Lab 8. Story generator

- You’re about to develop a program that can read a story and then write a new story in the same style as the original story.
- A new story will be based on a random selection of symbols from the original story.
Generating random stories

• Input story – old German saying:

What I spent, I had; what I saved, I lost; what I gave, I have.

• Consider each symbol to be the entire word

<table>
<thead>
<tr>
<th>Current symbol</th>
<th>Possible next symbols (list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>what</td>
<td>I, spent, had, saved, lost, gave, have</td>
</tr>
<tr>
<td>I</td>
<td>spent, had, saved, lost, gave, have</td>
</tr>
<tr>
<td>spent</td>
<td>I</td>
</tr>
<tr>
<td>saved</td>
<td>I</td>
</tr>
<tr>
<td>lost</td>
<td>what</td>
</tr>
<tr>
<td>gave</td>
<td>I</td>
</tr>
<tr>
<td>have</td>
<td>-</td>
</tr>
</tbody>
</table>

We can generate a new story:

• Pick symbol at random: I

• Pick at random any symbol that can follow I: lost

• After lost can be only: what

• Then: I

• Finally: have

I lost what I have
What is the best data structure to store symbol frequencies?

What I spent, I had; what I saved, I lost; what I gave, I have.

<table>
<thead>
<tr>
<th>Current symbol</th>
<th>Possible next symbols (list)</th>
</tr>
</thead>
<tbody>
<tr>
<td>what</td>
<td>I</td>
</tr>
<tr>
<td>I</td>
<td>spent, had, saved, lost, gave, have</td>
</tr>
<tr>
<td>spent</td>
<td>I</td>
</tr>
<tr>
<td>saved</td>
<td>I</td>
</tr>
<tr>
<td>lost</td>
<td>what</td>
</tr>
<tr>
<td>gave</td>
<td>I</td>
</tr>
<tr>
<td>have</td>
<td>-</td>
</tr>
</tbody>
</table>

We can generate a new story:

• Pick symbol at random: I

• Pick at random any symbol that can follow I: lost

• After lost can be only: what

• Then: I

• Finally: have

*I lost what I have*
What is the best data structure to store these data?

<table>
<thead>
<tr>
<th>Current symbol</th>
<th>Possible next symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b:3, c:4, n:12</td>
</tr>
<tr>
<td>b</td>
<td>a:33, o:21</td>
</tr>
<tr>
<td>c</td>
<td>a:44, r: 12</td>
</tr>
<tr>
<td>d</td>
<td>r: 12, o:23</td>
</tr>
</tbody>
</table>
Lab 8. Implement Hash Map – using Hash Table with separate chaining

- Stores (key, value) pairs
- Collisions are resolved by storing a list

\[(11, 5), (21, 3), (33, 1), \ldots\]

\[
\begin{array}{|c|c|}
\hline
\text{array index} & \text{List of (key, val) pairs} \\
\hline
0 & \text{} \\
1 & (11,5) \rightarrow (21,3) \\
2 & \text{} \\
3 & (33,1) \\
4 & \text{} \\
5 & \text{} \\
6 & \text{} \\
7 & \text{} \\
8 & \text{} \\
9 & \text{} \\
\hline
\end{array}
\]

\( H(key) = key \% 10 \)
Lab 8. hash table as a value

• Stores (key, value) pairs, where key is a symbol, and value is another hash table! (list of hash tables for separate chaining)

<table>
<thead>
<tr>
<th>Current symbol</th>
<th>Possible next symbols</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b:3, c:4, n:12</td>
</tr>
<tr>
<td>b</td>
<td>a:33, o:21</td>
</tr>
<tr>
<td>c</td>
<td>a:44, r: 12</td>
</tr>
<tr>
<td>d</td>
<td>r: 12, o:23</td>
</tr>
</tbody>
</table>

Key: current symbol

Value: another hash map (next symbol, count)
Graph ADT
Modeling with graphs

Lecture 24
By Marina Barsky
What is a graph?

A graph $G = (V, E)$ is an Abstract Data Type that consists of 2 collections:

• **Set** of objects (*vertices*, *nodes*)
  $$V = \{A, B, C, D, E\}$$

• **Relation** on set of objects (*edges*)
  $$E = \{(A, B), (A, C), (A, E), (B, D), (C, D), (C, E)\}$$

Running time of Graph algorithms uses **two** numbers:

• $n = |V|$
• $m = |E|$


Edge $e$ connects vertices $u$ and $v$

Vertices $u$ and $v$ are end points of edge $e$

Vertex $u$ and edge $e$ are incident

Two edges are also called incident, if they are incident to the same vertex

Vertices $u$ and $v$ are adjacent

Vertices $u$ and $v$ are neighbors

This is a vocabulary for undirected graph
The degree of a vertex

- The **degree** of a vertex is the number of its incident edges. I.e., the degree of a vertex is the number of its neighbors

- Let’s denote the degree of a vertex $v$ by $\deg(v)$

- The **degree of a graph** is sum of degree of its vertices. The degree of undirected graph with $m$ edges is $2m$
Example

The degree of $v$ is 6: $\text{deg}(v) = 6$

The degree of $v_6$ is 1: $\text{deg}(v_6) = 1$

The degree of this graph: $\text{deg}(G) = 2m = 12$
Directed graphs

Nodes: \{A,B,C,D\}

Edges (ordered pairs):
\{(A,C),(D,A),(B,D),(C,B)\}

These two graphs are different!
Graphs can model many things

Trivial:
• Mobile networks
• Computer networks
• Social networks

Non-trivial:
• Web pages
• States of the game
• ...
Graph: airlines

[Map of airline routes with cities labeled, showing connections between various cities such as Dakar, New York, Addis Ababa, and others. The map includes lines for ASKY Airlines and Ethiopian Airlines.]
Is there a direct flight from A to D?
- With one stop?
- With exactly two stops?

Graph: airlines

Graph of flights between 5 cities
Facebook graph
Facebook graph

Chandler

friends

Joey
Directed graph: one-way streets
Directed graph: followers
Directed graph: citations
Linked Open Data Diagram

DBpedia: structured cross-domain knowledge
Linked Open Data Diagram

- Media
- NY Times
- RDF link
Schizophrenia Protein–Protein Interaction (PPI)
Schizophrenia Protein-Protein Interaction (PPI)
A graph is **explicit** if all its vertices and edges are stored.

Often we work with an **implicit graph** which is conceptual or unexplored.

There are only $3^9 = 19,683$ different states in Tic-Tac-Toe. We can store the entire graph and compute the optimal strategy as a path through this graph.

The Rubik's Cube has 43 quintillion states. It can be solved without explicitly listing all vertices (states).
Modeling with graphs

Solving puzzles
A chess knight can move in an L shape in any direction
A chess knight can move in an L shape in any direction.
There are four knights on the 3×3 chessboard: the two white knights are at the two upper corners, and the two black knights are at the two bottom corners of the board.

The goal is to switch the knights in the minimum number of moves so that the white knights are at the bottom corners and the black knights are at the upper corners.

Try it out http://barsky.ca/knights/
Graph: nodes

Each position is a node in a graph
There is an edge between the nodes if you can go from one node to another by 1 knight move.
Does it help to solve the puzzle?
Unfold the graph!

All the nodes are on a circle
Solution

Do you see it now?
Solution

Move around the circle following legal edges
Solution

Until knights are in desired positions
Group activity
Graphs and puzzles

Consider the following puzzle:

• You can move each of the green or red pieces along the lines.
• The goal is to interchange the positions of the colored pieces in the minimum number of moves.

Draw a graph model which would help you to solve this puzzle.

What is the minimum number of moves? ______________