Algoritmos e Estruturas de Dados

Subset Sum Problem

Relatório

João Miguel Matos 103341 (50%)
Filipe Maia Antão 103470 (50%)
30-12-2021

Introdução ao problema

O problema que nos era proposto resolver para este projeto era o problema do soma de subconjuntos (subset sum problem). Este problema consiste em, dados um conjunto com vários números e uma soma, determinar se é possível executar a soma com os números dados pelo conjunto e se possível, quais os números que servem como parcelas para o cálculo dessa soma. É importante considerar que o conjunto contém números inteiros positivos apenas, ordenados de forma crescente. Este problema suscitou muito interesse na comunidade científica, tendo por isso muitas propostas de algoritmos para chegar a uma solução (Brute Force e Branch and Bound apoiando-se na recursividade, técnica de Horowitz e Sahni ou meet-in-the-middle, ou a técnica de Schroeppel and Shamir, etc...).

Neste projeto tentamos recriar os algoritmos Brute Force, Branch and Bound e Meet-In-The-Middle. Em todos eles, o objetivo é encontrar uma solução que consiste num vetor binário (zeros e uns) que representa os elementos do conjunto inicial que, somados, resultam na soma dada.

João Miguel Matos
Filipe Antão

Soluções encontradas

Brute Force

Este é o primeiro algoritmo que desenvolvemos. Apoiando-se na recursividade, este constrói o vetor um bit de cada vez, percorrendo o array dado de forma crescente, analisando todas as permutações possíveis para o vetor.

A função toma como argumento uma estrutura (BFData) que contém todos os dados necessários:

- *int n* – comprimento do array *p*;
- *integer_t *p* – ponteiro para o array *p*;
- *integer_t desiredSum* – soma pretendida;
- *int nextIndex* – índice do vetor de bits a ser processado pela função (ao chamar a função, o valor deve ser sempre 0);
- *integer_t partialSum* – soma dos valores considerados para a solução até ao momento (ao chamar a função, o valor deve ser sempre 0);
- *int *b* – ponteiro para o array que será populado com o vetor de bits correspondente à solução.

A recursividade tem dois escapes possíveis:

- Quando o valor de *partialSum* é igual ao valor de *desiredSum*. Isto é sinal que uma solução foi encontrada, portanto a função retorna o valor 1 e o array apontado por *b* estará já populado com o vetor de bits correspondente à solução.
- Quando o valor de *nextIndex* é igual ao valor de *n*. Aqui, o vetor de bits desta esta cadeira recursiva (agora finalizada) não representa solução, portanto a função retorna o valor 0.

Branch and Bound

Este pode ser considerado uma versão melhorada do primeiro algoritmo. A diferença essencial está nos dois novos casos triviais da recursividade. Neles, é analisado o estado corrente da solução – caso a soma parcial até ao momento seja maior que a soma pretendida, ou caso a soma parcial até ao momento mais a soma restante, tomando todos os bits restantes como um seja menor que a pretendida, a função cessa processamento posterior e retorna zero, já que não existe soma possível.

Para isto, a estrutura passada como argumento (BBData) contém o parâmetro adicional *remainingSum*, que contém o valor da soma dos valores que falta analisar.

À semelhança do algoritmo anterior, quando uma solução é encontrada, a função retorna 1 e o array apontado por *b* estará já populado com o vetor de bits correspondente a essa solução, e quando, finalizada a cadeia de recursão, não existe solução, a função retorna 0.

Meet in the Middle

Também conhecido com a técnica de Horowitz e Sahni.
Esta função recebe como argumentos:

- *int n* – comprimento do array *p*;
- *integer_t *p* – ponteiro para o array *p*;
- *integer_t desiredSum* – soma pretendida;
- *char *bin* – ponteiro para o array a ser populado pelo bit de vectores correspondente à solução.

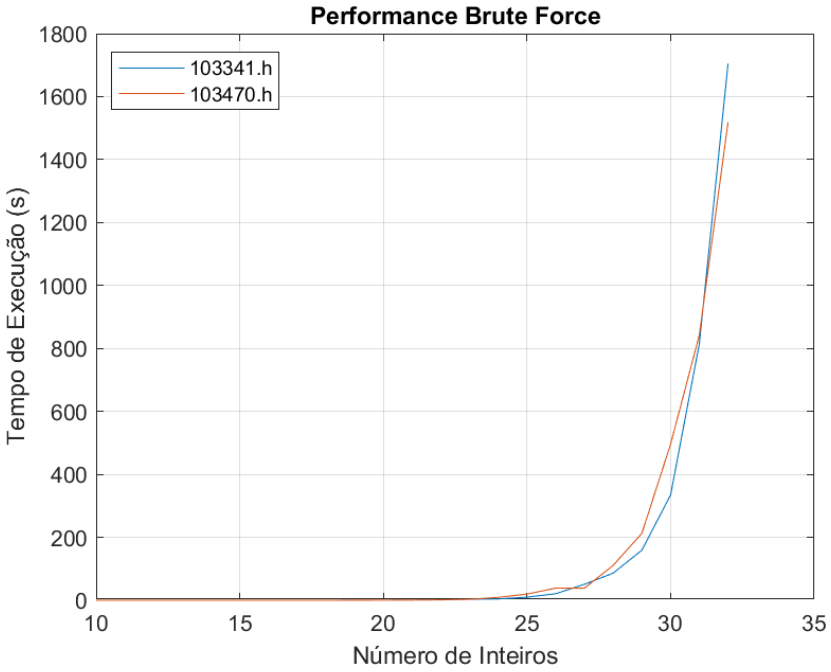
Nela, o array *p* é dividido em dois, e para cada uma dessas partes é criado e armazenado um array com todas as somas possíveis (arrays *a* e *b*, tamanhos n_a e n_b), utilizando um algoritmo de permutações dos vetores de bits, que são também armazenados (arrays *aBin* e *bBin*). Ambos os arrays são ordenados de forma crescente (o algoritmo de *sorting* utilizador é o quicksort adaptado do website https://www.programiz.com/dsa/quick-sort) e, para se obter a solução, percorre-se os arrays *a* e *b*, o primeiro a partir do índice *0* (início), e o segundo a partir do índice $n_b - 1$ (fim). Verificamos se $a_i + b_i = desiredSum$. Se sim, uma solução foi encontrada, então a função popula o array apontado por *bin* com a junção dos vetores de bits correspondentes a $a_i + b_i$ armazenados em *aBin* e *bBin*. Se a soma for demasiado grande, diminíu-se o índice de *b*. Se a soma for demasiado pequena, aumenta-se o índice de *a*. Caso algum dos arrays sair dos limites, não existe solução para os dados fornecidos.

Nesta função, decidimos utilizar um array de *char* em vez de *int* por questões de memória. Como não estávamos a conseguir computar todos os problemas dados, decidimos fazer essa mudança, já que um *char* ocupa 1 byte ao invés dos 4 bytes de uma *int*.

Análise das soluções encontradas

Brute Force

Este algoritmo percorre todas as situações possíveis no pior cenário. Dado que para cada inteiro podemos ter apenas duas opções (0 ou 1), concluimos assim que a complexidade do algoritmo é de O(2^N), sendo N o número de inteiros. Logo, o tempo de execução do algoritmo cresce de forma exponencial em função do número de inteiros, como é possível observar no gráfico representado (Observação: Como o tempo de execução era demasiado grande, não foi possível recolher valores de tempo para valores de n superiores a 32).

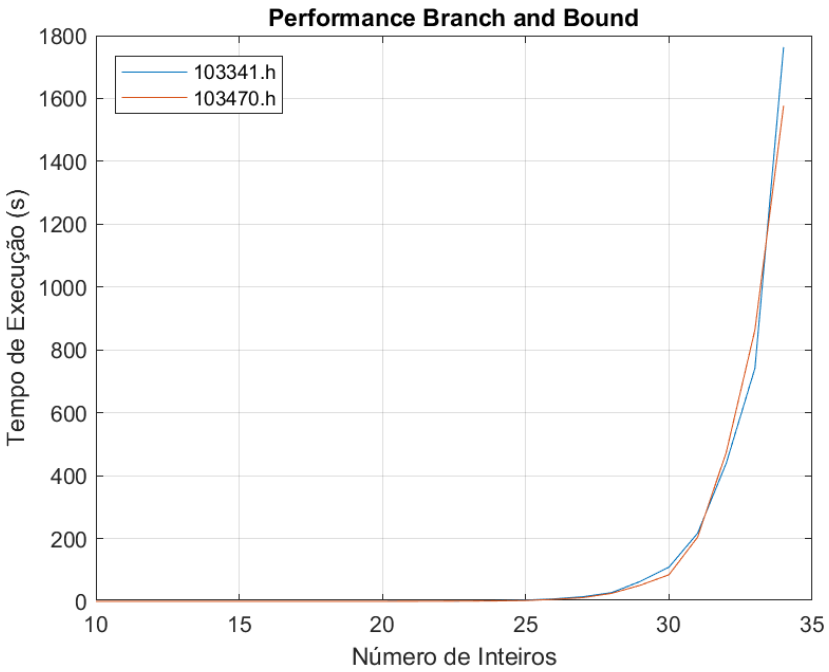


N	Tempo(103341.h)	Tempo(103470.h)	N	Tempo(103341.h)	Tempo(103470.h)
10	0.00026	0.000266	**22**	1.498343	1.316146
11	0.000589	0.000783	**23**	3.400088	3.049042
12	0.001180	0.001024	**24**	4.254520	9.267880
13	0.001883	0.002466	**25**	9.830943	19.634865
14	0.005028	0.004929	**26**	20.844932	38.606905
15	0.010587	0.008422	**27**	50.769689	38.392771
16	0.019641	0.19611	**28**	86.075633	110.941589
17	0.037737	0.045166	**29**	159.153207	212.584424
18	0.073003	0.071498	**30**	334.538435	495.838451
19	0.197814	0.157574	**31**	814.788577	842.382047
20	0.406625	0.357410	**32**	1704.890525	1519.302801
21	0.886781	0.665483			

João Miguel Matos
Filipe Antão

Branch and Bound

Este algoritmo é semelhante ao Brute Force, apenas interrompendo o cálculo para um determinado número de inteiros quando a soma é impossível com os cálculos já feitos. No entanto num pior cenário possível, terá de percorrer todos os casos possíveis sendo a complexidade O(2^N). O tempo de execução deste algoritmo cresce de forma exponencial. (Observação: Como o tempo de execução era demasiado grande, não foi possível recolher valores de tempo para valores de n superiores a 32).

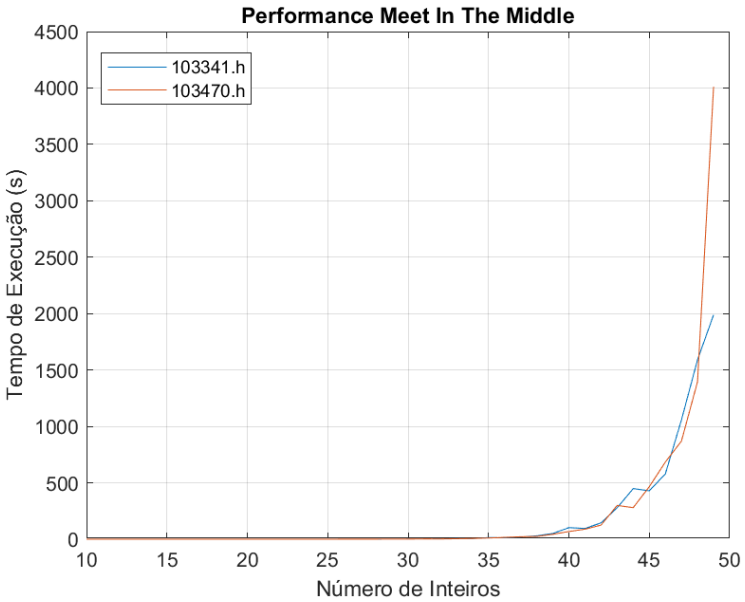


N	Tempo(103341.h)	Tempo(103470.h)	N	Tempo(103341.h)	Tempo(103470.h)
10	0.000085	0.000150	**23**	0.788883	0.886233
11	0.000281	0.000341	**24**	1.884009	1.514439
12	0.000524	0.000469	**25**	4.105963	2.772111
13	0.000868	0.000924	**26**	8.062917	6.490392
14	0.002108	0.001577	**27**	14.858841	11.939659
15	0.003497	0.003251	**28**	27.805003	25.576561
16	0.005932	0.005541	**29**	63.597179	51.391546
17	0.012985	0.021578	**30**	108.910439	84.795131
18	0.028299	0.019294	**31**	216.880176	204.477320
19	0.039796	0.056872	**32**	440.080007	474.691524
20	0.122002	0.070978	**33**	741.552437	863.709310
21	0.209529	0.211319	**34**	1763.692509	1577.058275
22	0.504800	0.507696			

João Miguel Matos
Filipe Antão

Meet In The Middle

Este algoritmo é mais eficiente para arrays de grande dimensão do que os dois algoritmos anteriores. Ao dividirmos o array em dois arrays de tamanho igual e calcular as somas parciais de todos os elementos a complexidade é O(2^N). Depois, damos sort com a função *quicksort()* que tem complexidade O(N^2). Por fim ao fazer os cálculos das somas temos complexidade O(N/2). Complexidade Final: O(2^N*N^3*(1/2)) . No entanto, não suporta arrays com mais do que 49 números inteiros (o programa retorna killed para n > 49).



N	Tempo(103341.h)	Tempo(103470.h)	N	Tempo(103341.h)	Tempo(103470.h)
10	0.000406	0.000427	**30**	1.478541	1.189185
11	0.000650	0.000712	**31**	2.425823	2.506681
12	0.000859	0.000873	**32**	2.476908	2.369058
13	0.001410	0.001562	**33**	6.132900	4.366565
14	0.001902	0.001970	**34**	5.555633	6.761485
15	0.003768	0.003555	**35**	14.434693	10.711662
16	0.005035	0.005629	**36**	13.604501	16.495458
17	0.008418	0.007846	**37**	20.153064	21.795100
18	0.010735	0.010776	**38**	29.961112	24.973794
19	0.017322	0.018786	**39**	50.976135	43.731178
20	0.026075	0.024051	**40**	102.524772	68.000000
21	0.040207	0.037918	**41**	95.376878	89.053430
22	0.060582	0.067825	**42**	144.884728	125.075419
23	0.085507	0.111810	**43**	278.088275	299.378846
24	0.116767	0.131013	**44**	448.630000	280.018263
25	0.212961	0.265363	**45**	429.079337	465.240683
26	0.286359	0.423662	**46**	578.827179	683.875798
27	0.434214	0.437235	**47**	1056.696078	869.505469
28	0.608936	0.647884	**48**	1589.892435	1395.060521
29	0.936382	1.289928	**49**	1988.592892	4011.089213

Apêndice

Código

```
//

// AED, November 2021

//

// Solution of the first practical assignement (subset sum problem)

//

// 103341 - João Miguel Matos

// 103470 - Filipe Antão

//


#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 "103470.h"

#endif


//

// include files

//


#include <math.h>

#include <stdio.h>

#include <stdlib.h>


#include "elapsed_time.h"

#include STUDENT_H_FILE


//

// custom data types

//

// the STUDENT_H_FILE defines the following constants and data types

//

//   #define min_n      24              --- the smallest n value we will

//   handle #define max_n      57             --- the largest n value we

//   will handle #define n_sums     20             --- the number of sums

//   for each n value #define n_problems  (max_n - min_n + 1)  --- the number of

//   n values

//

//   typedef unsigned long long integer_t;   ---  64-bit unsigned integer

//   typedef struct

//   {

//     int n;                        --- number of elements of the set

//     (for a valid problem, min_n <= n <= max_n) integer_t p[max_n]; --- the

//     elements of the set, already sorted in increasing order (only the first n

//     elements are used) integer_t sums[n_sums];             --- several

//     sums (problem: for each sum find the corresponding subset)

//   }

//   subset_sum_problem_data_t;            --- weights p[] and sums for a

//   given value of n

//

//   subset_sum_problem_data_t all_subset_sum_problems[n_problems]; --- // the

//   problems

//
```

```
//
// place your code here
//
// possible function prototype for a recursive brute-force function:
//   int brute_force(int n,integer_t p[n],int desired_sum,int
//   current_index,integer_t partial_sum);
// it sould return 1 when the solution is found and 0 otherwise
// note, however, that you may get a faster function by reducing the number of
// function arguments (maybe a single pointer to a struct?)
//

// auxiliary functions
void printIntArray(int n, int *b) {  // prints an int array
    for (int i = 0; i < n; ++i) {
        printf("%d", b[i]);
    }
    printf("\n");
}

void printTArray(int n, integer_t *b) {  // prints a integer_t array
    for (int i = 0; i < n; ++i) {
        printf("%llu\n", b[i]);
    }
}

void printCharArray(int n, char *b) {  // prints a char array
    for (int i = 0; i < n; ++i) {
        printf("%c", b[i]);
    }
}

integer_t sumAll(int n, integer_t p[n]) {  //
    integer_t sum = 0;
    for (int i = 0; i < n; i++) {
        sum += p[i];
    }
    return sum;
}

// 1st solution code
struct bruteForceData {
    int n;
    integer_t *p;
    integer_t desiredSum;
    int nextIndex;
    integer_t partialSum;
    int *b;
};
typedef struct bruteForceData BFData;

int bruteForce(BFData *data) {
    // trivial cases
    if (data->desiredSum == data->partialSum) {
        for (int i = data->nextIndex; i < data->n; ++i) {
            data->b[i] = 0;
        }
        return 1;
```


João Miguel Matos
Filipe Antão

```
    }

    if (data->nextIndex == data->n) {
        return 0;
    }

    // set next bit to zero
    data->b[data->nextIndex] = 0;

    BFData newData = {data->n, data->p,
                data->desiredSum, data->nextIndex + 1,
                data->partialSum, data->b};
    int result = bruteForce(&newData);
    if (result == 1) {
        return 1;
    }

    // set next bit to one - no solution found on zero
    data->b[data->nextIndex] = 1;
    newData.partialSum = data->partialSum + data->p[data->nextIndex];
    return bruteForce(&newData);
}

// 2nd solution code
struct branchAndBoundData {
    int n;
    integer_t *p;
    integer_t desiredSum;
    int nextIndex;
    integer_t partialSum;
    integer_t remainingSum;
    int *b;
};
typedef struct branchAndBoundData BBData;

int branchAndBound(BBData *data) {
    // trivial cases
    if (data->desiredSum == data->partialSum) {
        for (int i = data->nextIndex; i < data->n; ++i) {
            data->b[i] = 0;
        }
        return 1;
    }

    if (data->nextIndex == data->n) {
        return 0;
    }

    if ((data->partialSum + data->remainingSum) < data->desiredSum) {
        return 0;
    }

    if (data->partialSum > data->desiredSum) {
        return 0;
    }

    // set next bit to zero
    data->b[data->nextIndex] = 0;
```


João Miguel Matos
Filipe Antão

```
    BBData newData = {data->n, data->p, data->desiredSum, data->nextIndex + 1, data->partialSum, data->remainingSum - data->p[data->nextIndex + 1], data->b};

    int result = branchAndBound(&newData);
    if (result == 1) {
      return 1;
    }

    // set next bit to one - no solution found on zero
    data->b[data->nextIndex] = 1;
    newData.partialSum = data->partialSum + data->p[data->nextIndex];
    return branchAndBound(&newData);
    return 0;
}

// 3rd solution code
void swap(int x, int y, integer_t array[], char **bin) {
    //swap two integer_t elements
    integer_t temp = array[x];

    temp = array[x];
    array[x] = array[y];
    array[y] = temp;

    //swap corresponding binary entries
    char *tmp = bin[x];
    bin[x] = bin[y];
    bin[y] = tmp;

}

int partition(integer_t array[], int low, int high, char **bin, int *i) {
    // select the rightmost element as pivot

    integer_t *pivot = (integer_t *) malloc(sizeof(integer_t));

    *pivot = array[high];


    // pointer for greater element

    *i = (low - 1);

    // traverse each element of the array
    // compare them with the pivot
    for (int j = low; j < high; j++) {
      if (array[j] <= *pivot) {
        // if element smaller than pivot is found
        // swap it with the greater element pointed by i
        (*i)++;

        // swap element at i with element at j
        swap(*i, j, array, bin);
      }
    }

    // swap the pivot element with the greater element at i
    swap(*i + 1, high, array, bin);
```

```
    free(pivot);

    // return the partition point

    return (*i + 1);

}


void quickSort(integer_t array[], int low, int high, char **bin) { // Complexidade O(n log n)

    if (low < high) {

        // find the pivot element such that

        // elements smaller than pivot are on left of pivot

        // elements greater than pivot are on right of pivot

        //int pi = partition(array, low, high, bin);


        int *pi = (int *) malloc(sizeof(int));


        int *i = (int *)malloc(sizeof(int));


        *pi = partition(array, low, high, bin, i);


        free(i);


        // recursive call on the left of pivot

        quickSort(array, low, *pi - 1, bin);


        // recursive call on the right of pivot

        quickSort(array, *pi + 1, high, bin);


        free(pi);

    }

}


int meetinTheMiddle(int n, integer_t *p, integer_t desiredSum, char *bin) { // non-recursive

    int high = ceil(n / 2.0f), low = floor(n / 2.0f);

    int aSize = (int)pow(2, high), bSize = (int)pow(2, low);

    integer_t *a = (integer_t *)malloc(aSize * sizeof(integer_t)), *b = (integer_t *)malloc(bSize * sizeof(integer_t));

    integer_t sum;

    char **aBin = (char **)malloc(aSize * sizeof(char *)), **bBin = (char **)malloc(bSize * sizeof(char *));

    int i, j, k;


    // get all subset sums

    for (i = 0; i < aSize; ++i) {

        sum = 0;

        aBin[i] = (char *)malloc(high * sizeof(char));

        for (j = 0; j < high; ++j) {

            if (i & (1 << j)) {

                sum += p[j];

                aBin[i][j] = '1';

            } else {

                aBin[i][j] = '0';

            }

        }

        a[i] = sum;

    }


    for (i = 0; i < bSize; ++i) {

        sum = 0;

        bBin[i] = (char *)malloc(low * sizeof(char));

        for (j = 0; j < low; ++j) {
```


João Miguel Matos
Filipe Antão

```
        if (i & (1 << j)) {
          sum += p[j + high];
          bBin[i][j] = '1';
        } else {
          bBin[i][j] = '0';
        }
      }
      b[i] = sum;
    }

    // sort them and corresponding binary arrays
    quickSort(a, 0, aSize - 1, aBin);
    quickSort(b, 0, bSize - 1, bBin);

    // find solution
    i = 0;
    j = bSize - 1;
    while (i < aSize || j >= 0) {
      sum = a[i] + b[j];
      if (sum < desiredSum) {
        i++;
      } else if (sum > desiredSum) {
        j--;
      } else {  // found the solution
        for (k = 0; k < high; ++k) {
          bin[k] = aBin[i][k];
        }
        for (k = 0; k < low; ++k) {
          bin[k + high] = bBin[j][k];
        }

        for (int p = 0; p < aSize; ++p) free(aBin[p]);
        for (int p = 0; p < bSize; ++p) free(bBin[p]);
        free(a);
        free(b);
        free(aBin);
        free(bBin);

        return 1;
      }
    }

    return 0;  // array went out of bounds; didn't find a solution
}

//
// main program
//

int main(void) {
    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));
    //
    // place your code here
```


João Miguel Matos
Filipe Antão

```
  //

  int i, j, result;

  printf("\nTesting...\n");

  int n;

  integer_t *p, *sums;



  for (i = 0; i <= n_problems; ++i) {

    n = all_subset_sum_problems[i].n;

    p = all_subset_sum_problems[i].p;

    sums = all_subset_sum_problems[i].sums;


    int bf[n];

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

      BFData BFTest = {n, p, sums[j], 0, 0, bf};

      result = bruteForce(&BFTest);

    }

  }


  for (i = 0; i <= n_problems; ++i) {

    n = all_subset_sum_problems[i].n;

    p = all_subset_sum_problems[i].p;

    sums = all_subset_sum_problems[i].sums;


    int cbf[n];

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

      BBData BBTest = {n, p, sums[j], 0, 0, sumAll(n, p), cbf};

      result = branchAndBound(&BBTest);

    }


  }

  for (i = 0; i < n_problems; ++i) {

    n = all_subset_sum_problems[i].n;

    p = all_subset_sum_problems[i].p;

    sums = all_subset_sum_problems[i].sums;


    char cbf[n];

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

      result = meetInTheMiddle(n, p, sums[j], cbf);

    }


  }


  return 0;

}
```


João Miguel Matos
Filipe Antão

Bit Vectors

103341.h

n = 10

1001101101 | 0011011011 | 0010100011 | 1111111111

0101101111 | 0001011110 | 0110001010 | 1110000001

0100011010 | 0111001111 | 0110111111 | 0010111101

0011110000 | 0110101110 | 0011001010 | 1111101000

0010100000 | 0001011000 | 0111001010 | 0011111100

---

n = 11

00011111111 | 00001000111 | 11000010011 | 01000011000

00110011111 | 10010001000 | 00110110110 | 10011111000

10001000110 | 01100110011 | 01011101100 | 11111000000

11000001111 | 10100101110 | 10010001100 | 11111111101

10101000011 | 00101000011 | 00100010101 | 01001101000

---

n = 12

100001101110 | 011011000111 | 100111010101 | 111011111101

001010000110 | 100100000001 | 110011000011 | 001010101100

000100010101 | 101011010101 | 000011001110 | 010000100101

011011010001 | 011100110101 | 101010101011 | 101100111110

110011100000 | 110100001000 | 001001010010 | 010101100001

---

n = 13

0101011100010 | 0000010010010 | 0010101010111 | 1001110010100

0100001110100 | 0010000101110 | 1111100001111 | 0100000111001

1110101001001 | 0001011011110 | 0100001011010 | 0011001010001

0000110100111 | 0111110000100 | 0101001000100 | 1100101110100

1110011010110 | 0100101010100 | 1011011001000 | 0000111000110

---

n = 14

10001111101100 | 00011110100011 | 10000011001001 | 01101110010101

01001111110010 | 10100111111001 | 10001000011110 | 00100000001111

10011001101110 | 11001011100101 | 01011010101101 | 11000110101111

11111011100110 | 11110001111001 | 00111011111010 | 00101000111000

11001011010011 | 10101011100001 | 10101110010010 | 00111011010011

---

n = 15

111000001111101 | 010101100000100 | 101110001000010 | 001010001110001

110101000010100 | 010011100011001 | 101100001011101 | 011010110100100

001011110001010 | 100101110101010 | 000101011111100 | 111001100011110

101100001100000 | 010011010001000 | 010110110010011 | 110101100000011

100100110110001 | 100000000010010 | 101101001011000 | 111001010001000

João Miguel Matos
Filipe Antão

---

n = 16

1100010010111010 | 0000100011001111 | 1100010010110000 | 0001110100111111

0011100011011110 | 0110101011110010 | 1100101101011001 | 1101010000101011

1100001111100010 | 0110100001001000 | 1100010001010001 | 1111000100110010

1011111111011010 | 1101100011010111 | 1000010000111000 | 0101100101010010

0000011011110111 | 0010110010011111 | 0010011110110110 | 1011111100011001

---

n = 17

10011101100011011 | 10001111111011111 | 10100001110100010 | 01100011011011110

10001010011100010 | 10001111100101111 | 10110010011001101 | 01000100001111101

11101101111001011 | 01100001101000111 | 01000111001101100 | 01101110110010001

00000000100000011 | 10000010100111100 | 10101111111000101 | 01000011000111011

10001011001011100 | 11000010000100111 | 10111100111100001 | 00011111110110011

---

n = 18

11000011001000110 | 01001000100110111 | 11001111111010010 | 01001000111010110

00111111011010100 | 00000101010110011 | 01000010100011100 | 01110100100101011

10010101100101000 | 11101111000101000 | 00110110100110011 | 00001011100000100

01101110010011111 | 01010001001111110 | 10100000110110101 | 10110100001011011

10000000010101011 | 11101001010110110 | 01110100010101011 | 10111011100100001

---

n = 19

00010001001011010 | 11100001111100011 | 10100000000101111 | 11111001010111011111

01000101100111001 | 00010111111010111 | 11100110000111101 | 10001100100011000010

01111110100111010 | 11101101011000011 | 10100110100110000 | 01000100110100100000

11101010111101011 | 00110011000010000 | 11010010010111001 | 00010010011100001111

01111100111011011 | 10010110011110111 | 00011100000011110 | 11111101110101000010

---

n = 20

00001001110100011000 | 11110101101101000010 | 10100011111010001100 | 10001100101111011110

11001001110000110100 | 01111001101000011110 | 00111111011001010010 | 10101110000101010000

11010001000101101010 | 01100100011111110101 | 00111101100110100001 | 10000110010101100001

11000101101010010011 | 01100000000000101001 | 00100011101111011000 | 01111101010000001101

10011000101011110100 | 01110100110100100011 | 11101100011011000110 | 11001001111110110001

---

n = 21

00100111010000000011101 | 11111000111110001010 | 10010110010111101011 0 | 11000000100011000011 0

11100011010110010011 0 | 10111000111000000011 10 | 00000101010110000010 0 | 11101001000100100010 0

01100111011100111000 | 11001011100000111011 0 | 01101110111100100110 0 | 10110000010100011111 0

11001110001010110101 0 | 11111011011101011101 | 11111110001100010110 0 | 00010110110001000 1000

00110011010110101110 0 | 01000000010010110010 0 | 10110010011001100011 1 | 01011100101000111010

---

n = 22

01110011011001001100110011 | 11000110001010000000011 | 11110101101010011111001 | 00000001001110011101110

João Miguel Matos
Filipe Antão

011010100110010111111100 | 001101011110101101000 | 000101000001100111111 0 | 101100010111100011011 0

110100000001000011010 0 | 000110110000101100100 0 | 101111001011111110011 0 | 010011111100101111000 1

111100101100001001101 1 | 100110000100000001101 10 | 100001001111011011010 0 | 100001001001000110100 0

111000010011100000110 1 | 110001100100101101000 1 | 000100100011111100101 1 | 111011111101100101101 0

---

n = 23

100100001010111111111000 | 111011001001110010011 11 | 110000100100101111101 01 | 100001001000001010111 10

101001011101000011101 11 | 110011110110000000101 00 | 110101011001010001011 00 | 111100110001001011110 10

100001011101100100000 111 | 101111100010011101000 01 | 110011000001011101010 00 | 000010001011010001011 0

001011010100100000111 10 | 111010010010111000001 01 | 100010000001011101110 10 | 111101101111011101100 10

110101000111010100111 | 100011100111010011111 11 | 011000010101000010110 10 | 000000011100010010010 00

---

n = 24

001011111100011011111101 | 000001011001100010011 0111 | 101010011010111000010 001 | 000101010101000000110000

100111111110100100100 01 | 000111011010100010001 001 | 100110110101000101111 111 | 101101100101100000100101

001101011000110110100001 | 000001011011101110000000 | 011001100011100001101 111 | 101000011001101110111001

111001001111110100000010 | 110011011011101100001001 | 101100001110100011101 111 | 000001010010001111100101

011000001000111111001110 | 000001110101010110000101 | 000110001001110101011011 | 100011001000000001011111

---

n = 25

011100011110000101110100 1 | 110000000110101011100 0010 | 101010110011000011000 0101 | 111110111100101000111 1101

011101110101100111101010 0 | 111001111001111101110 0110 | 000011011111000010 0101011 | 011101100001100011110 0101

000110111000010011010 0111 | 011011100000000111011 0101 | 101101010111110011001 1011 | 000000110001001111000 1010

100011110011100010100 1001 | 111110000101100110 1101000 | 000001011110001001001 0101 | 000101000011010010110 0010

101010011011000000101 0100 | 011010001011100100 0011 10 | 000000010101010111 01101 | 010010101001100110 11011

---

n = 26

101110110000100000010 10001 | 001010011100011000 10011 010 | 011010111010000010 10000 110 | 011110000110000111011 00100

000100110011010000110 01000 | 100100001011100110 00011 011 | 011000000101111100 0100 111 | 001010000000110000 10010 010

000100011100011000100 10011 | 010000101110010010 10000 001 | 011010111101000101 1000 011 | 010110100001100001 10011 000

110111001011100000011 11010 | 111101110001000 11100 0011 101 | 100111010001101000 0110 101 | 101011101101100101 01001 001

010010001111011010 1110010 | 011011111111010001 1110001 | 011001110100000 11111 010 101 | 111101101111011001 11011 110

---

n = 27

011001110000011001001 001011 | 100111011100000 11100 100100 | 110101111011100 11001 111 1010 | 001000110000 11111 010 000 111

101000111111011100 01101001 | 100011001010111 11110 011 00 | 110011010101110101101 11110 | 101111101111011 11010 111 11

010111011000001101110 10000 1 | 010010000010001 00011 110 00 011 | 111100111100000 00101 001 00 | 011100100010110101011 00101 1

100100110100011110111 00000 00 | 001111101010011 01101 100 00 011 | 100001101101001 11011 100 | 110111001101000 10001 1011 100

001011011110001001000 10011 0 | 111110010011011 10111 010 11 | 100100111100100 10011 100 011 | 101100101110101 10101 010 000

---

n = 28

010011101101011001101 01001 10 | 101110111001111 00111 001 10 | 100001110100110111010 0100 001 | 111110000001000001 00000 10100

011100110010101011001 01100 00 | 010000001010100011000 11110 | 000110010010011111001 10010 01 | 111101010100100110001 11010 1

101001001101001011010 11011 0 | 110101011010111101101 01000 | 110001111000100111011 00001 | 001011011010111011110 01010

100001100001110101100 10011 0 | 111100011001101000 11010 101 11 | 010011010010011011 00000 1 | 001000110101101111100 01100 1

001100110000110101110 10001 10 | 011011100111001111001 00111 | 010011001000111101101 01010 0 | 000100010100101101101 11011

João Miguel Matos
Filipe Antão

---

n = 29

100001100100110011100111110110 | 101000000100111011000101000010 | 001101111000111011001001110000 | 011111011000001011001101111111

101000011111001010001101000010 | 000111001100010111110101010000 | 001001001010110110010011010000 | 011000011111011010100111011110

001000001011010001100011101001 | 111101110000100111010010000011 | 111110000111001101101110110010 | 100111011100000110110111010011

001000010010011110001100010000 | 001001110111100001010011111011 | 110011101100000000111111110000 | 000000111010100010110101000011

011000110010100001011010110111 | 011011011011001011001000101 | 101100011010101010000000010010 | 000110110011110110111001111111

---

n = 30

011010001100001111000010001011 | 000010111011001000100000011001 | 000110000111111110101001101101 | 111111001001111001000111100101

010101100010101100001011011111 | 111101011000011100010101000010 | 011001001111110001000011111011 | 100000001110111101001000000101

111011010111011010000111110100 | 001010010111101110110100001111 | 001010100101111011011010000001 | 100101111101101100000111111101

100000010010011111101100101011 | 010100001011111110001111001001 | 011010110100010110100010001111 | 101100010011011000000100010100

010011100010111111010010011011 | 011001001000111011000010000100 | 100000001101001001011000010111 | 010101011001011110101000110100

---

n = 31

001110100010001011011100011111 | 000101111001111001011101011111 | 001110101111011000101000110001 | 111101011001001110111001001011

000110001100001101011000011000 1 | 101001111010101110010000001100 | 110001111111111011110100011101 | 110001010000011001011100100010

110100000001110011000000011101000 | 011100011001110100111010010101 | 111000100001110110110000100001 | 000000011001110010100010011011

010111100010001110101010111010 | 001011010010110101110110100100 | 110100110110100111001010100110 | 111001011111100100010000011000

110101110001011100111110100100 1 | 011100100110010111110000010010 | 111111100001001111000110100011 | 111111110001011011001001011000

---

n = 32

011111001111111010010011100001 00 | 111010101100001010111000111001 10 | 110000010000110010000000000111 1 | 010001000101000001100011101100 10

010010110110000110100001010001 01 | 001000010101011000101100010000 01 | 110001100111001000110010111010 1 | 001111100000011100101101101111 10

011101000011100000110101110010 10 | 111100110111111011010010001101 0 | 000111100010100000001100101010100 | 010100100110110011011011100001 01

101111010001001000101110110001 00 | 110011111010101101001000100011 11 | 111100101100111100010100010001 101 | 110110001111010010000100011101 10

101111111010101000100000111010 11 | 110011101000101001101010011011 00 | 101111010101010000111101111 00 | 100101011100000100000001100000 100

---

n = 33

010100011111100011110001010100011 | 101011101010010111111111110011000 | 111010000011001110011001010000011 | 001111000100111100011100010110110

111011001010000010101001110010 11 | 011110111101010110111011111000011 | 111100000100010100111100101100001 | 100010100010110111011111010010100

011000001011000111101001001011110 | 101110000010110000011101000100111 | 100011100000100110000110101100001 | 001011100000111111000010110110000

100111011110101111010111010110 01 | 101010000000000101001011000100110 | 001011101000000011100001101101100 | 001100111000100010001100110010000

100100100111110011001000111011001 | 000100101011011011101011100100100 | 001111001000010101110101010101011 | 001110000100000000110010110101110

---

n = 34

110110100101001110100100001111101 1 | 000001001111101110010000011001111 | 010011100100110111000010100010100 | 011100101110010101000011010010010 01

101011010000011001100101101100010 01 | 100001001000111001011010010010101 | 101111011111010101110010101000000 | 010001110101001100100011100111111

111000101001100011010000011011001 | 111111011001101000100110101101110 111 | 011001110111111101111111100101000 | 001111101001101001100001010011001 1

100111001100011010100101010010 0001 | 101011111001110001000001011111001 1 | 000001000110111011001001110110110 | 101011001000110010010101110101011 0

100011001010101110010110000100000 0 | 100000011101011001100101100100101 | 100111000111001110001101100111011 1 | 111100101101011000011101010011100 1

---

n = 35

000110001010100000011000001110000 10 | 011010011001101111010001000101001 0 | 001101010110011101010110010010100 | 110000110101001100000100101000001 1

João Miguel Matos
Filipe Antão

1100101101011010100100100001010011 | 1100111101100000101111000011101011 | 0111100010110011101110001010011110 | 1011100001011110111101000000111011

0011100000101001000011011110111011 | 1110100111000101000000001100011001 | 0100100110010110000101011010000111 | 0100101100011110100100100000111110

1101011101010101000010001001111101 | 0111010000000110000100110011101110 | 0111111100000100000111000000011010 | 0001100100100010000010010100101100

1011000011010010010110010110101101 | 0010000110010101001000101110001001 | 0100011011111111010001100001000100 | 1100001001101010011111001101010010

---

n = 36

0001110110010100110100110110001110000 | 1100111000011001100000010100001110000 | 1111111110001110001111100000000011011 | 0111011111001111101000100001001111001

0000101010100111001111101010001100 | 1100111111010001010100000111111111 | 1100010110010110000010001100010100010 | 0001100101001100100100101011001000000

1100110100011000110111001100110100011 | 0101000011110111100100110000011101110 | 1011010111010100100111001110000010000 | 0101101001011101110011001111111010111

1100110101001100011101101100100000000 | 1001100100001011100010101101100101111 | 0000110000000001001100001010111010101 | 1011001011101110011100110111010010111

1000011000110100100110010101101010000 | 1111110100000011100000010111000011101 | 0011011100001100010010101101110111011 | 0101001001111000111101011001011001111

---

n = 37

1101011000011110100111100000001110111 | 1110101001111110011001100001000000100 | 1010010000011000100101110001000101110 | 0010101111101110011011101100001100000

0010100110011001110001000000101011011 | 0010010100110010100011100101000111101 | 1001100111110101101101011100000001 | 1011100101100110001011001001000000

1011100110000110001111100100011000101 | 1011001100000110111010100000110001010 | 1001100101010110011011111110100100111 | 1111100110111000011100111011101100100

1010110010101111010110100000000101101 | 1001011111100101100001011001111001000 | 0101111000101001101100010000011101110 | 0000000001000101001110001010011111101

0101011101101011111001101000110101011 | 1101011111001100010110000011010100110 | 1000011001011000001010110111111111100 | 0000101111101111001111001100010001001

---

n = 38

0101001100000111100111101101110011010101 | 1000011101000000111001110111101011000 | 1000011111110011101101101101010001111 | 0110111111000110011011010001110111011

0111101001101011010000111010111101001 | 0101001101110001110011000101011001101 | 0110111101111011101110011001100011010 | 0111110010001100011111001001000110101

1100000111101011001000110000000111110 | 0100111001101000110100111101000001101 | 0010100110110011001100111100001010110 | 1001101000111000111110101011111011100

0011100010001001110001100101100110011 | 1111011111101001010001011010000110101 | 0111100001001010100010010111010000101 | 1101010000001100010000110100110011011

1000110001000101011010100101110000010 | 0011001010000100110011010110010111010 | 1111100101111110111111100111011110101 | 0011010000001110001110110010011010110

---

n = 39

1010100010101101011010100100011111011111 | 1001000000000110001010011000000100111 | 1001010010000101011100110010101111011 | 1010000110000011100011110100000011011010

1001000110011001011000101000001110111101 | 1010110101101110000100000001010111110 | 0101101111101110111010100101011111110 | 1001101101001010101010111000001100010

1101100000011100001011001000111110111111 | 0010100010000000101001001010111111011001 | 0011111100100010101010101100001101111 | 0001110110000011110101111001111010000100

1100111110110010111111010011101001010 | 0000100011001100011101101000100000001100 | 1101100011101001000100111101111000000 | 1110000101001101110101111001111000010000

1101001000010001001100100011010010011 | 0101101110110100100111100000011101101110 | 1101011110101011000001011110001011111 | 0101011110000101101000110110010011011110

---

n = 40

0101000011011100100011010111101000111111 | 1101101010011110101011100011011010000011 | 0101100110000001100000100100101101001011 | 1110001100011000110111110100110010010100

1001001011101111101100110010000101010111 | 0110100000001110111110101001000000100111 | 0001101101101011110100100101011111010 | 0110111100011110110100100010001011110010

0101011001100100111101110110101111100100 | 1000011100111100110101010111011100111011 | 0101011100000000001101110010000001101010 | 1111100000110110000011001000000100011111

1001001111111111011010001111110010100 | 1100111110000010100000001111000100101001 | 0101010001001001110110000111001001011111 | 1110110101101000010101100110011100001

0110100110000011101010011110001011001010 | 1101000010010111000100001110000101111011 | 1011001010000000001011101101010110000000 | 1100101011010101000000001101000001001100

---

n = 41

0000011111011010111110010111111110100011 | 0100110011111000010010000010111011000011 | 1000110011011111101110111011010111010100111 | 1100111011111110001011111110011101101111

1111010110000100000011011100100001010011000000100100001 | 0010001100100010100101100110000100100001 | 1011011010101100100101011011100010001011 | 0110110110000110101110000111111011111100

1011010110010100011000101011010011101001 | 0001110000001111001110000000101001111101 | 1110001010101100101011110001111001101011 | 0010000100100111100000100110000110000100

0011111010101110111010101100001101100000 | 1110000110000000110011100100011110110101 | 1111100011110111101001000011010101100011 | 1001000100001110000100001010011110010100

0000100001110101000010110110000110010111 | 1101101010110101001011001010111100011000 | 0000000001100100100000010011000000111110 | 0011111111011100110110110000111000111111

João Miguel Matos
Filipe Antão

---

n = 42

11111001101011101100001010111111010000 | 11010010001100100010000110111100010101110 | 11110000111001011100010000110110011011000 | 11110110111100001001010011010011001010011

11011100101101010111011001000100010000111 | 11110101100011101100100000010000000110110 | 11011011100011001011010010111100010111101 | 11010000000110011110001001101111010100000

01011001111000011011001001101011001001010 | 10011001001100010111001111101011010011001 | 01110101101011111111110001011011010001101 | 11010000000000101110001000101100000101000

00000100111001001101010010110001111100101 | 10010010111000011000011100101110000101101111 | 00001000000011001011101110000111111101101 | 01011001101001101110111011111001101111100011

01011110110111010110111101111011010010100 | 01010001101101101111000000000110010011010 | 01101101000010110000110010011111100001001 | 00110000011110000100000011110111000100101

---

n = 43

11111010101001000001011010001101000000011 | 00001001101010000000110011001101000001011 | 01110010011011011011000111011101101111110 | 00011010000010100011010111010011001000011000

00001000001011110001101101101100100001011110 | 10111100100100111000010001001010001001011110 | 00111011000101011001001101010110110101011 | 10111111110100010110011111000010100101010000

10011000001010001011010011001001101001001101 | 10001111010000000011110010110110100100010010 | 10110011100100000100011100101100010001100010 | 10100100000101111001000000010011111111100100

01110111011110110110100111110011101111101101 | 00011111010000010111111001110000110101000101 | 11001001011010110011011011000000001100001010 | 00010010000011001001000011010001111011011111

10111011111100001101111000000110110110000011 | 00110011010001110010000011100010100001100100 | 01001000001110111000101011000100111101101 | 01110011010001010110110100101011010110110100

---

n = 44

01100010011000110110110011011101001111010000 | 11000000011111010010000001000001000000011001 | 01011111001001000101110001001010001001111 | 10100101101101100000110000101000111010000010

00101110000101111000001011100000101000110011 | 11110101010111111001010111001001101110110000 | 01111111010001010111000001110011011110100 | 11101011001000110101111100100101011010010111

01011010001000000011101000101111111010000010 | 01001100001000001000001010010000001100011101 | 11100011001001010110011000110000100000111010 | 10000101000000000111000111110010101101001101

11111111001010001010100100101100010111101100 | 10011000110110010100110111100101111100100000 | 11100010111000100001100010001110100000111011 | 11000111010011111111100000000001011000001100

01100111000001111000110100101110101000111010 | 11101010001101011100001001010101011111110011 | 11010110000001011110011000111001000101000111 | 01111101111101100000111111000000001001101010

---

n = 45

01000011110001101111010001110000010010011011 | 10010111101101010011101100111100110001001 | 11111010101100100100100011101000001001100 | 00010011010001011010011111011111001101010010

11011101111101111001000001010100110111010100 | 10000000101100100010011101101111110001011010 | 10111101110011000110100010011011101101010100 | 01000100101001111111000100101111010110001011010

01001110000100011010000110101111011010010101 | 11010010111010111111100001000111001010101010011 | 10101100001011001110101010000011011001101000 | 10011011011100101010111001001011000110001001

11011101101011111000000110111111011110000101 | 11011000011011101110011010111001010100110111 | 10110010000000010100100001011001001110000101 | 00001011011001111111010111000000111011101111

00100011110001110001101100001001101100011101 | 11101111111011000101110111010011111101011110 | 10010010010110111110110011000010111001010010 | 10011101011001100000001001100101101000011011

---

n = 46

00110111100010001101111110101111010100000001 | 11010010010011010111100101000011010101010111 | 11001011001100001000100110101000101101010001 | 00001011100000011010111011010101101100001110

10010011000010001010100100011010110011101100 | 10011100111011010100011101100111100001010001100 | 01101000001011000111010111100010111001010101 | 01100110110111011100011000110111011010000100

01000100011001001110101011010010011101000011001 | 11001111000100101011000010001110101011101100 | 11111011000110110101011111000110011001101100 | 00000101000011111111001010110101001001100010

10000111000001101001000010001011100000010011110 | 01000010010110110000101010100111110000011110 | 00101010111001111000101110001100100001011111 | 11111010011000101100101001101010000110111001

00100101101100000101001111100111010110101100 | 01001010110110000000111010000011101000011010 | 10110011001011011110000111010000100000010001 | 00101111101111010101111011010000110111100000

---

n = 47

10011101000101010011011000000111000000001110100 | 00000100110010010000011100011011000010010001010 | 00110101011001110110011110000110010111011100 | 00101010110100101100010001100011011111100101010

11001100110111101001111111100000011111100110111 | 00101000010100100111110100101111110011110000111110 | 10011100111111101001010011100001011000000101011 | 01010110000011000011111101101111011111000011001

01101011001010011010101001001111101000100001101 | 10111111100000000000011110101011111111111101110 | 00001000110010000111100110010111000011100101101 | 10100011010011110101101011001011111001110101011

11011001011001111011000111101101010011111111110 | 01111001010000010100100100101110100000101010110 | 10010011110000000111111000001010010110110101011 | 00101000010100000001011001000010001011000100000

11101011110111000010110001101110011100001000010 | 01010000100101001110101101101000011111101100100 | 10100111011010011110010000000000110010000000010 | 11010011110000011111100111001010010010001101

---

n = 48

10000000000101101011000000011001011100000010011100 | 01001110011011100100110000011010111110000110101 | 01010010110111010010000011111010110011101011011 | 01010001100110011111000010000010101101000001000 1

00111110011111111110001110001101001001101101001 | 00100001111001001001001000000000011110110111111 | 01011001100011100010011101110100101000111001100 | 11001100001000011111011110010100111011110100111001001

00110100110100001010011100110000110100100111011 | 01101111001000011001010011100111000110010101010001 | 11011111010110000001100110101110010110101011001 | 00100011010100001000011110000010101100110100010

00000011001001100000101111000001100110000010010 | 11111000001100101010001001010001100101101010001101 | 10110011100111001101111110101010110010000111010 | 01001000011110001111000101111010110001110111010

10000000110100111000100000100011001100110101000 | 11001101101000010010111110001110110001000010110 | 00011011001110010011101101110100000111100000010 | 11000111001000111000101111000111000100011010101

---

n = 49

01100100110111011100001011011101110010100101000 | 10010000100100101100001001001110111011110001111 | 01001101010000011101010000001110111110011000011 | 11101111000001001011000001100010011111001110011010

00110011101011110000111100010101001010000100101001 | 11000110111100011111000110100000101010100001111 | 00010111111001001010111001110110010001000101 | 11101101010111001111100011011111000111010101100

00000111110001001000100000110100110110001011110001 | 11100100101100100100001011001011011011011110111111 | 00110101100010001111111011001001011001100000001110 | 00010001100100111010001011011001110000011100000000

01110000111011000100010111111111111000010110001 | 11001011100000110110110011010011101111011010100 1 | 01111000010010000001011000000000001111001111001 | 01011111111101000111111110101110110001011010101

11101101010111010111001111101000010110001111100 0 | 10001111100010110111000110000110100001010001111 | 11010010100010010100010011000110100001010010100 | 00000010011100001100100011101110001100000000000

103470.h

---

n = 10

0110011110 | 1001111110 | 1110000000 | 1011010100

1011100010 | 1101111001 | 0110111111 | 0011010011

1100110111 | 0101110000 | 1101110011 | 0110000110

0011100001 | 0010000111 | 0000101110 | 0100101100

0100101000 | 0101100110 | 0101010110 | 0011010001

---

n = 11

10001010011 | 11011110100 | 01001110011 | 01010000001

11011101100 | 10001000101 | 00110000101 | 11010100111

11110010011 | 10001110110 | 11111001110 | 10101101110

00110001100 | 01010001110 | 01101000101 | 10000011100

01101011010 | 10001000111 | 11100001011 | 11011111000

---

n = 12

111011100101 | 001101110000 | 110110000111 | 011110110111

010000010101 | 111010010011 | 111001000111 | 101100000011

001001001111 | 101011110111 | 001010001001 | 001010100111

000111101100 | 001011000010 | 001111011110 | 000101111111

101110001011 | 000001000011 | 000101111011 | 100000111101

---

n = 13

0110101100100 | 1010110111101 | 1111010100011 | 1110111110001

1001110111000 | 1010000000011 | 0101101000110 | 0110110100010

1100000001010 | 0100010010000 | 1000010010111 | 1101011100010

1110101101110 | 0001100011011 | 0110000011111 | 0010110101011

0110000010011 | 1001011000101 | 0110111001101 | 0000000001000

---

n = 14

01000100001011 | 11110110101111 | 11101010110110 | 00100110000111

01110111100111 | 01100110111001 | 01001011011110 | 10100011010101

João Miguel Matos
Filipe Antão

00010001100010 | 00101110000100 | 00110111011000 | 11010100011100

11111110000011 | 11100001001100 | 01100000110010 | 01101101101111

11110011101101 | 01100111110011 | 10100000010001 | 00111101010110

---

n = 15

011101111011111 | 010001110111101 | 111110000101111 | 011100001111010

011001000011010 | 101000010110010 | 010000100011001 | 000111100000110

101111110100001 | 100111101100001 | 000010000001100 | 101000001001010

100110000000010 | 111111111101011 | 001111100101000 | 001110010010111

000011000011101 | 100100001001011 | 000100001000010 | 011101100110010

---

n = 16

0100001110000110 | 1110011100011110 | 0001101011110000 | 0010111111100010

1001000111010000 | 0101011111111011 | 1100010000011110 | 1110000001010011

1010001100100111 | 0110111110001100 | 1001101111101101 | 0011101001001001

1111111101110110 | 1101110000101100 | 0010000011110010 | 1000010000010101

1101010111111000 | 0011011000011000 | 0101011001100111 | 0110000111100100

---

n = 17

01110001101000001 | 01010011011100010 | 01111001010000011 | 10101101011101100

11000000111100011 | 01001100101000001 | 11010000111001010 | 0011011000000011

11110111001000101 | 00100100110101011 | 10110011010100101 | 10010100111101011

10110100110010111 | 11000011011011011 | 10000011011101001 | 10001000100001111

11111011011100000 | 11001010100010101 | 01001100101101000 | 11010000011000110

---

n = 18

111010011010101100 | 011110110000100010 | 101000101010011001 | 001000011110000100

010011101011111010 | 000100110100111111 | 110000001101111011 | 111110111110101110

111110011111011111 | 011100100010100100 | 000001001010110000 | 010011010101000000

011011011111001100 | 011000001101001011 | 010000110111011110 | 010111100101100110

000001001000110100 | 000010010101101010 | 111100110101010000 | 110000100010100110

---

n = 19

1111100011110110101 | 1101011001010100000 | 0011101010010011101 | 0010010101100110001

0100111010111111011 | 1111100101110100000 | 0101100000111100100 | 0111010010011010000

1100001010001000010 | 1100011011001110101 | 0110001100001001110 | 0101111101100100101

1110110001001100110 | 1010100101111011110 | 1111001010111110110 | 0000101000110001101

0101100111011110111 | 0000000001111010001 | 0011010010000111000 | 0111111001111010001

---

n = 20

00100001101000100010 | 01110001100111111100 | 10000011001011101111 | 00000011101011000110

01100011101000110110 | 00000101010101001000 | 01000000000011111010 | 00100110111011010110

01111010010110101101 | 11111010100011101111 | 10011110100100000010 | 11000111010000010011

00111100100101100011 | 10111110110011010110 | 10000011111011110010 | 11011110010111011101

10110100001111111100 | 11100110110111110101 | 00101001000101100001 | 10110110111111101111

João Miguel Matos
Filipe Antão

n = 21

001011100011001010001 | 100110101001001110110 | 000110001000001111100 | 001011000110110101010

100101011011111100001 | 000010111000110111001 | 110010110000111011010 | 101010110101000001010

110001000001001011101 | 101001110101111010101 | 001011111100001111100 | 001110100001011111101

110100011011110000001 | 111010100011011010011 | 001011010011001000010 | 010110011110110100110

001100011110110111110 | 010110001101110010000 | 111000111000011111010 | 101101011101111001001

0.038166 | 21

---

n = 22

000000110111111010000 | 111001000110101011101 | 110100010111111111000 | 101100101101100110101 1

110101110010111011001 | 011000000001110010100 1 | 100110011110001111010 1 | 100001011010010011000

110101100010110111111 | 000011011100000000101 0 | 010111000011000111010 | 000001111100101111111

101111000111011101010 | 000011000100101100111 1 | 010101101110101000110 1 | 100100000011010111011

010011011000111000101 | 101000000110100011100 0 | 001010101100110010100 0 | 011111000001111000110 1

---

n = 23

101110100000111011001000 | 111111100001001000000 10 | 001101000010000010000111 | 111010001011101100010 10

101100100011110010011 | 001000111011100001010 1 | 110101010101100101011 10 | 110001110011101100111 00

011110000100000100010100 | 010101100111010000101011 | 101001111011100101111 01 | 000011010001110101010 00

101011111010101101101 | 100101001101100000111 1 | 011111100010110111101 11 | 011111000000011010000 00

001000101110110100011 01 | 111111011001001101010 10 | 011111011100001010000 100 | 010011010001011001100 11

---

n = 24

010000101010110101111010 | 001110001010010110000111 | 011110001011110111001001 | 001111010101111010100 110

100111010000000001011111 | 001110111100011011011 10 | 001111100110010011010110 | 011001110001010000000 010

001011000110101001001110 | 100010010100010110100101 | 111001011101010101111000 | 010110101101011101011011

001011001101010011101 11 | 111101101001111111101011 | 111110110110010000110000 | 110011100111100100010000

100001001101100011101000 | 000001101000110011000000 | 010111011011110010100011 | 100100101110101110101001

---

n = 25

010010101101100111000000 0 | 110100100010111000100011 | 010101000011001011110 0101 | 110101101010111111010011

101110100011111000010110 10 | 000011000011111010101000 0 | 101101010110110010111110 | 011101001100110110000001 0

010110101111011111111110 | 010111010010101010100110 | 110001001111011011110111 | 110100010100110111011001

010011001110111000011011 0 | 111010101011101000001100 0 | 100100111001000000 1001011 | 010101000001000101011000 0

101111011100100000100 1100 | 110010101111011000001010 0 | 111001101010000110001110 0 | 011001110001010010001111 0

---

n = 26

001001010010011100110011100110 | 100011100001011111101111101 | 111001001100101001011000000 | 000100111111101010100 1001

101011010011001100000011001 | 001110100011000010011011 10 | 100011101101100111001010 00 | 110110001100111000001011 10

001101111101101001110011011 | 010110100001011111011 10011 | 110011111101010000111010 0 | 010000011010001011100 10 10

010100101000101101110101 11 | 010100101011011001110000 001 | 010000100101000011100100000 | 010011011101110100111001 01

001001011101001001100011111 | 001111001000100010001111 1011 | 010010101110011010100011 01 | 001111001111001111100110 01

---

n = 27

001010001000100110000100010 | 001001100110111111000111010 | 010100010100100110001010001 | 001100000111100010010001100

João Miguel Matos
Filipe Antão

111001100111010100000100101 | 101100110110001101111001111 | 000000101011010010100000011 | 101010111110011010001110010

001001000011010000101011011 | 100001100000101111100001110 | 100010001001100101010110111 | 011001011010100001001100110

101111100011001100110100101 | 001000011011100010101100100 | 110100110111000100011001111 | 001001110010011011011000110

001010100101001001111100010 | 001000100101111010010110011 | 001011010110011001110110100 | 001010101101010001100001111

---

n = 28

111000101100111011001010000 | 100110100111011010011001100 | 110101011110100110111000000 | 110100010010100000110111110

110101010110011111001010011 | 010111101100111000110110111 | 011100000100011101110010100 1 | 101101100000000101011110001

110010000011000000111101111 | 000100101011100001000001000 | 000100001100000011100001010 | 100110010111000001101000111

100110010100001110110000011 | 100111101110011111011110100 | 011101100110111001110111001 | 111100111010010000100100111

000100110001000101101010000 | 110000000101001100100011000010 | 001110010010010111010101100 | 001000110100000101011110110

---

n = 29

100101000101010000001011001 10 | 101010111001010001111001000 11 | 101001100110100110011111100 10 | 101011101001000001101011000 11

101010100001101010101100 11001 | 111010110110011111100100 11 | 011101000111111110000110 1101 | 110101100011001010010110001 01

101100101101110110111101101 01 | 010000001000001111010000100 100 | 010111011010001101010100011 0 | 110101001101111100001011000 11

100001100111011001100101110 01 | 101010010101110001001010100 | 100010101111001000111010111000 | 000010001011111101110111000 11

000000111101011101000101111 00 | 011000100000000101000001111 10 | 000010110111101010111010010 | 001000111110111010011111100 11

---

n = 30

010011000001010000010111010011 | 011001011101101111101010011 1 | 010100100101000100100000001 110 | 100000000100100000011001111 1010

011010010111111110000100111000 | 111010010001101010100011110000 | 011101010001111100000100010000 | 111111010011001001010000111111

001001010010001010010111111010 | 110110100001011011101101101 11 | 101000000101011000010101100101 | 001000110001111100001011001 0001000

110111000010001011011000000011 | 111000101010001001100000100001 | 110110011100000100000000101001 | 110100011011110101100010110000

100001010110010011011001011000 | 110111110110100110100101100111 | 001011001111001011100111100010 | 011101001000000111001010011 00

---

n = 31

111010011010111001101101101010 00 | 010111100011010110001000011 0100 | 010011010101110100101011000 0010 | 100101111000010001011000 1000101

000101100000011010010010011111 0 | 001101010001111001000110011 1101 | 010111010110010011101111101101 | 111111100111010001101101110111

001101111100100011001011101110 | 001010100001111010100001001100 | 101111111010001010100010011001 | 010100111010010111100010001 0001

111010011101110111011000000 1100 | 010101101001011010000001101101 | 110111101011010111100010010011 1 | 101110111000111111110100011011

110000000000001111010110000 1100 | 110001111111010001111001010111 | 110000100010101111101010101100 | 110000001100101011001000000 1001

---

n = 32

001010101010101010100111100011 | 111011010001001111010010111001 10 | 101110111111011011000000011110 11 | 110001110010101110111111001100001

101110101110010111100011010100 11 | 111111000111110110111000100010 01 | 101000111010110100101000010001 101 | 110010101111000000011101011110

001010101000000101111010000110 11 | 001011000010011101101111100 1001 | 011000110101101101101101000100 11 | 010010010011110001111010010 0011

000110100101111010001010101001 01 | 001001101111011100110101011000 01 | 010011111101111000111111111101 | 010011110011011000011000011 1101

100000100000000010011101100 11010 | 010110100010111010101110100 11000 | 111001000011100100001011001 11 | 111011101100011101100111111 11000

---

n = 33

100001110111110100010000100111001 | 010101011101100000001111001001 1011 | 101101000101000111100011100101001 | 011010101110000011000000011000100

101111001000010100000010010010111 | 111101110000011111110011101001 011 | 001100011100001000111001001011110 | 011111000010101001001110101111111

111110111111111001110000110001101 | 111000010100101010011100011100001 | 001001000010111010111101111011011 | 100110100101110000011001010 01001001

100110110011110000011100010100001 | 101011101001110001011111001101000 | 101001001111001010100001101000110 | 011011110101011001101010001010 10

001010101100010010100000011100011 | 101000001111010110010001011111 | 000001111101111010011010010111001 | 010010110110011111011010001 0011111

João Miguel Matos
Filipe Antão

---

n = 34

0101000001011111101100101000110010 | 0001000100000011011001011110001111 | 0111111011101001000101010100011011 | 1100010000111011101001100000111110

0011010111000011111010010100000001 | 0101111010111000011110001011100000 | 1001001110100110000101110001000000 | 1110100110001110000010111111000100

0100010101101101110111011100010110 | 1100001111101001000001101110101110 | 1000100010000001001110000111001101 | 0100100001110000100111100101100010

0101100110000011011100011001110111 | 0000000011111101000000110101110011 | 0111011110100101001010111110001101 | 0001001011011010110100101100101000

1010110101101111101001101101111011 | 0000110110110101000001111000110001 | 1111010001010011001110000100091011 | 1011010010100110110010011001111011

---

n = 35

1000000111000011111001111010111010000 | 0111100001011101011000010111001000 | 1101101110011001110011001000101111 10 | 0000000011100001110011011010110010 0

1010101101111100000111001101010110 1 | 0110010000011000110010101000101100 01 | 0010001011110001010010001010100100 1 | 1001100010101001000000100110111011

0101000010100100100101000110110010 1 | 1111001111110110101000011001001010 0 | 0101001000100011110010010111011110 | 1010101011101111001101000100011110 1

1011101111010101001010000100101100 0 | 0011000011001111100111101000000100 1 | 1110000101000101001111010110111010 | 0101001101101001001000100100111100

1011000111101001111101011010001101 0 | 1011100100101000010000110111000011 0 | 0100110110001111100111100111000000 | 0010010000110110001110100011101110

---

n = 36

1010001101000011010010101100011001 01 | 1111000001000010111000111000111011 01 | 0101011110011011010100111001110011 | 0011000100110110010111111011101 1001

1100000000110111100110011111000100 01 | 0100001001110010100001110110100 1101 | 0100011101110101010110101011101100 | 1000010110001100010011000001011100 10

1111110101111010111010011100010110 01 | 0001111000010110011000100000101111 01 | 0000111000010100010000111001010000 10 | 0001111010000101110000110111111100 00

1110101101010010011000101010000111 0 | 1011010000111010110001101101111011 00 | 1111000100010100000101101110001111 | 0111111010000110010000001111000100

1101010010010010001010101001010000 100 | 0111101110001101000001110101111111 | 0000100011010001001101001100014001 | 1111100111011010100111010110010 1101

---

n = 37

1011110000100100100011010011011100011 | 0001011010011000000011101011111001101 | 1100111000101101010111110101001001 00 | 1001111010110011101100001101001101111

1001100100100011000111011000001110100 | 0001011111101010100111011010111101110 | 1100110101011010111110011110100101111 | 0010111010000000101110000110111000100

1100110111010000000101001001100001111 | 1110000100110111011100101001100100010 | 0100001111111101001100101010011101100 | 0110010010000101111110000010110110010 0

1110101111110100010101001001011111000 | 0101011110100000100010100011100100110 | 0101010111000001011011010111001111 0 | 0101001000100101110101011000010110001

1001100101011001000110110110001000000 | 0110100100001000011001010101111001110 | 0100011010101100000100010011100010011 | 0101110011011111111010110101011111101

---

n = 38

0001010010100000011111010001011010101 01 | 0001101101000100111001000100001110110 | 1111101000010001111110000101101100 0100 | 1010011101111110001010010111001100001 0

0011110111101110111000001111100011011 1 | 1001100101000110010100001110111000111 01 | 1111011100011010100111001011000010110 | 1000001010011000110001110110011010011 1

1011011111110110010001011111100011000 0 | 0111100101011001000101110000110000011 | 1101111111100100101010110100101011001 | 1100111110000111001010111111100101000

0110010111101110101100010001110001111 1 | 1011011111101011000101001010101110110 | 0110001001100011111001000100101110010 | 0011001101100100101011011010010011001 0

1011100101011001000100110101110011 10 | 0110110101100100010011101010111111 | 0110010110001000101000000100111000110 0 | 1000000001011100001111101111011101111

---

n = 39

1111011001011110101101010001100010001 1 | 1000001100000111010100110100010100101 10 | 1100100110101100011101010000101111 10 | 0001100101101000111001100111111011111 01

0000011101110100100110001000001000011 00 | 0101111101011111110100111111110110100 | 1000011100010110101111100110101001101 | 0010101111111000000011101011110001101 01

1110000010111010101001101011010010010 1 | 1111101110100000110011001010111011 01 | 1011111100000111001001101010001010110 01 | 0101011001111001111100110100000101111 11

0100011100101100001100010110111111110 10 | 1001000100111100111100110010110000111 0 | 1001110100101001100101100011111110010 | 0110110100111111111001001100011100101 10

0000010011001011011011011011000000010 | 0101010000000010010011111011000101 0 | 1001010101101100110110101101100011000 | 1010100111110111001100100001000101010 1

---

n = 40

0110001000010000010000110001101111010111 | 1100011111101000011110011001101011100000 | 0010100011111101110111010100001011010 10 | 1110100010010010011111011001011010001

João Miguel Matos
Filipe Antão

100101101000000110110100010100110101011 | 010100100001100011101000110101110010110 | 111001001110111100000011001010001011110 | 010010101101000111101000100011111001110

010011111111000111010100010100011111100 | 011011100001010011010110110011000100100 | 000011000110100010101111010010001000011 | 110101111010111010000110001010101101110

010101001100011000000000111010111100110 | 000001110011000000011111101111111111001 | 001001010000110000110000001101000011111 | 000000001000010100010100011000001100000

101100001000011011101001111011000010111 | 011101111101011111011010100000011001100 | 010000010011000101000010011011000011011 | 100000000010110101100100111101011101001

---

n = 41

111011110000010110011000110011010011110 | 101111010010001111110100011110000000110000 | 010110110011010111011000100011111011110 | 101111111001100110111011111100110100001

011111011000011010010110000001101100000011 | 000110110101111000110111111100011010000 | 001011000110101010010110111001010000010110 | 000110100101010101101000111100100111011111

000111110100111111111000100011011100011110 | 001010000101101100001000000001011110111111 | 111101110000000000010011111010001111001110 | 010010101010111110001001001100010110011111

010101001100001011110101011001111000011001 | 010110001101011101100001101110110011000100 | 101010110111111101001000111011101101011101 | 111000001011111111010000010100000100110111

100001101000100011101011011010011111001 | 001001011111101011000100101101011000010 | 101000001100001001000110100111000010100100 | 110001110100001101110010111001010011010101

---

n = 42

110010000111000100010101110001110001110001 | 010111011001111100011010111010001010001011 | 011000000100011101000111111011101111110000 | 110111001011000111000001100010000111010100

000101011110001010100100000010101110000011 | 110010010001100100001111000001011011010110 | 000010101110100101001111011111011100010001 | 100110010111110110100010100000110000101011

111100000000110111110001110100000110111010 | 111000011010001010111100111101101111011001 | 100001100110001000011010111010100001010110 | 100000000001010110011111110101011011001011

010010111011010100001111000011011001110011 | 101011101100111010111010011010000110000110 | 101111010011110000011010101010011100101000 | 010101001011010100110011010010000011111110

110001101101100100011110110110000001011000 | 111011111000101001101111111110001111110001 | 011000100001100000101001011010011100110 | 001100000011101011101011000001010111000000

---

n = 43

101101101010011010001000101000010101000100 | 110110010100001010110011111100100111011001 | 011101110110100011110110001100111010010111

111010010010011000011100000110000100111100 | 011100010011100010110110110010110101010111 | 011101001111001001010011011101010010101 | 000010111101010000111000100001010010010011

110101001001111111110001101110100000000011 | 001011001011000011101010001100000001101110 | 010110010000110000000100000100101011001001100110 | 011010110111010100010010100000010000011

111101000100100000101110101101101100110110 | 100001110101010101110011011101000110111111 | 101101110110101110010111010110010100111101 | 001011001111100111101101101110000011001101

110001100110000001001011000011101101101110 | 010110110101000001000001101100000100100101 | 001000101111101100101110001111010000110100101 | 111000110100101111010001010011111010111101

---

n = 44

111101101111111010100100110100011000001000011 | 001000100111011100101010110001110101101010000 | 110101000100101100001101000100100001100111011 | 111011001100001001110111010100110101011100101

001101000110010100010010101101111010000000111 | 100001110101001100011110001101001100000000100 | 100010110010010111111111011001000010001100100 | 011001000000110111110011110101110100000101100

010110000010011010011101110011010010111010010 | 011100110111011100111110100101001011001 | 010111010110110011011010011010000000111110 | 111100000110010110110001001111100110111001100

010111110111000011000110011100111000011110000 | 111111100110100100100011100001100101010110 | 011111110001101010001010010110000001001 | 010100010110010100001010111110100110001001

001100000110001000010100101111100101101110 | 010101111111010101110011111010010111000010 | 100000101100011100010000111010100111010011 | 000011011101000000110100000100110011000100

---

n = 45

110111100101100101100111010010111001111110101 | 000001100011100010101000100100001111110101010 | 100110110011000000100110110000001001110011100 | 011111010011000111000100100101101000011111100

010110111010111111010011101011010011000010111 | 110000001111010101110011001010011001000000000 | 111001111111001000011100011100001100111010000 | 000111111100001110010011000101111001010101010

000101000001100100010011010000010111100011111 | 111100001111101000011110000000011100001010000 | 010011101000000010100011101011000010010000111 | 011001011011101010001111110100000011010111001

101110001111000010100101000010001111110100101 | 010001011010001001001111100111010000011000 | 111010010110000101000011101111111110001100110 | 110000100011001111001010100010101111010010001

100100110010000001011010110100111100001000101 | 011101110010110001011100111111000100010100001 | 110010100101000011100100101010100011011100001 | 001010111000010011100001101001100110101011111

---

n = 46

111001100011101011011111010010001100101101 0 | 010111011001001111010010011110000100111111 | 001010110010100011100000110111101011100111100 | 101001000101001111010110010000110101010000001

111011001100011010001000001011000000110000100 1 | 000001001000101100010011110010010111010000101001101 | 011011001100000011101000101010000100101000000 | 110010111010101011010001111001100100000000111

110010100000011101011001111101101101111111101 1 | 010100101110110111101001000001011100000111 | 010001000000110111001011111100010000101000000 | 111010000011111010111001101000011001111100111

001000101001100000010110101100000001010100100 | 010001100001101000001001101111110100101000110 1 | 010010000001010100100111001001011000000000 | 011010001111010001000110000000110101000111001 0

011000101011001101111100100000000111001001010 1 | 100000010101010011100000001010101100001001111 10 | 100000110101010000101000001100011101001100 1 | 010010 100111101010101010011001011001111111001

João Miguel Matos
Filipe Antão

---

n = 47

111111011101011000101000111000001010111111000101 | 101011110111111001010101101100111010101111110101 | 110110101011010010011011001111111010010100111 | 011001000011101101011001001111111110110000000001

100101010011001010110111010111000000111010100111100 | 010110111100100110100010011100100111010011100011 | 010000010101101111010000011110101011111101000 | 000111100100001111110111110011000000000011001

010000111100110111011110001111000101000111110 | 111010110011111001011011100100010011010001011 | 111011100011110110100110111011111000010010000 | 100011010011001010110100101001010001101011110

111111011010111111110100100011111010100011101 | 010000111001100100010010010011101001000100010 | 011000100010000010111001011110000011001111101001 | 101010101001001000111111110100101101101000010

100101001101000011010111000111000111011101010 | 111011111111101100000000100010010000111110111101 | 001101110111110011010000101111111010111010011 | 100001110010011010011001111000011111011101001 01

---

n = 48

111011100111001110101101110010110101000100000111 | 001010111100001110110101010001100001110010110000 | 010101111001100001011011010010110100010111100 | 111110011010111101111010111011010111100011100

110011010111010110100111100111000110001101100101 | 010001111101000100000111101011101110010110101010 | 001111011111010000101011000010000111001111110010 | 011110100010001101110100111110010000110101100000

111101101101011011001000111100001110000010011110 | 001011100000010110101111001100001101011001111001 | 001100001100011100010110100101111011010111110111 | 111011011111010101001011011010110101010011001100

100000110001100011110000111101101111010100101011 | 011111000110010100010010101101111010100101101100 | 101111010111001101011011010000101101010110111001 | 001110011111111111000100110011100100101011000

001000001010001101010001110001111101011111101001 | 101010010000010101101001001100110101111000010001 | 011111011001111111001000100111000011101110111 | 001000010011001100011110011111010010111010110

---

n = 49

111110101001010111111000101010100011000011000011 | 010100100101010000000111111010111100111011110110 | 110100000111110110011000111000111010101000001 | 010000101010011011100011100111011110101110000100

100110010001100010000010001010011011100000011000 | 100011001000111101101111001010101010100001010101 | 001010011010110111010000010101110101110101011111 | 000000011101001010110110001101000010010100101

110000000000101010110000000000000111011100101011100 | 000101001010101011100010101001011011010101010110010 | 110010011011110110101111111110000011011111011000 | 111011101100000 100100011001010111010011010011110000

111100110110000001111011011010010101101101010000 | 110111000010110010000011100000111010101000011011 | 111100110010101011111101111001010110010111111011 | 111100110010001110011100000100001010010111110101

001100011011110100010101111101011011010000101010 | 101010110111001001101000011101101110010111010000 | 000000101010100100011010110001011000100011000110 | 011010001001011010001001000001110110110010010110

João Miguel Matos
Filipe Antão