Linked lists

Prof. Noah Snavely
CS1114
http://cs1114.cs.cornell.edu

Cornell University
Computer Science

Administrivia

- Assignment 2, Part 2 due tomorrow
  - Please don't wait until the last minute to finish (the last two problems are challenging)
- Assignment 3 will be posted tomorrow
  - Due in two weeks (Friday, 3/6)
- Prelim 1 next Thursday, 2/26 in class
  - Review session: Tuesday or Wednesday evening?
  - Topics include: running time, graphs, linked lists

Making quickselect fast on non-random input

- The version of quickselect in A2 can be **very** slow
- What is the problem?
- How can we fix it?

Conditionals with multiple branches

- What if we want the robot to correctly obey a traffic signal?

```
L = getLightColor();
if L == ‘red’
   robotStop();
end
if L == ‘green’
   robotDriveStraight(r, 10, 100);
end
if L == ‘yellow’
   robotDriveStraight(r, 100, 100);
end
if L ~= ‘red’ && L ~= ‘green’ && L ~= ‘yellow’
   fprintf(‘Unknown light color\n’);
end
```

Conditionals with multiple branches

- What if we want the robot to correctly obey a traffic signal?

```
L = getLightColor();
if L == ‘red’
   robotStop();
else
    if L == ‘green’
        robotDriveStraight(r, 10, 100);
    else
        if L == ‘yellow’
            robotDriveStraight(r, 100, 100);
        else
            fprintf(‘Unknown light color\n’);
        end
    end
end
```

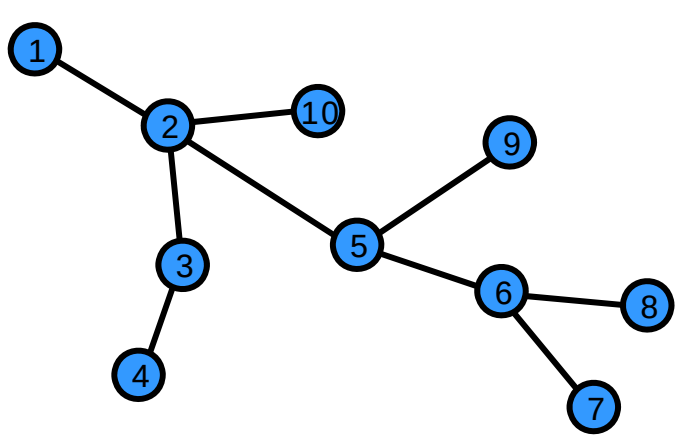
Conditionals with multiple branches

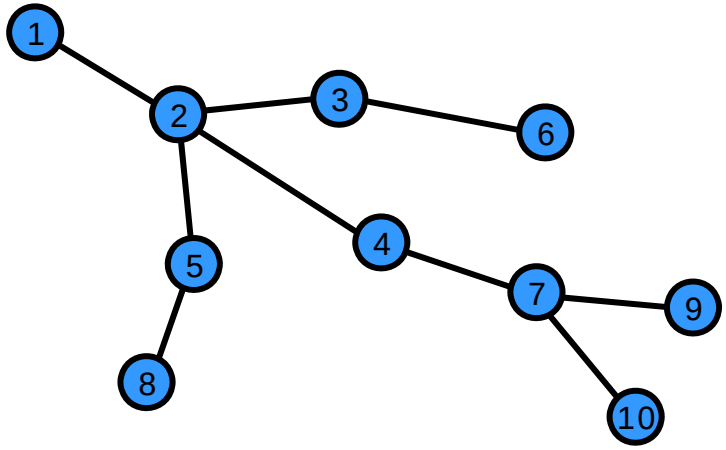
- What if we want the robot to correctly obey a traffic signal?

```
L = getLightColor();
if L == ‘red’
    robotStop();
elseif L == ‘green’
    robotDriveStraight(r, 10, 100);
elseif L == ‘yellow’
    robotDriveStraight(r, 100, 100);
else
    fprintf(‘Unknown light color\n’);
end
```

Last time

- Graph traversal





- Two types of todo lists:
  - Stacks → Depth-first search
  - Queues → Breadth-first search

Last time

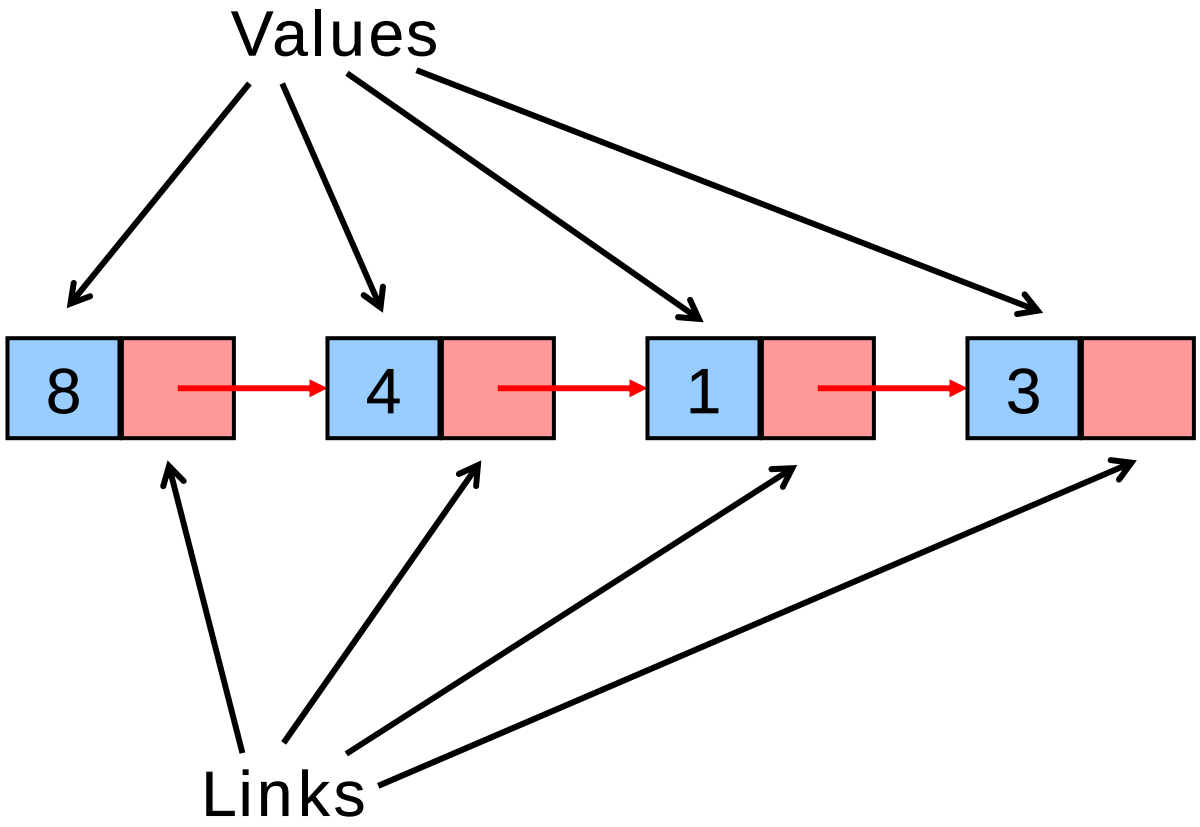
- Implementing a stack and queue using arrays
- What went wrong?
- Today we'll talk about a better approach

Linked lists

- Alternative to an array

- Every element (cell) has two parts:
  1. A value (as in an array)
  2. A link to the next cell

Linked lists



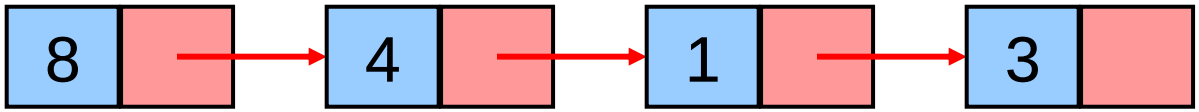
Linked lists as memory arrays

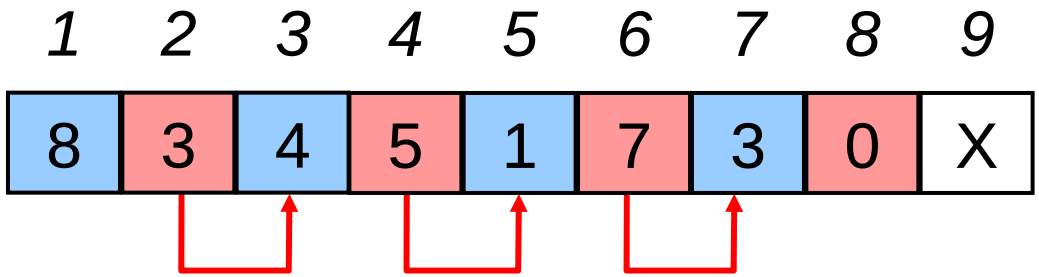
- We'll implement linked lists using M
- A cell will be represented by a pair of adjacent array entries

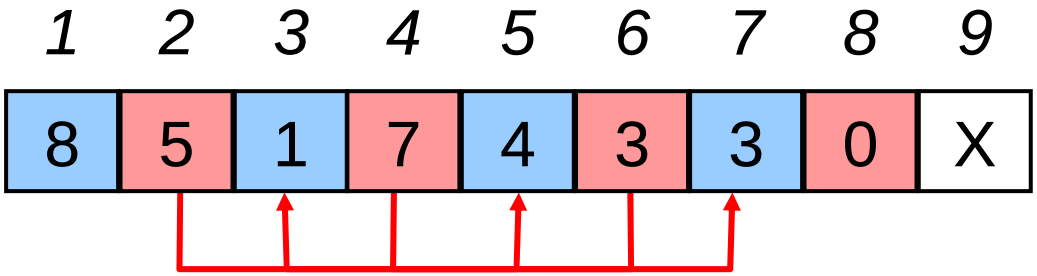
A few details

- I will draw odd numbered entries in blue and even ones in red
  - Odd entries are values
    - Number interpreted as list elements
  - Even ones are links
    - Number interpreted as index of the next cell
    - AKA *location*, *address*, or ***pointer***
- The first cell is M(1) and M(2) (for now)
- The last cell has 0, i.e. pointer to M(0)
  - Also called a “null pointer”

Example







Traversing a linked list

- Start at the first cell, **`[M(1),M(2)]`**
- Access the first value, **`M(1)`**
- The next cell is at location **`c = M(2)`**
- If **`c = 0`**, we're done
- Otherwise, access the next value, **`M(c)`**
- The next cell is at location **`c = M(c+1)`**
- Keep going until **`c = 0`**

Inserting an element – arrays

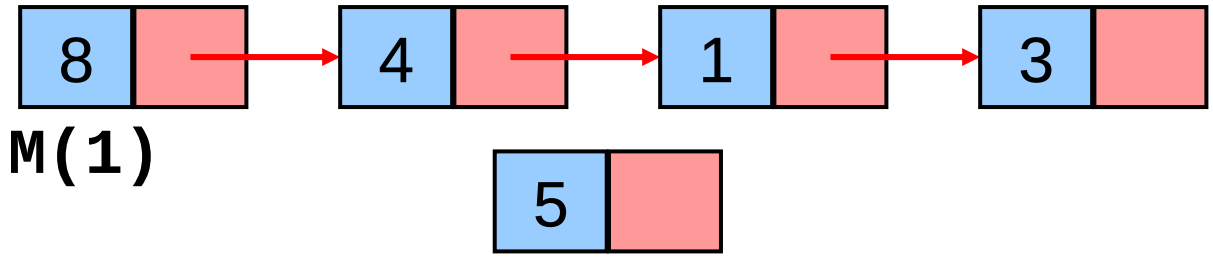
- How can we insert an element **x** into an array **A**?
- Depends where it needs to go:
  - End of the array:

```
A = [A x];
```

  - Middle of the array (say, between elements A(5) and A(6))?
  - Beginning of the array?

Inserting an element – linked lists

- Create a new cell and splice it into the list



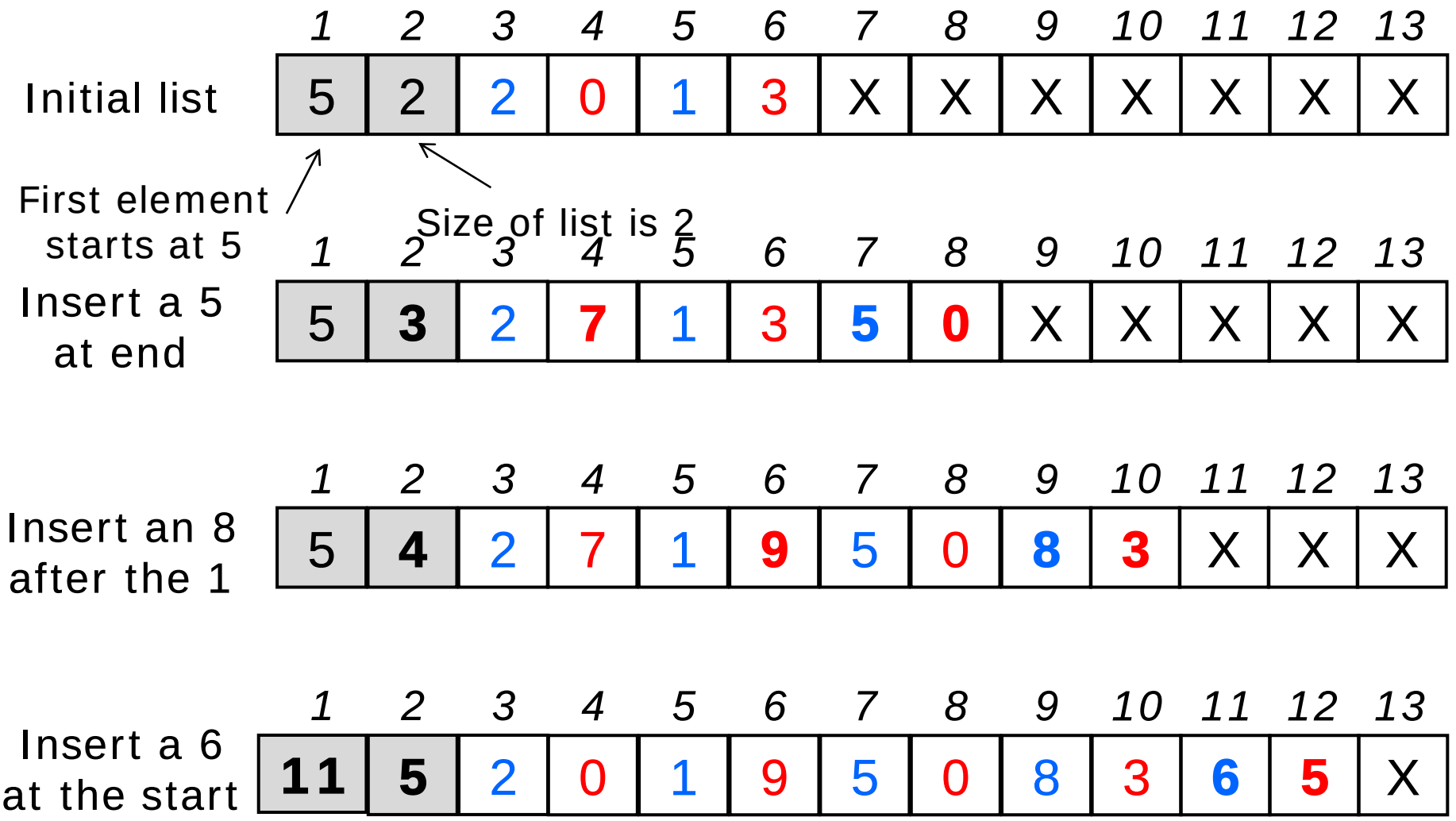
- Splicing depends on where the cell goes:
  - How do we insert:
    - At the end?
    - In the middle?
    - At the beginning?

Adding a header

- We can represent the linked list just by the initial cell, but this is problematic
  - Problem with inserting at the beginning

- Instead, we add a header – a few entries that are not cells, but hold information about the list
  1. A pointer to the first element
  2. A count of the number of elements

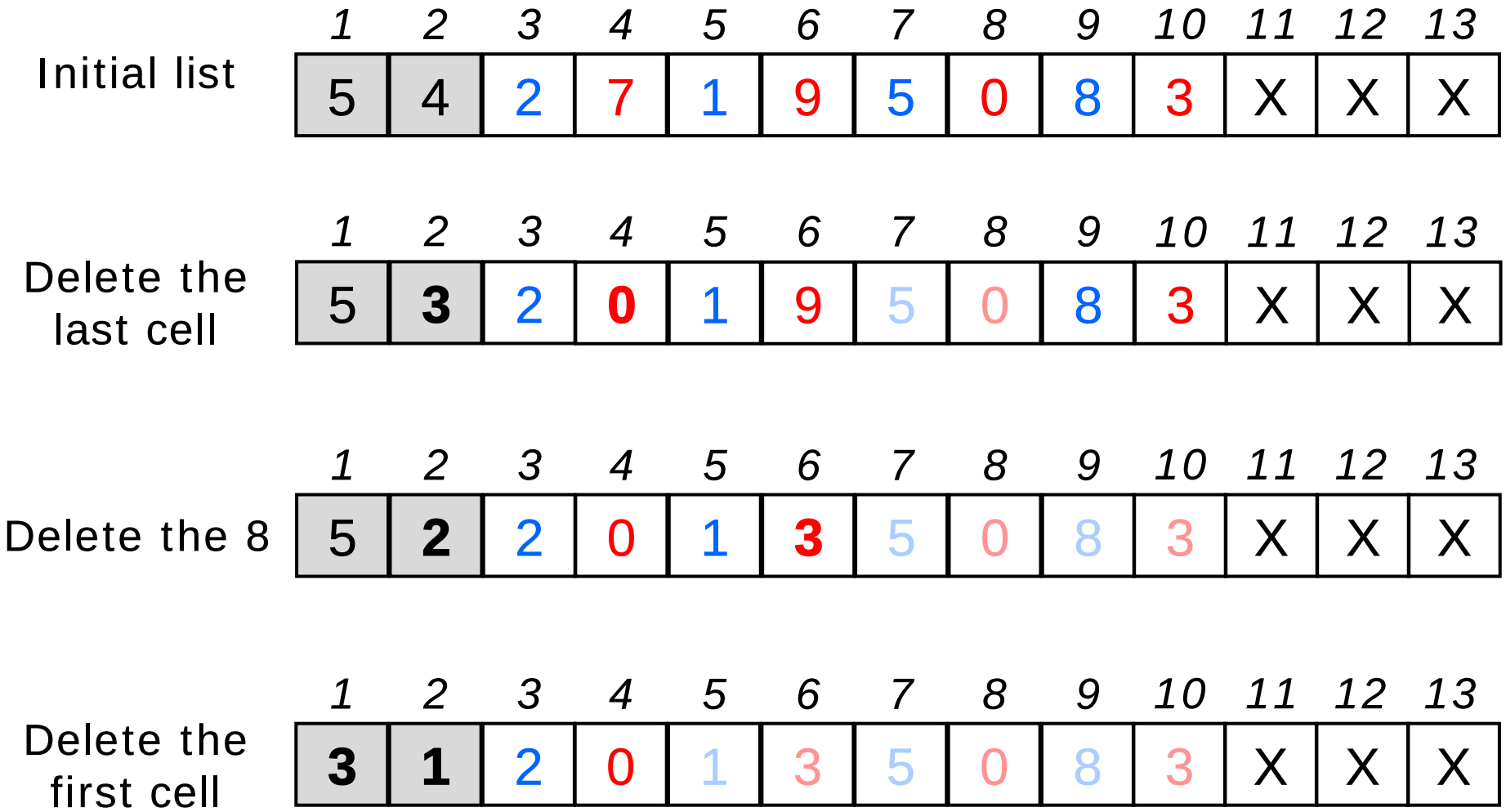
Linked list insertion



Linked list deletion

- We can also delete cells
- Simply update the header and change one pointers (to skip over the deleted element)
- Deleting things is the source of many bugs in computer programs
  - You need to make sure you delete something once, and only once

Linked list deletion



Linked lists – running time

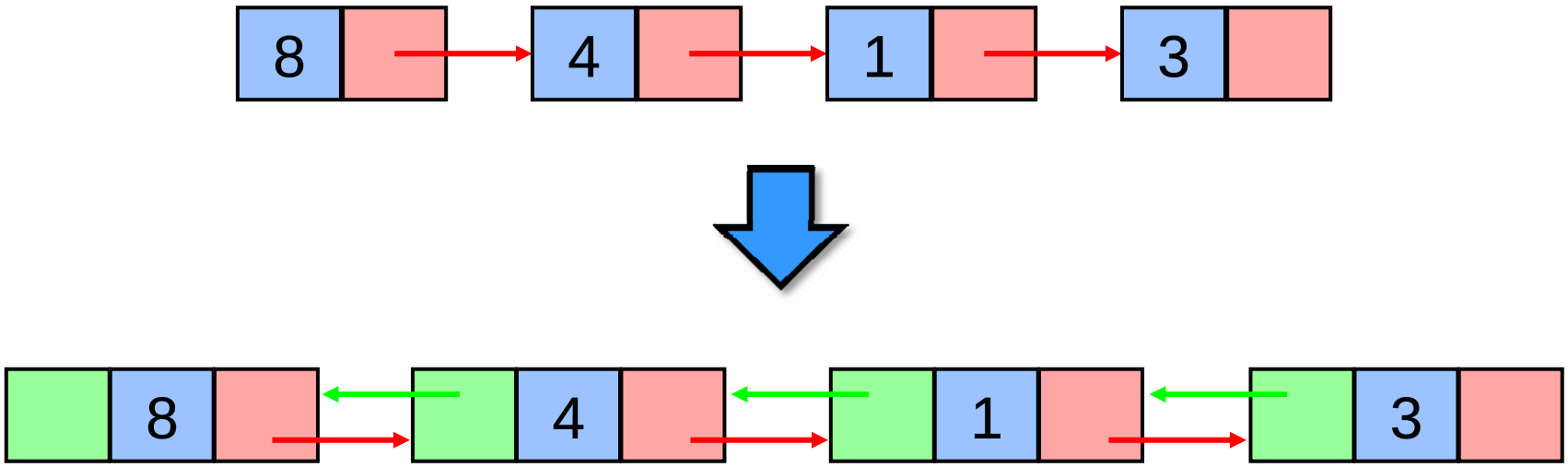
- We can insert an item (at the front) in constant (O(1)) time
  - Just manipulating the pointers
  - As long as we know where to *allocate* the cell
- We can delete an element (at the front) in constant time

Linked lists – running time

- What about inserting / deleting from the end of the list?

- How can we fix this?

Doubly linked lists



A doubly-linked list in memory