Algorithms and Data Structures: Week 3

Johnson rev. Bell (Michaelmas 2024)

Question 1

Write pseudocode for inserting into a doubly linked list L a piece of data v before node N. You can assume L has sentinel nodes. What would need to be changed in your solution if L did not have sentinels? There is no need to write a full solution for this.

Question 2

What does the following code do?

```
Input: integer k, integer A[1...n]
Output: ?
  integer c = 0
  for i = 1 to n do
    if A[i] = k then
      c = c + 1
    end if
  end for
  return c
```

Question 3

The ten high score entries for a video game can be stored in an array `H` in which each element is either `NULL` or an array with two entries, a score and a name. The elements of the array are ordered with respect to the scores. For example, using a 1-based indexing convention:

```
H = [[940,Mike],[880,Rob],[830,Jill],[790,Paul],
       [750,Anna],[660,Rose],[650,Jack],NULL,NULL,NULL]
```

So `H[1] = [940,Mike]`, `H[2][1] = 880` and `H[3][2] = Jill`.

Write pseudocode for an algorithm whose input is the high score array `H` and an additional entry `E = [newscore, newname]` and whose output is the correctly updated high score array. Alternatively, if you prefer, download the Jupyter notebook `ADSWeek3.ipynb` from the Jupyter server (or from Ultra) and follow the instructions to write your solution there. Note that if you are implementing this in Python, Python uses 0-based indexing for arrays.

Question 4

Describe a method for finding, by "link hopping", the middle node of a doubly linked list with head and tail sentinels and an odd number of nodes. The method should use as many "pointers" (i.e. references to nodes) as necessary, that can be defined by assignments such as

```
pointer1 = List.head
```

and can be updated by, for example,

```
pointer1 = pointer1.prev.
```

The method must not count anything and should not reference `List.size`.

Question 5

Suppose you are designing a multi-player game that has n interacting players. The winner of the game is the first player to meet all the others.

(a) Describe a data structure(s) that could be used to keep track of the meetings that have taken place.

(b) Write some pseudocode that will update the data structure(s) whenever two players meet and check whether one (or both) of them has won. (Note that when players meet they might have met before so you cannot just count the number of meetings a player has.)

Extension

Complete some or all of the tasks below if you have time

- Try to solve Problem 109 on Project Euler
- Look at the additional exercise in `ADSWeek3.ipynb` on implementing linked lists in Python.

Implementation Notes

Sometimes the quickest to way to get a feel for Python behaves is to try out a simple case, and make sure that the way the programming language works as you expect. What do you expect the following Python code to do? Do not run the code; make a guess first, and *then* run the code.

```
for i in range(5):
```

```
    print(i)
```

From this, we get a sense for what is needed in order to implement for loops of the following form in the pseudocode notation (which iterates over the integers from 1 to n, inclusive)

for i = 1 to n **do**
  print i
end for

Now, looking at the pseudocode in **Question 2**, ask how is the variable *i* being used? It is used as an index into the array of integers. We can infer quite reasonably from the header that the one-indexing convention is being used here. So the effect of the for loop is to range over the contents of the array in turn, and for each *i*, get the value at the *i*-th location of the array and do something with that.

You also should recall from CT that Python `list()`s use zero-indexing. You might also know that Python's `range()` can work in a mode where it takes two arguments: `start` and `end` (for more details see: `https://docs.python.org/3/library/stdtypes.html#typesseq-range`) What do these two facts about the language mean in practice for your Python implementation of **Question 2**? There are often multiple ways to implement the same pseudocode in Python or in any other language.

About Python implementation of attributes

Remember that in a linked list, a Node has two attributes: data and next. When you define a Node, you must state what the data is. Assuming there is an implementation of this kind of data structure in Python (you do not need to code this yourself; see the Jupyter Notebook for the implementation), you might use it as follows:

```
n1 = Node(7)
n2 = Node(4)
```

to create two nodes with data 7 and 4. If you want to recall what the data of a node is, you can write in Python (as you do in pseudocode).

```
n1.data
```

to get the data in `n1`.

In the implementation, the next attribute is initially `None` but you can update by, for example, writing:

```
n1.next = n2
```

And now you could write

```
n1.next.data
```

and get the `data` attribute of `n2`. In this way, one can start to build up the chain of next references that gives the linked list data structure its characteristic structure.