Licenciatura em Engenharia Informática

Algoritmos e Estruturas de Dados

10/01/2022

Trabalho realizado por:

Victor Melo 101099 (50%)

Airton Moreira 100480 (50%)

Introducão

Primeiro trabalho prático de AED , Merkle-Hellman cryptosystem , o objetivo desse trabalho é resolver os problemas de somas de conjunto , dado um determinado conjunto e uma determinada soma , encontrar todos os subconjuntos em que a soma dos elementos desse conjunto seja igual a soma desejada e construir um array de bits em que 1 e 0 fazem corresponder os elementos do conjunto que serão usados para encontrar a soma desejada , 1 significa que o elemento é usado e 0 significa que o elemento não é usado,

Alguns materiais adaptados para implementar as soluções:

1. Print sums of all subsets of a given set - GeeksforGeeks
2. Copying data using the memcpy() function in C - educative.io
3. Dynamic Memory Allocation in C - Geeks for Geeks

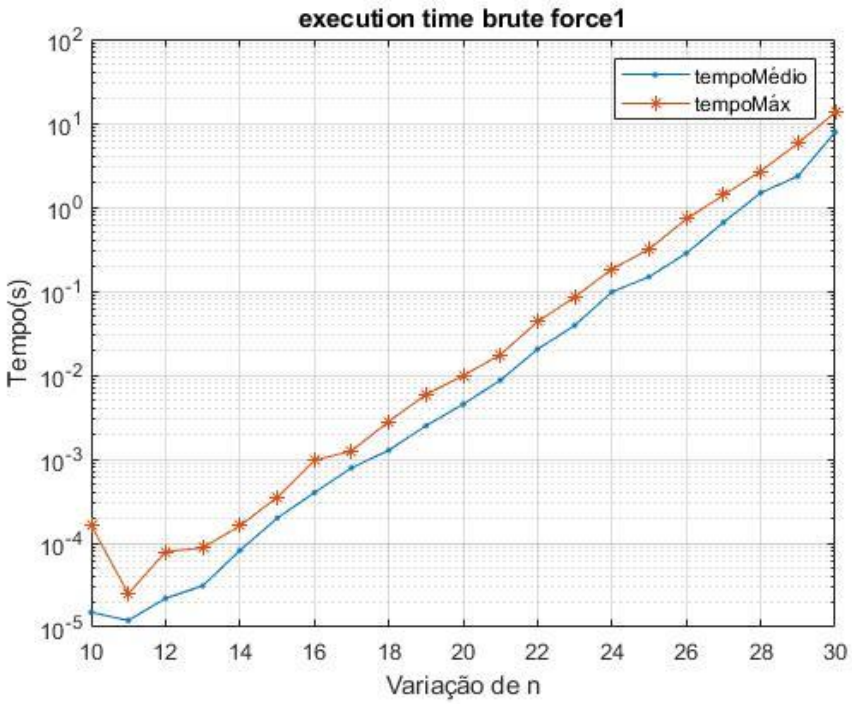
Métodos usados para encontrar as soluções

-Bruteforce

Na implementação por força bruta, o array do tipo integer_t *p corresponde ao set de números, onde o problema do subset_sum consiste em encontrar quais números do set p deverão ser utilizados para encontrar a soma desejada (desired_sum).

Na função brute_force, se a soma desejada for igual a soma parcial, popularemos entao o resto do array com zeros (prevenção de erros) e a função brute_force devolve o valor de 1, pois a soma foi encontrada. Já se o índice atual (current_index) for igual a n (tamanho do set p) a função retorna 0, pois já chegou ao fim. Então aplicamos a recursividade pela primeira vez onde chamamos brute_force com o current_index com o valor do próximo indice (current_index + 1), onde rejeitamos o próximo elemento do array de bits no índice atual (que corresponderá a quais indices do set p serão utilizados para obter a soma desejada) e se a variável sol_found, que corresponde se a solução foi encontrada (1) ou não (0), baseada no retorno de brute_force, for verdadeira (1), a funcão brute_force retorna 1, e o elemento b[current_inndex] será 1, pois a solução naquele índice foi encontrada, e por fim retornamos o resultado de brute_force com o indice atual sendo current_index + 1 e a soma parcial sendo a partial_sum (atual) + p[current_index]. Por fim, retornamos o array de bits.

A sua implementação no main com objetivo de otimizar e ganhar velocidade, alocamos logo o espaço em memoria para o integer_t *p (set de números a serem utilizados) com o malloc correspondente ao seu tamanho, após isso preencheremos p com o set fornecido no ficheiro .h correspondente a cada aluno, ou os ficheiros de teste fornecidos pelo Senhor Professor. Também alocaremos espaço em memória e preencheremos este espaço com zeros para o array de bits integer_t *b, onde será preenchido com zeros sem necessidade de um ciclo for.



-Bruteforce otimizado

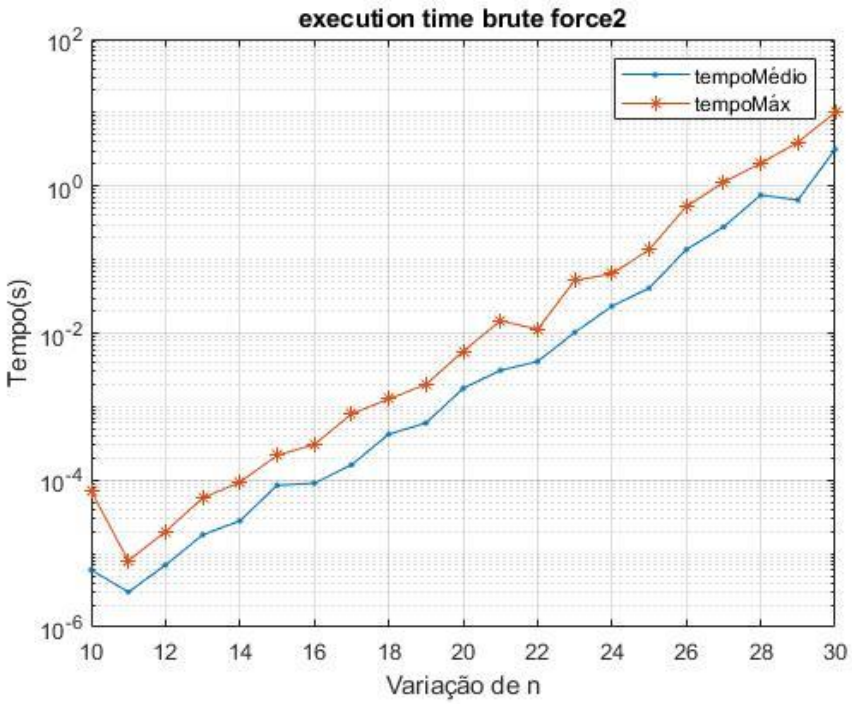
Em Relação ao brute_force1, o brute_force2 (otimizado) consiste em excluir os casos claros onde a soma não terá como ser encontrada, inicialmente excluímos os casos onde a soma parcial é maior que a soma desejada (a funcao retorna 0), após isso quando a soma parcial é maior que a soma desejada, o array de inteiros b será igual a mascara (mask) e a função retorna 1, e a ultima exclusão para otimizar a função, se o índice atual for menor que 0, a execução não tem como trabalhar com índices não existentes no set p, logo a função retorna 0.

Outra diferença é que o integer_t *b neste caso será utilizado como ponteiro para a variável passada, e não é mais um array.

Então se implementará a recursividade, onde se a chamada do brute_force2 com o indice atual = current_index - 1, soma parcial = partial_sum + p[current_index] e a máscara (que inicialmente tem o valor de 0) agora será a operação lógica para extrair o intervalo de bits do current_index, se esta recursão for verdadeira (retorna 1) a função finaliza-se e retorna 1, se não a brute_force2 é chamada recursivamente somente com o índice atual = current_index – 1. Posteriormente a função printBitArray consiste em um loop que irá da int I = 0 até o valor de quantity (inteiro que representa o tamanho do set p utilizado), onde o loop corre o seguinte código: putchar(x & (1u << I)) ? ‘ 1’ ‘0’), que faz:

Se (1 unsigned, apos feita a operação shift left logical I vezes) & (and logico) x (integer_t de input, utilizado como *b em brute_force2) forem verdadeiros, a função putchar escreve 1, se forem falsos, escreve 0, assim utilizando operações bitwise, o código fica mais rápido ao imprimir os bits que correspondem aos índices do set necessários para obter a soma do subset.

A sua implementação no main com objetivo de otimizar e ganhar velocidade, alocamos logo o espaço em memoria para o integer_t *p (set de números a serem utilizados) com o malloc correspondente ao seu tamanho, após isso preencheremos p com o set fornecido no ficheiro .h correspondente a cada aluno, ou os ficheiros de teste fornecidos pelo Senhor Professor.


execution time brute force2
tempoMédio
tempoMáx
Tempo(s)
Variação de n
10^2
10^0
10^-2
10^-4
10^-6
10
12
14
16
18
20
22
24
26
28
30

- Meet-in-the-middle attack

No meet-in-the-middle o array p é dividido em duas partes , set e set2, que contem todas as somas possíveis de cada um, para dividir o p, usamos o `length(p)/2 + 1`, para encontrar todas as somas possíveis de cada array usamos a função **subsetSums,** no total são 2**n subconjuntos. para cada elemento, consideramos duas escolhas, incluir no subconjunto e não incluir no subconjunto, os valores encontrados são armazenados nos arrays `arrayofallsumsSet1` e `arrayofallsumsSet2`, que depois são ordenadas usando a função **sort.**

Tendo os arrays `arrayofallsumsSet1` e `arrayofallsumsSet2` prontos, agora a função `m` é chamada com os arrays necessários e os seus respetivos tamanhos, o objetivo é encontrar `I` e `j` tais que os elementos dos arrays `arrayofallsumsSet1[i]` e `arrayofallsumsSet2[j]` somados são iguais a soma desejada, para encontrar o `I` e o `J` nos sets originais, é feito a chamada da função `brute_force`.

A sua implementação no main com objetivo de otimizar e ganhar velocidade, alocamos logo o espaço em memoria para o integer_t *p (set de números a serem utilizados) com o malloc correspondente ao seu tamanho, após isso preencheremos p com o set fornecido no ficheiro .h correspondente a cada aluno, ou os ficheiros de teste fornecidos pelo Senhor Professor.

Também alocaremos espaço em memória e preencheremos este espaço com zeros para o array de bits integer_t *b, onde será preenchido com zeros sem necessidade de um ciclo for.

Variação de n

Métodos usados para encontrar os gráficos das soluções

Cada solução gera um ficheiro.txt com o output de sua execução e os bits de cada aluno. Para a obtenção dos gráficos e dos ficheiros de output, limitamos a execução das soluções até o set de número 30. Após obtermos os ficheiros de output, desejamos retirar as informações relativas ao tempo de execução médio, o set que está tal execução e também o tempo médio de execução, para isso, ao correrrmos o loop for para cada set desejado, as double somaTempos e tempoMaximo têm valor zero, após isso, no loop para cada soma desejada a double time terá o valor de cpu_time(), e após calcular e fazer o output dos bits a variável time será a diferença de tempo para a realização desses calculos, a função somaTempos tem o valor de time incrementado a ela e se o tempo (time) for maior que o tempoMaximo de execucão, o tempoMaximo será time, agora, de volta ao loop de cada n (set p) o tempo médio será a média da somaTempos / n_sums e após isso, daremos o output do n (i + 10), tempoMedio e tempoMaximo.

Após isso utilizamos um script em bash para obtermos somente os outputs dos dados do tempo de execução, e daí utilizamos um script em matlab para gerarmos os gráficos desejados.

Códigos utilizados:

Brute_force e Brute_force2(otimizado)

```c
//
// AED, November 2021
//
// Solution of the first practical assignement (subset sum problem)
//
// 101099 Victor Melo  100480 Airton
//


#if __STDC_VERSION__ < 199901L
#error "This code must must be compiled in c99 mode or later (-std=c99)" // to handle the unsigned long long data type
#endif
#ifndef STUDENT_H_FILE
#define STUDENT_H_FILE "000000.h" // "101099.h" "100480.h"
#endif


//
// include files
//


#include <stdio.h>
#include <stdlib.h>
```

```c
#include "elapsed_time.h"
#include STUDENT_H_FILE


//
// main program
//


int brute_force(int n, integer_t *p, integer_t desired_sum, int current_index,
integer_t partial_sum, integer_t *b)
{
  if (desired_sum == partial_sum)
  {
    for (int i = current_index; i < n; i++)
    {
      b[i] = 0;
    }
    return 1;
  }


  if (current_index == n)
  {
    return 0;
  }
  // Nao usar o proximo elemento.(rejeitar o elemento na posicao current_index)
  b[current_index] = 0;
  int sol_found = brute_force(n, p, desired_sum, current_index + 1, partial_sum,
b);
  if (sol_found == 1)
```

```
  {
    return 1;
  }
  // Usar o prox elemento
  b[current_index] = 1;
  return brute_force(n, p, desired_sum, current_index + 1, partial_sum +
p[current_index], b);
}

int brute_force2(int n, integer_t *p, int current_index, integer_t partial_sum,
integer_t desired_Sum, integer_t mask, integer_t *b)
{
  if (partial_sum > desired_Sum)
    return 0;

  if (partial_sum == desired_Sum)
  {
    *b = mask;
    return 1;
  }

  if (current_index < 0)
    return 0;

  if (brute_force2(n, p, current_index - 1, partial_sum + p[current_index],
desired_Sum, (mask | (1 << (current_index))), b))
    return 1;
```

```
  else
    return brute_force2(n, p, current_index - 1, partial_sum, desired_Sum, mask,
b);
}


void printBitArray(integer_t x, int quantity)
{
  for (int i = 0; i <= quantity - 1; i++)
  {
    putchar(x & (1u << i) ? '1' : '0');
  }
}


//
// main program
//


int main(void)
{
  int brut = 1; // 1 for brute_force || 2 for brute_force2


  if (brut == 1)
  {
    freopen("brute_force.txt", "w", stdout); // output goes to .txt file
    for (int i = 0; i < n_problems; i++)
```

```
    {


      double somaTempos = 0;
      double tempoMaximo = 0;
      unsigned int n = all_subset_sum_problems[i].n;
      integer_t *p = malloc(n * sizeof(integer_t)); // the weights
      integer_t *b = calloc(n, sizeof(integer_t));  // bit's array -> Use it with
brute_force v1
      p = all_subset_sum_problems[i].p;


      if (n > 30)
      {
        continue;
      }


      for (int j = 0; j < n_sums; j++)
      {
        double time = cpu_time();
        integer_t desired_sum = all_subset_sum_problems[i].sums[j]; // the
desired sum


        brute_force(n, p, desired_sum, 0, 0, b);


        for (int i = 0; i < n; i++)
        {
          printf("%lld", b[i]);
        }
```

```
      printf("\n");


      time = cpu_time() - time;
      somaTempos += time;
      if (time > tempoMaximo)
      {
        tempoMaximo = time;
      }
    }
    double tempoMedio = somaTempos / n_sums;
    printf("time for: %d %f %f\n", i + 10, tempoMedio, tempoMaximo);
  }
}


if (brut == 2)
{
  freopen("brute_force2.txt", "w", stdout); // output goes to .txt file
  for (int i = 0; i < n_problems; i++)
  {


    double somaTempos = 0;
    double tempoMaximo = 0;
    unsigned int n = all_subset_sum_problems[i].n;
    integer_t *p = malloc(n * sizeof(integer_t)); // the weights
    p = all_subset_sum_problems[i].p;
```

```
if (n > 30)
{
  continue;
}
for (int j = 0; j < n_sums; j++)
{
  double time = cpu_time();
  integer_t desired_sum = all_subset_sum_problems[i].sums[j];
  integer_t x = 0;


  if (brute_force2(n, p, n - 1, 0, desired_sum, 0, &x))
  {
    printBitArray(x, n);
    printf("\n");
  }
  else
  {
    printf("error");
    exit(1);
  }


  time = cpu_time() - time;
  somaTempos += time;
  if (time > tempoMaximo)
  {
    tempoMaximo = time;
  }
}
```

```
      double tempoMedio = somaTempos / n_sums;
      printf("%d %f %f\n", i + 10, tempoMedio, tempoMaximo);



    }
  }


  return 0;
}
```

Meet in the Middle

```
//
// AED, November 2021
//
// Solution of the first practical assignement (subset sum problem)
//
// 101099 Victor Melo  100480 Airton
//


#if __STDC_VERSION__ < 199901L
#error "This code must must be compiled in c99 mode or later (-std=c99)" // to handle the unsigned long long data type
#endif
#ifndef STUDENT_H_FILE
#define STUDENT_H_FILE "000000.h"
```

```
#endif


#include <stdio.h>
#include <stdlib.h>
#include "elapsed_time.h"
#include STUDENT_H_FILE
#include <math.h>


#define ARRAYSIZE(a) (sizeof(a)) / (sizeof(a[0]))
#include <string.h>
#include <math.h>


int count;


int brute_force(int n, integer_t *p, integer_t desired_sum, int current_index,
integer_t partial_sum, integer_t *b)
{
    if (desired_sum == partial_sum)
    {
        for (int i = current_index; i < n; i++)
        {
            b[i] = 0;
        }
        return 1;
    }
```

```
    if (current_index == n)
    {
        return 0;
    }
    // Nao usar o proximo elemento.(rejeitar o elemento na posicao current_index)
    b[current_index] = 0;
    int sol_found = brute_force(n, p, desired_sum, current_index + 1,
partial_sum, b);
    if (sol_found == 1)
    {
        return 1;
    }
    // Usar o prox elemento
    b[current_index] = 1;
    return brute_force(n, p, desired_sum, current_index + 1, partial_sum +
p[current_index], b);
}


// Sort the array in ascending order
void sort(integer_t *arr, int length)
{
    integer_t temp = 0;


    for (int i = 0; i < length; i++)
    {
        for (int j = i + 1; j < length; j++)
        {
            if (arr[i] > arr[j])
```

```
            {
                temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
            }
        }
    }
}


/// ALL SUMS OF AN ARRAY
void subsetSums(integer_t *arr, int l, int r, integer_t sum, int allsums,
integer_t *array)
{
    // Print current subset
    if (l > r)
    {
        // printf("%d \n", count);
        array[count] = sum;
        count++;
        return;
    }


    // Subset including arr[l]


    subsetSums(arr, l + 1, r, sum + arr[l], allsums, array);


    // Subset excluding arr[l]
```

```
    subsetSums(arr, l + 1, r, sum, allsums, array);
}


void m(integer_t *subsets1, integer_t *subsets2, int sub1size, int sub2size,
integer_t desiredsum, integer_t total, int setarraysize, integer_t *set,
integer_t *set2, integer_t *b, integer_t *b1, integer_t *b2)
{
    // printf(" \nDesired sum: %lld ", desiredsum);
    if (desiredsum > total)
    {
        printf(" \n No soluction max desired sum");
        exit(1);
    }
    else
    {
        int test = 0;
        int j = sub1size - 1;
        int i = 0;
        while (test == 0)
        {


            if (subsets1[i] + subsets2[j] == desiredsum)
            {
                // printf("\n The soluction {%lld,%lld}", subsets1[i],
subsets2[j]);
                if (subsets1[i] != 0)
                {
                    brute_force(setarraysize, set, subsets1[i], 0, 0, b1);
                }
```

```
                if (subsets2[j] != 0)
                {
                    brute_force(setarraysize, set2, subsets2[j], 0, 0, b2);
                }

                memcpy(b, b1, setarraysize * sizeof(integer_t)); // copy
final_arraysize integer_t's from b1 to b[0]...total[setarraysize]
                memcpy(b + setarraysize, b2, setarraysize * sizeof(integer_t));

                // return b;

                test = 1; // stop the while loop
            }

            if (subsets1[i] + subsets2[j] < desiredsum)
            {
                i++;
            }

            if (subsets1[i] + subsets2[j] > desiredsum)
            {
                j--;
            }

            if (i > sub1size || j < 0)
            {
```

```
                printf("\nNO soluction ");
                test = 1;
            }
        }
    }
}

/*program to test printPowerSet*/
int main()
{

    freopen("meet_in_the_middle.txt", "w", stdout);

    // fprintf(stderr, "Program configuration:\n");
    // fprintf(stderr, "  min_n ....... %d\n", min_n);
    // fprintf(stderr, "  max_n ....... %d\n", max_n);
    // fprintf(stderr, "  n_sums ...... %d\n", n_sums);
    // fprintf(stderr, "  n_problems .. %d\n", n_problems);
    // fprintf(stderr, "  integer_t ... %d bits\n", 8 * (int)sizeof(integer_t));

    // printf("\n\t --- MEET IN THE MIDDLE IMPLEMENTATION --- \t\n");

    // set -> sera o array p dado pelo prof do indice 0 ao length(p)/2
    // set2 -> array p do indice lenght(p)/2 ao length(p)
```

```
    for (int i = 0; i < n_problems; i++)
    {
        int n = all_subset_sum_problems[i].n;
        int setarraysize = (n / 2) + 1;     // +1 to fix bug
        int allsums = pow(2, setarraysize); // todas as somas possiveis = tamanho
do arrayofallsumsSet1
        int final_arraysize = 2 * setarraysize;
        integer_t set[setarraysize];  // the weights from 0 to length(p)/2
        integer_t set2[setarraysize]; // weights from length(p)/2 to length(p)


        for (int k = 0, j = setarraysize; k < setarraysize; k++, j++)
        {
            set[k] = all_subset_sum_problems[i].p[k];
            set2[k] = all_subset_sum_problems[i].p[j];
        }


        // Array com todas as somas possiveis
        integer_t arrayofallsumsSet1[allsums];
        integer_t arrayofallsumsSet2[allsums];


        // find all sums
        subsetSums(set, 0, setarraysize - 1, 0, allsums, arrayofallsumsSet1);
        count = 0; // make count equal to 0 to avoid bug
        subsetSums(set2, 0, setarraysize - 1, 0, allsums, arrayofallsumsSet2);
        count = 0;
        // sort the arrays
        sort(arrayofallsumsSet1, allsums);
```

```
        sort(arrayofallsumsSet2, allsums);


        integer_t maxsum = 0;
        for (int i = 0; i < setarraysize; i++)
        {
            maxsum += set[i];
            maxsum += set2[i];
        }
        double somaTempos = 0;
        double tempoMaximo = 0;


        printf("\n--------------------- Array to work: %d i: %d ------------------
-\n", n, i);


        for (int j = 0; j < n_sums; j++)
        {


            double time = cpu_time();


            integer_t desiredsum = all_subset_sum_problems[i].sums[j]; // the
desired sum


            integer_t *b = malloc(final_arraysize * sizeof(integer_t));
            integer_t *b1 = calloc(setarraysize, sizeof(integer_t));
            integer_t *b2 = calloc(setarraysize, sizeof(integer_t));
```

```
            // m(arrayofallsumsSet1, arrayofallsumsSet2, allsums, allsums,
desired_sum, maxsum, b);
            m(arrayofallsumsSet1, arrayofallsumsSet2, allsums, allsums,
desiredsum, maxsum, setarraysize, set, set2, b, b1, b2);



            int fixbit;
            if (i % 2 == 0)
            {
                fixbit = 2;
            }
            else
            {
                fixbit = 1;
            }


            printf(" bits: ");
            for (int i = 0; i < (setarraysize * 2) - fixbit; i++) // -fixbits to
dont count the last  bits elements that is trash
            {
                // printf("%lld", b[i]);
                printf("%lld", b[i]);
            }


            printf("\n");
            time = cpu_time() - time;
            somaTempos += time;
            if (time > tempoMaximo)
```

```
                tempoMaximo = time;


            free(b);
            free(b1);
            free(b2);
        }
    }
    // fim do main
    return 0;
}
```

Script em bash para obter dados do tempo de execução

```
#!/bin/bash

input=$1".txt" while IFS= read -r line do echo "$line" | grep " " >>
"times_$1.txt" done < "$input"
```

Código Matlab

```
valores = load("NOME DO FICEHIRO.txt");
n1 = valores(1:end,1);       %primeira col
tmed1 = valores(1:end,2);    %segunda col
tmax1 = valores(1:end,3);    %terceira col


figure(5);
semilogy(n1,tmed1,'.-');
```

```
ylabel("Tempo(s)");

xlabel("Variação de n")

title('Comparação das Brute Forces')

hold on;

semilogy(n1,tmax1, '*-')

legend("tempoMédio", "tempoMáx", "tempoMédioOTIM", "tempoMáxOTIM");

grid on;

hold off;
```

1000480.h - Output brute_force

```
1100000001

1110101101

1111001010

1111001101

1011110100

0001111111

1011100000

0110111101

0100101011

0111010110

1001110000

1101101010

1111110010

0111111000

1001010101

1100000011
```

0001010010

1010001101

1101010101

1101010001

10 0.000013 0.000048

01010000101

10000110001

00011000101

11110000101

10000110000

01111111011

01110001110

10100011010

01001111100

00100010001

01001010011

11111000001

00101101000

01011110011

11011001010

11011010111

11001010100

01100001010

10101111111

10110110111

11 0.000015 0.000030

011111010110

000000110010

011100000100

000101101001

100000011101

101001001001

000000011011

101000001101

111010011100

011011101110

100001110100

001010101011

000011000111

011110000100

000101100110

011001110011

100111110101

101000001001

011110001000

111011001010

12 0.000024 0.000060

1101110000101

0001100110001

1011111101111

1011011001001

1100000100110

1100001100010

1101101010101

0011100101001

1010000000110

0110100100000

0010101100011

1101000011011

1111000011000

1010111110111

1111011001110

1101001101000

1111001010000

0111001010011

0110100010101

0010110000010

13 0.000083 0.000136

10101000110110

01001100011001

11100100001110

00111000100111

00101000010011

00110110110010

00000100110111

10000100001010

00100101100101

00011110111001

11101001010111

11010111101111

01101011011100

11111111001011

10001110100001

10101001000001

00101100110101

11001111001101

00110110100110

00101111000111

14 0.000101 0.000253

010111001101000

100100011001110

100110011011010

111111001111111

111000010101111

101010110101010

010000111110010

010101011111000

010010011011010

111101011010001

001110100000001

010000000000100

101010100100000

101110101110100

111100111110110

110010001101101

110010000011010

011011110100001

001110110000011

110111111001111

15 0.000288 0.000533

1011101011001100

1111100011101000

0100111011000101

1111010100001000

0010101101000011

0000011100000010

1001010001000000

1111010011111111

0111010101111101

0010110110010011

1001011110010001

0001100110110011

1000000011011000

1110110000100001

1010000101100111

1100011011110111

0100111001000110

0101110010001111

0101000000010111

0100100010001011

16 0.000408 0.000849

10110000011011001

11100101010010000

10101000100011110

01111110001110011

01111101001001100

11100011010111000

00111011010010101

11101010111100010

00100100110110010

00000011001110001

10111010101011111

11000100001010110

10101001110010110

00110001101000001

00011011001110011

10010111101011011

01100100010001000

11110100000101110

01000100101100110

10011000111110010

17 0.000763 0.001356

0001001100010110011

0101010100101100010

1101110011111010100

1000001001110100000

1000100000000011010

1111000111001000001

1110111100000101011

0000000100000011100

1101101101001000001

0010011001010110010

1011101011110001100

0010111111100011111

1111101000111100011

1010010101001000010

1010001110010100011

0110111000101010000

0100011100101010110

0001111010110101111

1000100100010011111

0010100001010110011

18 0.001429 0.002782

10111000001011000010

1100010000001100101

1010110000001101111

1011011010110101100

0111001111111110101

1010010110111011110

1101001001001100110

0011101100000100101

1001101000111100010

0100110110011011010

0000001100010101101

0100110010100101000

0110001110011101101

0001000110010100110

1100100110010010110

1010100101110111110

0101011001110010111

1100011100000111010

1000110100100011100

0111100000100111111

19 0.003644 0.006769

01111000110100111100

01011001110010101101

00100000111101000111

10001010011001110010

01100101000010011000

00000110011110111100

11010001011000001001

10110100001101001001

11010100111001111111

11001011000101111111

10100101001011011111

01000101100000011110

11001110000110111000

10100001010111011011

11010111111011011100

11001010101000000001

11010000100001011011

11100010101110100001

10100100011110110000

01111011011111001001

20 0.006922 0.010033

11011100101101000101

00111001110101100100

00110101111010101100

10000001100111010111

11100101000111011000

11100001011101000101 0

10110111110110001000 0

01011110001011000100 1

00000101011000100111 1

01100000001111100000 0

11111110101001011011 0

01100100000101011100 1

00100110101110000000 0

11101110100111100010 0

10110110110001011111 1

011000111001001

1000480.h - Output brute_force2

1100000001

1110101101

1111001010

1111001101

1011110100

0001111111

1011100000

0110111101

0100101011

0111010110

1001110000

1101101010

1111110010

0111111000

1001010101

1100000011

0001010010

1010001101

1101010101

1101010001

10 0.000004 0.000012

01010000101

10000110001

00011000101

11110000101

10000110000

01111111011

01110001110

10100011010

01001111100

00100010001

01001010011

11111000001

00101101000

01011110011

11011001010

11011010111

11001010100

01100001010

10101111111

10110110111

11 0.000005 0.000021

011111010110

000000110010

011100000100

000101101001

100000011101

101001001001

000000011011

101000001101

111010011100

011011101110

100001110100

001010101011

000011000111

011110000100

000101100110

011001110011

100111110101

101000001001

011110001000

111011001010

12 0.000008 0.000025

1101110000101

0001100110001

1011111101111

1011011001001

1100000100110

1100001100010

1101101010101

0011100101001

1010000000110

0110100100000

0010101100011

1101000011011

1111000011000

1010111110111

1111011001110

1101001101000

1111001010000

0111001010011

0110100010101

0010110000010

13 0.000007 0.000029

10101000110110

01001100011001

11100100001110

00111000100111

00101000010011

00110110110010

00000100110111

10000100001010

00100101100101

00011110111001

11101001010111

11010111101111

01101011011100

11111111001011

10001110100001

10101001000001

00101100110101

11001111001101

00110110100110

00101111000111

14 0.000027 0.000113

010111001101000

100100011001110

100110011011010

111111001111111

111000010101111

101010110101010

010000111110010

010101011111000

0100100011011010

1111010110100001

0011101000000001

0100000000000100

1010101001000000

1011101011110100

1111001111110110

1100100011101101

1100100000011010

0110111101000001

0011101100000011

1101111111001111

15 0.000078 0.000217

1011101011001100

1111100011101000

0100111011000101

1111010100001000

0010101101000011

0000011100000010

1001010001000000

1111010011111111

0111010101111101

0010110110010011

1001011110010001

0001100110110011

1000000011011000

1110110000100001

1010000101100111

1100011011110111

0100111001000110

0101110010001111

0101000000010111

0100100010001011

16 0.000066 0.000278

10110000011011001

11100101010010000

10101000100011110

01111110001110011

01111101001001100

11100011010111000

00111011010010101

11101010111100010

00100100110110010

00000011001110001

10111010101011111

11000100001010110

10101001110010110

00110001101000001

00011011001110011

10010111101011011

01100100010001000

11110100000101110

01000100101100110

10011000111110010

17 0.000234 0.000576

000100110001011011

010101010010110010

110111001111010100

100000100111010000

100010000000011010

111100011100100001

111011110000101011

000000010000011100

110110110100100001

001001100101011010

101110101111000110

001011111100011111

111110100011110011

101001010100100010

101000111001010011

011011100010101000

010001110010101110

000111101011010111

100010010001001111

001010000101011011

18 0.000299 0.001146

1011100000101100010

1100010000001100101

1010110000001101111

1011011010110101100

0111001111111110101

1010010110111011110

1101001001001100110

0011101100000100101

1001101000111100010

0100110110011011010

0000001100010101101

0100110010100101000

0110001110011101101

0001000110010100110

1100100110010010110

1010100101110111110

0101011001110010111

1100011100000111010

1000110100100011100

0111100000100111111

19 0.001039 0.003059

01111000110100111100

01011001110010101101

00100000111101000111

10001010011001110010

01100101000010011000

00000110011110111100

11010001011000001001

10110100001101001001

11010100111001111111

11001011000101111111

10100101001011011111

01000101100000011110

11001110000110111000

10100001010111011011

11010111111011011100

11001010101000000001

11010000100001011011

11100010101110100001

10100100011110110000

01111011011111001001

20 0.002400 0.010678

11011100101101000101

00111001110101100100

00110101111010101100

10000001100111010111

11100101000111011000

11100001011101000101

10110111111011000100

01011110001011000100

00000101011000100111

01100000001111100000

11111110101001011010

01100100000101011001

00100110101110000000

11101110100111100100

10110110110001011111

01100011100100101011

01001001110000011110

10101101001000010110

11111100101101001010

10001001000111100100

21 0.005949 0.021119

00001111010010101101

00010110010001001110

00010001010110000100

01

10101010011111111000

01001100100111110101

00100101010111000111

11011101000011100000110

10000011111001000011011

11011011010100010100100

11010110011111010011010

00101011111101001101010

10000110000110101011010

10001000110100111001010

01011111111101000111110

10011101000111001101100

00110000101110010100001

10010101010001011101000

11011111001010000000011

01101110000101010010000

10110101100111011011100

22 0.011933 0.030249

00101110110111111001011000

11111011101110101101010

01010000011101011111100

01000111010000100100100

00000001100011101010011

10111010001101010101010

01001101101101011101100

11100001110011100010100

10011000000011111001010

01000001000100010111101

01100100010100110100011

11010010111111110101000

11111101111001100000100

10000011010001111111010

010000111100111110101101

001010000111000010010001

010110001010110010100000

110111101011101101101101

001001010011111100010101

100011101100110011010011

23 0.033475 0.095809

110111110000001100101101

011111001000011001111011

100111010101001101110101

001011000111110000110100

010111100010110100110000

010111100001101111100010

110011010010010001001100

000000110000111001000111

010011011110010111100001

010000101110010101100101

011101011010111110110000

111011110001100110110100

010100111011100011011000

110000111111101111000100

010011011011000000000001

000100010110110111100001

110101001001100111111001

110000110101011000100000

001101110110000101100101

000011001010101111010110

24 0.061273 0.153618

101100110111110010001101 0

101101010000011101110001

111010111100101010000111 1

100001111011100000110100 0

001100111100010101010010 0

010101110011001010110110 1

001000000010001110010110 1

101100100101111000111001 0

111101111101011101100011 1

110100101101010011100010 0

001101101011100100001111 0

100100110000111010000111 0

100100110110100011000001 0

011001100111110011011100 1

011100100100000100110011 1

001000110110010101110101 0

000011101001011010010111 1

001001110110101110010011 1

010000001011010001110111 0

000011000111111101010111 0

25 0.090497 0.247674

100001101000100110110111 01

100101110010101110110110 11

011001111100101010111110 00

110111010101000100110101 00

100100100101000011011110 11

101000011101000101111001 11

100101111111001000101100 11

101101111001011000111010 00

101110110100110000010100 00

01100110010110100011011110

01110100011000010101011011

01110001100010111111011001

00111100000001100010011101

10100100100110111101000110

10101010100110001100111101

10001100001000110000011111

10001101010110011101010110

10001000100111101100110100

10000000000110011111000101

11011000010010110100001111

26 0.171342 0.579786

10110101001100000111111101

00001101000001111100000 1110

00011011100000010100 1000101

10011011100010001100011 0000

11011010100001111011010 0101

10001100011100101000011 0100

11111010111000000110110 1110

00111101101010000110000 0000

01111100010100110100111 1111

11110111111100010000010 1101

00011010110110100000001 1000

10110100101100100101011 0111

00010000100110001110111 1101

11110000101011001101111 1001

01010101010000101010011 1101

00001000100110010011010 1101

01101100110010000011000 0010

101111011101101101001101001

110111111000100101000100111

111101101011111010001000011

27 0.202319 0.528323

011101010110010111000011000 0

010100001010001101101011011 1

100010111010001000000000011 1

010101101001111101011000000 0

011111101110010000011101110 1

001011110000111010101011111

100100110010000000010101011

101111101100100010000001110 1

001001001011110001010111111 0

010100011101111001010011001 0

100011010101010000000011101 0

000101010101110111000100001 0

011011001101010000001110101 1

001010010111000011101001010 1

101011011100011010010100010 0

000110111011101111000111101 0

111011110111010110011000000 1

000101111100111000101100110 1

011010111001001100001101000 0

001110001101011100010010010 1

28 0.436026 1.982954

100100111000100100110101011 11

011010101000000100111010000 11

000011101000110001100010010 11

100100000101001110011100100 11

10101111000011100010111011111

10010010100101111101000110111

00110100001011011011110110001

01010001100010111101000000000

10100000011001100100001000010

01001100100000010011111011111

00111100011010010001100000110

00010101100000111010000011111

10100101000001010011111111011

01001000001010000001111011110

10111101101001111110111001111

11101111100101011101010000100

01101100010100101001010001011

00111101011000100000111011001

00011010010101011111011111100

10100001000010000101000101010

29 0.705967 5.343805

10001000101110100100100011010 0

01010101000101100100001110001 0

11000100000110010100110110011 0

11011101001011111110111011010 1

01110111000111001000101101010 1

00000101100111001101011011000 0

11011001010001111010001010111 1

11011100111010101010110001 11

00001100000101010100011101110 1

10101010011001011000000001100 0

00100001000010111101011011111 1

01001000001111011100000101101 1

00011000101101101100000111010 0

01101110001100100011111001111 0

11110100111111101011011010101 1

01010001101111101101100000101 0

00001001001110101000001010001 0

00101001000010001110010010111 0

00001000111101001110111001111 1

01000111010101011100001001010 1

30 1.876817 5.972353

1000480.h - Output meet_in_the_middle

1100000001

1110101101

1111001010

1111001101

1011110100

0001111111

1011100000

0110111101

0100101011

0111010110

1001110000

1101101010

1111110010

0111111000

1001010101

1100000011

0001010010

1010001101

1101010101

1101010001

10 0.000006 0.000030

01010000101

10000110001

00011000101

11110000101

10000110000

01111111011

01110001110

10100011010

01001111100

00100010001

01001010011

11111000001

00101101000

01011110011

11011001010

11011010111

11001010100

01100001010

10101111111

10110110111

11 0.000004 0.000004

011111010110

000000110010

011100000100

000101101001

100000011101

101001001001

000000011011

101000001101

111010011100

011011101110

100001110100

001010101011

000011000111

011110000100

000101100110

011001110011

100111110101

101000001001

011110001000

111011001010

12 0.000005 0.000007

1101110000101

0001100110001

1011111101111

1011011001001

1100000100110

1100001100010

1101101010101

0011100101001

1010000000110

0110100100000

0010101100011

1101000011011

1111000011000

1010111110111

1111011001110

1101001101000

1111001010000

0111001010011

0110100010101

0010110000010

13 0.000006 0.000007

10101000110110

01001100011001

11100100001110

00111000100111

00101000010011

00110110110010

00000100110111

10000100001010

00100101100101

00011110111001

11101001010111

11010111101111

01101011011100

11111111001011

10001110100001

10101001000001

00101100110101

11001111001101

00110110100110

00101111000111

14 0.000008 0.000011

010111001101000

100100011001110

100110011011010

111111001111111

111000010101111

101010110101010

010000111110010

010101011111000

010010011011010

111101011010001

001110100000001

010000000000100

101010100100000

101110101110100

111100111110110

110010001101101

110010000011010

011011110100001

001110110000011

110111111001111

15 0.000010 0.000013

1011101011001100

1111100011101000

0100111011000101

1111010100001000

0010101101000011

0000011100000010

1001010001000000

1111010011111111

0111010101111101

0010110110010011

1001011110010001

0001100110110011

1000000011011000

1110110000100001

1010000101100111

1100011011110111

0100111001000110

0101110010001111

0101000000010111

0100100010001011

16 0.000017 0.000042

10110000011011001

11100101010010000

10101000100011110

01111110001110011

01111101001001100

11100011010111000

00111011010010101

11101010111100010

00100100110110010

00000011001110001

10111010101011111

11000100001010110

10101001110010110

00110001101000001

00011011001110011

10010111101011011

01100100010001000

11110100000101110

01000100101100110

10011000111110010

17 0.000016 0.000023

0001001100010110011

0101010100101100010

1101110011111010100

1000001001110100000

1000100000000011010

1111000111001000001

1110111100000101011

0000000100000011100

1101101101001000001

0010011001010110010

1011101011110001100

0010111111100011111

1111101000111100011

1010010101001000010

1010001110010100011

0110111000101010000

0100011100101011110

0001111010110101111

1000100100010011111

0010100001010110011

18 0.000028 0.000045

10111000001011000010

11000100000011001010

1010110000001101111

1011011010110101100

0111001111111110101

1010010110111011110

1101001001001100110

0011101100000100101

1001101000111100010

0100110110011011010

0000001100010101101

0100110010100101000

0110001110011101101

0001000110010100110

1100100110010010110

1010100101110111110

0101011001110010111

1100011100000111010

1000110100100011100

0111100000100111111

19 0.000028 0.000043

01111000110100111100

01011001110010101101

00100000111101000111

10001010011001110010

01100101000010011000

00000110011110111100

11010000101100000 1001

10110100001101001001

11010100111001111111

11001011000101111111

10100101001011011111

01000101100000011110

11001110000110111000

10100001010111011011

11010111111011011100

11001010101000000001

11010000100001011011

11100010101110100001

10100100011110110000

01111011011111001001

20 0.000037 0.000053

11011100101101000101

00111001110101100100

00110101111010101100

10000001100111010111

11100101000111011000

11100001011101000101

10110111110110001000

01011110001011000 1001

00000101011000 1001111

01100000001111100000

11111110101001011 0110

01100100000101011 1001

00100110101110000000

11101110100111100 0100

10110110110001011 1111

011000111001001

101099.h - Output brute_force

1011110000

1110000101

0010100111

1111100111

1011010101

1101011101

1111110011

1101100010

0001001100

1100000101

0110010111

0000111000

0001101010

1001010111

1000101110

0111110001

1001101111

0001001001

0111000111

1011111111

time for: 10 0.000018 0.000227

01011100100

11101110111

10010111101

01110010010

10111100101

10100010110

00000001100

11000101000

01110000110

01101011010

10111001101

01110001100

01001010111

00011000111

00100010000

00101011001

00101111110

01110111100

11000110010

10100000010

time for: 11 0.000011 0.000035

111001110111

110110010111

011001010111

001110111110

000100001101

011110011100

101000101101

101011010001

011000011010

010101101110

011010000111

011100010100

001011010001

000011111000

111110001110

001001110111

001110110010

010101100101

001001100110

111010101010

time for: 12 0.000020 0.000041

0010000001111

1000010000010

0000010100001

0101101100010

0111000001100

1100111110010

0000100000000

0110001101111

1001111011100

0111100011001

1010101000000

0100100100011

0111000011110

0110001111011

0101011001010

1110011111100

0101001000100

0111000000100

0011110001000

0001001100100

time for: 13 0.000034 0.000084

00101011000010

01100000000000

11011101001100

01011000001100

01011001110101

00111100101010

01001100001001

10010011011110

01001111100101

00111111010010

00001000101010

10001011101010

10110010011100

00011011011011

01110000001101

00100100100100

10100000100100

11111010111000

11111001110000

01110011001111

time for: 14 0.000075 0.000159

101000110000011

010001001000100

111011111010001

001010010110100

101011110010111

001111001110100

110101101010110

101011001110000

010100000110011

010000011100101

010100111111011

101011010000001

000011110011001

000111101001100

000010011001101

100010101010001

011001001110001

001011100100000

110110000000001

001101010101101

time for: 15 0.000138 0.000314

1110111100100111

1111001101100110

0110011100100110

1101001000000000

0111101101011100

1010001010101101

1110011010000110

1111100101100001

1101000110000101

1001001011000001

0100001110110001

1100100100001111

1000010000000111

0110001010111010

0100000001000010

0101000011001010

1100111110101100

0011011001110101

1100101010100100

0001101010010000

time for: 16 0.000413 0.000850

10101111000110111

11001101001010011

00001111011010111

00010101110110100

01100011110110001

00001100001011100

00010111101101100

01000001010110000

11110100110110000

01100010000111110

11111101001111010

10010100110100110

01100001011011101

11110010100111101

11101101110101110

11011011110000110

01110111110001100

10000111000101111

00000001100010110

00111101100010101

time for: 17 0.000645 0.001294

100100110010110110

000001100100100100

000100001100001101

001001111001101011

000010100101000111

111011110000011011

011011101011101001

011001111010011000

100111100001001001

110100010110011101

011011111101110001

111011010000001110

011110110110101101

111100010101111011

011011001000011101

010001000111100110

000010010011100011

101010011010000100

011011111101110100

111110100100101011

time for: 18 0.001305 0.002716

0110010101101010110

1101001100011100001

1111100011111010110

0111110101100100000

0110111100110111110

1000010001101101111

1010100101011011001

0010011011111000011

1001111110011001110

0010111100111110100

0100100110110111000

1000110000000101011

1001000101010010100

0000110010010010111

1101111110010110100

0010000001000101101

0001001000000111010

0000101100001110011

1111100111000110100

1110001111100000110

time for: 19 0.002589 0.005144

10001110100001100110

00100000000110010111

10001001100101000010

10000100011011011100

00000011000111010111

11101101000000011110

01000000000100011100

11001101000110000110

00101101111111011110

00100100000100001110

00010100101101100001

00011011001110111001

00111001111010010001

01110011110110010000

01101011010011011110

00101001010111011111

11010111110101001111

11101001101100100001

01110011011110110111

00100000001110111001

time for: 20 0.004945 0.020835

10010010010010110011

11010101010011001100

00010010011010010101

10101011101110110110

00001000110101010111

11111010110100110100 0

00000010101101001110 0

10110101011111101111 1

10010111100111001001 1

11010111000111100111 1

11011111110110100011

10100101000111000010 1

00010001110011100000 1

00101001000100000111 0

10100111010100000001 1

00001101100110111000 1

11101001111010001011 0

00010000011000010001 0

00110111010010110001 1

11000100111011001101 0

time for: 21 0.010864 0.020813

10011101110100111010 10

01100110010110101001 01

10100001100100001101 11

01011010110101001111 01

00001100000110010101 01

11011010111010110010 00

00101011111011111001 10

00101000111110110000010

11110100011100111110100

10011110000111001111110

10010010111110010000101

10001111001100000001010

01100101110110000011100

01111000000101000000000

11001110011110001100011

10000110000000011111100

10000010100110000010101

00000000011101100000010

01100111110111000000001

01110111100111100011100

time for: 22 0.021285 0.044012

10100001010001110101101

01011011010010100100010

11101111100101110010000

00011101111101011000110

10000001011110100110010

00000111000000100010010

01101010100100000100101

01101111001101001101110

01010011000101010110111

10001001101100011101100

10010111011001111110111

11000000011011100001100

11001011111001001101000

00011010000111110101001

11110001101000100000010

011011001010011011110110

000101101010100110010 11

001010110110011001001 01

010001011011000110000 10

100010010010000110011 01

time for: 23 0.041304 0.095063

110001100101111100100101

011100011111101101111110

100000000111101110010111

011000001100111010001101

000001011000110101011010

000001001111111001010111

010100101000101011110110

110100000101000000111101

110010111000111111001011

010101111011101010110010

101000001110000011010100

110101110011001010101110

010100101000100111110011

111101000111100101111001

101110110101101110000101

011001001010110111000111

011111101101001011010011

111111011000101100110111

001100011110100011111001

001110011111011100111101

time for: 24 0.095769 0.202937

111101110101010110010 0001

110110010100111010111100000

000101111001101001111001 0

000111000000111000110000 1

010101011111011011001000 0

110101101110001101111011 0

001111000010100110100010 0

010011111001000001100100 0

101001100000010101000111 0

000010011000000101010111 0

110011111110011100111101 0

101001011100010011011111 0

000101011100100011011000 1

010011011000101001010010 0

101001110101000101110010 1

101010001110100001011110 1

010111111100001111101000 1

001011111001111100111101 1

000101011101110010010100 0

100000111011010100100101 1

time for: 25 0.175305 0.378433

010110110000111001100110 11

010000101011011101110111 00

001110110011100010111010 11

110110001100100001110001 10

011111111100101110000010 01

100011101001010101011000 00

010011001101111110000010 10

010000010110101101101101 01

010110010110000010010001 01

000011101110111101100000 10

11101110010101001111011000

00110001011001101011101101

11000101100011111100110001

01100110100111110101000011

10001000001100001011100010

00011101101111010101010110

01010111011111000110100011

01000000100101111011101010

10011110000111001110101100

10011100011100101011001111

time for: 26 0.380589 0.811496

00110011001001111001100000011

01110011011000111000100011O

01000010100011101111111111011

01001100010001001100110011

00000110110011000101101110l

00101100100000111100110100O

11001001110011000111110110O

10011011110110010100011101l

10000101111001001101101101O

01100011111001101110111101O

01110000111001100111000001O

11101010001100111001101000l

11111001111101000011011001O

11101010010011100110110111l

00101101011111101101000111

00101000010111011101110001O

00000111110101000111100110l

10001001000011000000101111l

000001110101111111000100001

001110111110011001101111110

time for: 27 0.629304 1.581326

111100100110101100001010011O

011001100100010101110010010O

001011100000001000000001100O

011111011010001110010001000O

010011010001110010000111110O

101110111000110010111000100O

100000100100001110111111010l

011111001100001011110101001O

000001011011000000101111010O

011100011101010001101001000l

011110011110000010000000111

110000001110010001010100100O

101100010101001000011011101O

110101001110000000011011001O

101110101101101011000010000l

000010010110001001100100100l

011100011100100100000110110l

001101101011001110100010000l

011001110011001100111011110l

110101000110101011001001101O

time for: 28 1.585058 3.083862

110001100001110001011010111OO

000110101000111100101010011ll

000000100111011110011111100l0

001101000111001111011110000l

010011001101110101001000110O1

00001100000111110111100111101

00110101011000111110111001101

11111110110110110110101100100

10001000100101101000010111101

01011011101001101001000111111

01100101110000011001110010011

11101000111100101100011010100

00011010011001111011111110010

01100100101110110001000101011

10000100000000000011001001100000

10011000010110110001101100000

01100101011110100110010010001

10100000010011011000100110100

00101010101100110001101101111

11111011011100111010111111010

time for: 29 2.659056 6.224059

00100110110111001001000000001100

10000001101101101110011101101O

11000111100100001000100111100l

11110001110001001010111000l010

00000110001100100010111000000001

00101010100011100111010110l100

10100001111100010010000010011l

11110101001100100110011010011O

10110000101110100010001001111l

00010010101001001111101111l000

11011010011001100011110111011O

00001001110111101010110111001O

11010100011111100101101111010l

10101001111110111111010011011 1

00011010011001110001111001010 0

01110010101001010011000110000 0

01011111111101111011110110000 0

01101101111000110101010011100 1

10001000100000000010011100111 00

10011110010110100101010011000 0

time for: 30 5.843165 12.156283

101099.h - Output brute_force2

1011110000

1110000101

0010100111

1111100111

1011010101

1101011101

1111110011

1101100010

0001001100

1100000101

0110010111

0000111000

0001101010

1001010111

1000101110

0111110001

1001101111

0001001001

0111000111

1011111111

10 0.000007 0.000109

01011100100

11101110111

10010111101

01110010010

10111100101

10100010110

00000001100

11000101000

01110000110

01101011010

10111001101

01110001100

01001010111

00011000111

00100010000

00101011001

00101111110

01110111100

11000110010

10100000010

11 0.000004 0.000014

111001110111

110110010111

011001010111

001110111110

000100001101

011110011100

101000101101

101011010001

011000011010

010101101110

011010000111

011100010100

001011010001

000011111000

111110001110

001001110111

001110110010

010101100101

001001100110

111010101010

12 0.000008 0.000019

0010000001111

1000010000010

0000010100001

0101101100010

0111000001100

1100111110010

0000100000000

0110001101111

1001111011100

0111100011001

1010101000000

0100100100011

0111000011110

0110001111011

0101011001010

1110011111100

0101001000100

0111000000100

0011110001000

0001001100100

13 0.000011 0.000047

00101011000010

01100000000000

11011101001100

01011000001100

01011001110101

00111100101010

01001100001001

10010011011110

01001111100101

00111111010010

00001000101010

10001011101010

10110010011100

00011011011011

01110000001101

00100100100100

10100000100100

11111010111000

11111001110000

01110011001111

14 0.000035 0.000116

101000110000011

010001001000100

111011111010001

001010010110100

101011110010111

001111001110100

110101101010110

101011001110000

010100000110011

010000011100101

010100111111011

101011010000001

000011110011001

000111101001100

000010011001101

100010101010001

011001001110001

001011100100000

110110000000001

001101010101101

15 0.000056 0.000174

1110111100100111

1111001101100110

0110011100100110

1101001000000000

0111101101011100

1010001010101101

1110011010000110

1111100101100001

1101000110000101

1001001011000001

0100001110110001

1100100100001111

1000010000000111

0110001010111010

0100000001000010

0101000011001010

1100111110101100

0011011001110101

1100101010100100

0001101010010000

16 0.000087 0.000327

10101111000110111

11001101001010011

00001111011010111

00010101110110100

01100011110110001

00001100001011100

00010111101101100

01000001010110000

11110100110110000

01100010000111110

11111101001111010

10010100110100110

01100001011011101

11110010100111101

11101101110101110

11011011110000110

01110111110001100

10000111000101111

00000001100010110

00111101100010101

17 0.000272 0.000720

100100110010110110

000000110010100100

000100001100001101

001001111001101011

000010100101000111

111011110000011011

011011101011101001

011001111010011000

100111100001001001

110100010110011101

011101111101110001

111011010000001110

011110110110101101

111100010101111011

011011001000011101

010001000111100110

000010010011100011

101010011010000100

011101111101110100

111110100100101011

18 0.000403 0.001833

0110010101101010110

1101001100011100001

1111100011111010110

0111110101100100000

0110111100110111110

1000010001101101111

1010100101011011001

0010011011111000011

1001111110011001110

0010111100111110100

0100100110110111000

1000110000000101011

1001000101010010100

0000110010010010111

1101111110010110100

0010000001000101101

0001001000000111010

0000101100001110011

1111100111000110100

1110001111100000110

19 0.001056 0.002818

10001110100001100110

00100000000110010111

10001001100101000010

10000100011011011100

00000011000111010111

11101101000000011110

01000000000100011100

11001101000110000110

00101101111111011110

00100100000100001110

00010100101101100001

00011011001110111001

00111001111010010001

01110011110110010000

01101011010011011110

00101001010111011111

11010111110101001111

11101001101100100001

01110011011110110111

00100000001110111001

20 0.001445 0.005457

10010010100101010110011

11010101010011100110 0

00010011001101001010 1

10101011110111011011 0

00001000110101011011 1

11111010110100110100 0

00000010101101001110 0

10110101011111101111 1

10010111100111001001 1

11010111000011110011 11

11011111111011010001 1

10100101000111000010 1

00010001110011100000 1

00101001000100000111 0

10100111010100000001 1

00001101100110111000 1

11101001111010001011 0

00010000011000010001 0

00110111010010110011
11000100111011001010
21 0.003056 0.010880
10011101110100111010 10
01100110010110101001 01
10100001100100001101 11
01011010110101001111 01
00001100000110010101 01
11011010111010110010 00
00101011111011111001 10
00101000111101100000 10
11110100011100111101 00
10011110000111001111 10
10010010111100100001 01
10001111001100000001 010
01100101110110000111 00
01111000000101000000 00
11001110011100011000 11
10000110000000111111 00
10000010100110000101 01
00000000011011000000 10
01100111110111000000 01
01110111100111100111 00
22 0.007950 0.023442
10100001010001110101 101
01011011010010100100 010
11101111100101110010 000
00011101111101011000 110
10000010111101001100 10

000001110000001000010010

011010101001000000100101

011011110011010011011110

010100110001010101110111

100010011011000011101100

100101110110011111110111

110000000110111000011100

110010111110010011101000

000110100001111101010001

111100011010001000000010

011011001010010111110110

000101101010100110010011

001010110110011001001011

010001011011000110000010

100010010010000011001101

23 0.012019 0.033068

110001100101111100100101

011100011111101101111110

100000000111101110010111

011000001100111010001101

000001011000110101011010

000001001111111001010111

010100101000101011110110

110100000101000000111101

110010111000111111001011

010101111011101010110010

101000001110000011010100

110101110011001010101110

010100101000100111110011

111101000111100101111001

101110110101101110000101

011001001010110111000111

011111101101001011010011

111111011000101100110111

001100011110100011111001

001110011111011100111101

24 0.028580 0.085257

111101110101010110010001

110110010100111011110000

000101111001101001111001

000111000000111000110001

010101011111101101100010000

110101101110001101111010

001111000010100110100010

010011111001000001100100

101001100000010101000111

000010011000000101010111

110011111110011100111101

101001011100010011011110

000101011100100011011001

010011011000101001010010

101001110101000101110010

101010001110100001011101

010111111100001111101001

001011111001111100111011

000101011101110010010100

100000111011010100100101

25 0.072658 0.191273

01011011000011100110011011

01000010101101110111011100

00111011001110001011101011

11011000110010000111000110

01111111110010111000001001

10001110100101010101100000

01001100110111111000001010

01000001011010110110110101

01011001011000001001000101

00001110111011110110000010

11101110010101001111011000

00110001011001101011101101

11000101100011111100110001

01100110100111110101000011

10001000001100001011100010

00011101101111010101010110

01010111011111000110100011

01000000100101111011101010

10011110000111001110101100

10011100011100101011001111

26 0.146703 0.427703

00110011001001110011000011

01110011011000111000100110

01000010100011101111111011

01001100010001001100110011

00000110110011000101101101

00101100100000111100110100

11001001110011000111110100

10011011110110010100011011

10000101111001001011011010

01100011111001101110111010

01110000111001100111000010

11101010001100111001101001

11111001111101000011011001 0

11101010010011100110110111 1

00101101011111110110100001 11

00101000010111011101110001 0

00000111110101000111100110 1

10001001000011000000101111 1

00000111010111111100010000 1

00111011111001100110111111 0

27 0.256804 0.778189

11110010011010110000101001 10

01100110010001010111001001 00

00101110000000010000000011 000

01111101101000111001000100 00

01001101000111001000011111 00

10111011100011001011100010 00

10000010010000111011111101 01

01111100110000101111010100 10

00000101101100000010111101 00

01110001110101000110100100 01

01111001111000001000000001 11

11000000111001000101010010 00

10110001010100100001101110 10

11010100111000000001101100 10

10111010110110101100001000 01

00001001011000100110010010 01

01110001110010010000001101101

00110110101100111010000100001

01100111001100110011101111101

11010100011010101100100110010

28 0.343930 0.791127

11000110000111000101101011100

00011010100011111001010100111 1

00000010011101111001111110010

00110100011100111101111000001

01001100110111010100100011001

00001100000111111011110011110 1

00110101011000111110111001101

11111110110110110110101100100

10001000100101101000010111101

01011011101001101001000111111

01100101110000011001110010011

11101000111100101100011010100

00011010011001111011111110010

01100100101110110001000101011

10000100000000001100100110000

10011000010110110001101100000

01100101011110100110010010001

10100000010011011000100110100

00101010101100110001101101111

11111011011100111010111111010

29 1.288769 4.481757

00100110110111001001000000 1100

10000001101101101110011101 1010

11000111100100001000100111100 1

111100011100010010101110001010

000001100011001000101110000001

001010101000111001110101101100

101000011111000100100000100111

111101010011001001100110100110

101100001011101000100010011111

000100101010010011111011111000

110110100110011000111101110110

000010011101111010101101110010

110101000111111001011011110101

101010011111101111110100110111

000110100110011100011110010100

011100101010010100110001100000

010111111111011110111101100000

011011011110001101010100111001

100010001000000001001110011100

100111100101101001010100110000

30 2.724731 8.781207

101099.h - Output meet_in_the_middle

1011110000

1110000101

0010100111

1111100111

1011010101

1101011101

1111110011

1101100010

0001001100

1100000101

0110010111

0000111000

0001101010

1001010111

1000101110

0111110001

1001101111

0001001001

0111000111

1011111111

01011100100

11101110111

10010111101

01110010010

10111100101

10100010110

00000001100

11000101000

01110000110

01101011010

10111001101

01110001100

01001010111

00011000111

00100010000

00101011001

00101111110

01110111100

11000110010

10100000010

111001110111

110110010111

011001010111

001110111110

000100001101

011110011100

101000101101

101011010001

011000011010

010101101110

011010000111

011100010100

001011010001

000011111000

111110001110

001001110111

001110110010

010101100101

001001100110

111010101010

0010000001111

1000010000010

0000010100001

0101101100010

0111000001100

1100111110010

0000100000000

0110001101111

1001111011100

0111100011001

1010101000000

0100100100011

0111000011110

0110001111011

0101011001010

1110011111100

0101001000100

0111000000100

0011110001000

0001001100100

00101011000010

01100000000000

11011101001100

01011000001100

01011001110101

00111100101010

01001100001001

10010011011110

01001111100101

00111111010010

00001000101010

10001011101010

10110010011100

00011011011011

01110000001101

00100100100100

10100000100100

11111010111000

11111001110000

01110011001111

101000110000011

010001001000100

111011111010001

001010010110100

101011110010111

001111001110100

110101101010110

101011001110000

010100000110011

010000011100101

010100111111011

101011010000001

000011110011001

000111101001100

000010011001101

100010101010001

011001001110001

001011100100000

110110000000001

001101010101101

1110111100100111

1111001101100110

0110011100100110

1101001000000000

0111101101011100

1010001010101101

1110011010000110

1111100101100001

1101000110000101

1001001011000001

0100001110110001

1100100100001111

1000010000000111

0110001010111010

0100000001000010

0101000011001010

1100111110101100

0011011001110101

1100101010100100

0001101010010000

10101111000110111

11001101001010011

00001111011010111

00010101110110100

01100011110110001

00001100001011100

00010111101101100

01000001010110000

11110100110110000

01100010000111110

11111101001111010

10010100110100110

01100001011011101

11110010100111101

11101101110101110

11011011110000110

01110111110001100

10000111000101111

00000001100010110

00111101100010101

100100110010110110

000000110010100100

000100001100001101

001001111001101011

000010100101000111

111011110000011011

011011101011101001

011001111010011000

100111100001001001

110100010110011101

011101111101110001

111011010000001110

011110110110101101

111100010101111011

011011001000011101

010001000111100110

000010010011100011

101010011010000100

011101111101110100

111110100100101011

0110010101101010110

1101001100011100001

1111100011111010110

0111110101100100000

0110111100110111110

1000010001101101111

1010100101011011001

0010011011111000011

1001111110011001110

0010111100111110100

0100100110110111000

1000110000000101011

1001000101010010100

0000110010010010111

1101111110010110100

0010000001000101101

0001001000000111010

0000101100001110011

1111100111000110100

1110001111100000110

10001110100001100110

00100000000110010111

10001001100101000010

10000100011011011100

00000011000111010111

11101101000000011110

01000000000100011100

11001101000110000110

00101101111111011110

00100100000100001110

00010100101101100001

00011011001110111001

00111001111010010001

01110011110110010000

01101011010011011110

00101001010111011111

11010111110101001111

11101001101100100001

01110011011110110111

00100000001110111001

100100101001010110011

110101010100111001100

000100110011010010101

101010111101110110110

000010001101010110111

111110101101001101000

000000101011010011100

101101010111111011111

100101111001110010011

110101110001111001111

110111111110110100011

101001010001110000101

000100011100111000001

001010010001000001110

101001110101000000011

000011011001101110001

111010011110100010110

000100000110000100010

00110111010010110011

11000100111011001 1010

10011101110100111010 10

01100110010110101001 01

10100001100100001101 11

01011010110101001111 01

00001100000110010101 01

11011010111010110010 00

00101011111011111001 10

00101000111101100000 10

11110100011100111101 00

10011110001110011111 10

10010010111100100001 01

10001111001100000010 10

01100101110110000111 00

01111000000101000000 00

11001110011100011001 11

10000110000000111111 00

10000010100110000101 01

00000000011011000000 10

01100111110111000000 01

01110111100111100111 00

10100001010001110101 101

01011011010010100100 010

11101111100101110010 0000

00011101111101011000 110

10000001011110100110 010

00000111000000100010 010

01101010100100000100 101

011011110011010011011110

010100110001010101101111

100010011011000111011100

100101110110011111101111

110000000110111000011100

110010111110010011010000

000110100001111101010001

111100011010001000000010

011011001010010111101110

000101101010100110010011

001010110110011001001101

010001011011000110000010

100010010010000011001101

110001100101111100100101

011100011111110110111111110

100000000111101110010111

011000001100111010001101

000001011000110101011010

000001001111111001010111

010100101000101011110110

110100000101000000111101

110010111000111111001011

010101111011101010110010

101000001110000011010100

110101110011001010101110

010100101000100111110011

111101000111100101111001

101110110101101110000101

011001001010110111000111

011111101101001011010011

111111011000101100110111

001100011110100011111001

001110011111011100111101

111101110101010110010000 1

110110010100111011110000 0

000101111001101001111001 0

000111000000111000110000 1

010101011111101101100100 00

110101101110001101111011 0

001111000010100110100010 0

010011111001000001100100 0

101001100000010101000111 0

000010011000000101010111 0

110011111110011100111101 0

101001011100010011011111 0

000101011100100011011000 1

010011011000101001010010 0

101001110101000101110010 1

101010001110100001011110 1

010111111100001111101000 1

001011111001111100111101 1

000101011101110010010100 0

100000111011010100100101 1

010110110000111001100110 11

010000101011011101110111 00

001110110011100010111010 11

110110001100100001110001 10

011111111100101110000010 01

100011101001010101101100000

010011001101111111000001010

010000010110101101101101101

010110010110000001001000101

000011101110111110110000010

111011100101010011111011000

001100010110011010111101101

110001011000111111100110001

011001101001111101010000011

100010000011000010111100010

000111011011111010101010110

010101110111111000110100011

010000001001011111011101010

100111100001110011101101100

100111000111001010110011111

001100110010011100110000011

011100110110001110001000110

010000101000111011111111011

010011000100010011001100011

000001101100110001011011101

001011001000001111001101000

110010011100110001111101100

100110111101100101000111011

100001011110010011011011010

011000111110011011101111010

011100001110011001110000010

111010100011001110011010001

111110011111010000110110010

111010100100111001101101111

00101101011111110110100000111

00101000010111011011100010

00000111110101000111100 1101

10001001000011000000 1011111

00000111010111111 1000100001

00111011111001100 1101111110

11110010011010110000 10100110

01100110010001010111001001 00

00101110000000100000000011000

01111101101000111001000 10000

01001101000111001000011111 00

10111011100011001011100 01000

10000010010000111011111110101

01111100110000101111010100 10

00000101101100000010111101 00

01110001110101000110100 10001

01111001111000001000000000111

11000000111001000101010010 00

10110001010100100000110111010

11010100111000000000110110010

10111010110110101100000100001

00001001011000100110010 01001

01110001110010010000011 01101

00110110101100111010001000 01

01100111001100110011101 11101

11010100011010101100100 11010

11000110000111000101101011100

00011010100011111001010100 1111

00000010011101111001111110010

00110100011100111101111000001

01001100110111010100100011001

00001100000111110111100111101

00110101011000111110111001101

11111110110110110110101100100

10001000100101101000010111101

01011011101001101001000111111

01100101110000011001110010011

11101000111100101100011010100

00011010011001111011111110010

01100100101110110001000101011

10000100000000000110010011000 0

10011000010110110001101100000

01100101011110100110010010001

10100000010011011000100110100

00101010101100110001101101111

11111011011100111010 11