CSSE 220 Day 23

Exam Review
Hardy Efficiency
Doubly-linked lists

CSSE 220 Day 23

- Reminder: Exam #2 is this Friday
  - Can start 7:15 am
  - One piece of paper with handwritten notes for the first part.
  - Same resources as last time for programming part.
- Markov Milestone 2 due Saturday 5 PM
- Begin thinking about Spell-check program
- Please do the Mini-project partner surveys this morning if you haven’t yet

Today's Agenda

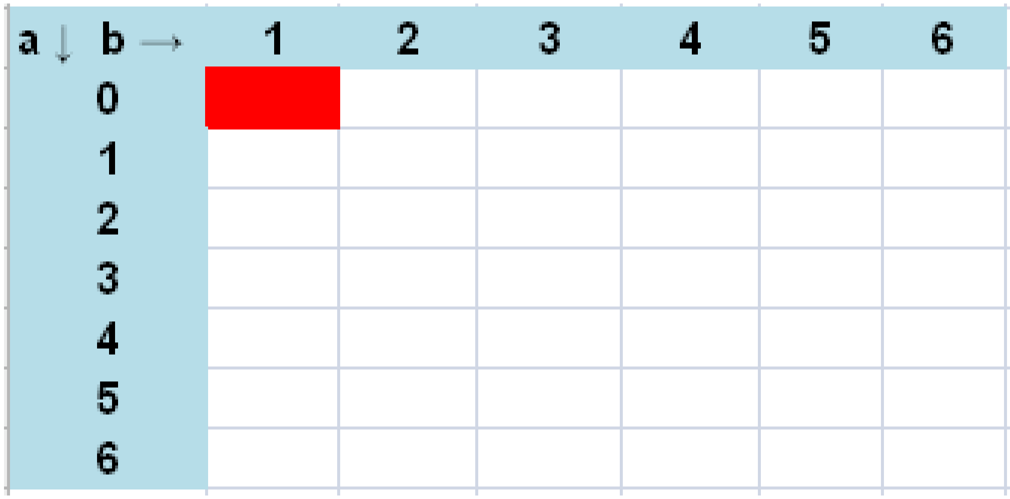
- Answers to your questions in preparation for the exam
- A look at some Hardy solutions
- and empirical analysis.
- More on Linked Lists

Answers to your questions

- Abstract Data Types and Data Structures
- Collections and Lists
- Markov
- Exam
- Material you have read
- Anything else

A Hardy Algorithm

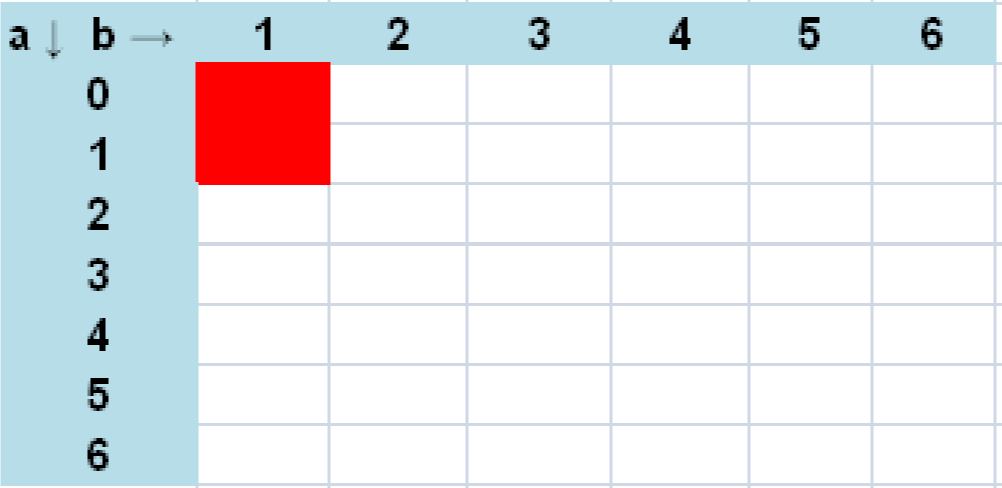
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1						
2						
3						
4						
5						
6						

A Hardy Algorithm

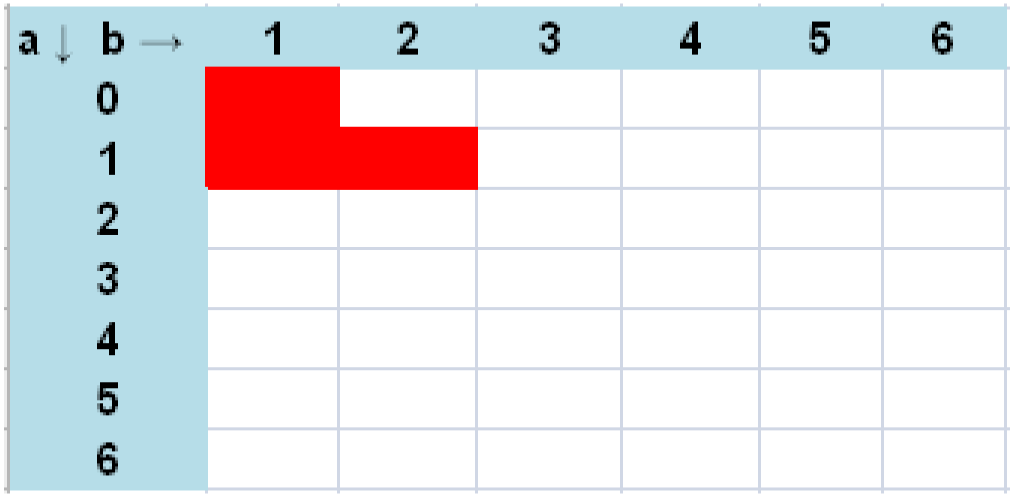
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥					
2						
3						
4						
5						
6						

A Hardy Algorithm

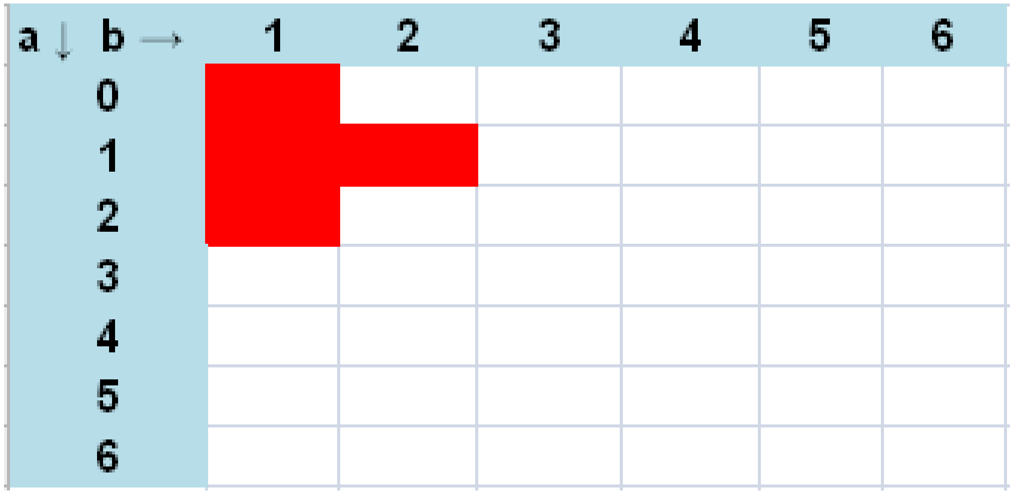
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥	🟥				
2						
3						
4						
5						
6						

A Hardy Algorithm

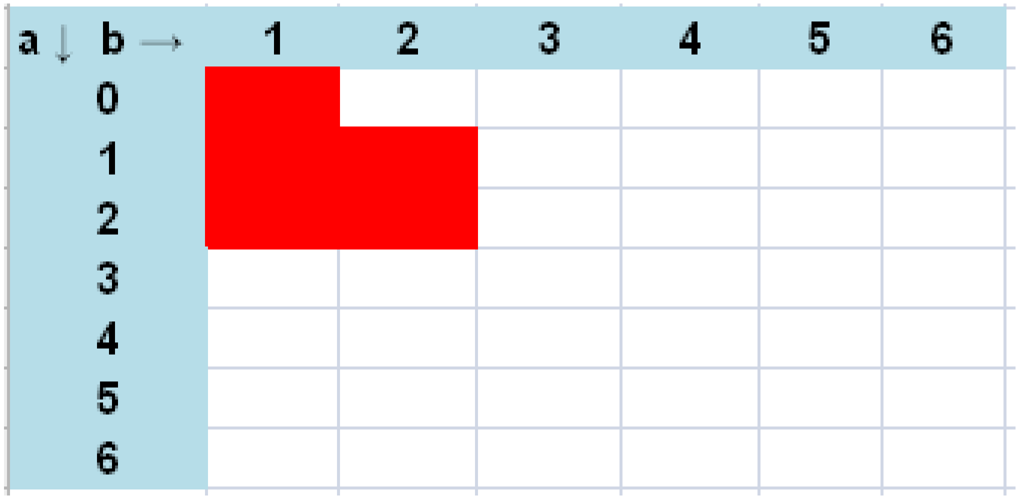
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥	🟥				
2	🟥					
3						
4						
5						
6						

A Hardy Algorithm

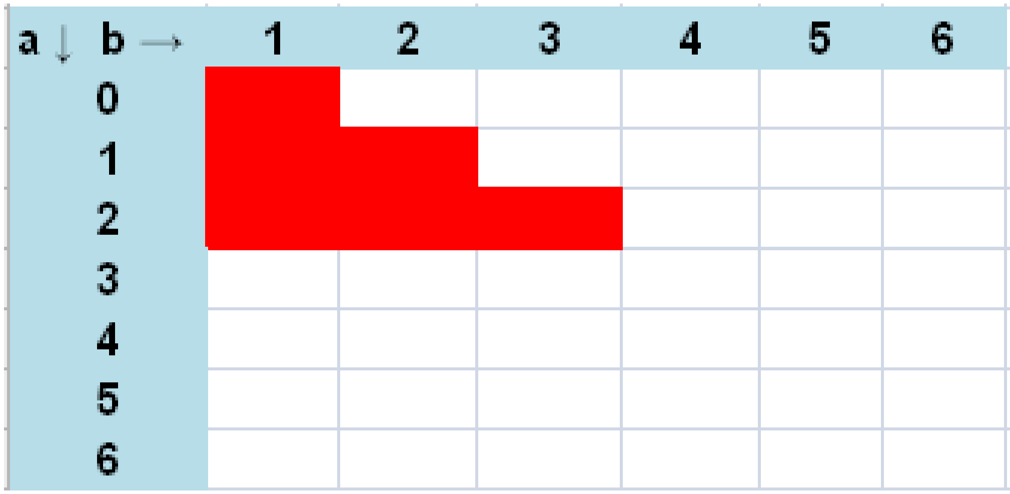
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥	🟥				
2	🟥	🟥				
3						
4						
5						
6						

A Hardy Algorithm

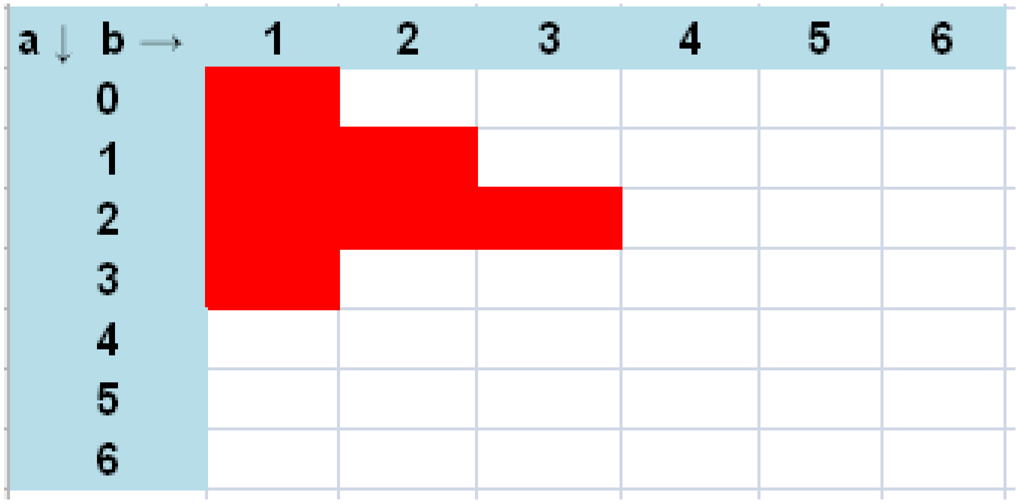
- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥	🟥				
2	🟥	🟥	🟥			
3						
4						
5						
6						

A Hardy Algorithm

- total = $a^3 + b^3$.
- One way to move through **a** and **b** loops:



a ↓ b →	1	2	3	4	5	6
0	🟥					
1	🟥	🟥				
2	🟥	🟥	🟥			
3	🟥					
4						
5						
6						

Hardy Algorithm basic idea

- Go through the values of a and b in the order just described
- When we calculate each total
  - Look in table if we have seen that total before
  - If not, record its triple: (a, b, total) in table.
  - If so, record in the duplicates table
- When we get N items in the duplicates table
  - They may not be the N smallest. Sort them
  - See if we can find any others with sums smaller than the max of those N.
    - If, so, they will all have a **b** that is less than the cube root of this max. Find all of those and add to duplicates table.
- Sort again and pick out the Nth one.

Hardy Code

- Look at them together
- Ask questions about anything you don't understand.
- I'll ask you questions.
- We'll show some timing computations.
- Then see how much of a speed-up we get by using a faster data structure

Break

- An inside joke

An iterator for LinkedLists

```
 public Iterator<T> iterator() {
     return new LinkedListIterator();
   }

   class LinkedListIterator implements Iterator<T> {

     private ListNode<T> current, previous;

     private LinkedListIterator() {
       current = header;
     }

     public boolean hasNext() {
       return current.next !=null;
     }


```

```
public T next() {
      T val = (current.next.element);
      previous = current;
      current = current.next;
      return val;
    }

    public void remove() {
      if (previous == null)
        throw new NoSuchElementException("You can only call an iterator's remove method after a call to next()");
      previous.next = current.next;
      current = previous;
      previous = null;
    }
  }
```

Doubly-linked list

- Each node has two pointers, **prev** and **next.**
- There is one other new node, **tail**, whose **prev** pointer points to the node containing the last element of the list.
- This makes `remove()` easier to write
    - and it also makes an efficient **ListIterator** possible.

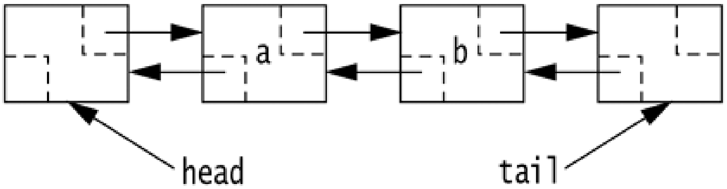


figure 17.15

A doubly linked list

Rest of class

- Work on LinkedLists
- Work on Markov justification