***shCRYPTARITHMS* DENGAN MENGGUNAKAN ALGORITMA *BRUTE-FORCE***

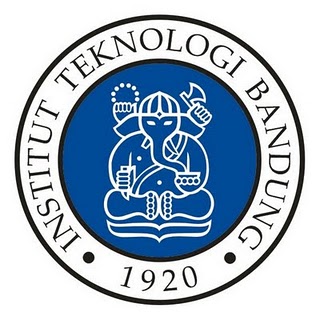
TUGAS KECIL

Diajukan sebagai tugas dari mata kuliah Strategi Algoritma di jurusan Informatika Institut Teknologi Bandung

Oleh :

**Dery Rahman Ahaddienata**

**13515097**

****

**PROGRAM STUDI INFORMATIKA  
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG  
2017**

1. **Deskripsi Persoalan** *Cryptarithmetic* (atau *cryptarithm*) adalah sebuah *puzzle* penjumlahan di dalam matematika dimana angka diganti dengan huruf. Setiap angka dipresentasikan dengan huruf yang berbeda. Deskripsi permainan ini adalah: diberikan sebuah penjumlahan huruf, carilah angka yang merepresntasikan huruf-huruf tersebut. Contohnya

|  |  |  |
| --- | --- | --- |
| Permasalahan | Solusinya | Keterangan |
| SEND  MORE+  ----------  MONEY | 9567  1085+  ----------  10652 | S = 9 | E = 5  N = 6 | D = 7  M = 1 | O = 0  R = 8 | Y = 2 |
| JUNE  JULY+  ----------  APRIL | 7924  7906+  ----------  15830 | J = 7 | L = 0  U = 9 | Y = 6  N = 2 | A = 1  E = 4 | P = 5 R = 8 | I = 3 |
| FORTY  TEN  TEN+  ----------  SIXTY | 29786  850  850+  ----------  31486 | F = 2 | E = 5  O = 9 | N = 0  R = 7 | S = 3  T = 8 | I = 1 Y = 6 | X = 4 |

Maksimum huruf yang unik adalah 10 buah.

1. **Algoritma Brute-Force**

Pemecahan masalah *Cryptarithms* menggunakan algoritma *Brute Force* secara garis besar adalah sebagai berikut

1. Baca semua operand dan hasil berupa *string*, kemudian simpan.
2. Petakan setiap *character* dalam *string* tadi menjadi index [0..9].
3. Buat *array of integer* yang berukuran 10, dan masukan angka dari 0 sampai 9.
4. Lakukan permutasi pada array tersebut.
5. Konversi operand yang berupa string menjadi sebuah angka, kemudian jumlahkan semua operand yang ada. NB : Misalnya pemetaan index ‘A’ adalah 0, f(‘A’)=0, sedangkan angka yang berada dalam array 0 adalah 4, ArrPermut[0]=4. Berarti nilai ‘A’ adalah 4.
6. Konversi hasil yang berupa string menjadi sebuah angka.
7. Lakukan pengecekan, jika jumlah semua operand sesuai dengan hasil, maka tulis jawabannya kedalam output file. Kemudian kembali lakukan dari langkah ke 4, sampai permutasi selesai.

Pemecahan secara sistematis adalah sebagi berikut :

1. Baca semua *operand*, dan masukan ke dalam *array of string*. *Array* dengan *index* 0, merupakan operand pertama, 1 merupakan operan kedua, dan seterusnya. Baca hasil yang berupa *string*, dan masukan kedalam variabel *string*.
2. Masukan setiap huruf semua *operand* dan masukan kedalam fungsi yang dapat memetakan setiap huruf menjadi index. Misal f(‘D’)=0, f(‘R’)=1. Pemetaan ini digunakan untuk mengarahkan sebuah karakter menuju *index* dari *array* permutasi yang bersesuaian. Misalnya dalam khasus SEND + MORE = MONEY, kita perlu memetakan karakter-karakter SENDMORY.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Char | ‘S’ | ‘E’ | ‘N’ | ‘D’ | ‘M’ | ‘O’ | ‘R’ | ‘Y’ |

1. Buat sebuah *array of integer* dengan panjang 10. Kemudian isi angka dari 0 sampai 9. *Array* ini digunakan untuk meng*generate* semua permutasi yang mungkin sebanyak 10!.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Permutasi selanjutnya

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 9 | 8 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

.

.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Val | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. Setiap permutasi angka-angka tersebut lakukan konversi semua operand dan hasil menjadi angka.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Val | 9 | 5 | 6 | 7 | 1 | 0 | 8 | 2 | 3 | 4 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|  | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ | ↑ |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| Char | ‘S’ | ‘E’ | ‘N’ | ‘D’ | ‘M’ | ‘O’ | ‘R’ | ‘Y’ |

1. Kemudian cek, apakah semua operand dijumlahkan dengan menggunakan nilai tersebut, hasilnya sesuai atau tidak. Jika sama, maka tampilkan hasilnya kemudian catat waktunya. Lakukan terus hingga permutasi berakhir.

Tambahan : Algoritma Next Permutation

1. Dari belakang, cari 2 angka yang menurun pertama menggunakan iterator i dan ii. Contoh :

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | i | ii |  |
|  |  |  |  |  |  |  |  | ↓ | ↓ |  |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 8 | 7 | 9 | 6 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. Cari angka dari belakang dengan menggunakan iterator j, dimana angka tersebut lebih besar dari angka yang ditunjukan oleh iterator i.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | i | ii | j |  |
|  |  |  |  |  |  |  |  | ↓ | ↓ | ↓ |  |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 8 | 7 | 9 | | 6 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 |

1. Kemudian *swap* antara angka yang ditunjuk iterator i dan j.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | i | ii | j |  |
|  |  |  |  |  |  |  |  | ↓ | ↓ | ↓ |  |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 7 | | 6 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 |

1. Selanjutnya *reverse* angka yang ditunjuk dari iterator ii sampai index yang paling akhir.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | i | ii | j |  |
|  |  |  |  |  |  |  |  | ↓ | ↓ | ↓ |  |
| Val | 0 | 1 | 2 | 3 | 4 | 5 | 8 | 9 | 6 | | 7 |
| Idx | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | | 9 |

1. Lakukan berulang kali hingga iterator i menunjuk ke index paling awal, dalam hal ini 0.
2. **Source Code**

|  |
| --- |
| Cryptarithms.cpp |
| /\*  Author : Dery Rahman A  NIM : 13515097  Date : 23-01-2017  Topic : Solving Cryptarithms using Brute-Force Algorithm  \*/  #include <iostream>  #include <fstream>  #include <string>  #include <time.h>  using namespace std;  class Map{  private:  char c[10];  int val[10], last;  public:  Map(){  for(int i=0;i<10;i++)  val[i]=-1;  last=0;  }  void setMap(char x){  for(int i=0;i<10;i++){  if(c[i]==x) break;  if(val[i]==-1) {  c[i]=x; val[i]=last; last++; break;  }  }  }  int getN(){  return last;  }  int getVal(char x){  for(int i=0;i<10;i++)  if(c[i]==x) return val[i];  return -1;  }  };  void reverse(int i, int j, int arr[10]){  int arrTemp[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1}, k=0, ii=i;  for(;i<j;++i,++k)  arrTemp[k]=arr[i];  k--;  for(;k>=0;k--)  arr[ii++]=arrTemp[k];  }  bool next\_permutation(int first, int last, int arr[10]){  if (first==last)  return false;  int i=first; ++i;  if (i == last)  return false;  i = last; --i;    while(true){  int ii = i; --i;  if (arr[i] < arr[ii]){  int j = last;  while (!(arr[i] < arr[--j]));  int temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;  reverse(ii, last, arr);  return true;  }  if (i == first){  reverse(first, last, arr);  return false;  }  }  }  bool isSame(int arrPrev[10], int arr[10], int N){  for(int i=0;i<N;i++)  if(arrPrev[i]!=arr[i]) return false;  return true;  }  void assignArr(int arrPrev[10], int arr[10], int N){  for(int i=0;i<N;i++)  arrPrev[i]=arr[i];  }  void updateMap(string opr,Map &f){  for(int i=0;i<opr.length();i++)  f.setMap(opr[i]);  }  int getValOpr(string opr, Map f, int arr[10]){  int sum=0, pengali=1;  for(int i=opr.length()-1;i>=0;i--,pengali\*=10)  sum+=arr[f.getVal(opr[i])]\*pengali;  return sum;  }  bool isMeetReq(string opr[10], int size, Map f, int arr[10], int oprVal[10]){  int sum=0,i=0;  if(arr[f.getVal(opr[size-1][0])]==0) return false;  for(;i<size-1;i++){  if(arr[f.getVal(opr[i][0])]==0) return false;  oprVal[i]=getValOpr(opr[i],f,arr);  sum+=oprVal[i];  }  return sum==getValOpr(opr[i],f,arr);  }  int main(){  int arr[10]={0,1,2,3,4,5,6,7,8,9};  Map f;  fstream file("input.txt", ios\_base::in);  ofstream outfile; outfile.open("output.txt");  string opr[10],x,res,separator;  int size=0;  while(file >> x) {  cout << x << endl;  if(x[0]!='-'){  if(\*(x.end()-1)=='+')  x.erase(x.end()-1);  opr[size++]=x;  } else{  separator=x;  file >> x;  cout << x << endl;  opr[size++]=x;  }  updateMap(x,f);  }  cout << endl;  int oprVal[10],arrPrev[10]={-1,-1,-1,-1,-1,-1,-1,-1,-1,-1};  bool found=false;  float t;  clock\_t tStart = clock();  do{  if(isMeetReq(opr, size, f, arr, oprVal)){  if(isSame(arrPrev,arr,f.getN())) continue;  t=(double)(clock() - tStart)/CLOCKS\_PER\_SEC;  int resVal=0,i=0;  for(;i<size-2;i++){  outfile << oprVal[i] << endl; cout << oprVal[i] << endl;  resVal+=oprVal[i];  }  outfile << oprVal[i] << "+" << endl; cout << oprVal[i] << "+" << endl;  outfile << separator << endl; cout << separator << endl;  resVal+=oprVal[i];  outfile << resVal << endl << endl; cout << resVal << endl << endl;  assignArr(arrPrev,arr,f.getN());  found=true;  }  } while(next\_permutation(0, 10, arr));    if(!found){  outfile << "Tidak ditemukan kombinasi yang mungkin\n\n"; cout << "Tidak ditemukan kombinasi yang mungkin\n\n";  }  outfile << "Waktu yang diperlukan : " << t << endl; cout << "Waktu yang diperlukan : " << t << endl;  outfile.close();  return 0;  } |

1. **Contoh Masukan dan Keluaran**

|  |  |
| --- | --- |
| Input | Output |
| Screenshot from 2017-01-24 23:42:16 | Screenshot from 2017-01-24 23:42:22 |
| Screenshot from 2017-01-24 23:42:44 | Screenshot from 2017-01-24 23:42:53 |
| Screenshot from 2017-01-24 23:43:18 | Screenshot from 2017-01-24 23:43:25 |
| Screenshot from 2017-01-24 23:43:37 | Screenshot from 2017-01-24 23:43:43 |
| Screenshot from 2017-01-24 23:43:56 | Screenshot from 2017-01-24 23:44:02 |
| Screenshot from 2017-01-24 23:44:20 | Screenshot from 2017-01-24 23:44:27 |
| Screenshot from 2017-01-24 23:44:39 | Screenshot from 2017-01-24 23:44:46 |
| Screenshot from 2017-01-24 23:45:04 | Screenshot from 2017-01-24 23:45:11 |
| Screenshot from 2017-01-24 23:45:28 | Screenshot from 2017-01-24 23:45:34 |
| Screenshot from 2017-01-24 23:45:51 | Screenshot from 2017-01-24 23:45:59 |
| Screenshot from 2017-01-24 23:46:16 | Screenshot from 2017-01-24 23:46:23 |
| Screenshot from 2017-01-24 23:46:30 | Screenshot from 2017-01-24 23:46:38 |

1. **Cek List**

|  |  |  |
| --- | --- | --- |
| Poin | Ya | Tidak |
| 1. Program berhasil dikompilasi | V |  |
| 1. Program berhasil *running* | V |  |
| 1. Program dapat membaca file masukan dan menuliskan luaran. | V |  |
| 1. Solusi *cryptarithmetic* hanya benar untuk persoalan *cryptarihtmetic* dengan dua buah *operand*. | V |  |
| 1. Solusi *cryptarithmetic* benar untuk persoalan *cryptarihtmetic* untuk lebih dari dua buah operand. | V |  |