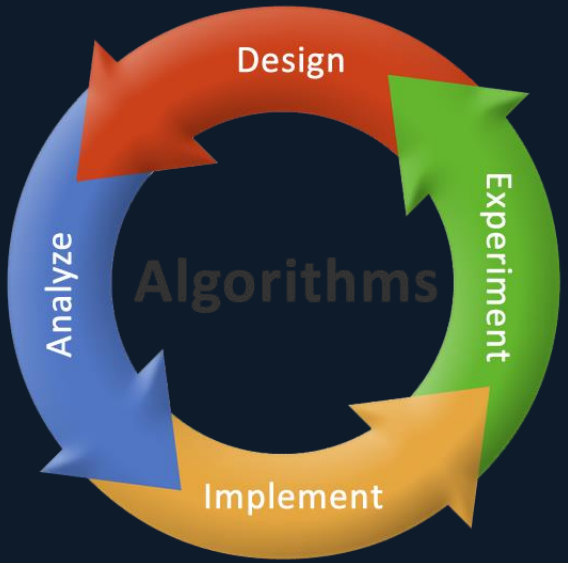
Algorithm Design and Pseudocode



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Algorithms, what are they?

- In mathematics and computer science an algorithm is a self-contained step-by-step set of operations to be performed
- Some examples:
  - Bubble sort
  - Insertion Sort
  - Binary Search
- A computer programmer implements algorithms using a specific programming language

Why use them?

- When solving complex problems it helps to separate the problem from specific implementation details.
- Once we have our algorithm expressed in a computer language agnostic form it's much easier to implement it in any language we choose
  - Assuming we know how to read the algorithm

Writing algorithms in plain English

- Write the instructions you want to follow as clearly as possible in English
- Make sure you cover all possible cases
- The more precise your algorithm the easier it will be to convert it into code

Example: searching a list of random numbers for the largest in English

1. If there are no numbers in the set then there is no highest number
2. Assume the first number in the set is the largest number in the set
3. For each remaining number in the set: if this number is larger than the current largest number, set the largest number to this one
4. When there are no numbers left in the set to check return the largest number found

Writing Algorithms in pseudocode

- Whilst we can write algorithms any way we want there are some standards which make them easier to write and understand.
- By adopting a standard, sharing of algorithms becomes easier
- If we write an algorithm using a formal language then we say we are using a *pseudocode*
- There are many useful algorithms, which are written in pseudo code, available for your use
- Learning to read and write pseudocode is a useful skill

Example: searching a list of random numbers for the largest in a more formal pseudocode.

Algorithm: Largest Number
Input: A List of numbers L.
Output: The largest number in the list L

If L.size = 0 **return** null
Largest ← L[0]
For each item **in** L, do
    if item > Largest, **then**
        Largest ← item
Return Largest

Pseudocode notes:

- It looks a bit like code but it's not quite as precise
- Note the use of "*for each*", "*return*", "*do*","*then*" etc.
- Note the use of ← to indicate becomes "*equal to*".
- Pseudocode can also make use of functions. For example we might make use of the LargestNumber function in another pseudocode fragment

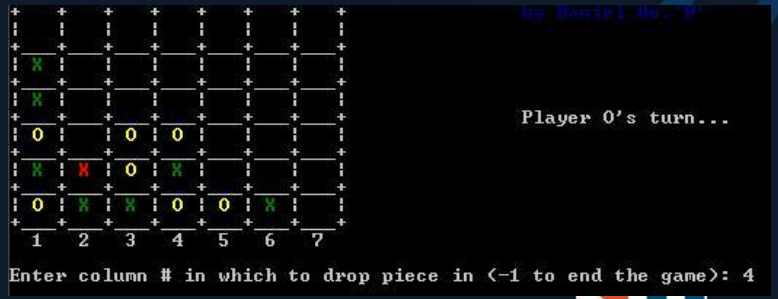
Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Another example: Connect Four

- Structured English

1. Ask player for column
2. Read column
3. If column full, go back to step 1
4. Check height of highest marker in selected column.
5. Place new marker above current highest.
6. Check for a winner
7. If no winner, repeat steps 1 – 6 for other player.



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Exercise!

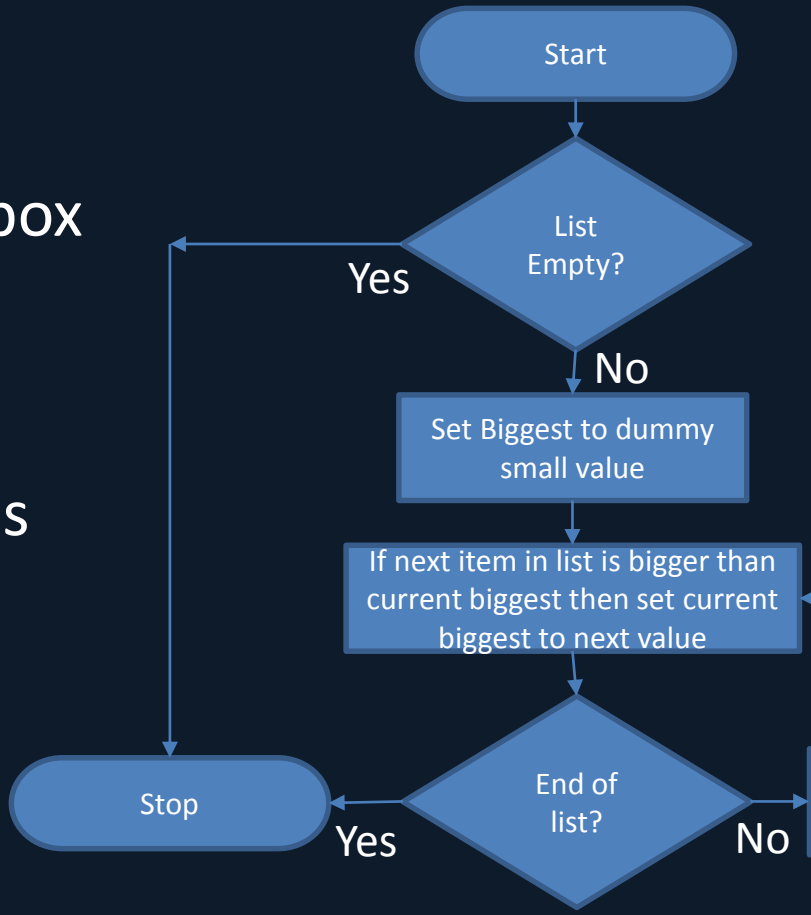
- Write the pseudocode for the match 4 example as per the earlier pseudocode example

Representing code using diagrams

- A third way to represent algorithms is by using diagrams
- Flow charts where very popular back in the early days of computing
- These days they don’t get used as much
- But they can still be useful

Flow Charts

- The shape of the box is important
- Still makes use of English
- Can have functions



Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX

Why use flow charts?

- Sometimes it's easier to visualize a problem when it's in diagram form.
- Flow charts don't have to be very formal
  - Just sketching out program flow in a simple pencil diagram, or on the white board, can often help to understand a complex problem
- The secret is to pick a suitable level of detail
  - Too much detail and the diagram is unreadable
  - Too little detail and it's too superficial to be useful

Abstraction

- Abstraction is the process of taking away characteristics from something to reduce it to a subset of essential characteristics
- Pseudocode is an abstract form of program code
  - It contains enough detail for us to code the algorithm
  - But does not contain implementation details

What's missing in pseudocode?

- For example in the previous example the pseudocode does not specify what sort of data type the list is
  - It could be a fixed array, dynamic array, linked list etc
  - This information is irrelevant to the algorithm
  - However you will need to think about this when you implement it
- The programmer often needs to adapt the pseudocode to make use of the features that their chosen programing language offers.

Developing an Algorithm

- When faced with a problem it is useful to work out these things:
  - Start state
  - Goal state
  - Inputs
  - Outputs
  - Constraints

Developing an Algorithm - Example

- Sorting an array of integers

  - Start state: unordered array of integers
  - Goal state: sorted array of integers
  - Inputs: an array
  - Outputs: an array (either to screen or returned from a function)
  - Constraints: size of the array out must be the same as the size of the array in

Developing an Algorithm

- Once you know the inputs and outputs, you can begin writing line-by-line instructions in English or pseudocode
- Any line that is too vague can be broken down into its own algorithm. These become your functions.
- Eg. “Check for a winner” from the Connect Four example would be broken down into another algorithm
- This is called functional decomposition

The Algorithm is not the same as the problem description

- Note that producing an accurate description of a problem is not equivalent to producing an algorithm
- In our previous example for finding the largest number the algorithm given is one solution to the problem, in this case *an iterative solution*.
- There are other solutions and they are expressed using different algorithms
- Exercise: Think of another algorithm for the above problem, write it in English and formal pseudocode

Picking the best algorithm for the job

- If there are several different algorithms which can be used to solve a problem which do you pick?
- Each algorithm has pros and cons
- Understand these and pick the most appropriate on a case by case basis. Things which effect your choice:
  - The type and organization of data (maybe the data is pre-sorted?)
  - The type of hardware (parallelization has a bearing)
  - Is your goal, speed, code compactness or something else?

References

- Stanford - Intro to Algorithms
- http://en.wikipedia.org/wiki/Algorithm
- Data Structures and Algorithms for Game Developers – Allen Sherrod

Canberra Institute of Technology

aie
SPECIALIST EDUCATORS IN GAMES, ANIMATION & FILM VFX