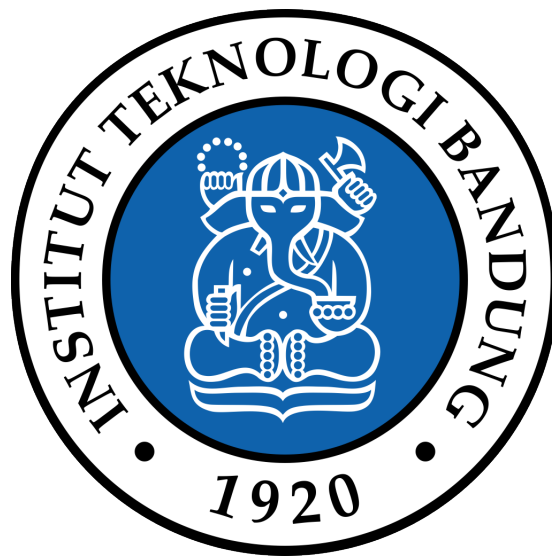


Laporan Tugas Kecil I
Dekripsi *Cryptarithmic* dengan *Brute Force*



Nama : Josep Marcello
NIM : 13519164
Kelas : K-03
Dosen : Prof. Dwi Hendratmo Widyantoro
Bahasa : C++

PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
BANDUNG
2021

1 Algoritma *Brute Force*

1. Proses dekripsi *cryptarithmic* dimulai dengan pembuatan himpunan/matriks berukuran $10! \times 10$ yang berisi kemungkinan permutasi himpunan yang terdiri dari 10 angka (0, 1, 2, ..., 9).
2. Selanjutnya, pilih salah satu soal untuk didekripsi lalu dicari huruf-huruf uniknya.
3. Kemudian, setiap huruf unik akan di-*assign* sebuah nilai yang diambil dari setiap anggota himpunan dari matriks yang dibuat pada langkah 1.
Misalnya jika anggota himpunannya adalah {5, 2, 0, 3, 4, 1, 8, 9, 7, 6} dan huruf uniknya terdiri dari {C, T, A}, maka C = 5, T = 2, A = 0, dan anggota himpunan lainnya “dibuang”¹.
4. Selanjutnya, diperiksa ada atau tidak huruf depan *operand* yang di-*assign* ke angka 0.
 - Jika ada, kembali ke langkah sebelumnya dan pilih anggota himpunan lain, tetapi
 - jika tidak ada, lanjutkan ke langkah selanjutnya.
5. Kemudian, setiap huruf pada setiap operand akan diganti dengan angka yang sudah di-*assign*-kan ke huruf yang bersangkutan kemudian angka itu akan dijumlahkan ke suatu variabel.
6. Hasil diubah ke dalam bentuk angka juga.
7. Lalu, variabel yang menjumlahkan semua *operand* angka pada langkah 5 diperiksa sudah sama dengan hasil yang diubah ke angka atau belum.
 - Jika tidak sama dan masih ada anggota himpunan hasil permutasi lain yang belum diuji, kembali ke langkah 3 dengan memilih anggota himpunan lain yang belum diuji, tetapi
 - jika sama atau sudah tidak ada anggota himpunan hasil permutasi lain yang belum diuji, lanjutkan ke langkah selanjutnya.
8. Pada langkah ini, program sudah selesai bekerja jika semua soal sudah didekripsi, tetapi jika masih ada soal lagi, program akan kembali ke langkah 2.

2 Source Code Program

Listing 1: main.cpp

```
1  /* Nama      : Josep Marcello
2   * NIM       : 13519164
3   * Tanggal  : 20 Januari 2021
4   */
5
6  #include <chrono> // itung waktu eksekusi
7  #include <utility> // pairs
8  #include <vector> // vector
9  #include <stdlib.h> // exit(), free(), malloc()
10 #include <sys/types.h> // exit codes
11 #include <stdio.h> // printf(), puts(), scanf()
12 #include <iostream> // string, cout
13 #include <unordered_map> // unordered_map
14 #include <fstream> // file ops
15
16 #define MAX_UNIQUE_LETTERS 10
17 #define debug1() puts("males belajar tapi...")
18 #define debug2() puts("pengen kaya")
19 #define debug3() puts("udah stres")
20 #define cel() puts("")
21
22 // *** DEKLARASI FUNGSI-FUNGSI ***
23
24 /**
25  * Fungsi untuk membuat semua kemungkinan permutasi dari suatu vektor
26  *
27  * @tparam T tipe data yang disimpan pada vektor
28  * @param vec vektor yang ingin dibuat permutasinya
29  * @returns vektor yg berisi vektor-vektor hasil permutasi
30  */
```

¹ Algoritma ini bisa menyebabkan rekalkulasi (misalnya jika anggota himpunannya adalah {5, 2, 0, 1, 3, 4, 6, 7, 8, 9}, maka C = 5, T = 2, A = 0), tapi setelah pengujian, cara ini bisa menyebabkan perhitungan lebih cepat jika diberikan beberapa soal sekaligus karena hanya perlu membuat satu enumerasi semua kemungkinan permutasi.

```

31 template <typename T>
32 std::vector<std::vector<T>> permute_vec(std::vector<T> vec);
33
34 /**
35  * Fungsi untuk menghasilkan enumerasi permutasi-permutasi yang mungkin dari
36  * angka-angka dalam range [0..lim)
37  *
38  * Mis: lim = 2, maka output:
39  * [[0,1], [1,0]]
40  *
41  * lim = 3:
42  * [[0,1,2], [0,2,1], [1,0,2], [1,2,0], [2,0,1], [2,1,0]]
43  *
44  * @param lim batas atas angka
45  */
46 std::vector<std::vector<int>> generate_permutated_numbers(int lim);
47
48 /**
49  * Fungsi untuk mendekripsi Cryptarithmic
50  *
51  * @param soal soal yang mau didekripsi
52  * @param permutatedNumbers vektor berisi vektor-vektor kumpulan
53  * permutasi-permutasi yang mungkin dari vektor angka [0..MAX_UNIQUE_LETTERS]
54  * @returns sebuah pair berisi solusi benar dan jumlah kasus yang dikerjakan,
55  * jika tidak ada solusi jumlah kasus adalah -1
56  */
57 std::pair<std::vector<int>, int> decrypt_cryparithm(std::vector<std::string> soal,
58 std::vector<std::vector<int>> permutatedNumbers);
59
60 /**
61  * Fungsi untuk mendapatkan huruf-huruf unik dari soal
62  *
63  * @param soal soal yang ingin dicari huruf-huruf uniknya
64  */
65 std::vector<char> unique_letters(std::vector<std::string> soal);
66
67 /**
68  * Fungsi untuk menuliskan isi vector of ints menjadi dalam bentuk penjumlahan
69  *
70  * @param vec vector yang ingin dituliskan outputnya
71  */
72 void vector_formatted_print(std::vector<int> vec);
73
74 /**
75  * Fungsi untuk menuliskan isi vector of strings menjadi dalam bentuk penjumlahan
76  *
77  * @param vec vector yang ingin dituliskan outputnya
78  *
79  * @overload
80  */
81 void vector_formatted_print(std::vector<std::string> vec);
82
83 /**
84  * Fungsi untuk membaca file (sesuai format pada spek) lalu memisahkannya
85  * berdasarkan soal
86  *
87  * @param *fileName string yang berisi nama file soal
88  * @param *output vector dari vector yang menampung soal-soal (tiap elemen
89  * adalah soal)
90  */
91 void parse_file(char* fileName, std::vector<std::vector<std::string>>& output);
92
93 /**
94  * Fungsi untuk menghapuskan whitespaces (' ', '\t', '\n') dari awal C string
95  *
96  * @param *strToStrip pointer ke C string yang ingin di-strip
97  * @returns std::string yang sudah dihapuskan whitespace-nya
98  */
99 std::string strip_at_beginning(char* strToStrip);
100
101 // *** END ***
102
103 int main(int argc, char *argv[])
104 {
105     /// Vektor untuk nyimpen semua soal
106     std::vector<std::vector<std::string>> semuaSoal;

```

```

106 if (argc == 1)
107 {
108     /*
109     fprintf(stderr, "Penggunaan: %s [nama file soal]\n", argv[0]);
110     exit(EX_USAGE);
111     */
112
113     /// string berisi nama file soal
114     char* namaFile;
115     namaFile = (char *) malloc(128 * sizeof(char));
116
117     printf("Masukkan nama file: ");
118     scanf("%s", namaFile);
119     getchar();
120     parse_file(namaFile, &semuaSoal);
121     free(namaFile);
122 }
123 else parse_file(argv[1], &semuaSoal);
124
125 std::chrono::steady_clock sc;
126
127 /// Vektor untuk menyimpan semua jawaban
128 std::vector<std::vector<int>> answers(semuaSoal.size());
129
130 /// awal perhitungan waktu semua soal
131 auto start = sc.now();
132 /// Vektor untuk menyimpan semua kemungkinan permutasi dari [0..9]
133 std::vector<std::vector<int>> permutedNumbers =
134 generate_permuted_numbers(MAX_UNIQUE_LETTERS);
135 /// akhir perhitungan waktu pembuatan permutasi list
136 auto permEnd = sc.now();
137 auto permTS = static_cast<std::chrono::duration<double>>(permEnd-start);
138 printf("Waktu pembuatan semua kemungkinan permutasi adalah: %lf.\n\n", permTS.count());
139 for (std::vector<std::vector<std::string>>::iterator it =
140     semuaSoal.begin(); it != semuaSoal.end(); ++it)
141 {
142     /// awal hitungan waktu
143     auto partialStart = sc.now();
144
145     /// counter iterasi
146     int i = it - semuaSoal.begin();
147     /// jumlah kasus yg diuji
148     int cases;
149
150     // dekripsi dan tuliskan hasil
151     std::pair<std::vector<int>, int> result = decrypt_cryparithm(*it, permutedNumbers);
152     answers[i] = result.first;
153     cases = result.second;
154
155     puts("-----");
156
157     vector_formatted_print(*it);
158     printf("\n\n");
159     if (cases > -1)
160     {
161         vector_formatted_print(answers[i]);
162         printf("\n");
163     }
164     else puts("Tidak ada solusi.");
165
166     /// akhir hitungan waktu
167     auto partialEnd = sc.now();
168     auto partialTimeSpend =
169 static_cast<std::chrono::duration<double>>(partialEnd-partialStart);
170     printf("Soal ke-%d membutuhkan: %lf detik (%lf detik jika dihitung dengan waktu
171 permutasi).\n", i+1, partialTimeSpend.count(), permTS.count() + permTS.count());
172     if (cases > -1)
173         printf("Jumlah kasus yang diuji adalah %d.\n\n", cases);
174     else
175         printf("\n\n");
176 }
177
178 // akhir perhitungan waktu semua soal
179 auto end = sc.now();
180 auto timeSpend = static_cast<std::chrono::duration<double>>(end-start);

```

```

179     printf("Total waktu 1 kali permutasi, eksekusi dekripsi %lu soal, dan menuliskan output
180     adalah %lf detik.\n",
181         semuaSoal.size(), timeSpend.count());
182 }
183
184 template <typename T>
185 std::vector<std::vector<T>> permutate_vec(std::vector<T> vec)
186 {
187     if (vec.size() == 0) return {{}};
188     else if (vec.size() == 1) return {vec};
189     else if (vec.size() == 2) return {vec, {vec[1], vec[0]}};
190
191     /// vektor yang menampung hasil semua permutasi
192     std::vector<std::vector<T>> newVec;
193     /// elemen pertama vektor
194     T first = vec[0];
195     /// tail yang sudah dipermutasi
196     std::vector<std::vector<T>> permuted = permutate_vec(std::vector<T>(vec.begin()+1,
197         vec.end()));
198
199     // tambahkan first ke setiap hasil permutasi tail
200
201     /// elemen dari permuted (tail yang sudah dipermutasi)
202     for (std::vector<T> p: permuted)
203     {
204         for (size_t i = 0; i < p.size() + 1; ++i)
205         {
206             /// vektor yang akan dipush ke newVec
207             std::vector<T> toBePushed(p.begin(), p.begin()+i);
208             toBePushed.push_back(first);
209             toBePushed.insert(toBePushed.end(), p.begin()+i, p.end());
210
211             newVec.push_back(toBePushed);
212         }
213     }
214
215     return newVec;
216 }
217
218 std::vector<std::vector<int>> generate_permutated_numbers(int lim)
219 {
220     /// vektor untuk menyimpan angka-angka pada vektor
221     std::vector<int> numbers(lim);
222     for (int i = 0; i < lim; ++i)
223         numbers[i] = i;
224
225     std::vector<std::vector<int>> hasil = permutate_vec(numbers);
226
227     return hasil;
228 }
229
230 std::pair<std::vector<int>, int> decrypt_cryparithm(std::vector<std::string> soal,
231     std::vector<std::vector<int>> permutedNumbers)
232 {
233     // proses perisapan dan inisialisasi
234
235     /// vektor untuk menyimpan huruf-huruf unik
236     std::vector<char> letters = unique_letters(soal);
237     /// vektor untuk menyimpan huruf pertama dari tiap operand
238     std::vector<char> firstLetters(soal.size());
239     /// unordered_map yang memetakan huruf ke angka
240     std::unordered_map<char, int> numberFromLetter;
241     /// counter jumlah kasus
242     int cases = 0;
243
244     // bikin vektor huruf pertama
245     for (std::vector<std::string>::iterator it = soal.begin();
246         it != soal.end();
247         ++it)
248         firstLetters[it - soal.begin()] = ((*it)[0]);
249
250     // probably not needed, but wut teh hecc
251     if (letters.size() > MAX_UNIQUE_LETTERS)
252     {

```

```

252     std::cerr << "Banyak huruf berbeda (unik) maksimum adalah "
253             << MAX_UNIQUE_LETTERS << '.';
254     exit(EX_DATAERR);
255 }
256
257 /// vektor u/ nampung operands yg udh diubah ke dalam bentuk bilangan
258 std::vector<int> operandInNumbers(soal.size());
259
260 // proses dekripsi
261
262 /// variabel untuk menyimpan sum dari semua operand
263 int sum = 0,
264 /// variabel untuk menyimpan sum 'yang seharusnya'
265 realSum = 0;
266
267 /// numbers vektor yang berisi angka [0..9] yang sudah dipermutasi
268 for (std::vector<int> numbers: permutedNumbers)
269 {
270     sum = 0;
271     realSum = 0;
272     operandInNumbers.clear();
273     // map huruf ke angka
274     for (size_t i = 0; i < letters.size(); ++i)
275         numberFromLetter[letters[i]] = numbers[i];
276
277     // periksa huruf pertama ada yg bernilai 0 atau ngga
278
279     /// Penanda apakah loop perlu dilanjutkan atau tidak
280     bool stopThyLoop = false;
281     for (char c: firstLetters) stopThyLoop = numberFromLetter[c] == 0;
282
283     if (stopThyLoop) continue;
284
285     // ubah operand-operand menjadi angka
286     for (size_t i = 0; i < soal.size(); ++i)
287     {
288         int curNum = 0;
289         for (size_t j = 0; j < soal[i].size(); ++j)
290             curNum = curNum*10 + numberFromLetter[soal[i][j]];
291
292         if (i != soal.size()-1)
293             sum += curNum;
294         else
295             realSum = curNum;
296
297         operandInNumbers.push_back(curNum);
298     }
299
300     if (sum == realSum) break;
301     else cases++;
302 }
303
304 if (sum != realSum) cases = -1;
305
306 return std::make_pair(operandInNumbers, cases);
307 }
308
309 std::vector<char> unique_letters(std::vector<std::string> soal)
310 {
311     /// vector untuk nyimpen huruf-huruf unik
312     std::vector<char> letters;
313
314     /// array untuk nandain huruf apa aja yg udah dipake
315     bool areLettersUsed[] = {
316         false, false, false, false, false, false,
317         false, false, false, false, false, false,
318         false, false, false, false, false, false,
319         false, false, false, false, false, false,
320         false, false, false, false, false, false
321     };
322
323     for (std::string operand: soal)
324     {
325         for (char c: operand)
326         {
327             if (!areLettersUsed[c - 'A'])

```

```

328         {
329             letters.push_back(c);
330             areLettersUsed[c - 'A'] = true;
331         }
332     }
333 }
334
335 return letters;
336 }
337
338 void vector_formatted_print(std::vector<int> vec)
339 {
340     size_t longest = 0;
341     for (int operand: vec)
342         if (longest < std::to_string(operand).size()) longest =
            std::to_string(operand).size();
343
344     for (size_t i = 0; i < vec.size()-2; ++i)
345     {
346         for (size_t j = 0; j < longest - std::to_string(vec[i]).size(); ++j) // ngasih spasi
347             std::cout << " ";
348         std::cout << vec[i] << '\n';
349     }
350     for (size_t j = 0; j < longest - std::to_string(vec[vec.size()-2]).size(); ++j) // ngasih
        spasi
351         std::cout << " ";
352     std::cout << vec[vec.size()-2] << "+\n";
353
354     for (size_t i = 0; i < longest+1; ++i)
355         std::cout << '-';
356     std::cout << '\n';
357
358     for (size_t j = 0; j < longest - std::to_string(vec[vec.size()-1]).size(); ++j) // ngasih
        spasi
359         std::cout << " ";
360     std::cout << vec[vec.size()-1];
361 }
362
363 void vector_formatted_print(std::vector<std::string> vec)
364 {
365     size_t longest = 0;
366     for (std::string operand: vec)
367         if (longest < operand.size()) longest = operand.size();
368
369     for (size_t i = 0; i < vec.size()-2; ++i)
370     {
371         for (size_t j = 0; j < longest - vec[i].size(); ++j) // ngasih spasi
372             std::cout << " ";
373         std::cout << vec[i] << '\n';
374     }
375     for (size_t j = 0; j < longest - vec[vec.size()-2].size(); ++j) // ngasih spasi
376         std::cout << " ";
377     std::cout << vec[vec.size()-2] << "+\n";
378
379     for (size_t i = 0; i < longest+1; ++i)
380         std::cout << '-';
381     std::cout << '\n';
382
383     for (size_t j = 0; j < longest - vec[vec.size()-1].size(); ++j) // ngasih spasi
384         std::cout << " ";
385     std::cout << vec[vec.size()-1];
386 }
387
388 std::string strip_at_beginning(char* strToStrip)
389 {
390     while ((*strToStrip == ' ' || *strToStrip == '\t' || *strToStrip == '\n')
391           && (*strToStrip != '\0')) strToStrip++;
392
393     return strToStrip;
394 }
395
396 void parse_file(char* fileName, std::vector<std::vector<std::string>>& output)
397 {
398     /// variabel untuk menyimpan file
399     std::fstream input;
400     input.open(fileName, std::ios::in);

```

```

401 if (input.is_open())
402 {
403     /// menyimpan baris dari file yang lagi mau diparse
404     std::string line;
405
406     while(getline(input, line))
407     {
408         /// vektor buat nyimpen operand-operand yang dibaca
409         std::vector<std::string> operands;
410         /// buat ngecek masih ngerjain ngeparse soal atau bukan
411         bool isMasihParseSoal = true;
412         /// buat ngecek udah operand terakhir atau belum
413         bool isReadingLastOperand = false;
414
415         do
416         {
417             /// operand yang lagi dibaca, sesudah di-strip di depan
418             std::string operand = strip_at_beginning(&(line[0])).c_str();
419
420             if (isReadingLastOperand)
421             {
422                 isMasihParseSoal = false;
423                 operands.push_back(operand);
424             }
425             else if (operand.empty() || operand[0] == '-')
426             {
427                 isReadingLastOperand = operand[0] == '-';
428                 continue;
429             }
430             else if (*(operand.end()-1) == '+')
431             {
432                 operand.resize(operand.size()-1);
433                 operands.push_back(operand);
434             }
435             else operands.push_back(operand);
436         } while(isMasihParseSoal && getline(input, line));
437
438         output->push_back(operands);
439     }
440
441     input.close();
442 }
443 else
444 {
445     std::cerr << "Gagal membuka file " << fileName << ".\n";
446     exit(EX_NOINPUT);
447 }
448 }
449 }

```


3 Hasil Pengujian

3.1 *Input*

```
> cat test/soal.txt
NUMBER
NUMBER+
-----
PUZZLE

    TILES
PUZZLES+
-----
PICTURE

    CLOCK
    TICK
    TOCK+
-----
PLANET

    COCA
    COLA+
-----
OASIS

    HERE
    SHE+
-----
COMES

DOUBLE
DOUBLE
    TOIL+
-----
TROUBLE

    NO
    GUN
    NO+
-----
HUNT

    THREE
    THREE
    TWO
    TWO
    ONE+
-----
ELEVEN

    CROSS
    ROADS+
-----
DANGER

    MEMO
    FROM+
-----
HOMER

    BRUH
    MOMENT+
-----
BBBBBBB
```

Figure 1: Masukan program (11 soal)

3.2 Output

```
> bin/main test/soal.txt
Waktu pembuatan semua kemungkinan permutasi adalah: 22.608529.

-----
NUMBER
NUMBER+
-----
PUZZLE

201689
201689+
-----
403378
Soal ke-1 membutuhkan: 3.466867 detik (26.075396 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 1213714.

-----
TILES
PUZZLES+
-----
PICTURE

91542
3077542+
-----
3169084
Soal ke-2 membutuhkan: 5.623392 detik (28.231921 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 2487929.

-----
CLOCK
TICK
TOCK+
-----
PLANET

90892
6592
6892+
-----
104376
Soal ke-3 membutuhkan: 6.858476 detik (29.467004 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 3097000.

-----
COCA
COLA+
-----
OASIS

8186
8106+
-----
16292
Soal ke-4 membutuhkan: 3.085873 detik (25.694402 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 1322093.

-----
HERE
SHE+
-----
COMES

9454
894+
-----
10348
Soal ke-5 membutuhkan: 5.667656 detik (28.276185 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 3028549.
```

Figure 2: Luaran program dekripsi untuk bagian permutasi dan soal 1 sampai 5

```

DOUBLE
DOUBLE
  TOIL+
-----
TROUBLE

798064
798064
  1936+
-----
1598064
Soal ke-6 membutuhkan: 5.446043 detik (28.054572 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 2172063.

-----
NO
GUN
  NO+
-----
HUNT

  87
  908
  87+
-----
1082
Soal ke-7 membutuhkan: 2.729912 detik (25.338441 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 1050033.

-----
THREE
THREE
  TWO
  TWO
  ONE+
-----
ELEVEN

84611
84611
  803
  803
  391+
-----
171219
Soal ke-8 membutuhkan: 3.972570 detik (26.581099 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 1340580.

-----
CROSS
ROADS+
-----
DANGER

  96233
  62513+
-----
158746
Soal ke-9 membutuhkan: 6.023432 detik (28.631960 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 2936303.

-----
MEMO
FROM+
-----
HOMER

  8485
  7358+
-----
15843
Soal ke-10 membutuhkan: 2.406747 detik (25.015275 detik jika dihitung dengan waktu permutasi).
Jumlah kasus yang diuji adalah 851279.

-----
BRUH
MOMENT+
-----
BBBBBBB

Tidak ada solusi.
Soal ke-11 membutuhkan: 7.095224 detik (29.703752 detik jika dihitung dengan waktu permutasi).

Total waktu 1 kali permutasi, eksekusi dekripsi 11 soal, dan menuliskan output adalah 74.985223 detik.

```

Figure 3: Luaran program dekripsi untuk soal 6 sampai 11

3.3 Tabel Penilaian

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan (no syntax error)	✓	
2. Program berhasil <i>running</i>	✓	
3. Program dapat membaca file masukan dan menuliskan keluaran	✓	
4. Solusi <i>cryptarithmic</i> hanya benar untuk persoalan <i>cryptarithmic</i> dengan dua buah <i>operand</i>		✓
5. Solusi <i>cryptarithmic</i> benar untuk persoalan <i>cryptarithmic</i> untuk lebih dari dua buah <i>operand</i>	✓	

Link ke repository Github

Link ke repository: https://github.com/jspmarc/Tucil1_13519164.