Summary on application of knowledge in this course

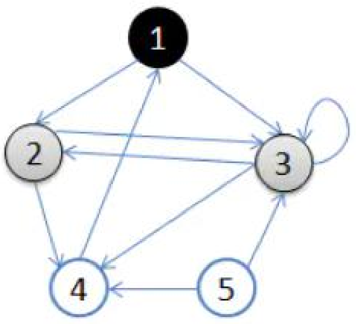
1. Data structure

In Search.java file

(1) Graph:

Think of the 40*24 square pieces in the map as nodes, node connect each node to form the final graph.

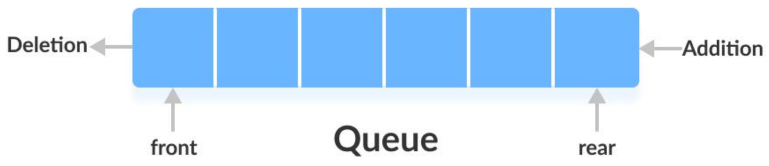
we create nodes and associations between node and node by SearchNode class, and graph-based we use bfs traversal algorithms. And we use BFS traversal algorithm based on the graph.



(2) Queue:

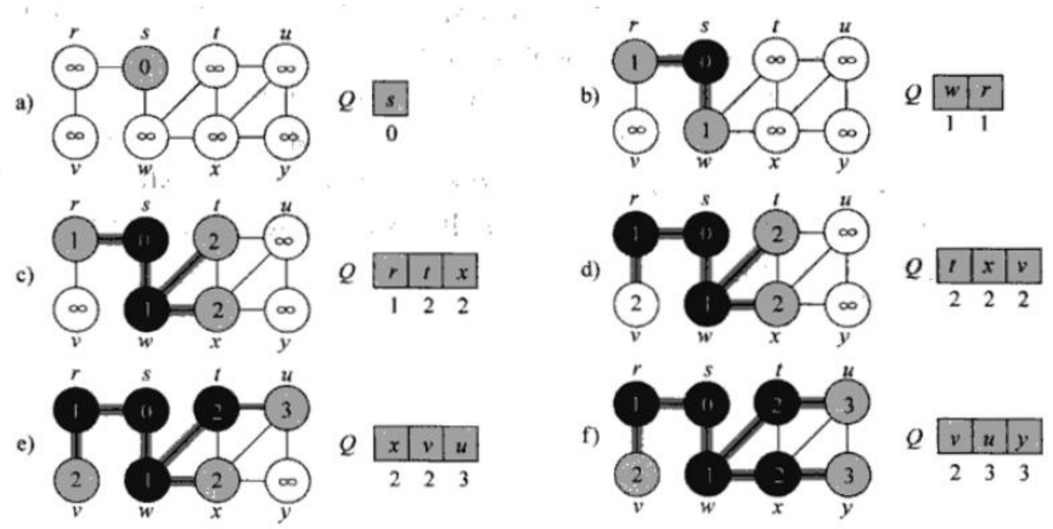
The queue is a special linear structure. It only allows delete operations at the front of the table (front) and inserts at the back of the table (rear). The end that performs the insertion operation is called the end of the team, and the end that performs the deletion operation is called the head of the team. Therefore, the queue is also called the "first in first out" (FIFO-first in first out) linear table. And the LinkedList class implements the Queue interface.

In bfs() method of Search.java, we use “Queue” to store successors found each time.



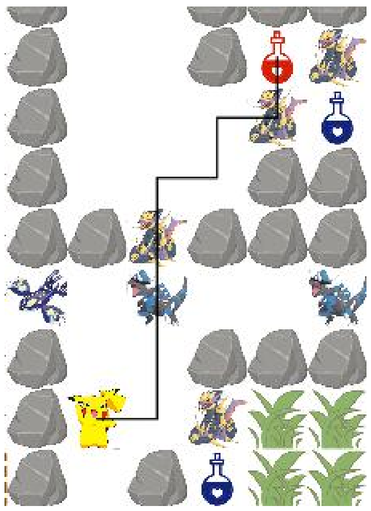
2. Algorithm

Consider each position in map as a node, and treat it as a graph, use the Breadth First Search to find a goal node. In Search.java file, the bfs() method uses the "Breadth First Search"(explore all possible options recursively at each step) algorithm to find the shortest path to the target.



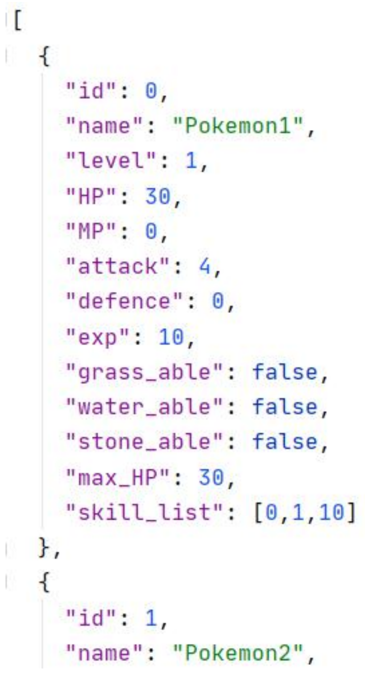
The child nodes obtained by expanding the nodes are added to a first-in-first-out queue. The nodes that have already been expand are placed in explored (set type) containers.

In GUI.java, we create "search" button, and press it can use bfs()method of Search.java to find the shortest path to the target. Here is the actual effect in following picture.



3. Performance

Pokemon.json and Enemy.json files



```
[
  {
    "id": 0,
    "name": "Pokemon1",
    "level": 1,
    "HP": 30,
    "MP": 0,
    "attack": 4,
    "defence": 0,
    "exp": 10,
    "grass_able": false,
    "water_able": false,
    "stone_able": false,
    "max_HP": 30,
    "skill_list": [0,1,10]
  },
  {
    "id": 1,
    "name": "Pokemon2",
```

JSON Is a data exchange format. Use a text format completely independent of the programming language to store and represent data. A concise and clear hierarchy makes json an ideal data exchange language. Easy to read and write, but also easy to machine analysis and generation, and effectively improve the efficiency of network transmission.

Therefore, we use JSON, this kind of open standard for data format, to store initial attributes data of Enemy and Pokemon, and make them become the Persistent data. In addition, some loadFromJson() methods in GUI.java we wrote in order to gain Enemy or Pokemon data from .json file.

4. Design by contract

jml(): add comments to the java code to determine what the method performs and describe the expected functionality of the method.

For example, in GUI.java:

```
/*@ requires id>=0 @*/
public void page3_initial(int enemy_id,int[] beforeBattlePos) {
```

We use “requires” specify the precondition for method.