Cycle Detection Algorithm: the Hare and the Tortoise

Frank the Giant Bunny

August 27, 2014

Question Using only two pointers, decide whether a singly-linked list L contains a loop or not.

Answer The simplest answer is to maintain two pointers: the `hare` moving *two* steps at a time and the `tortoise` moving *one* step at a time.

```
Hare-and-Tortoise($L$)
  // Input: a linked list $L$
  // Output: ‘yes’ if $L$ contains a loop and ‘no’ otherwise.
  Two pointers hare and tortoise are initially located at the head of $L$.
  Repeat
    If the hare reaches the end of $L$ within two steps, return ‘no’.
    Else
      Move the hare two steps forward and the tortoise one step forward.
      If two pointers are located on the same node, return ‘yes’.
    EndIf
  EndRepeat
```

Why it works If the list L doesn't contain a loop, the hare will reach the end of L sooner than the tortoise does and the algorithm will eventually return ‘no’. If the list L contains a loop, it results in a ρ shape when diagrammed.

- Once the hare and tortoise enter the cycle part of the ρ shape, they are trapped in the loop and keep rotating around the cyclic path.
- And the hare enters the cyclic part sooner than the tortoise does.
- Assume that the hare and tortoise are separated in d steps when the tortoise enters the cycle.
- As the tortoise moves only one step while the hare moves two steps, their distance in the cyclic path keeps decreasing by one for each iteration of the `Repeat` block of the algorithm. Hence, two pointers will eventually collide and the algorithm returns ‘yes’.

Ask yourself What if the hare moves *three* steps forward at a time while the tortoise moves *one* step at a time? What about other steps? Does the algorithm still work?