PSA EXTRA: Putting intelligence into your connect 4 game
Optional, but highly encouraged! – DUE WEDNESDAY FOR EVERYONE! (no additional slip day)

Not so much “guidance” for PSAEXTRA, but THERE ARE STILL LAB-HOURS
(if you’re doing PSA Extra!! Or want to practice any other thing!)

- Google search for recursion

  Did you mean: recursion

  Recursion - Wikipedia, the free encyclopedia
  Recursion is the process of repeating items in a self-similar way.
  The surfaces of two mirrors are exactly parallel with each other.
  Formal definitions of recursion
  - Recursion in language
  - Recursion (computer science)
  - Recursive data types
  - Recursive algorithms
  - Structural versus g
1) Add AI player “variable”
2) As usual, human-player clicks cell → make the move.
3) Right after... make your AI player “decide” its nextMove → then make that move.

```java
C4Player
C4Player( char ch, String tbt, int plyIn )
String toString()
char oppCh() // opposite checker than current turn
double scoreBoard(C4Board b)
double[] scoresFor(C4Board b)
int tiebreakMove(double[] scores)
int nextMove(C4Board b) // gives the column index where the computer wants to play given a board b...
```
PSA-EXTRA: The **C4Player**

**C4Player**

- **C4Player( char ch, String tbt, int plyIn )**
- **String toString()**
- **char oppCh()** // opposite checker than current turn
- **double scoreBoard(C4Board b)**
- **double[] scoresFor(C4Board b)**
- **int tiebreakMove(double[] scores)**
- **int nextMove(C4Board b)** // gives the column index where the computer wants to play given a board b ...

**DEMO!**

**HOW DO I DECIDE THAT!!?!?!?!**
- **THE GOAL!** Is to make a recursive “exploration” to find the move that gets you the highest score
- You can **START with A “DUMMY” method that decides without much “intelligence”**.
PSA-EXTRA: The C4Player

- **DEADLINE WEDNESDAY** for everyone! (no additional slip, that’s already one slip for everyone that tries this PSA)

- **Replace lowest PSA2 to PSA7 score.** Get up to 3 star points if you don’t get to improve your PSA score (because it’s already so good!)

- **DO IT FOR THE SAKE OF LEARNING!** Study and run our suggested code for PSA4 and PSA6. Practice with recursion. Can you make this? It’s very challenging BUT fun!

**C4Player**

- `C4Player( char ch, String tbt, int plyIn )`
- `String toString()`
- `char oppCh() // opposite checker than current turn`
- `double scoreBoard(C4Board b)`
- `double[] scoresFor(C4Board b)`
- `int tiebreakMove(double[] scores)`
- `int nextMove(C4Board b) // gives the column index where the computer wants to play given a board b ...`

**DEMO!**
public static int find( String[] myList, String toFind, int startIndex )
{
    if ( startIndex >= myList.length ) {
        return -1;
    }
    if ( toFind.equals( myList[startIndex] ) ) {
        return startIndex;
    }

    // Recursive step here . . .

    WHAT ELSE DO WE NEED once we have designed the “base case”?
Ever used a telephone book?

• If I want to look up someone’s number and their last name starts with Z, where should I start? Do you really do like “find” method?

• If I want to look up someone’s number and their last name starts with G where should I start?
Binary Search algorithm (only works for “ordered” data)

Look in the middle of the data list (array)

- If that element is what you are looking for – return its index
- If not
  - If what you are looking for “comes before” in the ordering, look in the half on the left
  - Else (what you are looking for “comes after” in the ordering), look in the half on the right
Binary Search example

The method will return the *position* of the *item if it is found*

```java
public static int binarySearch( int[] theList, int toFind, int low, int high)
    low will start at 0, high will start at theList.length-1
```

Assuming the array pictured above is `myArray` these would be some example calls:

- `binarySearch( myArray, 4, 0, 8 )`
- `binarySearch( myArray, 5, 0, 8 )`
Imagine the numbers are arranged like this...
Binary Search example

```
2  4  6  8  10  12  14  16  18
0  1  2  3  4  5  6  7  8
```

Algorithm “pseudo-code”:
1. calculate the midpoint, mid, between low and high
2. if theList[mid] == toFind, return mid
3. if toFind is larger than theList[mid], recurse on the larger half of the list
4. else if toFind is smaller than theList[mid], recurse on the smaller half of the list

Trace the values of high, low and mid when you call binarySearch with this list above and the value 8. Draw more stack frames if you need them.

EVERYONE GET PAPER AND PEN!!!
Algorithm “pseudo-code”:
1. calculate the midpoint, mid, between low and high
2. if theList[mid] == toFind, return mid
3. if toFind is larger than theList[mid], recurse on the larger half of the list
4. else if toFind is smaller than theList[mid], recurse on the smaller half of the list

Trace the values of high, low and mid when you call binarySearch with this list above and the value 8. Draw more stack frames if you need them.

HOW MANY STACK frames do we need?
A. 1
B. 2
C. 3
D. 4
public static int binarySearch( int[] list,
        int toFind, int low, int high ) {
    int mid = (low+high) / 2;

    if (toFind < list.get( mid ))
        return ______________________________
    else if (toFind == list.get( mid ))
        return ______________________________
    else // toFind is > list.get(mid)
        return ______________________________
public int binarySearch( ArrayList<Integer> list,
                      int toFind, int low, int high ) {

  int mid = (low+high) / 2;
  if (toFind < list.get( mid ))
  return binarySearch( list, toFind, low, mid-1 );

  else if (toFind == list.get( mid ))
  return mid;

  else // toFind is > list.get(mid)
  return binarySearch( list, toFind, mid+1, high );
}

Which is the base case in the method above?
public int binarySearch( ArrayList<Integer> list, int toFind, int low, int high ) {
    int mid = (low+high) / 2;
    if (toFind < list.get(mid))
        return ________________________________
    else if (toFind == list.get(mid))
        return ________________________________
    else // toFind is > list.get(mid)
        return ________________________________
}

When will this method not work?
A. When the element you are looking for is the first element in the list
B. When the element you are looking for is the last element in the list
C. When the element you are looking for is not in the list
D. When the element you are looking for is less than 0
E. It will always work
public int binarySearch( ArrayList<Integer> list,
        int toFind, int low, int high ) {
    // base case for not in list

    int mid = (low+high) / 2;
    if (toFind < list.get( mid ))
        return binarySearch( list, toFind, low, mid-1 );
    else if (toFind == list.get( mid ))
        return mid;
    else // toFind is > list.get(mid)
        return binarySearch( list, toFind, mid+1, high );
}

Add the missing base case!
Exam 4 – NEXT TUESDAY

A bit more of programming than exam3, but you’ll get the method list with everything you may need (no need to memorize!!)

• Interfaces, Abstract classes

• Event Listeners
  – No need to memorize names/lists of methods! We’ll give you the list

• Recursion
  – Main ideas: base case, recursive case, stack frame
  – Simple examples: mathematical operations, drawing examples