國立清華大學
NATIONAL TSING HUA UNIVERSITY

Tic-Tac-Toe Tree

Data Structures Assignment

NTHU EE and CS

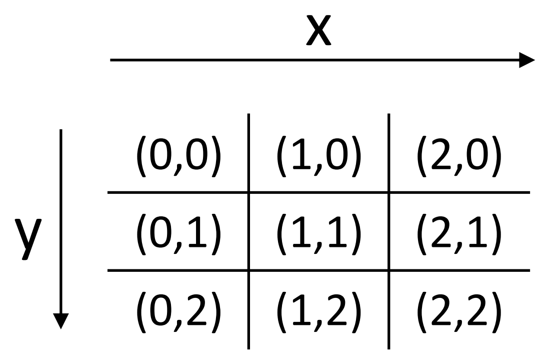
https://acm.cs.nthu.edu.tw/problem/12251/

Overview

- Given
  - A series of nodes representing the possible steps in Tic tac toe.
- Task
  - Convert the input data into a tree
  - Report “Win” if there is a path in the tree with a ‘winning’ status
    - Output the final game status
  - Otherwise, report “Tie”.
    - Print out the moves based on the Postorder Traversal method

Specification

- Each node of the tree consists of:
  - ID
  - The parent node ID
    - -1 represents null for root node
  - The move
    - Position (x,y) and the player mark {O, or X}
- To simplify the game, each node will only have at most two possible children
- The player who succeeds in placing three of their marks in a horizontal, vertical, or diagonal row wins the game



Output

- If there is a 'winning’ path in the tree, ouput:
  - ‘Win’, followed by new line
  - The tic-tac-toe grid with the moves on the winning path. Empty squares will be represented with ‘_’ .Positions are separated with whitespaces. There is an endline at the end of each line.

- Else, output:
  - ‘Tie’, followed by new line
  - For each node, traversed in postorder traversal, output:
    - Position x, position y and Mark{O, X}, separated by whitespaces, followed by new line

Sample Input 1

ID

Parent ID

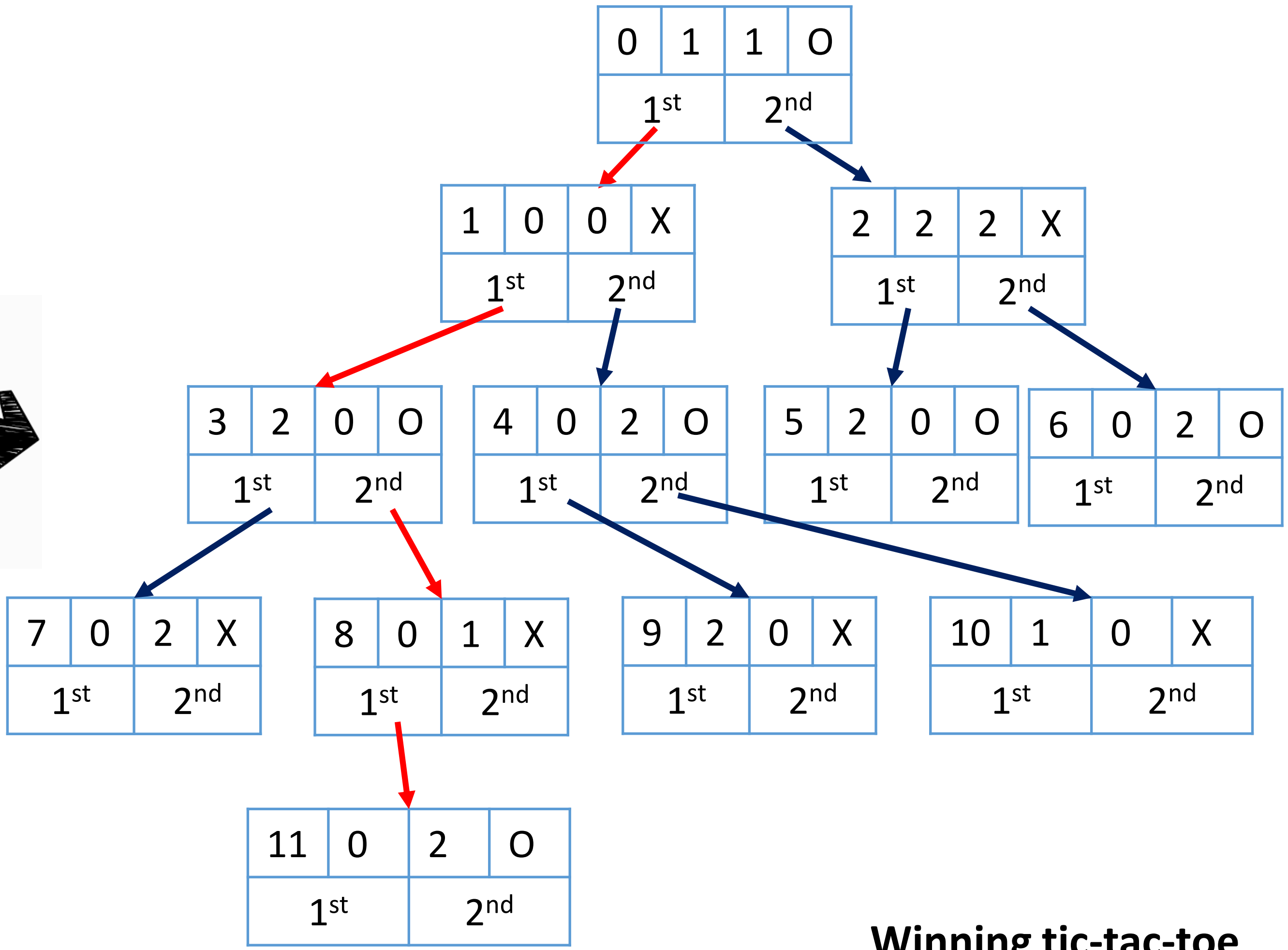
Move

Number of nodes (≥ 1)

Steps

```
12
0 -1 1 1 O
1 0 0 0 X
2 0 2 2 X
3 1 2 0 O
4 1 0 2 O
5 2 2 0 O
6 2 0 2 O
7 3 0 2 X
8 3 0 1 X
9 4 2 0 X
10 4 1 0 X
11 8 0 2 O
```

```
12
0 -1 1 1 O
1 0 0 0 X
2 0 2 2 X
3 1 2 0 O
4 1 0 2 O
5 2 2 0 O
6 2 0 2 O
7 3 0 2 X
8 3 0 1 X
9 4 2 0 X
10 4 1 0 X
11 8 0 2 O
```



Tree Node Format

ID	X Pos	Y Pos	Mark
1st Child		2nd Child	

Winning path

$\xrightarrow{\hspace{3em}}$

Winning tic-tac-toe

X	_	O
X	O	_
O	_	_

Sample Output 1

```
Win↲
X _ O ↲
X O _ ↲
O _ _ ↲
```

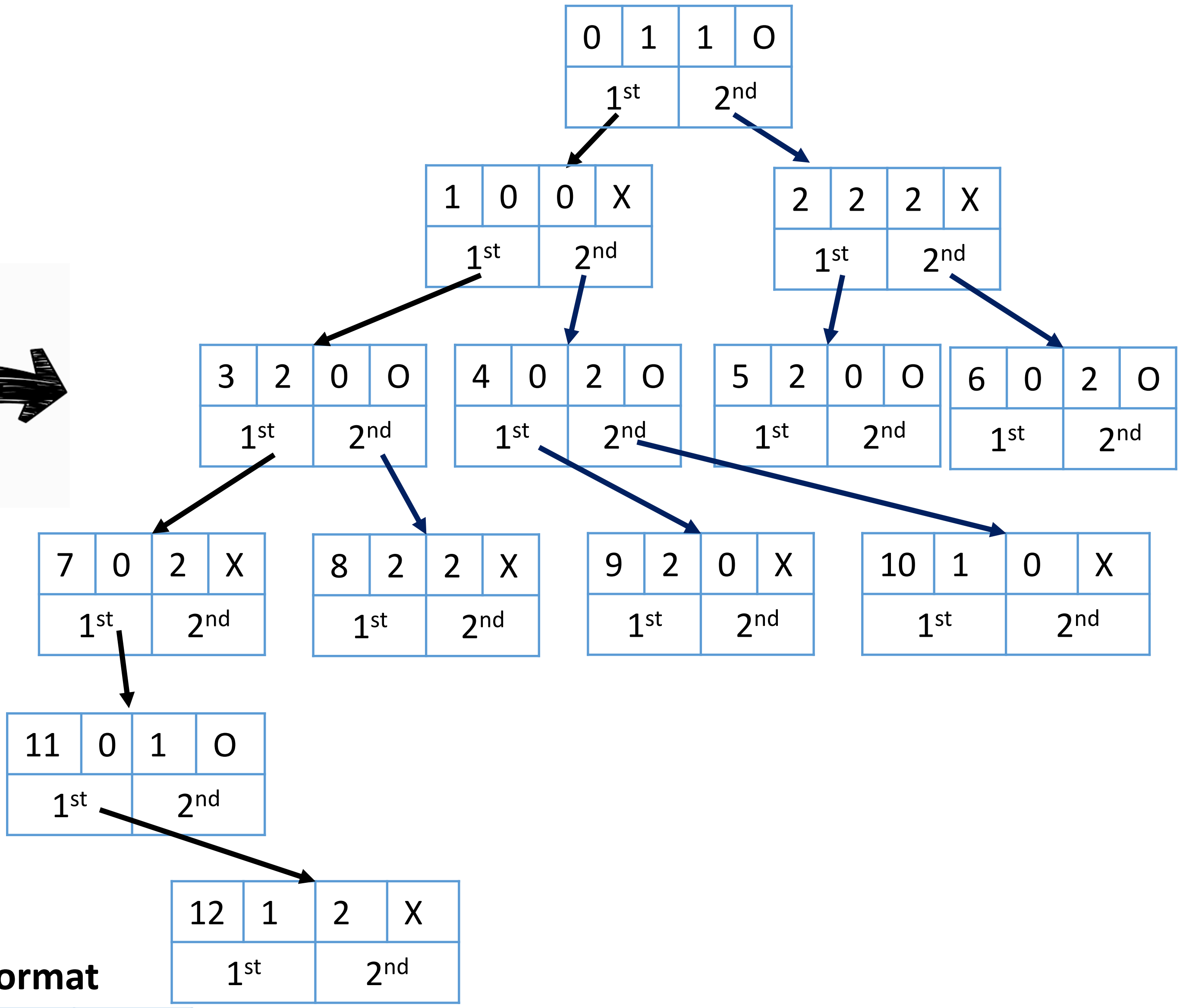
Sample Input 2

Number of nodes (≥ 1)

Steps

```
13 ↵
0 -1 1 1 O ↵
1 0 0 0 X  ↵
2 0 2 2 X  ↵
3 1 2 0 O  ↵
4 1 0 2 O  ↵
5 2 2 0 O  ↵
6 2 0 2 O  ↵
7 3 0 2 X  ↵
8 3 2 2 X  ↵
9 4 2 0 X  ↵
10 4 1 0 X ↵
11 7 0 1 O ↵
12 11 1 2 X ↵
```

```
13
0 -1 1 1 O
1 0 0 0 X
2 0 2 2 X
3 1 2 0 O
4 1 0 2 O
5 2 2 0 O
6 2 0 2 O
7 3 0 2 X
8 3 2 2 X
9 4 2 0 X
10 4 1 0 X
11 7 0 1 O
12 11 1 2 X
```



Tree Node Format

ID	X Pos	Y Pos	Mark
1st Child		2nd Child	

Sample Output 2

```
Tie↵
1 2 X↵
0 1 O↵
0 2 X↵
2 2 X↵
2 0 O↵
2 0 X↵
1 0 X↵
0 2 O↵
0 0 X↵
2 0 O↵
0 2 O↵
2 2 X↵
1 1 O↵
```

Notes

- A tree will have at most one 'winning’ path
- You don't need to keep track whose turn it is to move {X,O}
- The resulting trees will not be balanced, full nor complete