CS 1114:
Data Structures – Implementation: part 1

Prof. Graeme Bailey

http://cs1114.cs.cornell.edu

(notes modified from Noah Snavely, Spring 2009)

Cornell University
Computer Science

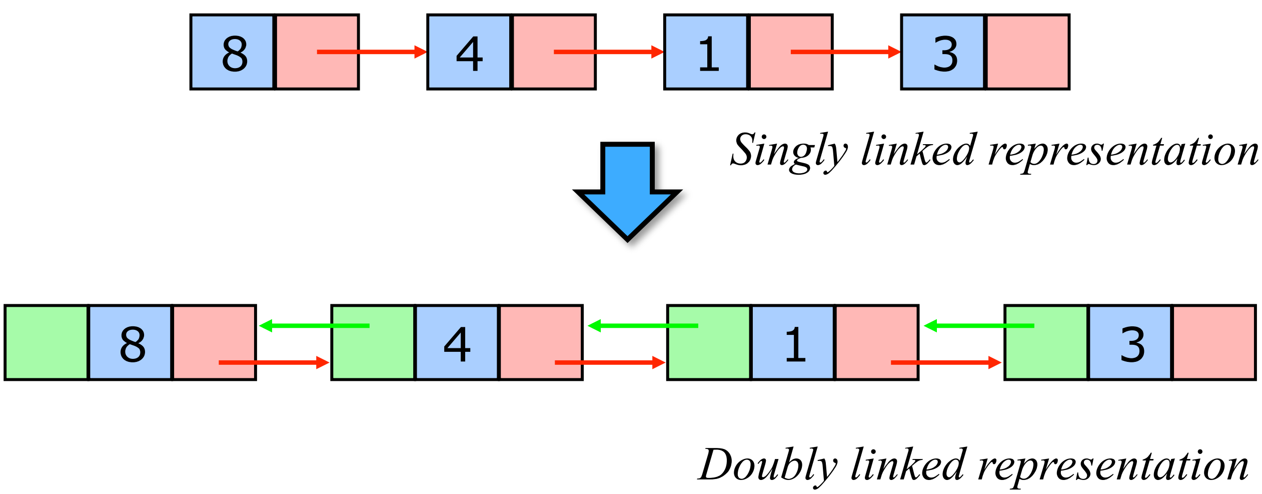
Linked lists – running time

- 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
  - If we need to insert an item *inside* the list, then we must first *find* the place to put it.
- We can delete an element (at the front) in constant time
  - If the element isn't at the front, then we have to *find* it … how long does that take?

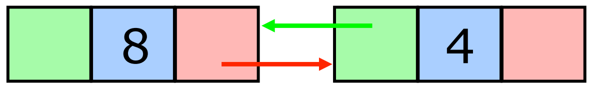
Linked lists – running time

- Finding the place to insert/delete …
- Easier case:
  - What about inserting / deleting from the *end* of the list?
- How can we fix this?

Doubly linked lists



A doubly-linked list in memory



1	2	3	4	5	6	7	8	9		
4	7	2	0	8	7	4	4	0	X	X

First element

Last element

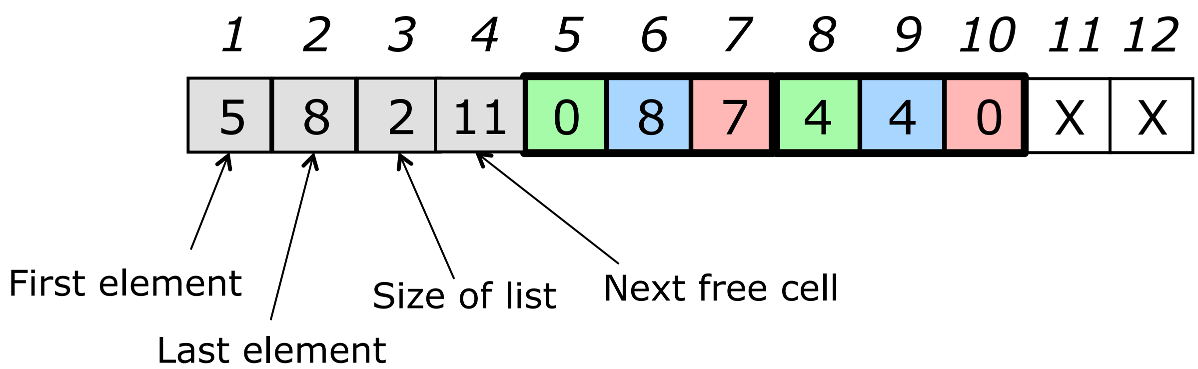
Size of list

Doubly-linked list insertion

	1	2	3	4	5	6	7	8	9	10	11	12	13
Initial list	0	0	0	X	X	X	X	X	X	X	X	X	X
Insert a 5 at end	*4*	*4*	*1*	*0*	5	*0*	X	X	X	X	X	X	X
Insert an 8 at the start	*7*	*4*	*2*	*7*	5	*0*	*1*	8	*4*	X	X	X	X

Memory allocation

- When we need a new cell, how do we know where to find it?
- We' ll keep track of a "free pointer" to the next unused cell after the list



Doubly-linked list insertion

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
	5	5	1	8	0	5	0	X	X	X	X	X	X	X

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Insert a 9 at the start	8	5	2	11	8	5	0	0	9	5	X	X	X	X

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Delete the last element	8	8	1		8	5	0	0	9	0	X	X	X	X

Memory allocation

- Current strategy: when we need more storage, we just grab locations at the end
- What can go wrong?

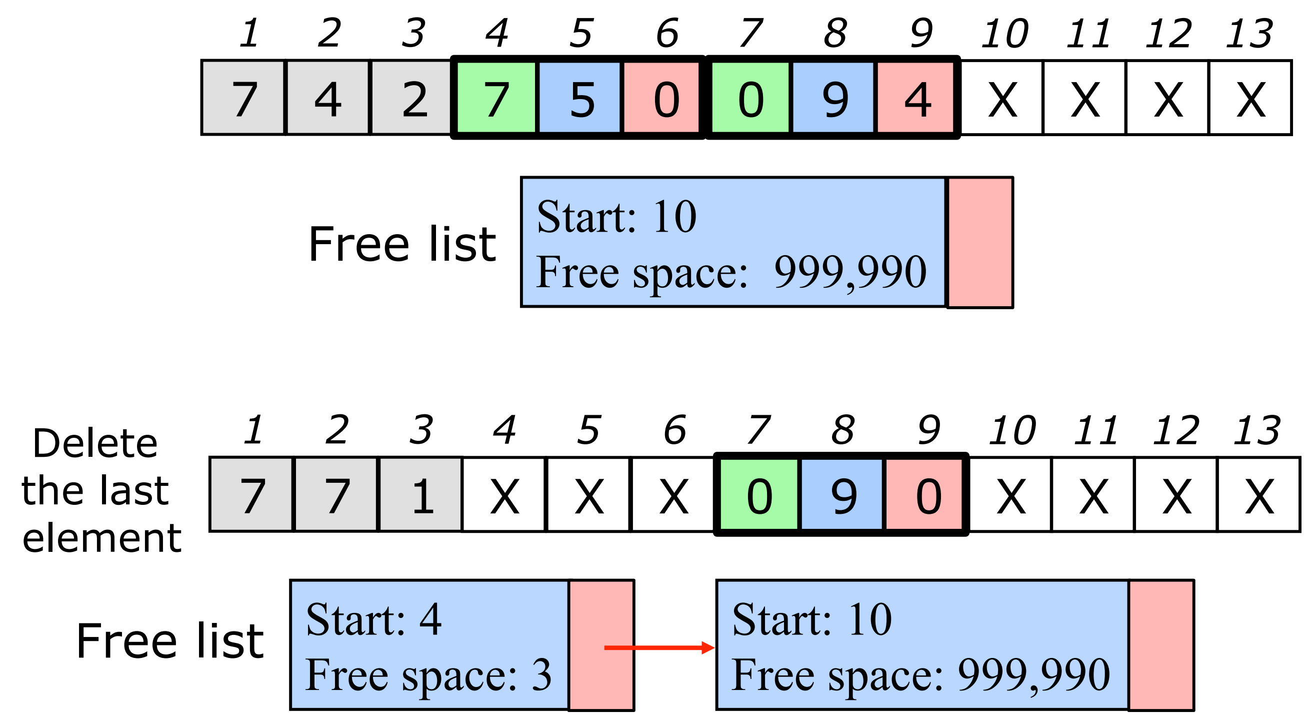
- When we delete items from a linked list we change pointers so that the items are inaccessible
  - But they still waste space!

Memory allocation

- Strategy 1: Computer keep tracks of free space at the end

- Strategy 2: Computer keeps a linked list of free storage blocks ("freelist")
  - For each block, stores the size and location
  - When we ask for more space, the computer finds a big enough block in the freelist
  - What if it doesn't find one?

Maintaining a freelist



Allocation issues

- Surprisingly important question:
  - Which block do you supply?
  - The smallest one that the users request fits into?
  - A larger one, in case the user wants to grow the array?

Memory deallocation

- How do we give the computer back a block we're finished with?

- Someone has to figure out that certain values will never be used ever ("garbage"), and should be put back on the free list

  - If this is too conservative, your program will use more and more memory ("memory leak")
  - If it's too aggressive, your program will crash ("blue screen of death")

Memory deallocation

- Two basic options:

1. Manual storage reclamation
   - Programmer has to explicitly free garbage
   - Languages: C, C++, assembler

2. Automatic storage reclamation
   - Computer will notice that you're no longer using cells, and recycle them for you
   - Languages: Matlab, Java, C#, Scheme