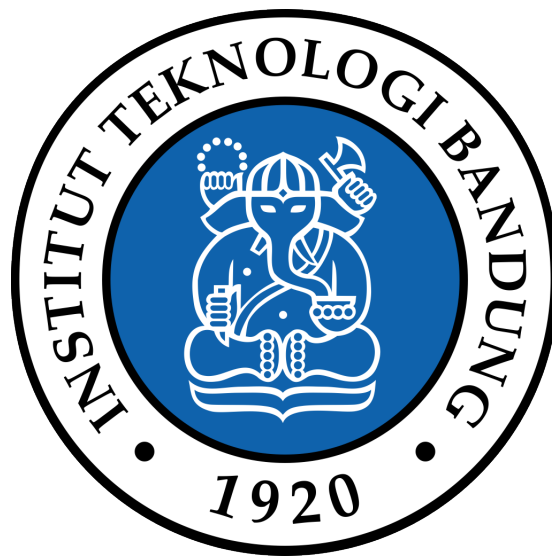


Laporan Tugas Kecil I
Dekripsi *Cryptarithmic* dengan *Brute Force*



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, untuk soal yang ingin didekripsi, 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 sama, lanjutkan ke langkah selanjutnya, tetapi
 - jika tidak sama, kembali ke langkah 2 dengan memilih anggota himpunan yang lain.
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  */
31 template <typename T>
32 std::vector<std::vector<T>> permutate_vec(std::vector<T> vec);
33
```

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.

```

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  */
56 std::pair<std::vector<int>, int> decrypt_cryparithm(std::vector<std::string> soal,
57 std::vector<std::vector<int>> permutatedNumbers);
58
59 /**
60  * Fungsi untuk mendapatkan huruf-huruf unik dari soal
61  *
62  * @param soal soal yang ingin dicari huruf-huruf uniknya
63  */
64 std::vector<char> unique_letters(std::vector<std::string> soal);
65
66 /**
67  * Fungsi untuk menuliskan jawaban sesuai dengan spek
68  *
69  * @param soal vektor yang berisi soal yang ingin diprint, hasil parse parse_file()
70  * @param answer jawaban dari soal yang ingin diprint, hasil decrypt_cryparithm()
71  */
72 void print_answer(std::vector<std::string> soal, std::vector<int> answer);
73
74 /**
75  * Fungsi untuk membaca file (sesuai format pada spek) lalu memisahkannya
76  * berdasarkan soal
77  *
78  * @param *fileName string yang berisi nama file soal
79  * @param *output vector dari vector yang menampung soal-soal (tiap elemen
80  * adalah soal)
81  */
82 void parse_file(char* fileName, std::vector<std::vector<std::string>>& output);
83
84 /**
85  * Fungsi untuk menghapuskan whitespaces (' ', '\t', '\n') dari awal C string
86  *
87  * @param *strToStrip pointer ke C string yang ingin di-strip
88  * @returns std::string yang sudah dihapuskan whitespace-nya
89  */
90 std::string strip_at_beginning(char* strToStrip);
91
92 // *** END ***
93
94 int main(int argc, char *argv[])
95 {
96     /// Vektor untuk nyimpen semua soal
97     std::vector<std::vector<std::string>> semuaSoal;
98
99     if (argc == 1)
100     {
101         /*
102         fprintf(stderr, "Penggunaan: %s [nama file soal]\n", argv[0]);
103         exit(EX_USAGE);
104         */
105
106         /// string berisi nama file soal
107         char* namaFile;
108         namaFile = (char *) malloc(128 * sizeof(char));

```

```

109     printf("Masukkan nama file: ");
110     scanf("%s", namaFile);
111     getchar();
112     parse_file(namaFile, &semuaSoal);
113     free(namaFile);
114 }
115 else parse_file(argv[1], &semuaSoal);
116
117 std::chrono::steady_clock sc;
118
119 /// Vektor untuk menyimpan semua jawaban
120 std::vector<std::vector<int>> answers(semuaSoal.size());
121
122 /// awal perhitungan waktu semua soal
123 auto start = sc.now();
124 /// Vektor untuk menyimpan semua kemungkinan permutasi dari [0..9]
125 std::vector<std::vector<int>> permutedNumbers =
126     generate_permutated_numbers(MAX_UNIQUE_LETTERS);
127 /// akhir perhitungan waktu pembuatan permutasi list
128 auto permEnd = sc.now();
129 auto permTS = static_cast<std::chrono::duration<double>>(permEnd-start);
130 printf("Waktu pembuatan semua kemungkinan permutasi adalah: %lf.\n\n", permTS.count());
131 for (std::vector<std::vector<std::string>>::iterator it =
132     semuaSoal.begin(); it != semuaSoal.end(); ++it)
133 {
134     /// awal hitungan waktu
135     auto partialStart = sc.now();
136
137     /// counter iterasi
138     int i = it - semuaSoal.begin();
139     /// jumlah kasus yg diuji
140     int cases;
141
142     // dekripsi dan tuliskan hasil
143     std::pair<std::vector<int>, int> result = decrypt_cryparithm(*it, permutedNumbers);
144     answers[i] = result.first;
145     cases = result.second;
146
147     print_answer(*it, answers[i]);
148     printf("\n");
149
150     /// akhir hitungan waktu
151     auto partialEnd = sc.now();
152     auto partialTimeSpend =
153     static_cast<std::chrono::duration<double>>(partialEnd-partialStart);
154     printf("Soal ke-%d membutuhkan: %lf detik.\n", i+1, partialTimeSpend.count());
155     printf("Jumlah kasus yang diuji adalah %d.\n\n", cases);
156 }
157
158 // akhir perhitungan waktu semua soal
159 auto end = sc.now();
160 auto timeSpend = static_cast<std::chrono::duration<double>>(end-start);
161
162 printf("Total waktu permutasi, eksekusi dekripsi %lu soal, dan menuliskan output adalah
163 %lf detik.\n",
164     semuaSoal.size(), timeSpend.count());
165 }
166
167 template <typename T>
168 std::vector<std::vector<T>> permute_vec(std::vector<T> vec)
169 {
170     if (vec.size() == 0) return {{}};
171     else if (vec.size() == 1) return {vec};
172     else if (vec.size() == 2) return {vec, {vec[1], vec[0]}};
173
174     /// vektor yang menampung hasil semua permutasi
175     std::vector<std::vector<T>> newVec;
176     /// elemen pertama vektor
177     T first = vec[0];
178     /// tail yang sudah dipermutasi
179     std::vector<std::vector<T>> permuted = permute_vec(std::vector<T>(vec.begin()+1,
180     vec.end()));
181
182     // tambahkan first ke setiap hasil permutasi tail
183
184     /// elemen dari permuted (tail yang sudah dipermutasi)

```

```

181     for (std::vector<T> p: permuted)
182     {
183         for (size_t i = 0; i < p.size() + 1; ++i)
184         {
185             /// vektor yang akan dipush ke newVec
186             std::vector<T> toBePushed(p.begin(), p.begin()+i);
187             toBePushed.push_back(first);
188             toBePushed.insert(toBePushed.end(), p.begin()+i, p.end());
189
190             newVec.push_back(toBePushed);
191         }
192     }
193
194     return newVec;
195 }
196
197 std::vector<std::vector<int>> generate_permuted_numbers(int lim)
198 {
199     /// vektor untuk menyimpan angka-angka pada vektor
200     std::vector<int> numbers(lim);
201     for (int i = 0; i < lim; ++i)
202         numbers[i] = i;
203
204     std::vector<std::vector<int>> hasil = permute_vec(numbers);
205
206     return hasil;
207 }
208
209 std::pair<std::vector<int>, int> decrypt_cryparithm(std::vector<std::string> soal,
210 std::vector<std::vector<int>> permutedNumbers)
211 {
212     // proses perisapan dan inisialisasi
213
214     /// vektor untuk menyimpan huruf-huruf unik
215     std::vector<char> letters = unique_letters(soal);
216     /// vektor untuk menyimpan huruf pertama dari tiap operand
217     std::vector<char> firstLetters(soal.size());
218     /// unordered_map yang memetakan huruf ke angka
219     std::unordered_map<char, int> numberFromLetter;
220     /// counter jumlah kasus
221     int cases = 0;
222
223     // bikin vektor huruf pertama
224     for (std::vector<std::string>::iterator it = soal.begin();
225          it != soal.end();
226          ++it)
227         firstLetters[it - soal.begin()] = ((*it)[0]);
228
229     // probably not needed, but wut teh hecc
230     if (letters.size() > MAX_UNIQUE_LETTERS)
231     {
232         std::cerr << "Banyak huruf berbeda (unik) maksimum adalah "
233         << MAX_UNIQUE_LETTERS << ' ';
234         exit(EX_DATAERR);
235     }
236
237     /// vektor u/ nampung operands yg udh diubah ke dalam bentuk bilangan
238     std::vector<int> operandInNumbers(soal.size());
239
240     // proses dekripsi
241
242     /// numbers vektor yang berisi angka [0..9] yang sudah dipermutasi
243     for (std::vector<int> numbers: permutedNumbers)
244     {
245         operandInNumbers.clear();
246         // map huruf ke angka
247         for (size_t i = 0; i < letters.size(); ++i)
248             numberFromLetter[letters[i]] = numbers[i];
249
250         // periksa huruf pertama ada yg bernilai 0 atau ngga
251
252         /// Penanda apakah loop perlu dilanjutkan atau tidak
253         bool stopThyLoop = false;
254         for (char c: firstLetters) stopThyLoop = numberFromLetter[c] == 0;
255

```

```

256         if (stopThyLoop) continue;
257
258         /// variabel untuk menyimpan sum dari semua operand
259         int sum = 0,
260         /// variabel untuk menyimpan sum 'yang seharusnya'
261         realSum = 0;
262
263         // ubah operand-operand menjadi angka
264         for (size_t i = 0; i < soal.size(); ++i)
265         {
266             int curNum = 0;
267             for (size_t j = 0; j < soal[i].size(); ++j)
268                 curNum = curNum*10 + numberFromLetter[soal[i][j]];
269
270             if (i != soal.size()-1)
271                 sum += curNum;
272             else
273                 realSum = curNum;
274
275             operandInNumbers.push_back(curNum);
276         }
277
278         if (sum == realSum) break;
279         else cases++;
280     }
281
282     return std::make_pair(operandInNumbers, cases);
283 }
284
285 std::vector<char> unique_letters(std::vector<std::string> soal)
286 {
287     /// vector untuk nyimpen huruf-huruf unik
288     std::vector<char> letters;
289
290     /// array untuk nandain huruf apa aja yg udah dipake
291     bool areLettersUsed[] = {
292         false, false, false, false, false, false,
293         false, false, false, false, false, false,
294         false, false, false, false, false, false,
295         false, false, false, false, false, false,
296         false, false, false, false, false, false
297     };
298
299     for (std::string operand: soal)
300     {
301         for (char c: operand)
302         {
303             if (!areLettersUsed[c - 'A'])
304             {
305                 letters.push_back(c);
306                 areLettersUsed[c - 'A'] = true;
307             }
308         }
309     }
310
311     return letters;
312 }
313
314 void print_answer(std::vector<std::string> soal, std::vector<int> answer)
315 {
316     size_t longest = 0;
317     for (std::string operand: soal)
318         if (longest < operand.size()) longest = operand.size();
319
320     for (size_t i = 0; i < soal.size()-2; ++i)
321     {
322         for (size_t j = 0; j < longest - soal[i].size(); ++j) // ngasih spasi
323             std::cout << " ";
324         std::cout << soal[i] << '\n';
325     }
326     for (size_t j = 0; j < longest - soal[soal.size()-2].size(); ++j) // ngasih spasi
327         std::cout << " ";
328     std::cout << soal[soal.size()-2] << " +\n";
329
330     for (size_t i = 0; i < longest+2; ++i)

```

```

332     std::cout << '-';
333     std::cout << '\n';
334
335     for (size_t j = 0; j < longest - soal[soal.size()-1].size(); ++j) // ngasih spasi
336         std::cout << " ";
337     std::cout << soal[soal.size()-1] << '\n';
338
339     std::cout << '\n';
340     std::cout << '\n';
341
342     for (size_t i = 0; i < answer.size()-2; ++i)
343     {
344         for (size_t j = 0; j < longest - std::to_string(answer[i]).size(); ++j) // ngasih
345             spasi
346             std::cout << " ";
347         std::cout << answer[i] << '\n';
348     }
349
350     for (size_t j = 0; j < longest - std::to_string(answer[answer.size()-2]).size(); ++j) //
351         ngasih spasi
352         std::cout << " ";
353     std::cout << answer[answer.size()-2] << " +\n";
354
355     for (size_t i = 0; i < longest+2; ++i)
356         std::cout << '-';
357     std::cout << '\n';
358
359     for (size_t j = 0; j < longest - soal[soal.size()-1].size(); ++j) // ngasih spasi
360         std::cout << " ";
361     std::cout << answer[answer.size()-1] << '\n';
362 }
363
364 std::string strip_at_beginning(char* strToStrip)
365 {
366     while ((*strToStrip == ' ' || *strToStrip == '\t' || *strToStrip == '\n')
367         && (*strToStrip != '\0')) strToStrip++;
368
369     return strToStrip;
370 }
371
372 void parse_file(char* fileName, std::vector<std::vector<std::string>>>* output)
373 {
374     /// variabel untuk menyimpan file
375     std::fstream input;
376     input.open(fileName, std::ios::in);
377
378     if (input.is_open())
379     {
380         /// menyimpan baris dari file yang lagi mau diparse
381         std::string line;
382
383         while(getline(input, line))
384         {
385             /// vektor buat nyimpen operand-operand yang dibaca
386             std::vector<std::string> operands;
387             /// buat ngecek masih ngerjain ngeparse soal atau bukan
388             bool isMasihParseSoal = true;
389             /// buat ngecek udah operand terakhir atau belum
390             bool isReadingLastOperand = false;
391
392             do
393             {
394                 /// operand yang lagi dibaca, sesudah di-strip di depan
395                 std::string operand = strip_at_beginning(&(line[0])).c_str();
396
397                 if (isReadingLastOperand)
398                 {
399                     isMasihParseSoal = false;
400                     operands.push_back(operand);
401                 }
402                 else if (operand.empty() || operand[0] == '-')
403                 {
404                     isReadingLastOperand = operand[0] == '-';
405                     continue;
406                 }
407                 else if (*(operand.end()-1) == '+')

```

```

406         {
407             operand.resize(operand.size()-1);
408             operands.push_back(operand);
409         }
410         else operands.push_back(operand);
411     } while(isMasihParseSoal && getline(input, line));
412
413     output->push_back(operands);
414 }
415
416     input.close();
417 }
418 else
419 {
420     std::cerr << "Gagal membuka file " << fileName << ".\n";
421     exit(EX_NOINPUT);
422 }
423 }

```


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
```

Figure 1: Masukan program (10 soal)

3.2 Output

```
> bin/main test/soal.txt
Waktu pembuatan semua kemungkinan permutasi adalah: 23.702763.

NUMBER
NUMBER+
-----
PUZZLE

201689
201689+
-----
403378

Soal ke-1 membutuhkan: 3.616433 detik.
Jumlah kasus yang diuji adalah 1213714.

TILES
PUZZLES+
-----
PICTURE

91542
3077542+
-----
3169084

Soal ke-2 membutuhkan: 5.698419 detik.
Jumlah kasus yang diuji adalah 2487929.

CLOCK
TICK
TOCK+
-----
PLANET

90892
6592
6892+
-----
104376

Soal ke-3 membutuhkan: 6.995424 detik.
Jumlah kasus yang diuji adalah 3097000.

COCA
COLA+
-----
OASIS

8186
8106+
-----
16292

Soal ke-4 membutuhkan: 3.182077 detik.
Jumlah kasus yang diuji adalah 1322093.

HERE
SHE+
-----
COMES

9454
894+
-----
10348

Soal ke-5 membutuhkan: 5.576527 detik.
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: 5.447500 detik.
Jumlah kasus yang diuji adalah 2172063.

NO
GUN
NO+
-----
HUNT

87
908
87+
-----
1082

Soal ke-7 membutuhkan: 2.773171 detik.
Jumlah kasus yang diuji adalah 1050033.

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

84611
84611
803
803
391+
-----
171219

Soal ke-8 membutuhkan: 4.019753 detik.
Jumlah kasus yang diuji adalah 1340580.

CROSS
ROADS+
-----
DANGER

96233
62513+
-----
158746

Soal ke-9 membutuhkan: 6.032639 detik.
Jumlah kasus yang diuji adalah 2936303.

MEMO
FROM+
-----
HOMER

8485
7358+
-----
15843

Soal ke-10 membutuhkan: 2.453551 detik.
Jumlah kasus yang diuji adalah 851279.

Total waktu permutasi, eksekusi dekripsi 10 soal, dan menuliskan output adalah 69.498728 detik.

```

Figure 3: Luaran program dekripsi untuk soal 6 sampai 10

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 luaran	✓	
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

<https://github.com/jspmarc/Tucil-1-Stima>