Application of Domino's Tiling Theory to calculate the probability of tatami mat arrangements in rectangular rooms, the number of mats required, and the areas where a full sheet of tatami mat cannot be arranged

Institute: Samsenwittayalai School | Thailand(TH)
Students: Tinnapat Sittisuwan, Saknarin Srima, Saran Laiwisetkul
Supervisor : Mr. Sutthiphan Hiranpruk

Abstract

In Japanese folklore, mats with a ratio of 1:2 are popular. The mat is called Tatami. After the organizers have studied Tatami, the arrangement of tiles in one room can be arranged in many different ways. This made the organizers interested in counting the number of possible ways to arrange the tiles in one room. So we built a computer program that can calculate the things we care about. There are 3 items as follows: 1. The number of methods for arranging the Tatami tiles into the room size 2*n and 3*n. 2. The total number of tiles used in the arrangement and 3. The remaining room space after the arrangement. tile By using knowledge of Domino's Tiling Theorem, Recursive Function, Fibonacci sequence, and computer programming. therefore became "Application of Domino's Tiling Theory to calculate the probability of tatami mat arrangements in rectangular rooms, the number of mats required, and the areas where a full sheet of tatami mat cannot be arranged". The results showed that the programming output showed consistent results for all three experiments: a comparison with the results of a mathematical formula according to Domino's tiling Theorem, a comparison with the results of a mathematical formula of the form. General and comparison with the results of the distribution of all possible number of methods. And part of the program can be used or extended in industrial applications, the cost of tatami mats according to the amount that can be placed in 2*n and 3*n rooms, is used as a reference for work that brings Knowledge of mathematics and computers, as well as being able to apply data and program formats to further developments.

Introduction

Dominoes , the well known board game. It's a tile-based board game that played with the two-sided rectangular domino pieces. the side of the domino piece divided into to two square end. Each end contain the numbers of dots (pips) from 0 to 6 dots. A set of domino contain 28 domino pieces, sometimes called a deck or a pack. How to play dominoes is to blocking them by matching the domino piece that have the same number of dots. We found that the play of the dominoes resemble the process of placing Tatami mats of the Japanese. Then we use the method of Domino Tiling to solve the process of placing the Tatami mats.

Objectives

We are interested in studying the number of possibilities for creating a tatami mat. By arranging according to the Domino tiling theory, we observe that we can apply the Tatami tiling pattern to the theory through coding and building it into an application.

Limitations

Since we configured the room width to have two formats, 2 x n, and 3 x n, we cannot quantify the number of variations of tatami mat arrangements in other styles of rooms. Including displaying the tile arrangement in a custom format (custom), we can't calculate the number of possible ways to arrange it. But we still can find out the remaining space and the number of mats required.

Literature review

- **Tatami Basics and Widely Proportions**

  Tatami, plural tatami**,** or tatamis, rectangular mat used as a floor covering in Japanese houses. It consists of a thick straw base and a soft, finely woven rush cover with cloth borders. A tatami measures approximately 180 by 90 cm (6 by 3 feet) and is about 5 cm (2 inches) thick. In *shinden* and *shoin* domestic architecture, tatami completely cover the floor.

Figure1 Details of Tatami mats (Japanese mat)

- **Fibonacci number**

Fibonacci numbers are used to create technical indicators using a mathematical sequence developed by the Italian mathematician, commonly referred to as "Fibonacci," in the 13th century. The sequence of numbers, starting with zero and one, is created by adding the previous two numbers. For example, the early part of the sequence is 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89,144, 233, 377, and so on. Fibonacci numbers are strongly related to the golden ratio: Binet's formula expresses the nth Fibonacci number in terms of n and the golden ratio, and implies that the ratio of two consecutive Fibonacci numbers tends to the golden ratio as n increases. Fibonacci numbers are also closely related to Lucas numbers, which obey the same recurrence relation and with the Fibonacci numbers form a complementary pair of Lucas sequences.

$$\textit{Fibonacci sequence } 0,1,1,2,3,5,8,13,21,34,55\ldots$$

- **Recursive Function**

A recursive function is a function in code that refers to itself for execution. Recursive functions can be simple or elaborate. They allow for more efficient code writing, for instance, in the listing or compiling of sets of numbers, strings or other variables through a single reiterated process.

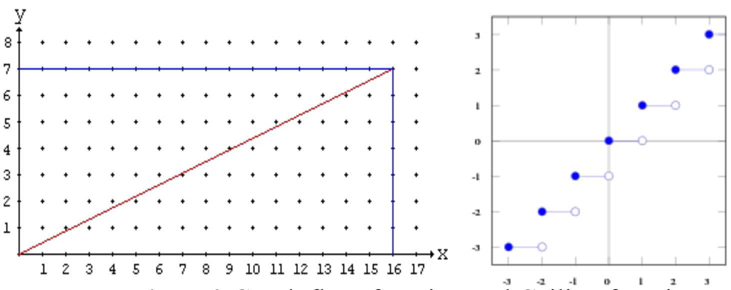
Recursive functions in code often rely on loop setups, where the initial variable is called on multiple times while being altered by the loop. Simple examples of a recursive function include the factorial, where an integer is multiplied by itself while being incrementally lowered. Many other self-referencing functions in a loop could be called recursive functions, for example, where n = n + 1 given an operating range. In addition to simpler recursive functions, programmers and others have come up with much more elaborate functions that also work through principles of recursion. Some, like the Fibonacci sequence, have applications to finance and other areas, where others remain esoteric and largely exclusive to the IT community.

- **Domino's tiling Theorem**

Domino tiling of space in the Euclidean plane is the tessellation of space by dominoes [1 × 2]. A shape formed by the merging of two squares converging edge-to-edge. in the same way It is a perfect match in a point graph formed by placing the vertex in the center of each square of the area. and connect two vertices when they match adjacent squares. and can find the number of arrangements according to the following formula.

$$A_n = 2^{2n^2}\prod_{i=1}^{n}\prod_{j=1}^{n}\left[\cos^2\left(\frac{i\pi}{2n+1}\right)+\cos^2\left(\frac{j\pi}{2n+1}\right)\right] \quad or \quad A_{m,n} = \prod_{j=1}^{m/2}\prod_{k=1}^{n/2}\left[4\cos^2\left(\frac{j\pi}{m+1}\right)+4\cos^2\left(\frac{k\pi}{n+1}\right)\right]$$

- **Floor and Ceiling function**



Method Floor and Ceiling Function : The floor function floor(x) is defined as the function that gives the highest integer less than or equal to x. **The graph of floor(x) is shown beside**. The domain of floor(x) is the set of all real numbers, while the range of floor(x) is the set of all integers. According to the reference image from Desmos, the website for plotting graphs. Similarly, the **ceiling function** maps *x* to the least integer greater than or equal to *x*, denoted ceil(*x*).

Figure2 Graph floor function and Ceiling function

- **Application of tiling**

There is no research that found how to calculate the leftover area in the room, so we applied the tiling theory to an application by coding to find the area remained in the room.

Mathematical knowledge

Floor function and Ceiling function, Recursive function, Summation and double summation $\prod_j^m \prod_k^n$, Basic probability, Basic Fibonacci, Basic Trigonometry, Exponential function, Basic Function.

Computational knowledge

Basic of C programming, nested loop, for loop, Extended math library(math.h), Extended math string (string.h). Fibonacci and recursive function, c Function, application flowchart, if-else condition.

Methods

- **Design the variables and formula**

There are four input variables: *r_w (room width), r_l (room length), t_w (Tatami width) and t_l (Tatami length).* The secondary variable s is *r_area* and *t_area*. Which can find the area according to the equation $r_{area} = r_{width} \times r_{length}$ and $t_{area} = t_{width} \times t_{length}$. You can use scanf() to get the values and printf(). Show values at the end of the program.

- **Design the user interface (UI)**

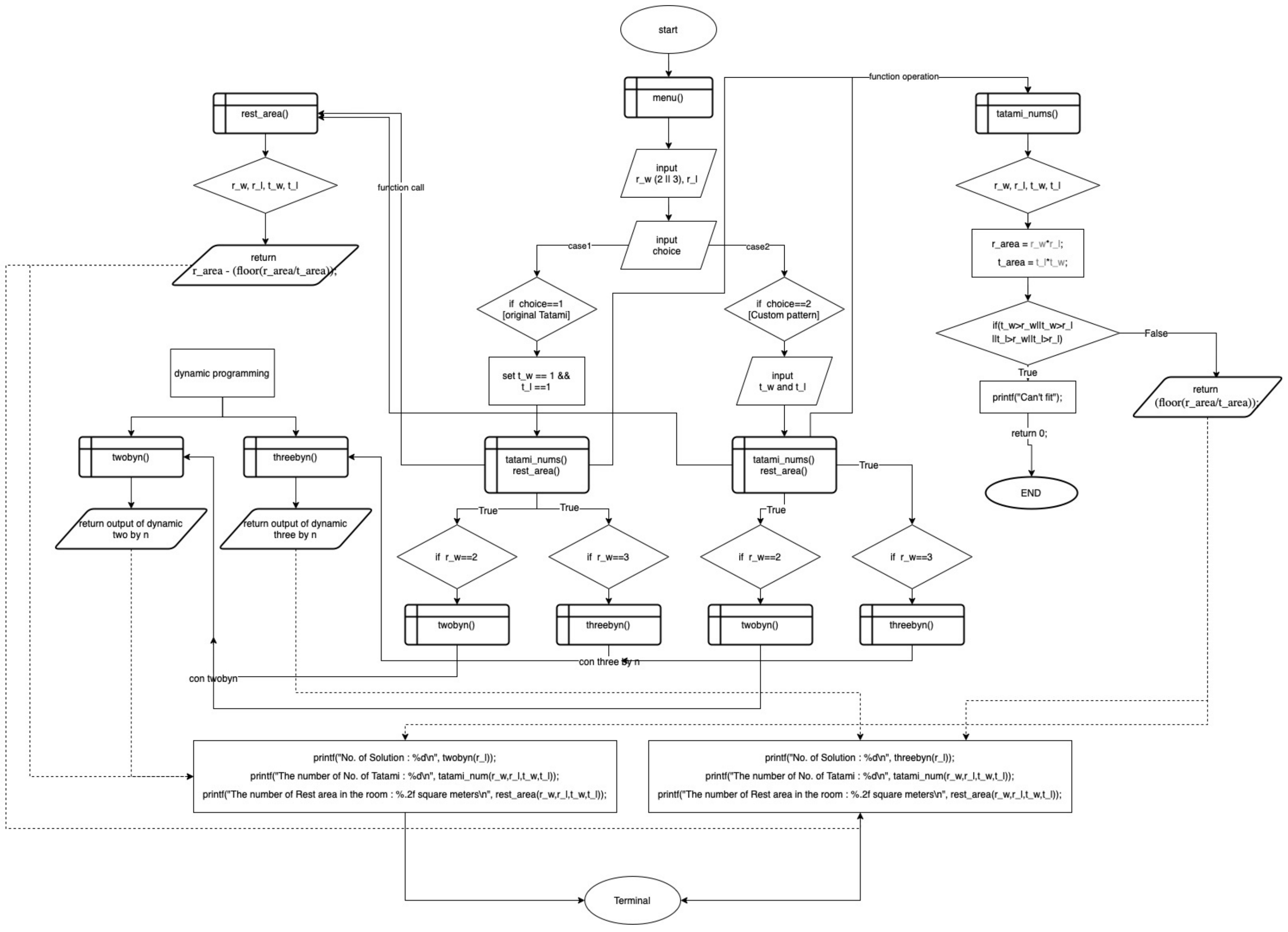
```
------- Welcome to the tatami Calculator -------
Enter the room width (2 or 3) : 2
Enter the room length : 2
**--**--**--**--**
Enter number
1: Tatami Original Size
2: Custom Size
**--**--**--**--**
1
```

```
No. of Solution : 2
The number of No. of Tatami : 2
The number of Rest area in the room : 0.00 square meters
```

User interface design to make it easy to input variable values and output 3 values: No. of solution, No. of Tatami, and left area.

- **Code the program**

build program accordingly Flowchart of Tatami Calculator Programs (C-Language Programming) with Dynamic Programming (2 by n and 3 by n room size) and to make it easier to display values The following functions are created: menu(void), tatami_nums(), rest_area(), twobyn() and threebyn(). All functions return int except menu(void).

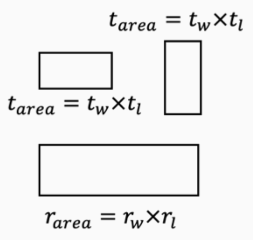


```
printf("No. of Solution : %d\n", twobyn(r_l));
printf("The number of No. of Tatami : %d\n", tatami_num(r_w,r_l,t_w,t_l));
printf("The number of Rest area in the room : %.2f square meters\n", rest_area(r_w,r_l,t_w,t_l));
```

```
printf("No. of Solution : %d\n", threebyn(r_l));
printf("The number of No. of Tatami : %d\n", tatami_num(r_w,r_l,t_w,t_l));
printf("The number of Rest area in the room : %.2f square meters\n", rest_area(r_w,r_l,t_w,t_l));
```

Figure x: *Flowchart of Tatami Calculator Programs*

- **Code the program Tatami Calculator Programs (C-Language Programming)**



$t_{area} = t_w \times t_l$
$t_{area} = t_w \times t_l$
$r_{area} = r_w \times r_l$

```c
#include <stdio.h>
#include <math.h>
#include <string.h>

int r_area, t_area;
int r_w, r_l, t_w, t_l;
int x=1;
void menu(void);

int tatami_num(int r_w, int r_l, int t_w, int t_l){
    r_area = r_w*r_l;
    t_area = t_l*t_w;

    if(t_w>r_w||t_w>r_l||t_l>r_w||t_l>r_l){
        printf("Cant fit");
        return 0;
    }
    else return(floor(r_area/t_area));
}

float rest_area(int r_w, int r_l, int t_w, int t_l){
    return (r_area - t_area*floor(r_area/t_area));
}

int twobyn(int n){
    if (n <= 2)
      return n;
    return twobyn(n - 1) + twobyn(n - 2);
}

int threebyn(int n){
    int A[n + 1], B[n + 1];
    A[0] = 1, A[1] = 0, B[0] = 0, B[1] = 1;
    for (int i = 2; i <= n; i++) {
        A[i] = A[i - 2] + 2 * B[i - 1];
        B[i] = A[i - 1] + B[i - 2];
    }
    return A[n];
}

int main(){
    menu();
    if(r_w == 2){
        printf("No. of Solution : %d\n", twobyn(r_l));
    }
    if(r_w == 3){
        printf("No. of Solution : %d\n", threebyn(r_l));
    }
    printf("The number of No. of Tatami : %d\n", tatami_num(r_w,r_l,t_w,t_l));
    printf("The number of Rest area in the room : %.2f square meters\n",
rest_area(r_w,r_l,t_w,t_l));
    return 0;
}

void menu(void){
    int choice;
    printf("------ Welcome to the tatami Calculator ------\n");
    printf("Enter the room width (2 or 3) : ");
    scanf("%d", &r_w);
    printf("Enter the room length : ");
    scanf("%d", &r_l);
    printf("**--**--**--**--**\nEnter number\n1: Tatami Original Size\n2: Custom
Size\n**--**--**--**--**\n");
    scanf("%d", &choice);
    if(choice == 1){
        t_w = 1;
        t_l = 2;
    }
    if(choice == 2){
        printf("Enter the Tatami width : ");
        scanf("%d", &t_w);
        printf("Enter the Tatami width : ");
        scanf("%d", &t_l);
    }
}
```

```c
int tatami_num(int r_w, int r_l, int t_w, int t_l){
    r_area = r_w*r_l;
    t_area = t_l*t_w;

    if(t_w>r_w||t_w>r_l||t_l>r_w||t_l>r_l){
        printf("Cant fit");
        return 0;
    }
    else return(floor(r_area/t_area));
```

The Tatami_num function is used to find the number of Tatami or Custom tiles in a single room layout.

```c
float rest_area(int r_w, int r_l, int t_w, int t_l){
    return (r_area - t_area*floor(r_area/t_area));
}
    else return(floor(r_area/t_area));
```

The Tatami_num function calculates the remaining area of the placement of Tatami or Custom tiles in a room.

```c
int main(){
    menu();
    if(r_w == 2){
        printf("No. of Solution : %d\n", twobyn(r_l));
    }
    if(r_w == 3){
        printf("No. of Solution : %d\n", threebyn(r_l));
    }
    printf("The number of No. of Tatami : %d\n",
tatami_num(r_w,r_l,t_w,t_l));
    printf("The number of Rest area in the room : %.2f
square meters\n", rest_area(r_w,r_l,t_w,t_l));
    return 0;
}
```

The main function calls other functions, including local variables.

```c
int twobyn(int n){
    if (n <= 2)
      return n;
    return twobyn(n - 1) + twobyn(n - 2);
}

int threebyn(int n){
    int A[n + 1], B[n + 1];
    A[0] = 1, A[1] = 0, B[0] = 0, B[1] = 1;
    for (int i = 2; i <= n; i++) {
        A[i] = A[i - 2] + 2 * B[i - 1];
        B[i] = A[i - 1] + B[i - 2];
    }
    return A[n];
}
```

Function to quantify the arrangement of Tatami and Custom tiles using knowledge of Fibonucci and Recursive Programming. This function is dynamic programming.

∴ and return the value to int.

count(n) = n if n = 1 or n = 2 count(n) = count(n-1) + count(n-2)

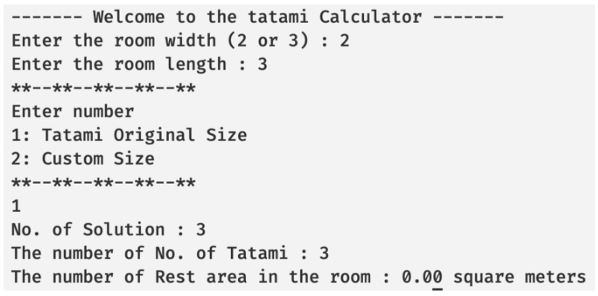
```c
//variable in this program
r_w == room width
r_l == room length
t_w == tatami width
t_l == tatami length
```

There are four input variables: r_w(room width), r_l(room length), t_w(Tatami width) and t_l(Tatami length). You can use scanf() to get the values and printf(). Show values at the end of the program.

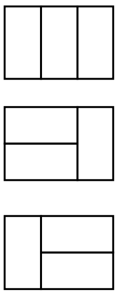
>>Printf() and Scanf() is the stdio output of C.

- **Test the program and verify the result**

*Input case 1(r_w=2, r_l=3, t_w and t_l [original size 1*2])*



```
------- Welcome to the tatami Calculator -------
Enter the room width (2 or 3) : 2
Enter the room length : 3
**--**--**--**--**
Enter number
1: Tatami Original Size
2: Custom Size
**--**--**--**--**
1
No. of Solution : 3
The number of No. of Tatami : 3
The number of Rest area in the room : 0.00 square meters
```

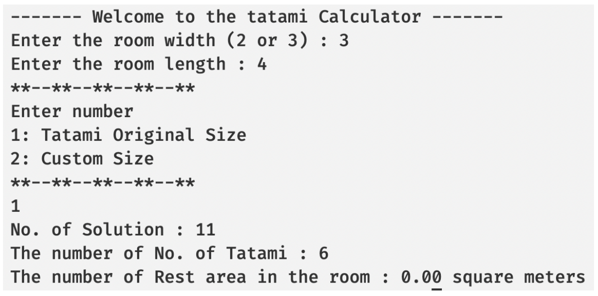


$3\ solutions$

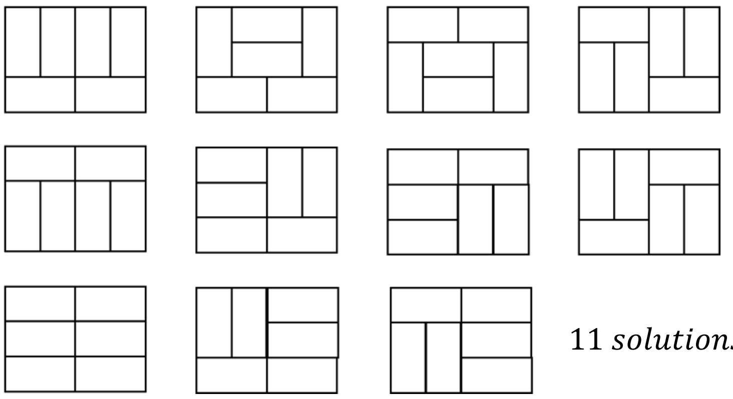
$$f_n = f_{n-1} + f_{n-2} \text{ when } f_1 = 1, f_2 = 2$$

From the fibonucci number and domino's tiling formula is $f_n = f_{n-1} + f_{n-2}$ when $f_1 = 1, f_2 = 2$ In this case the input(n) is 3 so $f_n = f_{n-1} + f_{n-2}$ is $f_3 = f_2 + f_1 = 1 + 2 = 3\ solutions.$

*Input case2(r_w=3, r_l=4, t_w and t_l [original size 1*2])*



```
------- Welcome to the tatami Calculator -------
Enter the room width (2 or 3) : 3
Enter the room length : 4
**--**--**--**--**
Enter number
1: Tatami Original Size
2: Custom Size
**--**--**--**--**
1
No. of Solution : 11
The number of No. of Tatami : 6
The number of Rest area in the room : 0.00 square meters
```

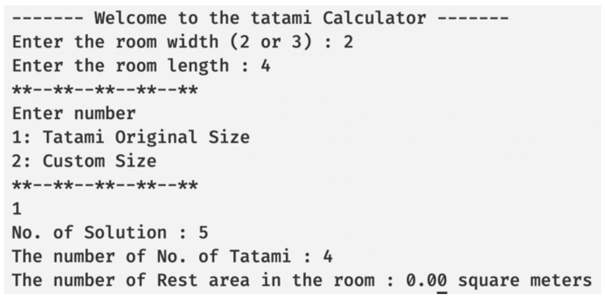


$11\ solutions$

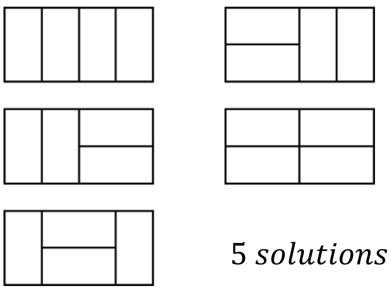
When the room width = 3 the fibonucci number and domino's tiling formula is $f_0 = 0\ and\ g_0 = 0; f_n = f_{n-1} + 2g_{n-1}\ and\ g_n = f_n + g_{n-1}\ and\ f_n = 4f_{n-1} - f_{n-2}$.

$$f_0 = 0\ and\ g_0 = 0; f_n = f_{n-1} + 2g_{n-1}\ and\ g_n = f_n + g_{n-1}\ and\ f_n = 4f_{n-1} - f_{n-2}.$$

*Input case2(r_w=2, r_l=4, t_w and t_l [original size 1*2])*



```
------- Welcome to the tatami Calculator -------
Enter the room width (2 or 3) : 2
Enter the room length : 4
**--**--**--**--**
Enter number
1: Tatami Original Size
2: Custom Size
**--**--**--**--**
1
No. of Solution : 5
The number of No. of Tatami : 4
The number of Rest area in the room : 0.00 square meters
```

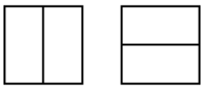


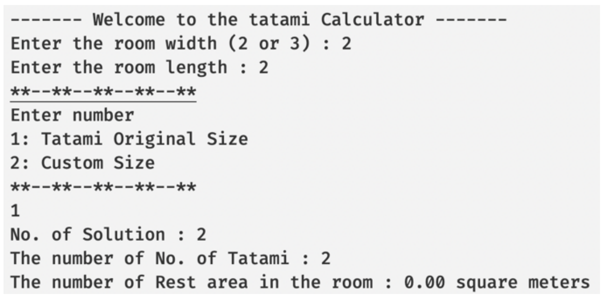
$5\ solutions$

In this case the input(n) is 4 so $f_n = f_{n-1} + f_{n-2}$ is $f_4 = f_3 + f_2 = 3 + 2 = 5\ solutions$. Same as the result from this computer program.

*Input case2(r_w=2, r_l=2, t_w and t_l [original size 1*2])*

$2\ solutions$





```
------- Welcome to the tatami Calculator -------
Enter the room width (2 or 3) : 2
Enter the room length : 2
**--**--**--**--**
Enter number
1: Tatami Original Size
2: Custom Size
**--**--**--**--**
1
No. of Solution : 2
The number of No. of Tatami : 2
The number of Rest area in the room : 0.00 square meters
```

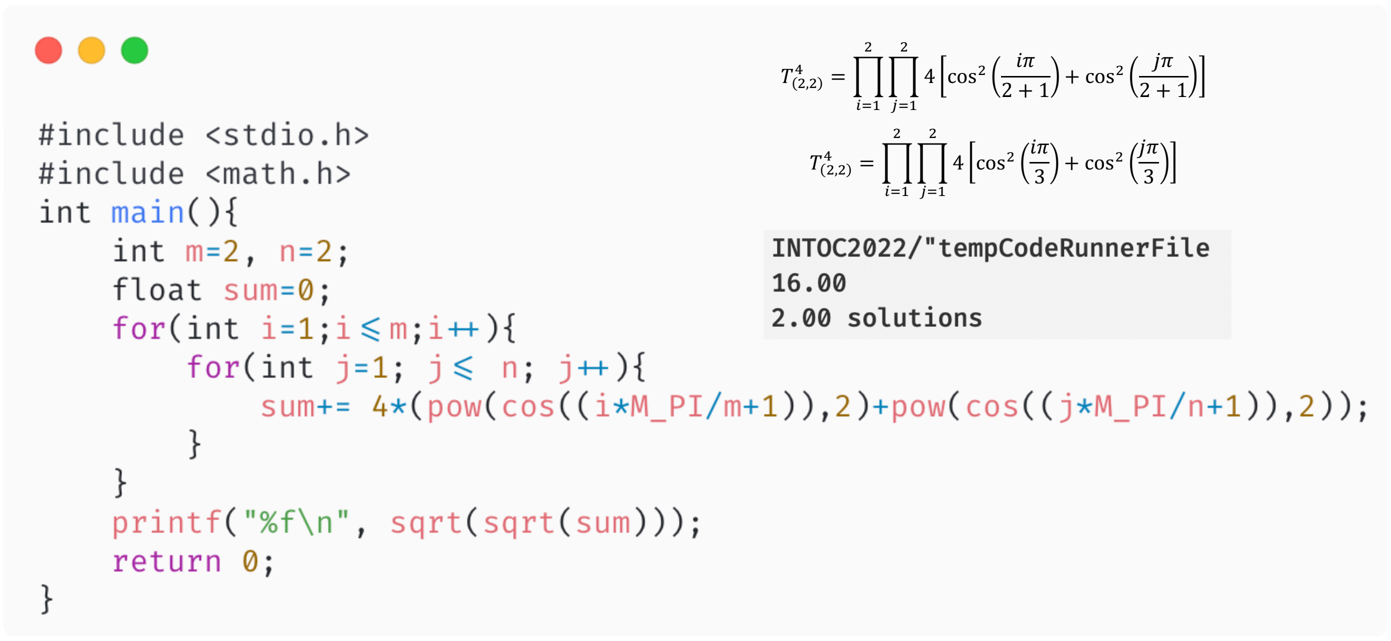
From the fibonucci number and domino's tiling formula is $f_2 = 2$ but we can proof the number of the solutions by using the domino's tiling formula

$$A_n = 2^{2n^2} \prod_{i=1}^{n} \prod_{j=1}^{n} \left[\cos^2\left(\frac{i\pi}{2n+1}\right) + \cos^2\left(\frac{j\pi}{2n+1}\right)\right]$$

When in the form of $2n \times 2n$. So in this case n=1. Then the value n=1 is substituted into the equation.

After substituting the values into the equation, it was found that $A_n = 2^2 \prod_{i=1}^{1}\prod_{j=1}^{1}\left[\cos^2\left(\frac{\pi}{3}\right)+\cos^2\left(\frac{\pi}{3}\right)\right]$. $\prod_{i=1}^{n}\prod_{j=1}^{n} k$, It can have a value of k because it is a sum from n=1 to n=1. Then $A_n = 2^2\left[\cos^2\left(\frac{\pi}{3}\right)+\cos^2\left(\frac{\pi}{3}\right)\right]$. So $A_n = 2^2\left[\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right]$. $A_n = 2^2\left[\left(\frac{1}{2}\right)^2+\left(\frac{1}{2}\right)^2\right] = 2^2\left[\frac{1}{2}\right] = 2\ solutions$. And another formula is the formula for calculating the arrangement of dominos or dimers first obtained by Kasteleyn in 1961. Also known as the formula $T_{(m,n)}$. Calculating the following formula will take quite a long time. The authors therefore write a program to calculate the total number of methods. Using a for loop to calculate sigma.

$$T^4_{(m,n)} = \prod_{i=1}^{n}\prod_{j=1}^{m} 4\left[\cos^2\left(\frac{i\pi}{n+1}\right)+\cos^2\left(\frac{j\pi}{m+1}\right)\right]$$



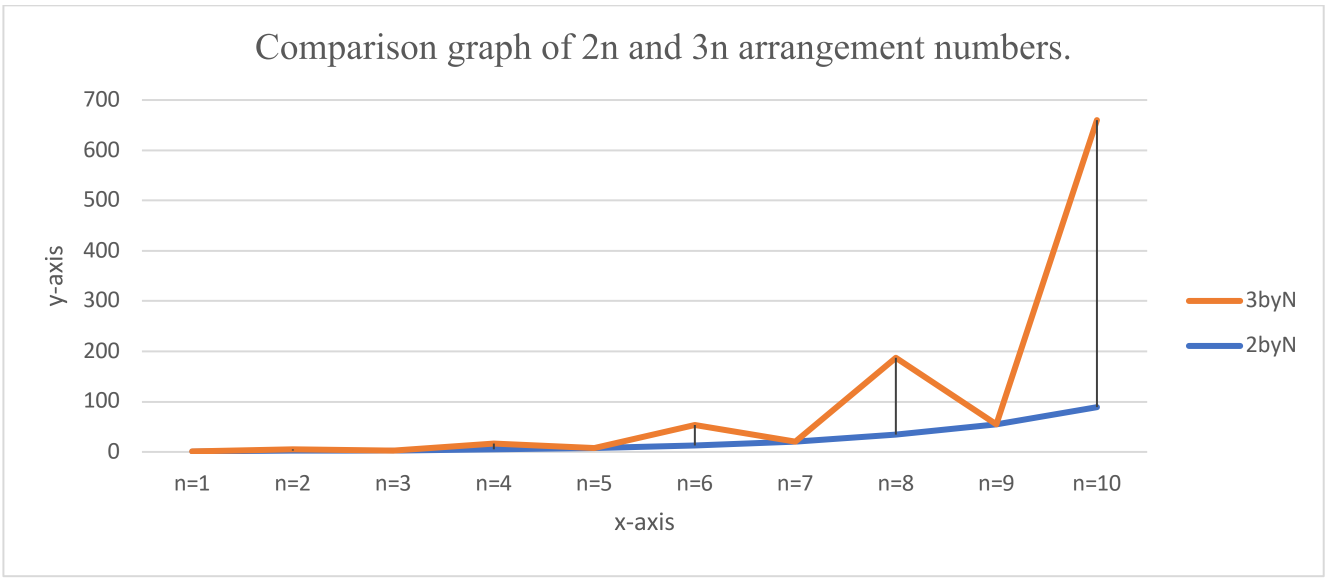
$$T^4_{(2,2)} = \prod_{i=1}^{2}\prod_{j=1}^{2} 4\left[\cos^2\left(\frac{i\pi}{2+1}\right)+\cos^2\left(\frac{j\pi}{2+1}\right)\right]$$

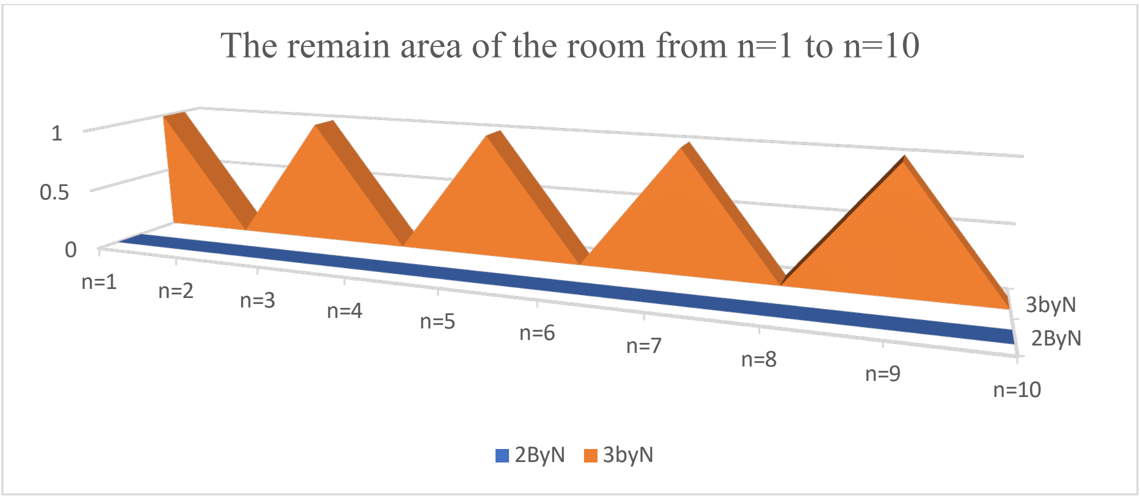
$$T^4_{(2,2)} = \prod_{i=1}^{2}\prod_{j=1}^{2} 4\left[\cos^2\left(\frac{i\pi}{3}\right)+\cos^2\left(\frac{j\pi}{3}\right)\right]$$

```
#include <stdio.h>
#include <math.h>
int main(){
    int m=2, n=2;
    float sum=0;
    for(int i=1;i<=m;i++){
        for(int j=1; j<= n; j++){
            sum+= 4*(pow(cos((i*M_PI/m+1)),2)+pow(cos((j*M_PI/n+1)),2));
        }
    }
    printf("%f\n", sqrt(sqrt(sum)));
    return 0;
}
```

```
INTOC2022/"tempCodeRunnerFile
16.00
2.00 solutions
```

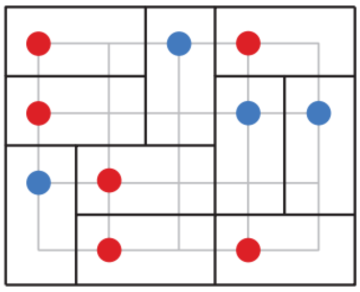
Conclusion
The result of the program for counting the number of tiling methods is equal to the calculation using domino tiling theorems. $A_n = 2^{2n^2}\prod_{i=1}^{n}\prod_{j=1}^{n}\left[\cos^2\left(\frac{i\pi}{2n+1}\right)+\cos^2\left(\frac{j\pi}{2n+1}\right)\right]$, $T^4_{(m,n)} = \prod_{i=1}^{n}\prod_{j=1}^{m} 4\left[\cos^2\left(\frac{i\pi}{n+1}\right)+\cos^2\left(\frac{j\pi}{m+1}\right)\right]$ and list all probability counts of patterns.

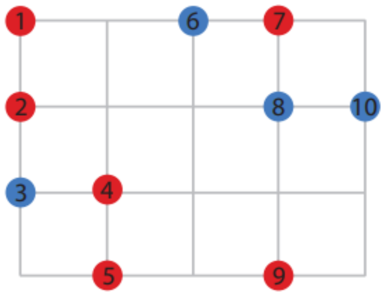




Discussion

When we compare our method with that of Patch Kessler (Arranging Tatami Mats, December 23, 2008), it is noticeable that our algorithms are different. Patch Kessler uses the Dot method through Matlab, where each colored dot indicates a direction. different The program will calculate the possibility of arranging points that represent the orientation of the tatami mats to fit the available space. where the red dot represents the horizontal position. The blue dot represents the vertical position. Our approach is to use programming using knowledge of Fibonucci and recursive functions.





Suggestions

- The cost of arranged tiles in one room can be calculated by the number of pieces of tile.
- This program can be applied in industry.
- This information can be used in other research studies.
- This project can be used as a model or as a case study in applying mathematics to our daily life.

References

- combinatorics - Dominos (2×1 on $2 \times n$ and on $3 \times 2n$). (n.d.). Mathematics Stack Exchange. Retrieved April 1, 2022, from *https://math.stackexchange.com/questions/1317885/dominos-2-times-1-on-2-times-n-and-on-3-times-2n*
- *Random Domino Tilings - Wolfram Demonstrations Project*. (n.d.). Demonstrations.wolfram.com. Retrieved April 2, 2022, from *https://demonstrations.wolfram.com/RandomDominoTilings/*
- Kessler, P. (2008). *Arranging Tatami Mats*. *https://www.mechanicaldust.com/Documents/TatamiMats_01.pdf*
- Ruskey, F., & Woodcock, J. (2009). Counting Fixed-Height Tatami Tilings. *The Electronic Journal of Combinatorics*, *16*(1). *https://doi.org/10.37236/215*
- Erickson, A., & Ruskey, F. (n.d.). *Domino Tatami Covering is NP-complete*. *https://arxiv.org/pdf/1305.6669.pdf*
- Klarner, D., & Pollack, J. (1980). Domino tilings of rectangles with fixed width. *https://doi.org/10.1016/0012-365X(80)90098-9*
- Audibert, P. (2010). *Mathematics for informatics and computer science*. Iste ; Hoboken, Nj.Borys, K. (n.d.). *DOMINO TILING*. Retrieved April 4, 2022, from *http://math.uchicago.edu/~may/REU2015/REUPapers/Borys.pdf*
- Rué, J. (n.d.). *Domino tilings of the Aztec Diamond*. Retrieved April 5, 2022, from *https://upcommons.upc.edu/bitstream/handle/2117/87721/Snapshot-2015-016.pdf*
- Kenyon, R. (2000). Conformal invariance of domino tiling. *https://projecteuclid.org/journals/annals-of-probability/volume-28/issue-2/Conformal-invariance-of-domino-tiling/10.1214/aop/1019160260.full*
- Aamand, A., Abrahamsen, M., Ahle, T., & Rasmussen, P. (2020). *Tiling with Squares and Packing Dominos in Polynomial Time*. *https://thomasahle.com/papers/tiling.pdf*
- *Recursive Functions Computability and Logic*. (n.d.). Retrieved April 7, 2022, from *https://homepages.hass.rpi.edu/heuveb/Teaching/Logic/CompLogic/Web/Presentations/Recursive%20Functions.pdf*
- Alhazov1', A., Morita1, K., & Iwamoto1, C. (n.d.). *A Note on Tatami Tilings*. Retrieved April 9, 2022, from *https://www.kurims.kyoto-u.ac.jp/~kyodo/kokyuroku/contents/pdf/1691-01.pdf*
- Kamtue, S., & Yang, B. (n.d.). *DOMINO-TILING PROBLEM*. Retrieved April 11, 2022, from *https://math.mit.edu/research/undergraduate/spur/documents/2013Kamtue.pdf*