Word Clouds cont.

ArrayList

- Constructors

```
ArrayList lst1 = new ArrayList();
ArrayList lst2 = new ArrayList(int initialSize);
ArrayList<String> strList = new ArrayList();
```

- Parameterized type
  - use if you know the type of the list and the list type is not mixed
- Methods

```
size()                          // Returns the num of items held.
add(Object o)                   // Appends o to end.
add(int idx, Object o)          // Inserts o at pos idx.
remove(int idx)         // Removes item at pos idx.
get(int idx)                    // Gets items at idx. No removal.
set(int idx, Object o)          // Replaces item at idx with o.
clear()                         // Removes all items.
isEmpty()               // true if empty.
toArray()                 // returns an array that contains
                          // the contents of the list
```

Removing items from ArrayList while iterating

- When an item is removed from an `ArrayList`, the list shrinks and the indices are renumbered behind the removed item
- Why doesn't this removal work?

```
for (int i=0; i<lst.size(); i++) {
    lst.remove(i);
}
```

- Must remove from the back to the front

```
for (int lst.size()-1; i>=0; i--) {
    lst.remove(i);
}
```

The word class

```
class Word {
  // Each Word is a pair: the word, and its frequency
  String word;
  int freq;
  Word(String newWord) {  // Constructor
    word = newWord;
    freq = 1;
  } // Word()
  String getWord() {
    return word;
  } // getWord()
  int getFreq() {
    return freq;
  } // getFreq()
  void incr() {  // increments the word count
    freq++;
  } // incr()
  String toString() {  // print representation of Word objects
    return "<"+word+", "+freq+">";
  }
} // class Word
```

Make the set using an ArrayList

```
ArrayList<Word> wordFrequency = new ArrayList();

// Compute the wordFrequency table using tokens
for (String t : tokens) {
  // See if token t is already a known word
  int index = search(t, wordFrequency);
  if (index >= 0) {
    wordFrequency.get(index).incr();
  }
  else {
    wordFrequency.add(new Word(t));
  } // if
} // for
```

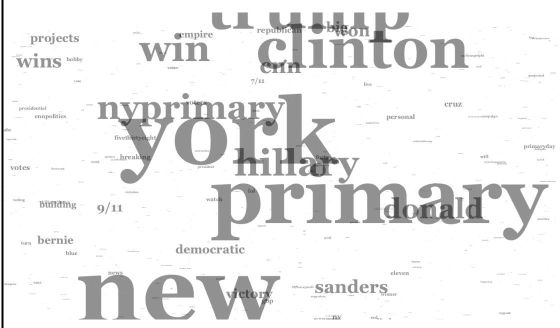
Stop words removal

- The most common short function words
  - the, is, a, at, which, on, etc
  - usually filtered out
- Usually given in a additional file and read in
- The list is not unique or definitive

```
fileText = loadStrings("stopwords.txt");
stopwords = new ArrayList(fileText.length);

for (int i=0; i < fileText.length; i++) {
  stopwords.add(fileText[i].toLowerCase());
}
```

makeUnique without stop words

```
void makeUnique(String[] words) {
  uniqueWords = new ArrayList();

  for (int i=0; i < words.length; i++) {
   if (!stopwords.contains(words[i])) {
      int idx = contains(words[i], uniqueWords);

      if (idx < 0) {
        uniqueWords.add(new Word(words[i]));
      }
      else {
        uniqueWords.get(idx).inc();
      }
    }
  }
}
```



Sorting

- Any process of arranging items in sequence
- Build-in `sort()`
  - Works on arrays of simple types, i.e. `int`, `float` and `String`
  - `float[] a = {3.4, 3.6, 2, 0, 7.1};`
  - `a = sort(a);`
  - `String[] s = {"deer", "elephant", "bear", "aardvark", "cat"};`
  - `s = sort(s, 3);`
- Convenient, but not very flexible
- Recall that we have an `ArrayList<Word>`

Implement your own sort

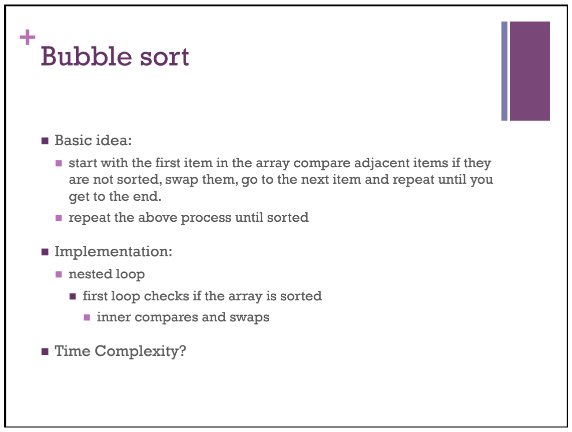
- Many sorting algorithms
- Bubble Sort
  - Looks at items in successive pairs
  - Swap if in the wrong order
- Selection Sort
  - Scan a list start to end and find the value that should come first
  - Swap that item into the first position
  - Repeat scanning and swapping starting at the second position in the list
- Insertion Sort

Sorting (implement your own)

- Easy to code (but slow)
  - Selection Sort
  - Bubble Sort
  - Insertion Sort
- Animations
  - https://www.cs.usfca.edu/~galles/visualization/ComparisonSort.html
  - http://www.sorting-algorithms.com/

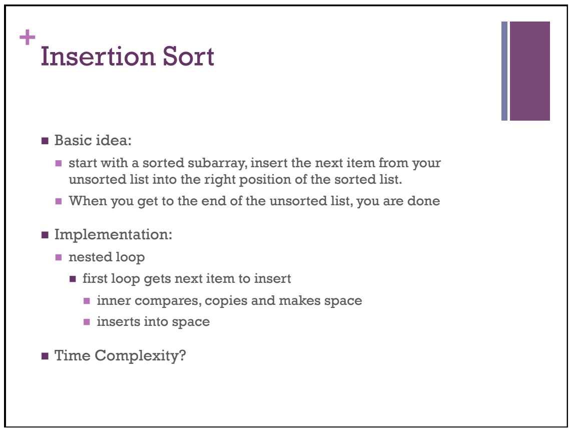
Selection sort

- Basic idea:
  - step forward on each item of the array starting with the first item, if there is a smallest item in front of the item being stepped on, then swap the two items. Repeat until you've stepped on every item.
- Implementation:
  - nested loop
    - first loop marks the current item
      - inner loop finds the smallest item between the current item and the last item inclusively, then swaps the items
- Time Complexity?



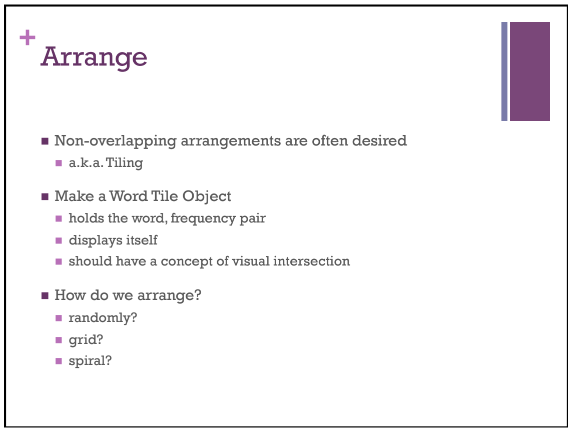
Bubble sort

- Basic idea:
  - start with the first item in the array compare adjacent items if they are not sorted, swap them, go to the next item and repeat until you get to the end.
  - repeat the above process until sorted
- Implementation:
  - nested loop
    - first loop checks if the array is sorted
      - inner compares and swaps
- Time Complexity?



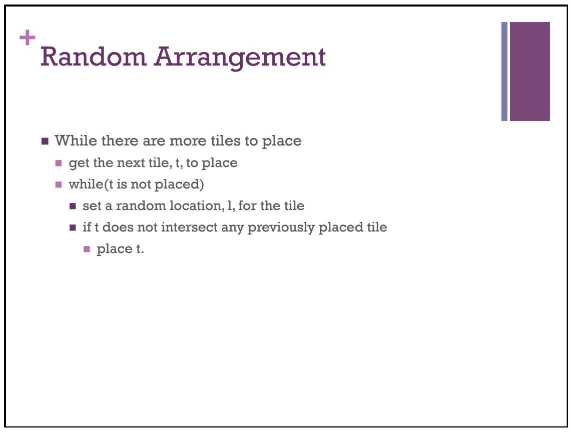
Insertion Sort

- Basic idea:
  - start with a sorted subarray, insert the next item from your unsorted list into the right position of the sorted list.
  - When you get to the end of the unsorted list, you are done
- Implementation:
  - nested loop
    - first loop gets next item to insert
      - inner compares, copies and makes space
      - inserts into space
- Time Complexity?



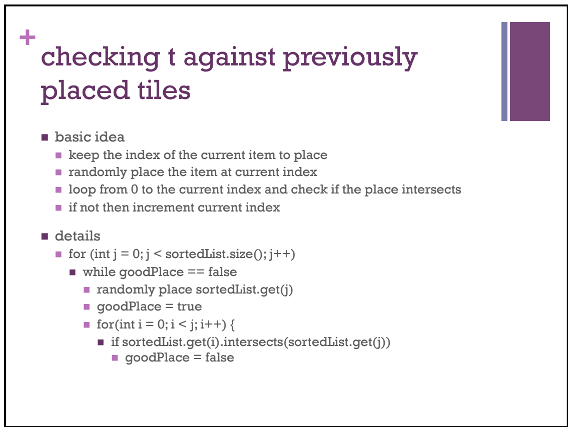
Arrange

- Non-overlapping arrangements are often desired
  - a.k.a. Tiling
- Make a Word Tile Object
  - holds the word, frequency pair
  - displays itself
  - should have a concept of visual intersection
- How do we arrange?
  - randomly?
  - grid?
  - spiral?



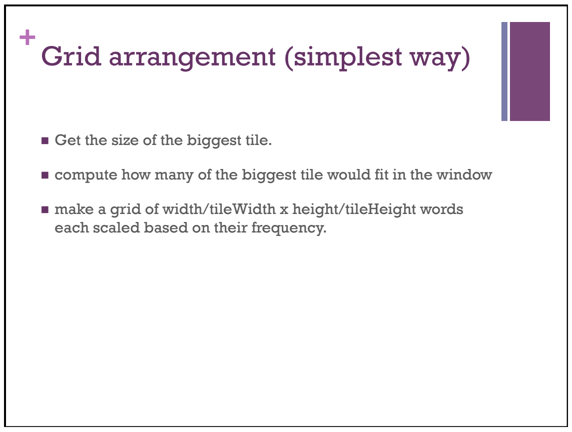
Random Arrangement

- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set a random location, l, for the tile
    - if t does not intersect any previously placed tile
      - place t.



checking t against previously placed tiles

- basic idea
  - keep the index of the current item to place
  - randomly place the item at current index
  - loop from 0 to the current index and check if the place intersects
  - if not then increment current index
- details
  - for (int j = 0; j < sortedList.size(); j++)
    - while goodPlace == false
      - randomly place sortedList.get(j)
      - goodPlace = true
      - for(int i = 0; i < j; i++) {
        - if sortedList.get(i).intersects(sortedList.get(j))
          - goodPlace = false



Grid arrangement (simplest way)

- Get the size of the biggest tile.
- compute how many of the biggest tile would fit in the window
- make a grid of width/tileWidth x height/tileHeight words each scaled based on their frequency.

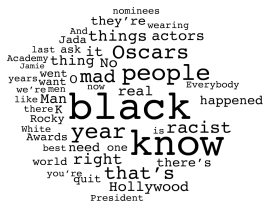
Grid arrangement (slightly tougher way)

- Get the size of the biggest tile.
- compute how many, M, of the biggest tile would fit in the sketch
- if N > M, then change the maximum font size of a tile so that a grid of the largest tile size would allow for N tiles on the sketch
- make a grid based on new tile sizes.

Spiral Arrangement

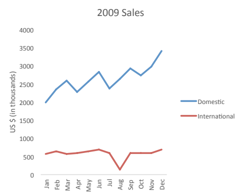
- Sort the tiles from largest to smallest.
- While there are more tiles to place
  - get the next tile, t, to place
  - while(t is not placed)
    - set location, l, for the tile to be at the current spiral location
    - if t does not intersect any previously placed tile
      - place t.
    - update the current spiral position outward by a fixed step size.

Chris Rock @ Oscars



Sales Data (US $ in thousands)

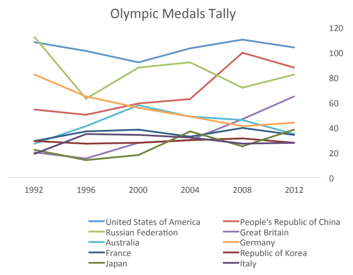
Region	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Domestic	1983	2343	2593	2283	2574	2838	2382	2634	2938	2739	2983	3413
International	574	636	573	593	644	679	593	139	599	583	602	690



Top Medals in Olympics by Country (1992-2012)

Country	2012	2008	2004	2000	1996	1992
United States of America	104	110	103	92	101	108
People's Republic of China	88	100	63	59	50	54
Russian Federation	82	72	92	88	63	112
Great Britain	65	47	30	28	15	20
Australia	35	46	49	58	41	27
Germany	44	41	49	56	65	82
France	34	40	33	38	37	29
Republic of Korea	28	31	30	28	27	29
Japan	38	25	37	18	14	22
Italy	28	27	32	34	35	19

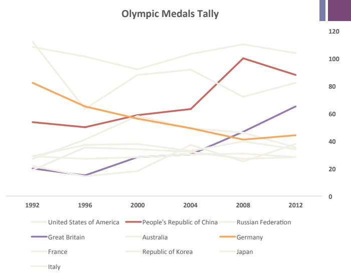
Top Medals in Olympics (1992-2012)



Top Medals in Olympics (1992-2012)

Focus on

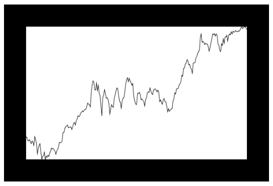
- Germany
- China
- Great Britain



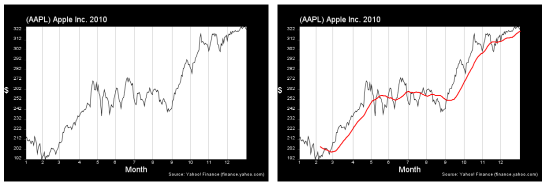
Example: Visualizing Time Series

Raw Data

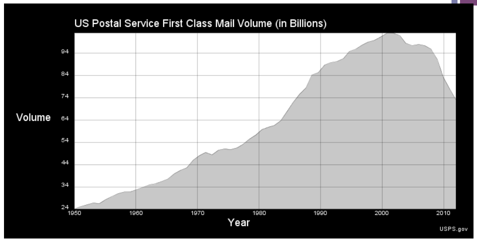
```
1,4,2010,213.43,214.50,214.01,17633200
1,5,2010,214.60,215.59,214.38,21496600
1,6,2010,214.38,215.23,210.97,19720000
1,7,2010,211.75,212.00,210.58,17040400
1,8,2010,210.30,212.00,211.98,15986100
…
12,27,2010,322.85,325.44,324.68,8922000
12,28,2010,325.91,326.66,325.47,6283000
12,29,2010,326.22,326.45,325.29,5826400
12,30,2010,325.48,325.51,323.66,5624800
12,31,2010,322.95,323.48,322.56,6911000
```



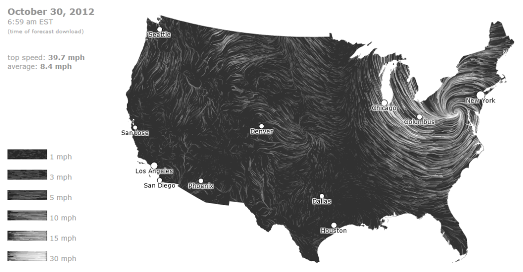
Visualizing Time Series



Visualizing Time Series



Data Visualization - Horizons


October 30, 2012
6:59 am EST
(time of forecast download)
top speed: 39.7 mph
average: 8.4 mph
1 mph
3 mph
5 mph
10 mph
15 mph
30 mph
Seattle
San Jose
Denver
Chicago
Columbus
New York
Los Angeles
San Diego
Phoenix
Dallas
Houston