**CMSC 132:**
Object-Oriented Programming II

**Bags, Markov Chains, and Random Text Generation**
Department of Computer Science
University of Maryland, College Park

---

**Random Text Generation Project**

**Goal**
- Read in text
- Generate similar semi-random text

**Approach**
1. Build DenseBag to store word frequencies
2. Use DenseBag to build Markov chain
3. Use Markov chain to generate semi-random text

---

**DenseBag**

**Properties**
- Like a Set
- But can contain duplicates

**Examples**
- \{ 1, 3, 1, 1, 3, 5 \}
- \{ 1, 1, 1, 3, 3, 5 \}
- \{ three 1's, two 3's, one 5 \}
- All represent same DenseBag

**DenseBag<E> Operations**

**Efficiency**
- Most operations should take \(O(1)\)
- If using hashing
- \(\text{choose(Random r)}\) may take \(O(|\text{unique items}|)\)

**Iterator**
- Iterates over all elements
- Order is undefined

**Examples**
- Given DenseBag<Integer> \(x = \{ 1, 1, 1, 3, 3, 5 \}\);
- \(x. \text{getUniqueElements()}\) \(\rightarrow \{ 1, 3, 5 \}\)
- \(x. \text{getCount}(1)\) \(\rightarrow 3\)
- \(x. \text{choose}(r)\) \(\rightarrow 1\) (50%), 3 (33%) or 5 (17%)
Markov Chain For Text

- Application of Markov chain
  - Represent probability of word following each word
  - Based on actual frequencies found in text

- Example
  - In the text “a b a c a b a b”
    - Word a is followed by b (75%) or c (25%)
    - Markov chain for words following a

```
    a
    75% 25%
  ---->  b  c
```

Higher-Order Markov Chain

- Application
  - Can represent probability of word following each group of words (order-k for k consecutive words)

- Example
  - In the text “a b a c a b a b”
    - Words ba are followed by b (50%) or c (50%)
    - Represent with following Markov chain

```
    a
    50% 50%
  ---->  b  c
```

Markov Text Generation

- Approach (for order-n Markov text)
  1. Generate higher-order Markov chains
     - Analyze “training” text(s)
  2. Represent Markov chains as DenseBags
  3. Connect DenseBags
     - To build probabilistic transition table
  4. Use transition table to generate text

Handling Start & End of Text

1. Use empty string(s)
   - Start text generation with “”
   - End text if “” generated
   - “” → “a”
   - “a”, “” → “a”, “a”
   - “a” → “”
2. Augment input with <Start> & <End> markers
   - “a b a c” → “<Start> a b a c <End>”
   - Start text generation with <Start>
   - End text if <End> generated