# Report of Group Project for Tangram Pieces Matching and Recognition

Subject Name: Fundamentals of Artificial Intelligence

Teacher: Associate Professor Tsang

Class: D2

Group Number: 05

Group Member:

> Wang Hongbo 18098532-I011-0018
>
> Li Yichu 18098533-I011-0103

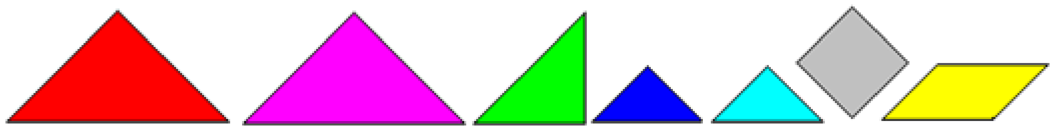
Date: 2021/05/13

## Introduction

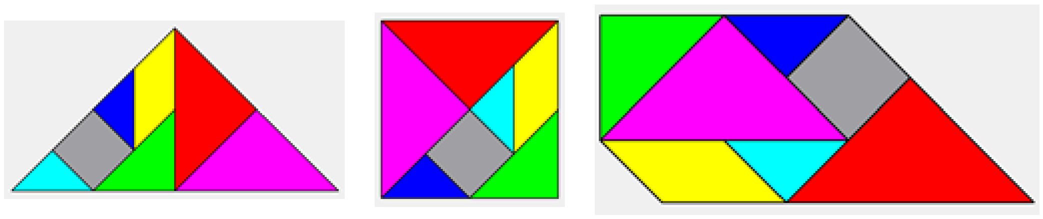
Tangram is an ancient and classic game that most of us have played before. However, in the past, we just use mind and hands to match figures. Now, we can use the relevant knowledge of artificial intelligence to let the computer automatically match the graphics. Our group used four AI search method to solve the tangram problem independently, two uninformed search (DFS, BFS) and two informed search (Greedy search, A* search). By using these search method, we can get the matching method of the goal figure in a very short time (may less than 0.2s).

## Problem Formulation of Tangram Problem

**Initial state**:



**Goal state**:



- States: pieces configurations
- Initial state: scattered pieces
- Operators: moves of the unused pieces
- Goal: assemble to the target figure
- Type of the problem: configuration search

## Solved Problems

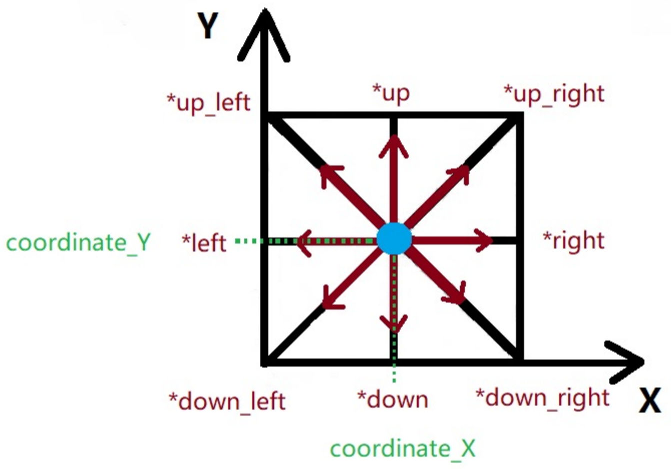
In the process of completing the project, our group member had met some tricky problems and solved them one by one eventually.

- How to record the thirteen convex figures

  The first question we met is how to record the thirteen convex figures that we want to match. Our solution is to record the coordinate information of all points contained in the thirteen convex figures. First, we store the vertex coordinates of the convex figures, then, we find all the points contained in the thirteen convex figures by using mathematical formula. In this case, we have already find all the coordinate information of all points contained in the thirteen convex figures. Then, we need to determine the connection relationship of all the points contained in the thirteen convex figures. Because, each convex figure can be divided into 32 half squared areas. Although a square can have two diagonals to bisect the square, there is only one possible way to divide the square into two half squared areas. Thus, this is also a problem we need to solve. We carefully observed these convex figures and found that, at any two adjacent points, one point can connect to at most three points while the other one can connect to at most one point. So, we can traverse every point contained in the convex figures and use a structure to store the connection relationship and other useful information of each point:

```
struct Point
{
    int coordinate_X;
    int coordinate_Y;
    Point *up;
    Point *down;
    Point *left;
    Point *right;
    Point *up_left;
    Point *up_right;
    Point *down_left;
    Point *down_right;
    bool arrived;
};
```

Each point has 8 pointers to point to its up, down, left, right, upper left, upper right, lower left and lower right, like this:



Finally, we can know all the useful information of the thirteen convex figures.

- How to let the seven tangram pieces to match the target figure

  After we calculate and confirm the order of traverse coordinates, we are going to match the target figure by using the seven tangram pieces. In this process, we use the current coordinate by traversed as a specific vertex of the one tangram piece that needs to be matched. In each coordinate and each piece, it can rotate to form four cases. But how can we let computer to know under what circumstances can the piece be matched successfully? That is also a tricky question that we must solve. Our solution is to match the vertex points first. If all the vertex points of the current tangram piece of the exact rotation degree can be found in the target figure, it also check whether all the inner connection relationships can be found in the target figure. If this condition is also satisfied, we can put the tangram piece at this place and continue to match the target figure by using the next tangram piece. We use the backtracking algorithm to execute recursively. If the seven-layer recursion of the tangram is executed smoothly, we can obtain a feasible solution.

- How to avoid overlapping of several tangrams

  After matching the first tangram, we put the tangram piece at this place and continue to match by using other tangram pieces. We need to avoid the overlapping of tangrams. Otherwise, the final solution cannot be correct because this does not meet the rules. At the beginning, we decided to delete the inner points of the current tangram piece if it successfully matched. However, after we simulated several situations of the matching procedure,  we found that, under some circumstances, the overlapping situation cannot be fully avoided. Because, the vertices on the edge of the tangram can still be used and overlapping can still happens although the possibility has reduced to a certain extent. Then, we consider to delete the inner connection relationship of the current tangram piece after the current tangram piece be matched at the convex figure successfully. That means, we only keep the outer edge's connection relationship and delete the inner points' connection relationship. Then, the overlapping of several tangrams can be fully avoided.

## Search Algorithms Analysis

Our group used two uninformed search (DFS, BFS) and two informed search (Greedy search, A* search) to search the feasible solutions. Now, we will introduce and analysis them.

## DFS and BFS

We used DFS and BFS search method to calculate the order of traverse coordinates, which is necessary because we must use a certain rule to traverse the points of entire target figure to match tangram pieces. While the order of traverse can affect the efficiency of search.

In both search method, we start our search from the most bottom left point. In DFS method, we define the deepest node is located at the most right point. If the x-axis coordinates are the same, first find the point with the largest y-axis. In BFS method, also, for each point, it search the left side points at the same time, then, the next point does the same operation until traverse all the points in the target figure.

### Properties of DFS

**Completeness**: Yes. The order is reached if it exists. Because the depth level is limited according to the thirteen convex figures. It is worth to mention that in some other circumstance of using DFS, the answer may be "No", because infinite loops can occur.

**Optimality**: Yes. In the tangram problem, all feasible solutions are optimal solutions.

**Space Complexity**:

$$O(3d).$$

3 is the maximum branching factor and "d" means the maximum depth of the points.

**Time Complexity**:

$$O(d^3)$$

3 is the maximum branching factor and "d" means the maximum depth of the points.

### Properties of BFS

**Completeness**: Yes. The order is reached if it exists. Because the point can go to all the adjacent points, and those adjacent points can also go to all their adjacent points. Then, by using this method, all the points can be reached if it exists.

**Optimality**: Yes. In the tangram problem, all feasible solutions are optimal solutions.

**Space Complexity**:

$$O(d^3).$$

3 is the maximum branching factor. BFS will store all the traversed points in the memory. "d" means the maximum depth of the points.

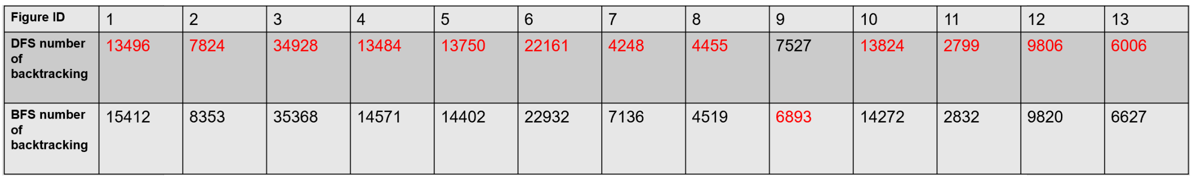
**Time Complexity**:

$$O(3^d)$$

3 is the maximum branching factor and "d" means the maximum depth of the points.

### Comparison of DFS and BFS

Although the time complexity of DFS and BFS is on the same level, there are some difference between those two uninformed search method. We tested those two search method and evaluate the efficiency by using the number of backtracking. And get the result as below:



| Figure ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DFS number of backtracking | 13496 | 7824 | 34928 | 13484 | 13750 | 22161 | 4248 | 4455 | 7527 | 13824 | 2799 | 9806 | 6006 |
| BFS number of backtracking | 15412 | 8353 | 35368 | 14571 | 14402 | 22932 | 7136 | 4519 | 6893 | 14272 | 2832 | 9820 | 6627 |

For the thirteen convex figure, DFS's average number of backtracking is 11869.85 and BFS's average number of backtracking: 12549.00.

In conclusion, although the time complexity of DFS and BFS is on the same level, DFS is 5.75% faster than BFS. Also, when we talk about space complexity, DFS is still better than BFS. So, when using informed search to solve problems, it is more recommended to use DFS.

## Greedy Search

Greedy search is the first informed search method of our project. The traverse order is based on DFS, while the efficiency is faster than DFS. Because the idea of greedy search is that the node seems to be the closest to the goal is searched first. In the tangram problem, the policy of "using the node that seems to be the closest to the goal" is to give priority to finding the tangram piece with the largest area. After the process of matching, the unsolved space is minimized. Unlike the uninformed search that we introduced before, it can wisely choose the order of pattern matching, in this case, the efficiency can be improved.

### Properties of Greedy Search

**Completeness**: Yes. The solution is reached if it exists. Because greedy search can only change the search solution order of the seven tangram pieces. It can finally get all the solution after the backtrack is fully finished. It is also needs to mention that, the answer of completeness may be "No" in some other problems such as the shortest path problem by using greedy search, because the nodes that seem to be the best choices can lead to cycles. Then, they will loop forever.

**Optimality**: Yes. In the tangram problem, all feasible solutions are optimal solutions. That does not means the answer of optimality can always be "Yes". Greedy search in some other problems cannot guarantee to be the best solution. Because the evaluation function disregards the cost of the path built so far.

**Space Complexity**:

$$O(7a).$$

"a" represents the total solutions of the specific convex figure, the range is between 10 to 1000. 7 means the positions of the seven tangram pieces.

**Time Complexity**:

$$O(b^7)$$

"b" represents the maximum branching factor and 7 is the maximum depth of the state space.

**Evaluation Function**:

$$f(n) = h(n) = S_(RemainingArea)$$

## Comparison of Greedy Search and DFS

Greedy search has made improvements on the basis of DFS. After tested the thirteen convex figures, we surprisingly found the efficiency has improved dramatically.

| Figure ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| DFS Execution Time | 901.86 | 465.54 | 2307.03 | 879.40 | 923.13 | 1487.00 | 490.95 | 282.54 | 437.64 | 895.13 | 167.21 | 842.34 | 355.13 |
| Greedy Search Execution Time | 86.85 | 47.49 | 300.04 | 86.86 | 88.26 | 214.96 | 58.44 | 85.50 | 79.70 | 152.44 | 20.27 | 433.53 | 41.10 |

The average execution time of the thirteen figures by using DFS is 802.68 ms (0.8027 second) and the average execution time of the thirteen figures by using greedy search is 130.42 ms (0.1304 second). So, the greedy search is 615.46% faster than DFS. So, using greedy search can dramatically improve the efficiency compare with uninformed search, because greedy search can choose the next node wisely, instead of choose it with no order.

## A* Search

During the process of searching, we can judge whether this configuration can finally get to the result. If it cannot find a possible solution after this operation, it will give up at this moment. Also, on the basis of greedy search, although the order of pattern matching is chosen by the size of tangram pieces, the size can be the same for the three tangram pieces: MT, SQ and PA, all are combined with four small triangles. Then, how can we determine the order of the three tangram pieces? We can build mathematical models for the three tangram pieces. We can know that, for the same sized tangram pieces, the longer the perimeter and the more edges there are, the more likely the other tangram pieces that may be matched, then the more likely the state will not go backtrack. Then, we can finally give the order for the three tangram pieces: PA is the first, MT is the second and SQ is the last of the three tangram pieces.

### Properties of A* Search

**Completeness**: Yes. The solution is reached if it exists. Because A* search can only change the search solution order of the seven tangram pieces. It can finally get all the solution after the backtrack is fully finished.

**Optimality**: Yes. In the tangram problem, all feasible solutions are optimal solutions.

**Space Complexity**:

$$O(7a).$$

"a" represents the total solutions of the specific convex figure, the range is between 8 to 112. 7 means the positions of the seven tangram pieces.

**Time Complexity**:

$$O(b^7)$$

"b" represents the maximum branching factor and 7 is the maximum depth of the state space.

**Evaluation Function**:

$$y(n) = I_(InvalidElementPosition) \times \infty - C - E - S_(AreaOfCurrentPiece)$$

C means the perimeter of the tangram pieces and E means the edges of the tangram pieces. When there is no way to find any possible solution under this state, it will give up instantly.

### Comparison of A* Search and Greedy Search

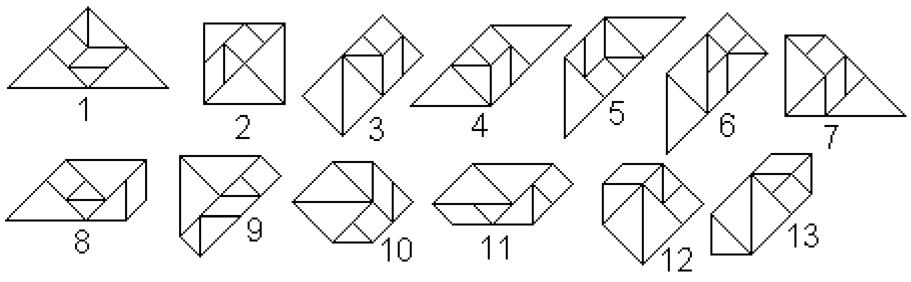
A* search has made improvements on the basis of greedy search. The execution efficiency can be improved.

| Figure ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Greedy Search Execution Time | 86.85 | 47.49 | 300.04 | 86.86 | 88.26 | 214.96 | 58.44 | 85.50 | 79.70 | 152.44 | 20.27 | 433.53 | 41.10 |
| A* Search Execution Time | 71.75 | 44.06 | 253.95 | 72.80 | 70.40 | 184.85 | 50.18 | 80.20 | 71.03 | 136.44 | 18.16 | 409.62 | 38.50 |

The average execution time of the thirteen figures by using greedy search is 130.42 ms (0.1304 second) and the average execution time of the thirteen figures by using A* search is 115.53 ms (0.1155 second). So, the A* search is 12.89% faster than greedy search. So, using A* search can improve the efficiency compare with greedy  search. Compare with greedy search, A* search is more recommended in the tangram problem.

## Number of Goal States of Each Convex Figures

After we finished the project, we run our program one by one and record the number of goal states of each convex figure. The figure ID is according to the picture of our project description as below:



The number of goal states of each convex figures varies greatly, the range is from the minimum 8 to the maximum 112. The specific number of goal states of each convex figures is show below:

| Figure ID | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Number of Goal States | 24 | 16 | 72 | 24 | 24 | 60 | 20 | 36 | 32 | 48 | 8 | 112 | 16 |

## Contributions

The project is written by Wang Hongbo and Li Yichu. Both group member designed  the project framework and implementation method, including the solved problems we mentioned above. The matching procedure, DFS and BFS and greedy search is written by Wang Hongbo. A* search, backtrack algorithm, UI design, report and user manual is written by Li Yichu. Both of them own the copyright and the right of explanation.

# Conclusion

## Strengths

- The execution efficiency is high. By using A* search, the most efficient search method, the average execution time for the thirteen convex figures is 115.53 ms (0.1155 second).
- The idea of the project is very clear. If you want, you can let the tangram to match any figure as you want by using the seven tangram pieces without changing too much code.
- By using backtrack algorithm, the space complexity is pretty small. That means, running this program will not use too many memory space.

## Weakness

- Although the space complexity is pretty small, we are temporarily unable to support multiple search at one time. That means, if user want to search another figure after the first search. He/She needs to restart the program. We believe the remaining bug is caused because of unsuccessfully freed memory. We tried to fix this bug, however, our ability of using C++ language is limited and unable to fix this problem temporarily. This is a pity for us and this is also the largest remaining problem of our project in our eyes.

## Thoughts

During the completion of this project, we understood better about AI search method and improved our programming ability. We also reviewed some of the techniques we learned in class, such as problem formulation, algorithm analysis and backtrack algorithm. Also, we have also encountered some problems, which may not be explained in class or explained in detail. In this case, we learned to use google to expand our horizon. We realized that knowledge is not only limited to what the teacher said in class and we improved our problem-solving ability during this process. Tangram, we used to play before, but we never thought about solving this problem by using algorithms. This project let us to learn to use code and algorithms to solve real-life problems. In conclusion, we can say that completing this project helped us a lot.