CodeVS Rules

April 27, 2012

1 Outline of Rules

The content of the problem is modeled on a tower defense game.

The game's objective is to build towers and repulse the enemies shown on the map, so that they do not make it to the defended squares. Contestants will create programs to determine, in place of players, how to build towers to defend the defended squares efficiently under a variety of conditions.

1.1 Map

The map has a rectangular shape, filled with squares. There are four types of squares on the map: empty squares, obstacle squares, enemy appearance squares, and defended squares. Of these, towers can be built on empty squares only. In addition, enemies can move on empty squares, enemy appearance squares, and defended squares only. Contestants will be assigned map sizes and layouts in advance. On a single map, enemies will appear in groups over several separate times. The cycle of repulsing one of these groups of enemies is referred to as a "level." Maps and levels are referred to using the format in the example "5-2," which indicates the second level of the fifth map.

1.2 Enemies

At predetermined times, enemies will appear in enemy appearance squares, targeting defended squares. If an enemy reaches a defended square as part of an attack on a tower before the enemy's life reaches zero, then the player's life decreases by one. The times at which enemies appear, their lives, and their speed of movement will be assigned in advance.

1.3 Towers

Players can spend money to build towers during the times between levels. Towers fire at the enemies they are able to attack. Once a tower has attacked, it cannot do so again for a certain period of time while it recharges. Players can build three types of towers, each different in terms such as its attacking strength, the time it takes to recharge, and its effect on enemies.

In addition, through strengthening, each tower can improve its capabilities up to four levels. The range, attacking power, time required to recharge, and presence or absence of special effects are assigned to each tower in advance.

1.4 Money, Lives

At the start of the game, the player is given a certain amount of money that he or she can use to build towers. While the player also may choose not to build a tower, if the defense provided by towers is too weak then the enemies will reach the defended square and the player will lose a life. If the player makes decisions that result in a negative balance of money or his or her life falls to zero or below, the game is over. Players need to think about arranging towers so that they can defeat the enemies efficiently using as little money as possible.

2 Flow of the Contest

2.1 Through the Submittal of Results

Players will use a client program provided by the competition to upload to a server the outputs calculated by their programs. Once a contestant has been authorized through account registration, the command for execution of the program prepared using the client will be designated. Since in each competition attempt this command will be executed and the information needed in decision-making will flow into the program's standard input in a predetermined format, the program must output its decisions on how to build towers based on such information to the standard output in the predetermined format. There is no limit on number of attempts, so that contestants can attempt the problem as many times as they like until the allotted time runs out. They also can view replays to see the details of the results they have submitted to that point in time. Contestants will aim to get better results by improving their programs further based on such replays.

Program standard inputs will be sent individually for each level — that is, each time a group of enemies comes. The client will stand by until a program standard output response comes. When the conditions for ending the game, given below, are satisfied, then interaction with the program will end and the results to that point will be uploaded together to the server and then updated as needed. Separately from reception via the Web, a checking program will run on the server to check the answers submitted to see if they include any improper content, and an attempt will be authorized as a result subject to ranking when this program determines that it is a proper submission (that is, when it has verified that the contestant has not conducted any actions such as changing the problem or improperly manipulating the parameters).

2.2 Languages Used, Competition Environment

While there are no particular restrictions on programming languages used, they must make it possible to execute commands externally and they must be able to handle the standard inputs and outputs. In addition, since the client program is implemented in Java, another necessary condition is that the language be one that can be used to generate programs that can be executed on an OS that runs Java. As long as the program can be executed, contestants can take part by designating the execution command whether using a language that outputs a single executable file (C/C++,C#) or one that does not directly output a single executable file, such as an interpreter-based language (like Java, Python, or Ruby). See the file codevs_client.pdf for specific instructions on how to use the client.

2.3 Rankings

Results subject to ranking are the best results from each contestant, with the elements of results assigned importance in the following order:

- Number of levels cleared until satisfying the ending conditions
- Value of money remaining when the ending conditions were satisfied
- How soon was the date and time when submitted

For example, when five contestants have submitted the following results:

- A: Cleared by level 100-2, with 1000 units of money remaining, submitted at time 5/10,00:00
- B: Cleared all by level 250-4, with 2000 units of money remaining, submitted at time 5/30,00:00
- C: Cleared all by level 250-4, with 1000 units of money remaining, submitted at time 5/10,00:00
- D: Cleared by level 100-1, with 3000 units of money remaining, submitted at time 5/10,00:00
- E: Cleared by level 100-2, with 1000 units of money remaining, submitted at time 5/20,00:00

then they would be ranked in the following order, from highest to lowest: $B > C > A > E > D$.

3 More Detailed Rules of the Game

3.1 Maps

The game uses 250 maps in total. Since four levels of the game are played for each map, in total there are 1000 levels, and the last level is 250-4. Each time the map changes, the towers built on it all will be cancelled. (Money and player lives remaining will carry over to the next map.) The maximum size of a map is a total of 50 units both horizontally and vertically, and instead of using a fixed map every time, each time the game is played a slightly different layout will be used. Coordinates start from (0,0) at the upper left. There are four types of map panels: empty squares, obstacle squares, enemy appearance squares, and defended squares.

Figure1　Empty square

Figure2　Obstacle square

Figure3　Enemy appearance square

Figure4　Defended square

Each map is guaranteed to be surrounded by obstacle squares.

3.2 Types of Towers

There are three types of towers in total. Each differs in terms such as attacking power, range, time required to recharge, and whether it has any special effects.

3.2.1 Rapid Type

The rapid type of tower recharges very quickly but has low attacking power.

Figure5　Rapid Type

- Cost:10
- Attacking power:10
- Range:3 square
- Recharge time:10
- Special effects:none

3.2.2 Attacking Type

The attacking type of tower has strong attacking power but recharges slowly.

Figure6　Attacking Type

- Cost:15
- Attacking power:20
- Range:2 square
- Recharge time:100
- Special effects:none

3.2.3 Freezing Type

The freezing type of tower has low attacking power and recharges slowly but has the effect of freezing enemies in their tracks.

Figure7 Freezing Type

- Cost:20
- Attacking power:3
- Range:2 square
- Recharge time:20
- Special effects :stops enemies for a full 1/10 of recharge time

3.3 Strengthening Towers

Players can spend more money to strengthen towers already built. A single tower can be strengthened up to four times. In the range of strengthening attempts numbers ($1 \leq n \leq 4$), tower performance will strengthen in accordance with the formulas below:

3.3.1 Rapid Type

- Cost: initial construction cost$\times 3^n$
- Attacking power: initial attacking power
- Range: initial range $+n$
- Recharge time: initial recharge time $-2 \times n$

3.3.2 Attacking Type

- Cost: initial construction cost$\times 4^n$
- Attacking power: initial attacking power $\times 5^n$
- Range: initial range
- Recharge time: initial recharge time

3.3.3 Freezing Type

- Cost: initial construction cost$\times(1+n)$
- Attacking power: initial attacking power $\times(1+n)$
- Range: initial range $+n$
- Recharge time: initial recharge time

3.4 Recharge and Attack Cycles

Towers cycle through the three states of recharging, charged, and attacking in accordance with the rules below.

- At the start of a level, all towers are in charged status.

- Beginning the next time after reaching charged status, a tower will attack any enemies available to attack and then enter recharging status. (A tower cannot attack in the time at which it has reached charged status.)
- A tower in recharging status will reach charged status as soon as its recharging time has passed.

For example, a tower with a recharging time of 3, located in a place where there always are enemies available to attack, will behave as shown below.

Time	Charge Status	Attacking
0	Charged	
1	Recharging	○
2	Recharging	
3	Recharging	
4	Charged	
5	Recharging	○

3.5 Rules of Tower Behavior

A tower will attack an enemy using the methods below.

- When it is capable of attacking, it will list the enemies located in squares for which the Euclidean distance from the tower to the enemy appearance square is equal to or less than the tower's range. (While the animation makes it appear as if enemies move continually, in the program enemies are warped to specific squares each time they are capable of movement.)
- When there are multiple enemies within the tower's range, it will list the enemy that entered its range at the earliest time (employing the latest time if the enemy has left and then reentered the range).
- If more than one enemy has entered the range at the same time, then the tower will list the enemy that appeared on the map first.
- If more than one enemy appeared on the map at the same time, then the tower will choose the enemy given first in the standard input.
- After all towers capable of attacking have chosen their targets, then all towers at once will inflict the damage of their attacking power values on their targets.

3.6 Overlapping of Freezing Tower Attacks

Attack by a freezing tower renders the enemy unable to move for a full 1/10 of the recharging time. When attacking an immobile enemy with a freezing tower again, the period of immobility will increase for each time the enemy is attacked while immobile. The enemy ' s next move can be delayed by the amount of time it is rendered immobile by a freezing tower.

3.7 Rules on Building Towers

When building a tower as decided by the program, the following prohibitions apply:

- Towers may not be built on enemy appearance squares, defended squares, or obstacle squares (in other words, on other than empty squares), or outside the range of the map.
- Decisions may not be made that would result in a balance of money of less than zero.
- Towers may not be built on square in ways that would result in the appearance of enemy appearance square from which it is impossible to reach any defended square.

When any of the above prohibitions has been violated, the game will be over immediately. The third prohibition refers to, for example, the following situation:

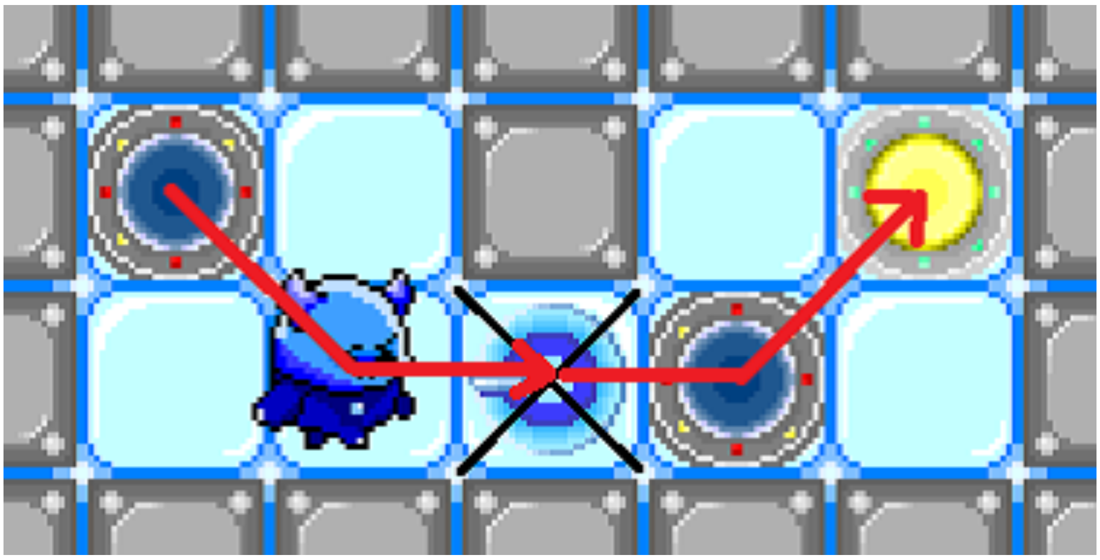

Figure8 Example of a tower layout resulting in the appearance of enemy appearance squares from which it is impossible to reach a defended square

While the enemy that has appeared in the enemy appearance square to the right can reach a defended square, the enemy that has appeared in the enemy appearance square to the left cannot reach a defended square by any route. The creation of such a square would assure the player's victory since the enemy would be unable to decrease the life of the player. This is why this rule prohibits acts that would result in the creation of even a single such enemy appearance square from which an enemy appearing there would be unable to reach even one defended square. The contestant's attempt will end at the level in which a tower has been built that would result in such a square. However, the presence of defended squares not reachable from any enemy appearance square is, on the other hand, acceptable.

3.8 Rules on Enemy Movement

Enemies appearing on the map have the following four parameters:

- Appearance coordinates (x,y)
- Appearance time
- Life
- Movement time

Enemies appear in their appearance coordinates at the appearance times and then aim for defended squares using routes determined by a predetermined algorithm. The steps by which the route an enemy will take is determined are shown below.

- Enemies recognize squares having no obstacles and no towers as passable squares.
- An enemy may move in the four diagonal directions only when all squares, including the two squares whose corners it will pass by on the diagonal route, are passable squares.
- An enemy will list all routes that lead to a defended square at the minimum cost, with the cost of moving up, down, left, or right being 10 and the cost of moving diagonally being 14.
- When the routes listed from the current square diverge, the enemy will give top priority to moving to the right and will prioritize available moves from there in the counter-clockwise direction (giving priority to the directions with the newest numbers below), moving in accordance with these priorities.

```
432
5.1
678
```

When an enemy moves in a diagonal direction, it will take the length of time determined through integer arithmetic by multiplying the time it takes for the enemy to move ordinarily in the up, down, left, or right directions by 14 and then dividing this by 10, rounding any remainder down. If the enemy ' s life reaches zero before it arrives at the defended square, then it will disappear on that spot. If the enemy arrives at a defended square before its life reaches zero, the player ' s life will decrease by one and the enemy will disappear. As shown in the gameplay steps below, while no damage is sustained if the life reaches zero at the time the enemy arrives, when time in which it cannot act due to a freezing tower remains at the time the enemy arrives then damage is sustained.

After appearing in a designated square at the appearance time, an enemy will move to adjoining squares repeatedly through warping at each movement time (if time in which the enemy cannot act is applied by a freezing tower as described above, this time in which it cannot act is added to the movement time). While the game ' s animation makes enemies

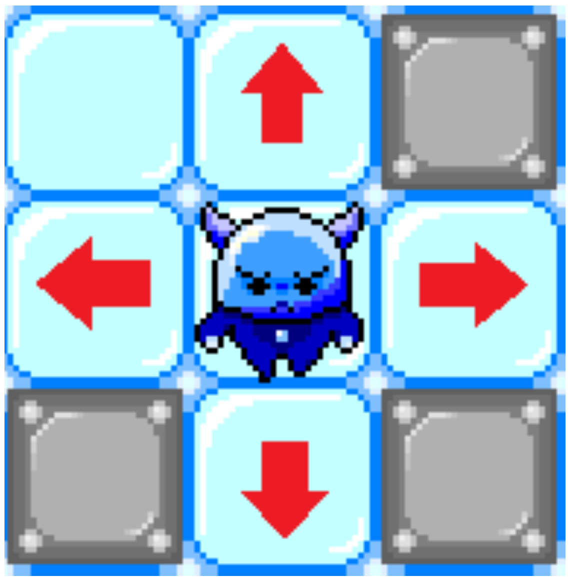
Figure9 Horizontal and vertical movement

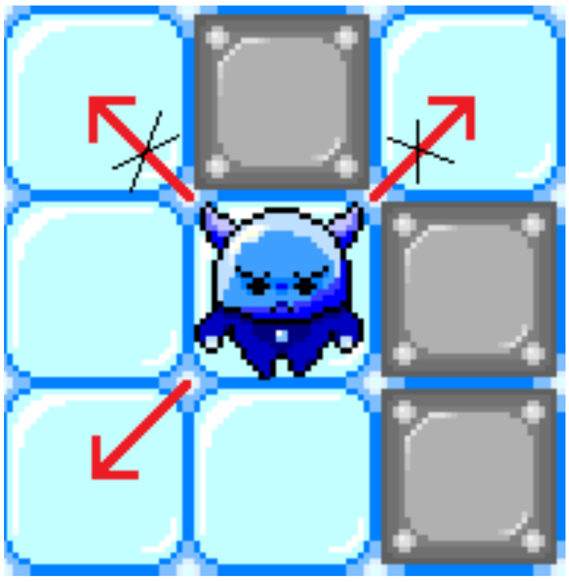
Figure10 Diagonal movement

appear to move continuously to make it easier to see their movement, within the game itself enemies always are positioned at integral coordinates, and this should be noted when thinking about tower range.

3.9 Game Ending Conditions

When any of the following conditions is satisfied, the game will end immediately and results will be submitted:

- When the entire map has been cleared
- When the player ' s life reaches zero
- When the rules on building towers (above) have been violated

In addition, in the following cases the game will end and the score either will not be sent or will be deleted from the records after being sent:

- When internal data such as map files have been manipulated inappropriately
- When the designated executable file or command cannot be executed
- When the standard input/output of the designated executable file or command cannot be obtained
- When game start and end requests cannot be sent due to a network error

3.10 Game Processing Steps

Game processing will take place in the following steps at each relevant time:

- At the appearance times, enemies will appear in the designated appearance squares.
- When an enemy can move, it will move to a neighboring square.

- All towers capable of attacking will determine their ranges following the designated rules.
- All towers capable of attacking will attack.
- When an enemy with a life of one or above has reached a defended square (even during a time in which it is unable to act), the player ’s life will decrease by one and the enemy will disappear.
- When the game ending conditions are satisfied, the game will end.
- When all enemies have appeared and then disappeared, the map will clear and the game will advance to the next level.

4 Program Input/Output Formats

4.1 Inputs

At the start, the number of maps S is provided as one line of data. Next, data is input for S units of maps.

Map data starts with provision of the map ’ s width and height (W, H) as a line of data separated by single-byte spaces. H is followed by a text string consisting of W characters. The text in row i, column j corresponds to the map data $F_{i-1,j-1}$ for the coordinates $(i+1, j+1)$, as follows:

- 0:empty square
- 1:obstacle square
- s:enemy appearance square
- g:defended square

Next, the map ’ s number of levels L is provided as a single line of data, and then data is provided for a number of L levels. Each set of these map inputs up here includes the line containing only the separator END.

Level data first provides in a single line separated by single-byte spaces the player ’ s life at the start of the level L_p, money M, number of towers T, and number of enemies E. The following T lines represent data for each tower. Tower data provides the tower ’s X coordinate X_t, its Y coordinate Y_t, the number of times the tower has been strengthened A_t, and the tower type C_t, in that order, in a single line. The tower type C_t can take the following values:

- 0:rapid tower
- 1:attacking tower
- 2:freezing tower

The next E lines provide data on each enemy. Enemy data provides the X coordinate of the enemy ’ s appearance X_e, the Y coordinate of the enemy ’ s appearance Y_e, the time the enemy appears T_e, the enemy ’ s life L_e, and the enemy ’ s movement time S_e, in that order, in a single line. Furthermore, a separator line containing the word END follows to separate the data provided to this point.

Standard input ends each time the sending of data for a single level is completed, and when the designated response format described below has been output to standard output data is sent again, this time for the next level. This flow repeats until the game ends. As designated above, input can be obtained in a format such as the following:

S
$W H$
$F_{0,0} F_{0,1} ... F_{0,W-1}$
$F_{1,0} ...$
...
$F_{H-1,0} ... F_{H-1,W-1}$
L
END
$L_p M T E$
$X_{t1} Y_{t1} A_{t1} C_{t1}$
$X_{t2} Y_{t2} A_{t2} C_{t2}$
...
$X_{tT} Y_{tT} A_{tT} C_{tT}$
$X_{e1} Y_{e1} T_{e1} L_{e1} S_{e1}$
$X_{e2} Y_{e2} T_{e2} L_{e2} S_{e2}$
...
$X_{eE} Y_{eE} T_{eE} L_{eE} S_{eE}$
END
$L_p M T E$
...
...
END
$W H$
$F_{0,0} F_{0,1} ... F_{0,W-1}$
$F_{1,0} ...$
...
$F_{H-1,0} ... F_{H-1,W-1}$
L
...

4.2 Outputs

Each time level data is received, the program outputs its decisions on how to build towers to standard output. The output first outputs the number T decided on, as a single line. The following T lines output the tower ' s X coordinate X_t, its Y coordinate Y_t, the number of times the tower has been strengthened A_t, and the tower type C_t, in that order and separated by single-byte spaces. The values of C_t have the following meanings:

- 0:rapid tower
- 1:attacking tower
- 2:freezing tower
- 3:blank (instruction to destroy tower)

These commands are executed in order from first to last. Commands are executed as shown in the following examples, in accordance with the coordinates for which they have been issued:

- Ex. 1: (1 1 0 0) in blank square (1,1)
  Build a rapid tower. Costs the amount of money required to build a rapid tower.
- Ex. 2: (1 1 2 0) in blank square (1,1)
  Build a rapid tower and strengthen it twice. Costs the amount of money required to build a rapid tower and to strengthen it the first and second times.
- Ex. 3: (1 1 3 0) at coordinates (1,1) on which is located a rapid tower strengthened twice
  Strengthen the tower once. Costs only the amount of money required to strengthen a rapid tower the third time.
- Ex. 4: (1 1 0 3) at coordinates (1,1) on which is located a rapid tower strengthened twice
  Destroy the existing rapid tower. Costs nothing.
- Ex. 5: (1 1 1 0) at coordinates (1,1) on which is located a rapid tower strengthened twice
  Destroy the existing rapid tower, build a new rapid tower, and strengthen it once. Costs the amount of money required to build a rapid tower and to strengthen it the first time.
- Ex. 6: (1 1 1 1) at coordinates (1,1) on which is located a rapid tower strengthened twice
  Destroy the existing rapid tower, build a new attacking tower, and strengthen it once. Costs the amount of money required to build an attacking tower and to strengthen it the first time.

As described above, each time you output level data do so in the following format:

T
$X_{t1} Y_{t1} A_{t1} C_{t1}$
$X_{t2} Y_{t2} A_{t2} C_{t2}$
...
$X_{tT} Y_{tT} A_{tT} C_{tT}$
T
$X_{t1} Y_{t1} A_{t1} C_{t1}$

...

...

4.3 Maximum Input Values

Input values are guaranteed not to exceed the values or ranges shown below. In addition, if output values include values outside the ranges below, either the game will end immediately or those outputs will be ignored.

$$\begin{aligned}
S &= 250 \\
L &= 4 \\
15 \leq W, H &\leq 24 \\
1 \leq L_p &\leq 10 \\
0 \leq M &\leq 1000000 \\
0 \leq X_t, X_e &\leq W - 1 \\
0 \leq Y_t, Y_e &\leq H - 1 \\
0 \leq A_t &\leq 4 \\
0 \leq C_t &\leq 3 \\
0 \leq T &\leq 576 \\
3 \leq E &\leq 21 \\
1 \leq T_e &\leq 30 \\
1 \leq L_e &\leq 27501 \\
3 \leq S_e &\leq 103
\end{aligned}$$