. 38-42, Feb. 2020, https://doi.org/10. 1137/1.9781611976014.7

[14] Wikipedia, "Greedy Algorithm," https://en.wikipedia.org/wiki/Greedy_algorithm, Retrieved Feb. 2022.

[15] A. Kumar and M. S. Visal, "Coin Change Making Problem Using Greedy Algorithm," https://www.youtube.com/watch?v=ZiQAqMR8jco, Retrieved Feb. 2022.

[16] M. Fienup, "Topics in Computing: Bioinformatics, Lecture 6(1-27-05) Slides, Divide-and-Conquer," http://www.cs.uni.edu/~fienup/cs188s05/lectures/lec6_1-27-05.htm, Retrieved Feb. 2022.

[17] P. Patel, "Change-Making Problem: Dynamic Method," https://medium.com/@pp7954296/change-making-problem-dynamic-method-4954a446a511, Retrieved Feb. 2022.

[18] A. Mehmood, "ASH CC Algo.: Coin Change Algorithm Optimization," International Journal of Computer Applications, Vol. 178, No. 15, pp. 1-9, May 2019.

[19] Terahertz, "The Coin-Change Problem," http://terahertzatheist.ca/2013/12/04/the-coin-change-problem/, Retrieved Feb. 2022.

[20] ProProfs, "What do You Know About the Change Making Problem?," https://www.proprofs.com/quiz-school/story.php?title=3dq-what-do-you-know-about-the-change-making-problem, Retrieved Feb. 2022.

[21] M. C. Patterson and B. Harmel, "The Coin Changing Problem as a Mathematical Model," Journal of Applied Quantitative Methods, Vol. 5, No. 2, pp. 298-301, Sep. 2010.

[22] Z. Chan, "Coin Change Problem(Variation of Change Making Problem)," Oct. 2014, http://www.udaychettiar.com/wp/2014/10/coin-change-problem-variation-of-change-making-problem/

[23] J. Shallit, "What This Country Needs is an 18¢ Piece," Mathematical Intelligencer, Vol. 25, No. 2, pp. 20-23, Feb. 2003.

[24] D. Anguin, "Making Change: Memorization and Dynamic Programming," http://zoo.cs.yale.edu/ classes/topics/topic-making-change, Retrieved Feb. 2022.

[25] Y. Suzuki and R. Miyashino, "Characterization of Canonical Systems with Six Types of Coins for the Change-Making Problem," arXiv:2111.12392, pp. 1-18, Nov. 2021.

[26] A. Adamaszek and M. Adamaszek, "Combinatorics of the change-making problem," European Journal of Combinatorics, Vol. 31, No. 1, pp. 47-63, Jan. 2010, https://doi.org/10.1016/j.ejc.2009.05.002

[27] S. Goebbels, F. Gurski, J. Rethmann, and E. Yilmaz, "Change-Making Problems Revisited: A Parameterized Point of View," Journal of Combinatorial Optimization, Vol. 34, pp. 1218-1236, May 2017, https://doi.org/10.1007/s10878-017-0143-z

[28] A. Kaswan, "Greedy Algorithm to Find Minimum Number of Coins," https://www.baeldung.com/cs/min-number-of-coins-algorithm, Retrieved Feb. 2022.

저 자 소 개

이 상 운(정회원)

- 1987년 : 한국항공대학교 항공전자공학과 (학사)
- 1997년 : 경상대학교 컴퓨터과학과 (석사)
- 2001년 : 경상대학교 컴퓨터과학과 (박사)
- 2003년 : 강원도립대학 컴퓨터응용과 전임강사
- 2004년 ~ 2007년 2월 : 국립 원주대학 여성교양과 조교수
- 2007년 3월 ~ 2015년 3월 : 강릉원주대학교 멀티미디어공학과 부교수
- 2015년 4월 ~ 현재 : 강릉원주대학교 멀티미디어공학과 정교수
- 관심분야 :소프트웨어 프로젝트 관리, 개발 방법론, 분석과 설계 방법론, 시험 및 품질보증, 소프트웨어 신뢰성, 인공지능과 빅데이터분석, 최적화 알고리즘
- e-mail : sulee@gwnu.ac.kr