Sorting Introduction: The Fellowship of Orderliness

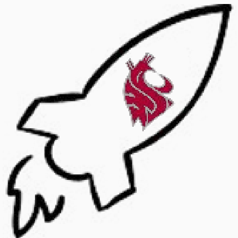
CptS 223 - Fall 2017 - Aaron Crandall

Today’s Agenda

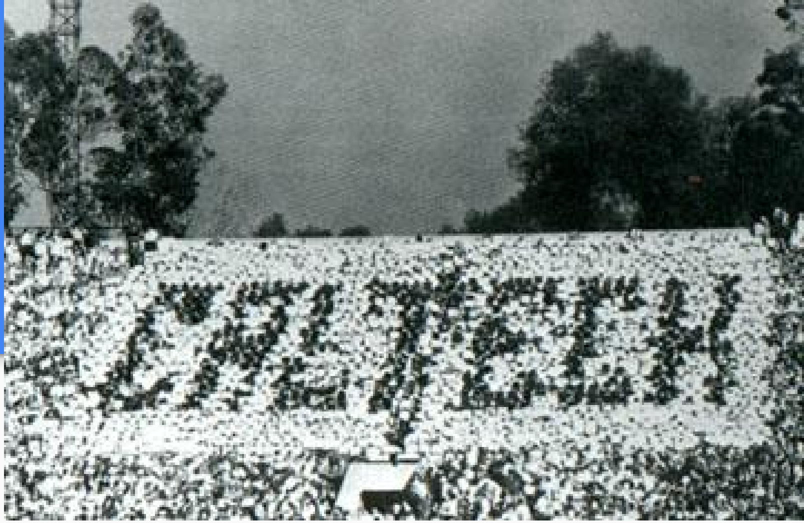
- Announcements
- Thing of the day
- Sorting is fun!
    - fginnorSstu!

Announcements

- You’ve got PA3 and MA4 in your hot hands, so keep hacking away.
- Yes, grading is continuing to catch up.
- There’s plenty of companies visiting campus this week.
  - Emsi was yesterday
  - Hulu and others are around. I sent you email about it



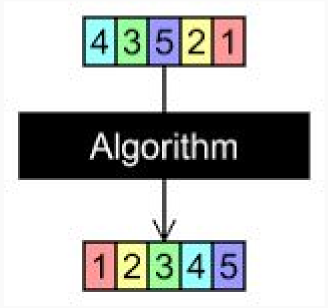
Great Rose Bowl Hoax

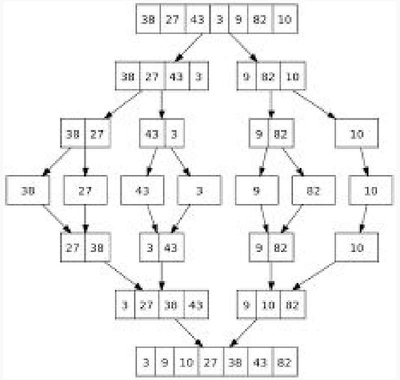


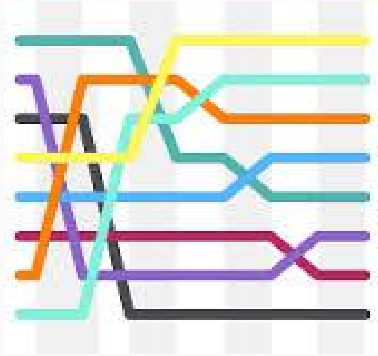
- Prank at the 1961 Rose Bowl CalTech
- UW Huskies playing Minn Golden Gophers
- Cheerleaders at halftime do a card stunt… it came up CALTECH, not UW
- Engineering CalTech student Lyn Hardy disguised himself as a reporter for a local LA high school, and asked Washington's head cheerleader. They learned that they would be able to trick unsuspecting Washington fans into holding up the incorrect signs by changing the 2,232 instruction sheets.
- Lolz

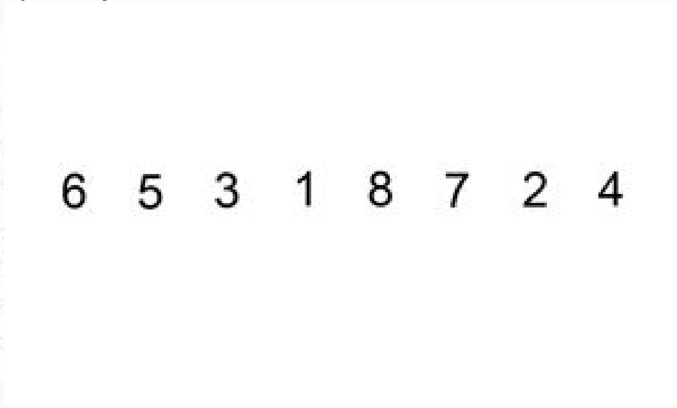
Sorting! - Chapter 7
Concerning Sorting

- A common need is to take unordered data and order it
- Requires comparing elements and re-ordering them as needed
- Speed in this operation remains highly valuable
  - Trees are often used for this when keeping data long term, not just one-off sorts









Background stuff

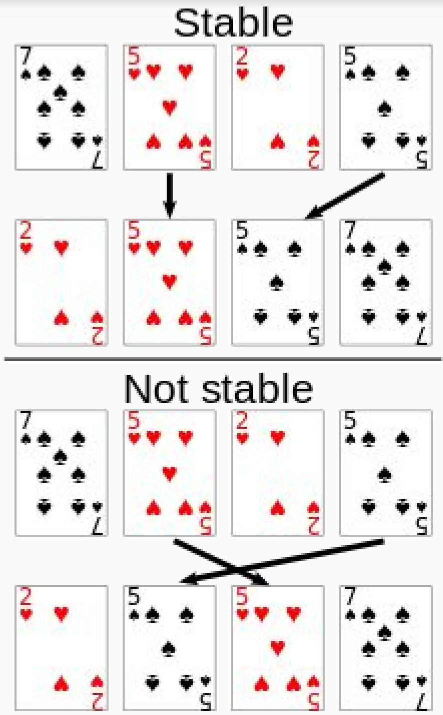
- We’ll assume everything fits into memory until stated otherwise
- There’s easy sorting algorithms in O(N^2) time:
  - Insertion sort, Bubble sort
- There’s more complex algorithms in O(N log N) time:
  - Quick sort, merge sort, heap sort
- Shellsort is cool and it’s in o(N^2) time (that’s actually little-o time)
- Code assumes we can always use ‘<’ and ‘>’ to compare elements
  - Also, that we have an assignment operator ‘=’
  - This is the field of comparison-based sorting

Why this minimum on O(N log N)?

- How many comparisons do you have to do to sort?
  - Seems to be Ω(N log N), so things must have that minimum bound
- There's exemptions to this rule for special cases (mainly special data):
  - Counting sort in O(N)
  - Radix sort in O(N)-ish
- Sorting by paired sums gets around the Ω(N log N) comparisons:
  - But gets Ω(N^2 log N) time, though with O(N^2) comparisons

The difference between Stable and Unstable sorting algorithms

- You'll see the terms Stable and Unstable sorting
- "Stable" means that items with the same value will retain their order from input to sorted order
- "Unstable" makes no guarantees about final ordering for equivalent items
- Why does this matter?
    - Some applications require stable sorting
    - See priority queues and such



What’s our interface like?

- All sorting algorithms here will have the same interface:
  - void sort(vector<Comparable> & a)
  - The only thing that changes is which algorithm happens under the hood
- The STL uses iterators to specify a subset (or all) of the vector to sort:
  - void sort(Iterator begin, Iterator end)

Worst Sorting Algorithm: Bogosort

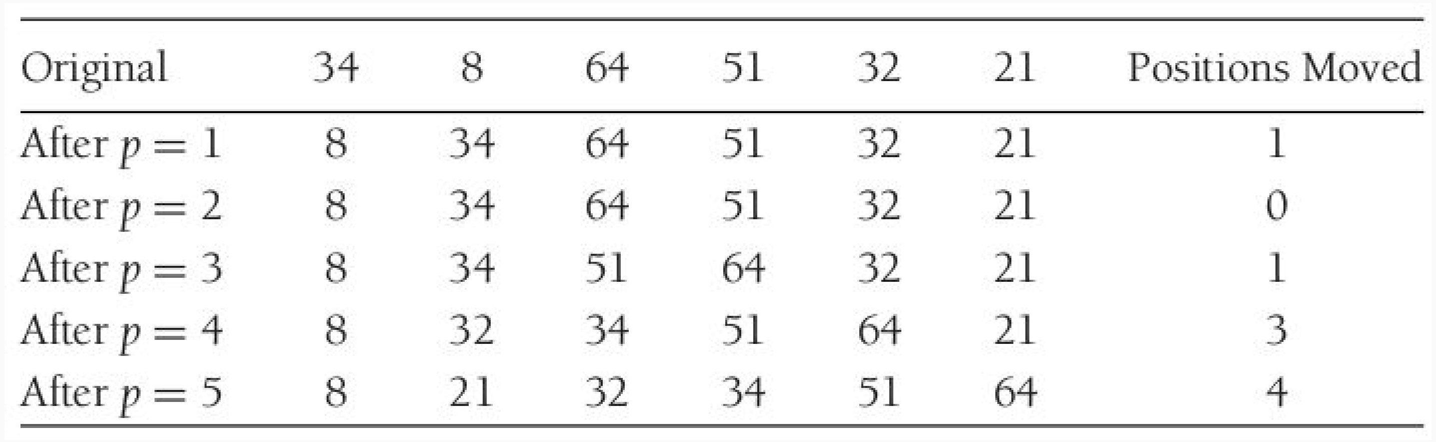
"permutation sort, stupid sort, slowsort, shotgun sort or monkey sort"

- Not a valid algorithm as it does not terminate in a finite amount of time
- Deterministic: permutate all possible orders, then check for sorted one
- Random: randomly shuffle list and check for ordered, repeat until sorted
- Psudocode: while not isInOrder(deck) then shuffle(deck)
- Expected number of comparisons asymptotically equivalent to
  - $(e - 1) * n!$
- Expected number of swaps:
  - $(n - 1) * n!$

https://www.desmos.com/calculator/mggie4b30q

Our first (well... second after BS) non-terrible algorithm: Insertion sort

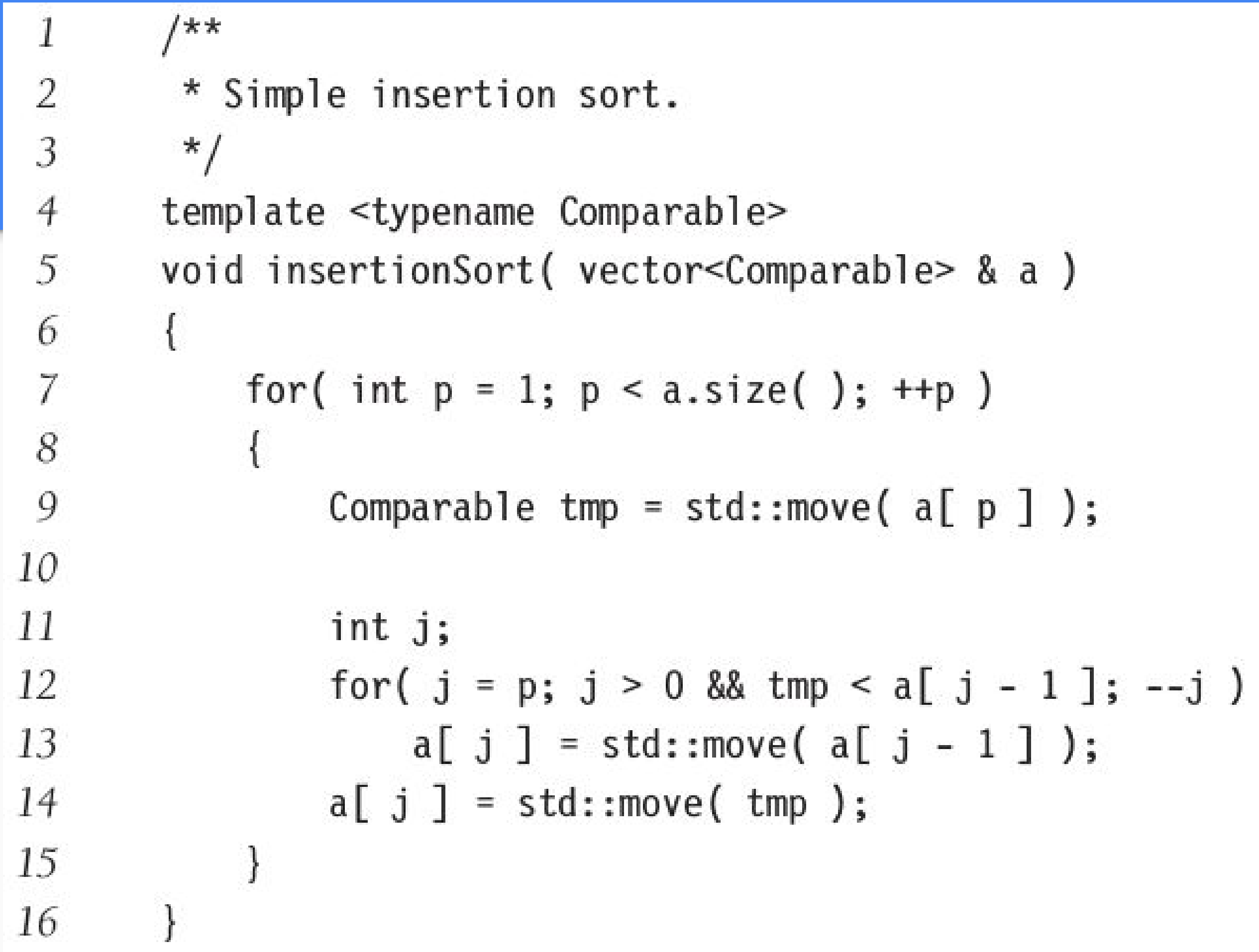
- Works down the vector, pushing elements up until they're no longer smaller then the next element in the vector
- It's effectively the same algorithm as percolating up in Heaps.



Original	34	8	64	51	32	21	Positions Moved
After $p = 1$	8	34	64	51	32	21	1
After $p = 2$	8	34	64	51	32	21	0
After $p = 3$	8	34	51	64	32	21	1
After $p = 4$	8	32	34	51	64	21	3
After $p = 5$	8	21	32	34	51	64	4

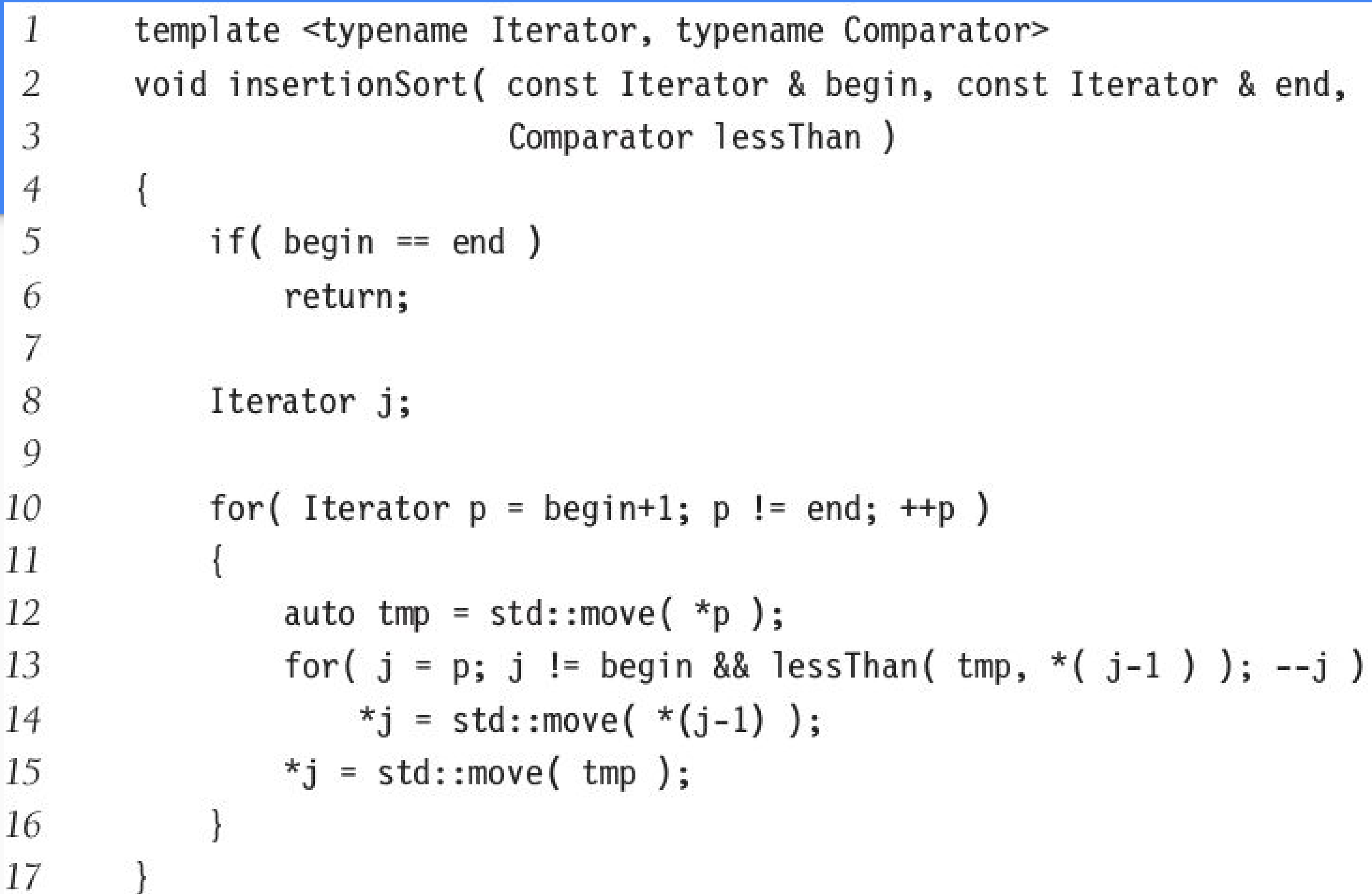
A Simple Insertion Sort

- Is it faster than BS?
- BS with swap flag?
- Is it stable?
- Issues with Vector?
- List version vs BS?
- Presorted speed?



```
/**
 * Simple insertion sort.
 */
template <typename Comparable>
void insertionSort( vector<Comparable> & a )
{
    for( int p = 1; p < a.size( ); ++p )
    {
        Comparable tmp = std::move( a[ p ] );

        int j;
        for( j = p; j > 0 && tmp < a[ j - 1 ]; --j )
            a[ j ] = std::move( a[ j - 1 ] );
        a[ j ] = std::move( tmp );
    }
}
```

STL version



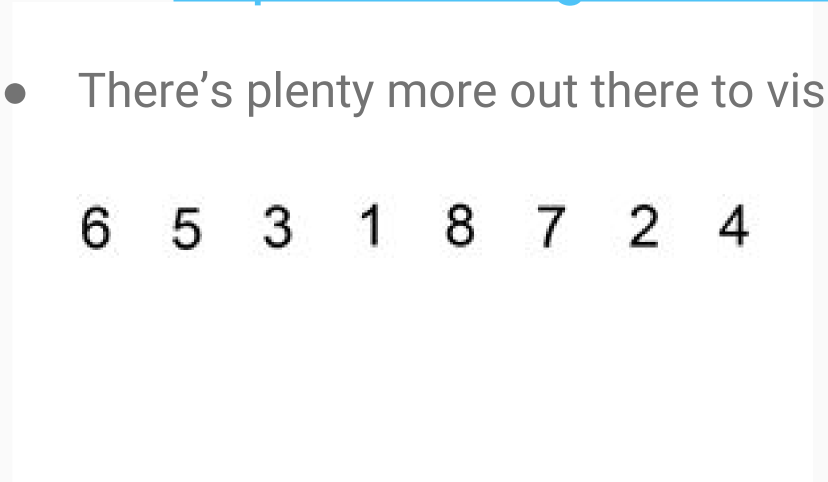
```
template <typename Iterator, typename Comparator>
void insertionSort( const Iterator & begin, const Iterator & end,
                    Comparator lessThan )
{
    if( begin == end )
        return;

    Iterator j;

    for( Iterator p = begin+1; p != end; ++p )
    {
        auto tmp = std::move( *p );
        for( j = p; j != begin && lessThan( tmp, *( j-1 ) ); --j )
            *j = std::move( *(j-1) );
        *j = std::move( tmp );
    }
}
```

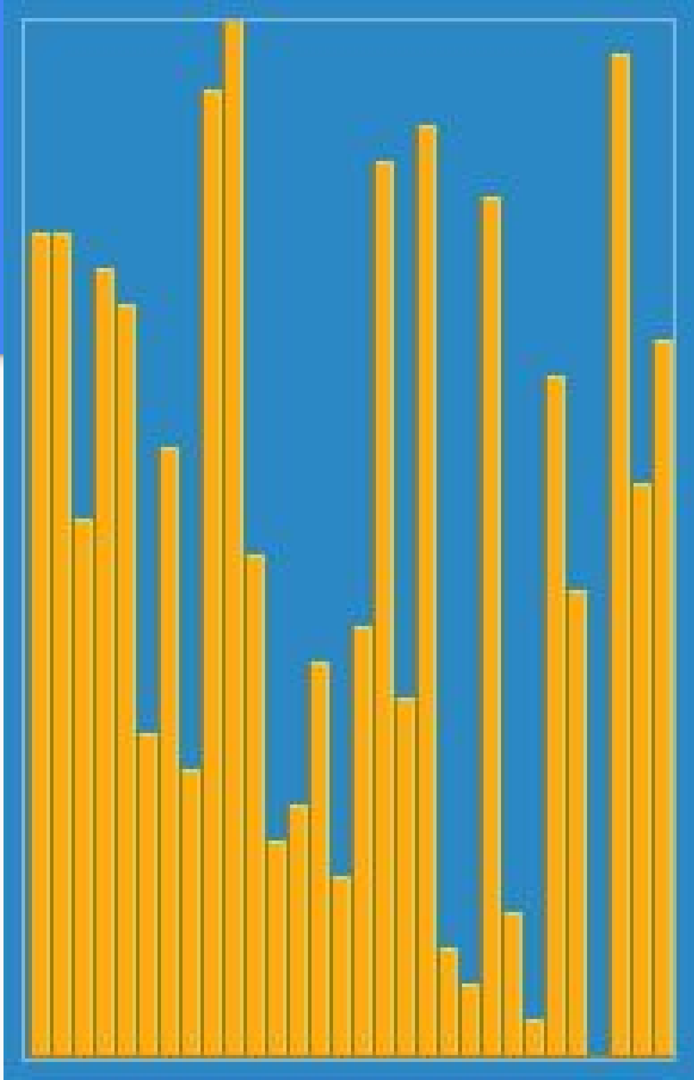
Back to Visualgo.net!

- People love to visualize sorting algorithms

  https://visualgo.net/sorting

- There's plenty more out there to visualize these





Analysis of Insertion sort

- Each loop can take N iterations, so it's O(N^2) worst case
- But… Inner loop only goes at most p+1 for each p, so a sum of the series:

$$\sum_{i=2}^{N} i = 2 + 3 + 4 + \cdots + N = \Theta(N^2)$$

- If data is already presorted, the inner loop only goes for 1 comparison each
  - This gives a total of O(1) * O(N) = O(N) operations
- Overall average case is O(N^2)
  - Along with many other simple sorting algorithms

Friday’s plan: moar sorting!

Shell sort notes and Heap sort in depth, plus how to analyze sorting algorithms

Worstsort:

> is a pessimal sorting algorithm that is guaranteed to complete, assuming that the universe does not burn itself out before completion of the algorithm; however, there is no computable limit to the inefficiency of the sorting algorithm, and therefore it is more pessimal than the other algorithms described herein.

BONUS TOD: Detecting Meter Maids with a RPi

- Guy parks on SF 2 hour limited street
- Camera & RPi to photo cars
- Motion detect for photo of cars & stuff
- Uses Tensor flow lib to image recog
- If 75%+ likely it's a meter maid:
  - Send SMS to phone via Twilio
  - Start clock on getting a ticket
- Bonus long term parking via tech
  - http://peoplesparking.space/