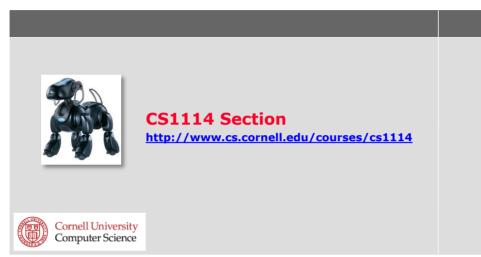
# **Sparse matrices, hash tables, cell arrays**



## **Useful new data types**

- Matlab has many useful data structures for handling different scenarios
- We'll cover a few that will be useful for A6:
  - Sparse matrices
  - Hash tables
  - Cell arrays

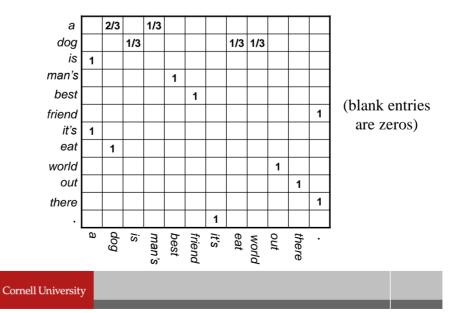


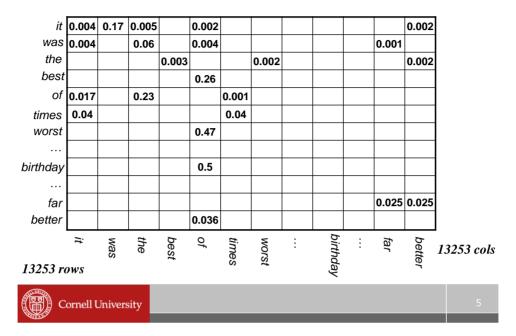
## **Transition matrices**

- For A6, you'll be creating very large matrices
- Storing these in memory will be an issue



## **Small transition matrix**





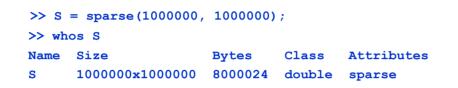
#### **Bigger example – "A Tail of Two Cities"**

# **Very large matrices**

- Jane Austen's Pride and Prejudice:
  - 8,828 unique words →
     8,828 x 8,828 transition matrix (77,933,584 entries)
- What about 1,000,000 words?
  - Matlab runs out of memory  $(1M \times 1M = 1T \text{ entries})$
  - Try this: >> zeros(1000000, 1000000);
- But the matrix is mostly empty
  - Most pairs (e.g. "and and") have zero probability

#### **Solution: sparse matrices**

- Matlab has a special type of sparse matrix
- Only stores the non-zero elements, and the position in the matrix of those elements
   A bit like a linked list





#### **Sparse matrices**

- Most operations on dense matrices work on sparse matrices
  - sometimes produce a sparse matrix, sometimes a dense matrix

```
S = sparse(1000000,1000000);
S(100,100) = 3; % S is still sparse
S = S + 1; % S is now dense
Error using +
Out of memory. Type HELP MEMORY for
your options.
```

```
Cornell University
```

## **Hash tables**

- Suppose we want to create a mapping from strings to numbers
  - E.g., from animals to number of legs

```
'human' → 2
'horse' → 4
'octopus' → 8
'centipede' → 100 (?)
```



## **Hash tables**

- We can use a *hash table* for this

   (Also called dictionary or associative array)
- Maps keys (e.g. `horse') to values (e.g. 4)
- Hash tables are interesting to implement, but we'll just use them as a tool

In Matlab:

```
>> hash = java.util.Hashtable;
```

- % For some reason in Matlab,
- % you can create Java objects

# **Hash tables**

 We can add key, value pairs to a hash table using *put* and retrieve values using *get* with the key

```
>> hash.put('horse', 4);
>> hash.push('octopus', 8);
% We just added two entries
% to the hash table
>> hash.get('horse')
ans = 10
```



## **Call arrays**

- Arrays can hold numbers or strings
- Q: What is the result of the following?
   [`abc', `def' ]
- Matlab has another kind of array: a cell array
  - A cell array can hold different types of objects
  - $-A = \{ 'abc', 'def', 103, [ 10 40 ; 40 10 ] \}$
  - We'll use these for A6 as well...