Vocabulary:
- bit, byte;
- binary codes.

There are $2^n$ strings of $n$ bits:

The **Huffman code** is often used for data compression. The main idea is that symbols that occur with higher probabilities (or frequencies) are encoded by shorter strings of bits. Given a collection of symbols with corresponding probabilities, one first constructs a **Huffman tree** as follows:

**Step 1.** Create a vertex for each symbol and add those vertices to the queue. They will form the lowest level of the tree.

**Step 2.** Select the two vertices with the lowest probabilities from the queue, say $u$ and $v$, and then:
- Create a new vertex, say $w$. (It will be above $u$ and $v$ in the tree.)
- Add edges from $w$ to $u$ and to $v$.
- Make the probability of $w$ be the sum of the probabilities of $u$ and $v$.
- Remove $u$ and $v$ from the queue.
- Add $w$ to the queue.

**Step 3.** Repeat Step 2 until there is only one vertex left in the queue.

The code for each symbol can be simply read off the Huffman tree (stepping to the left means a 0, stepping to the right means a 1).

**Example.** We will find a Huffman code for the following symbols and probabilities:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>0.40</td>
</tr>
<tr>
<td>B</td>
<td>0.20</td>
</tr>
<tr>
<td>C</td>
<td>0.35</td>
</tr>
<tr>
<td>D</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Create a vertex for each symbol (for convenience, we draw them in order of decreasing probability). The circle around a vertex indicates that it belongs to the queue.
The two vertices with the lowest probabilities are the ones representing the symbols B and D, so we “join” them to a new vertex (which we denoted by u):

\[
\begin{array}{cccc}
A & C & B & D \\
0.40 & 0.35 & 0.20 & 0.05 \\
\end{array}
\]

Notice that we removed B and D from the queue (i.e., they are not circled anymore). Among the circled vertices, the ones with the lowest probabilities are C and u, so we “join” them:

\[
\begin{array}{cccc}
A & C & B & D \\
0.40 & 0.35 & 0.20 & 0.05 \\
\end{array}
\]

Finally, we “join” the remaining two vertices in the queue:

\[
\begin{array}{cccc}
A & C & B & D \\
0.40 & 0.35 & 0.20 & 0.05 \\
\end{array}
\]

Here’s what the final tree looks like:

Now we can read off the codes for the symbols:

\[
\begin{array}{cccc}
A & C & B & D \\
0 \rightarrow 0 & 1 \rightarrow 10 & 0 \rightarrow 1 & 1 \rightarrow 111 \\
\end{array}
\]