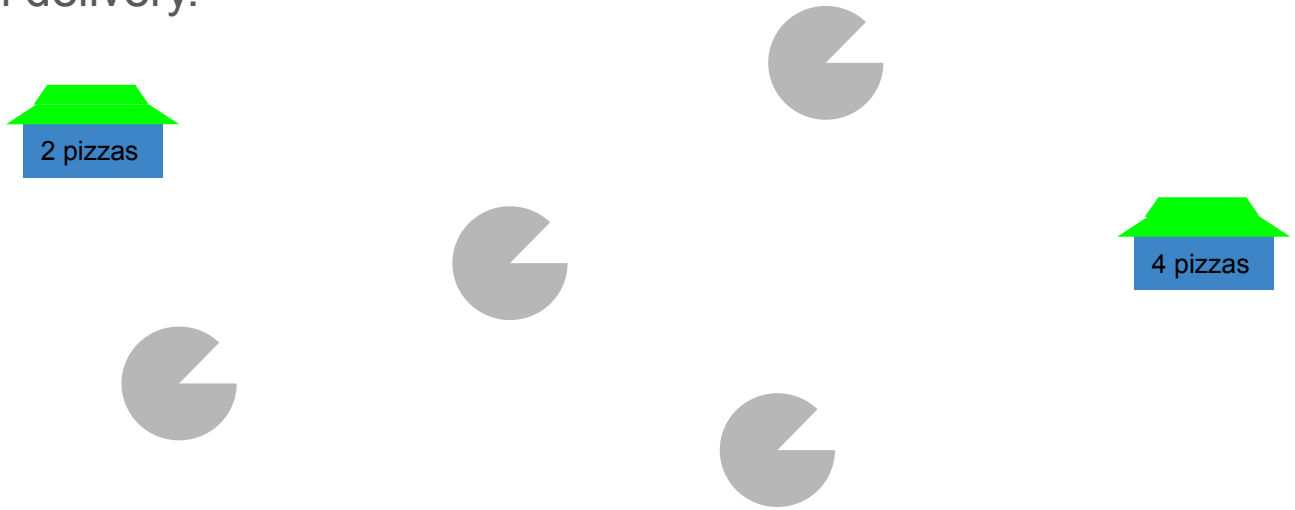
Other Greedy Problems

Pizza Delivery

You own two pizza places. Each place has some number of pizzas. There are people that have ordered pizzas. Find the least amount of gas needed to deliver all the pizzas assume you drive from the pizza joint to the person and back with each delivery.



Pizza Delivery: Strategy

How can we convert the problem to something easier?

Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used

Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used (assuming gas is a linear function of distance).

Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used (assuming gas is a linear function of distance).

We could choose for some place to go to the closest person.

Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used (assuming gas is a linear function of distance).

We could choose for some place to go to the closest person. That might not be ideal.

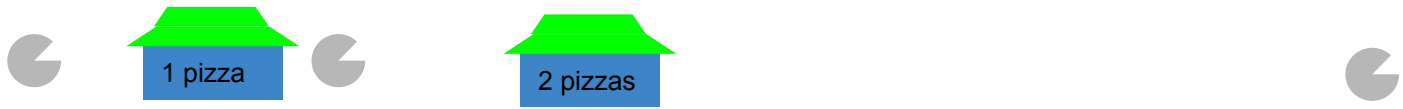
Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used (assuming gas is a linear function of distance).

We could choose for some place to go to the closest person. That might not be ideal.



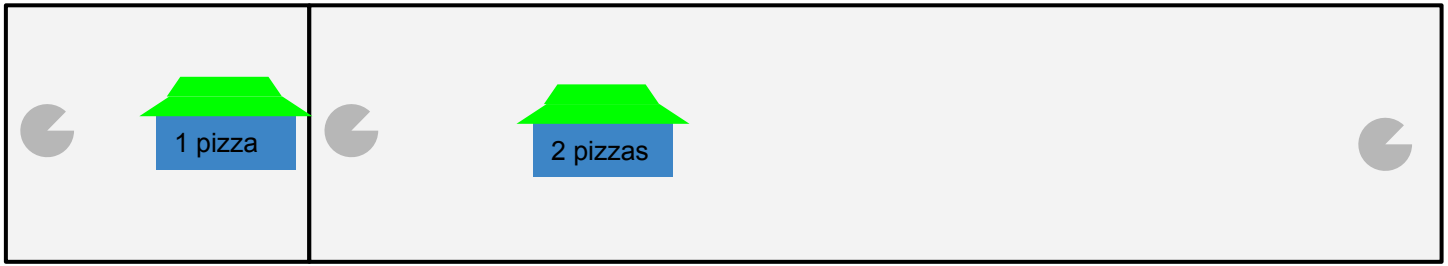
Pizza Delivery: Strategy

How can we convert the problem to something easier?

Turn each person into a pair of values (distance A, distance B).

How do we minimize the total gas used (assuming gas is a linear function of distance).

We could choose for some place to go to the closest person. That might not be ideal.



Pizza Delivery: Strategy 2

Sort by the difference in the distance.

Pizza Delivery: Strategy 2

Sort by the difference in the distance.

This strategy works!

Pizza Delivery: Strategy 2

Sort by the difference in the distance.

This strategy works!

People that have a significant difference of distances will try to receive food from the closer location.

Pizza Delivery: Strategy 2

Sort by the difference in the distance.

This strategy works!

People that have a significant difference of distances will try to receive food from the closer location.

If a location runs out of pizza, everyone else goes to the other location.

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Most characters are not used in a text file.

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Most characters are not used in a text file.

We are going to reduce the encoding for some text file.

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Most characters are not used in a text file.

We are going to reduce the encoding for some text file.

Suppose only 4 characters are used (like in DNA).

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Most characters are not used in a text file.

We are going to reduce the encoding for some text file.

Suppose only 4 characters are used (like in DNA).

How many bits would be needed for each character?

Simple File Encoding

Simple file encoding strategy uses 8 bits (2 bytes) per character.

256 possible characters.

Most characters are not used in a text file.

We are going to reduce the encoding for some text file.

Suppose only 4 characters are used (like in DNA).

How many bits would be needed for each character?

What if some characters occurred ***a lot*** more often than other characters?

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes.

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes.

If “E” occurs 100,000 times, but Z occurs 10 times,

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes

If “E” occurs 100,000 times, but Z occurs 10 times, give a code for “E” that is smaller than “Z”.

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes

If “E” occurs 100,000 times, but Z occurs 10 times, give a code for “E” that is smaller than “Z”.

Can we let “E” be just the bit-string “1”?

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes

If “E” occurs 100,000 times, but Z occurs 10 times, give a code for “E” that is smaller than “Z”.

Can we let “E” be just the bit-string “1”?
Yes, but **all** bit-strings for the other symbols must start with “0”.

Prefix-Free Encoding

A cool technique for encoding allows for variable length encoding strings.

- More common symbols are given shorter codes.
- Less common symbols are given longer codes

If “E” occurs 100,000 times, but Z occurs 10 times, give a code for “E” that is smaller than “Z”.

Can we let “E” be just the bit-string “1”?
Yes, but **all** bit-strings for the other symbols must start with “0”.

In general no code can be the prefix of another code.

Prefix-Free Encoding Tries

Tries are a good representation for a prefix-free encoding method.

Prefix-Free Encoding Tries

Tries are a good representation for a prefix-free encoding method.

- Every nodes should have two children.

Prefix-Free Encoding Tries

Tries are a good representation for a prefix-free encoding method.

- Every nodes should have two children.
- Every leaf should be a symbol.

Prefix-Free Encoding Tries

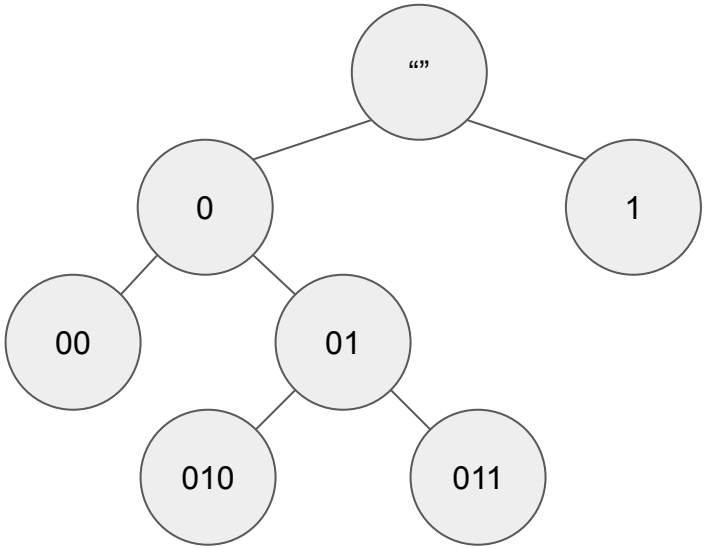
Tries are a good representation for a prefix-free encoding method.

- Every nodes should have two children.
- Every leaf should be a symbol.
- No symbol should occur in an internal node.

Prefix-Free Encoding Tries

Tries are a good representation for a prefix-free encoding method.

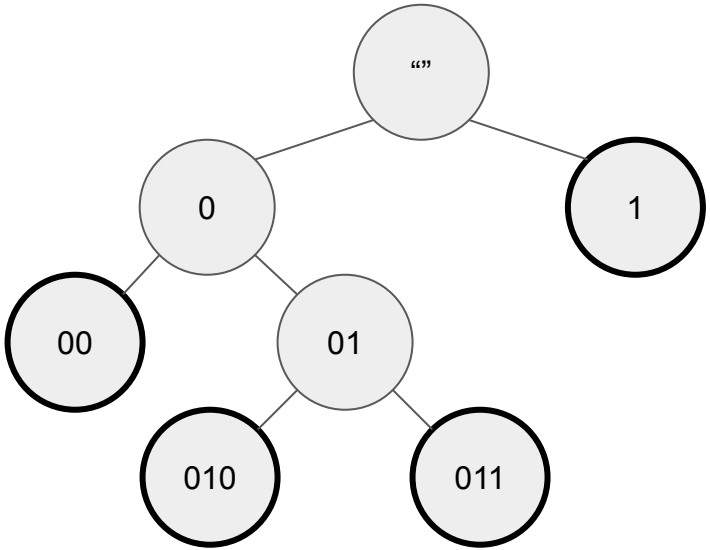
- Every nodes should have two children.
- Every leaf should be a symbol.
- No symbol should occur in an internal node.



Prefix-Free Encoding Tries

Tries are a good representation for a prefix-free encoding method.

- Every nodes should have two children.
- Every leaf should be a symbol.
- No symbol should occur in an internal node.



Example

Letter	A	C	T	G
Occurrences	30	100	10	5

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”
T would be “001”

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”
T would be “001”
G would be “000”

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”
T would be “001”
G would be “000”

Total would be 100 * 1 + 30 * 2 + 10 * 3 + 5 * 3

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”
T would be “001”
G would be “000”

Total would be 100 * 1 + 30 * 2 + 10 * 3 + 5 * 3 = 205

Example

Letter	A	C	T	G
Occurrences	30	100	10	5

We could let C be encoded using just a “1”
We could let A be encoded using a “01”
T would be “001”
G would be “000”

Total would be 100 * 1 + 30 * 2 + 10 * 3 + 5 * 3 = 205
(Originally 2 * (30 + 100 + 10 + 5) = 290)

Greedily Assigning Bit Codes

How could we assign the bit codes to an arbitrary alphabet?

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Largest Frequency = Smallest Code

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Largest Frequency = Smallest Code
1,000,000=1; 100,000=2; 10,000=3; 1,000=4; 100=5; 10=6; 1=6

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Largest Frequency = Smallest Code
1,000,000=1; 100,000=2; 10,000=3; 1,000=4; 100=5; 10=6; 1=6

Can this be broken?

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Largest Frequency = Smallest Code
1,000,000=1; 100,000=2; 10,000=3; 1,000=4; 100=5; 10=6; 1=6

Can this be broken? Yes, all equal.

Greedily Assigning Bit Codes

^^^

How could we assign the bit codes to an arbitrary alphabet?

What would be a good greedy strategy?

Largest Frequency = Smallest Code
1,000,000=1; 100,000=2; 10,000=3; 1,000=4; 100=5; 10=6; 1=6

Can this be broken? Yes, all equal.

Biggest first is not a good idea...

Smallest First “Assignment”

Look at the smallest frequency symbols.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.
- Merge the ***a*** and ***b*** together into a trie.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.
- Merge the ***a*** and ***b*** together into a trie.
- Set the frequency of this trie equal to the sum of ***a*** and ***b***.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.
- Merge the ***a*** and ***b*** together into a trie.
- Set the frequency of this trie equal to the sum of ***a*** and ***b***.
- Add the trie of ***a*** and ***b*** back into our list of symbols.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.
- Merge the ***a*** and ***b*** together into a trie.
- Set the frequency of this trie equal to the sum of ***a*** and ***b***.
- Add the trie of ***a*** and ***b*** back into our list of symbols.
- Repeat until one “symbol” is left.

Smallest First “Assignment”

Look at the smallest frequency symbols.

They need to have longer codes to ensure more frequent symbols get shorter codes.

- Remove the two least frequent symbols (***a***, ***b***) from the trie.
- Merge the ***a*** and ***b*** together into a trie.
- Set the frequency of this trie equal to the sum of ***a*** and ***b***.
- Add the trie of ***a*** and ***b*** back into our list of symbols.
- Repeat until one “symbol” is left.

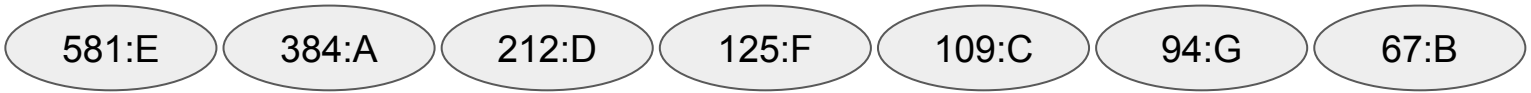
^^^ AND this works ^^^

“Huffman” Encoding Example

Letter	A	B	C	D	E	F	G
Occs.	384	67	109	212	581	125	94

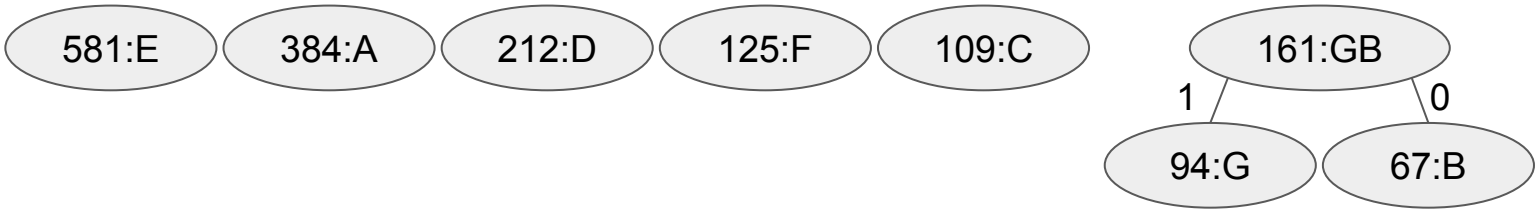
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



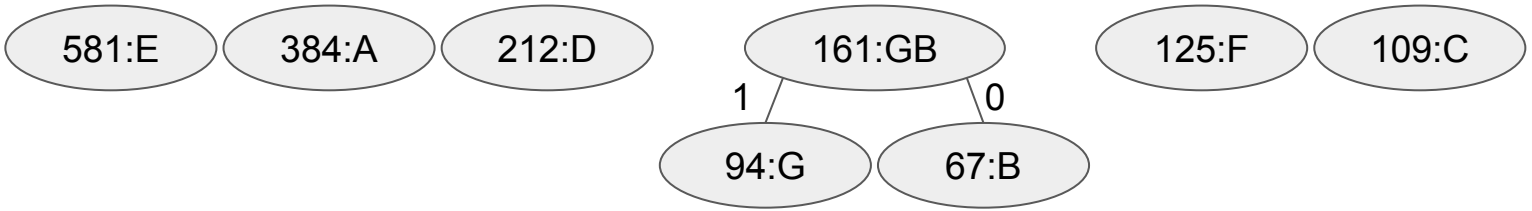
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



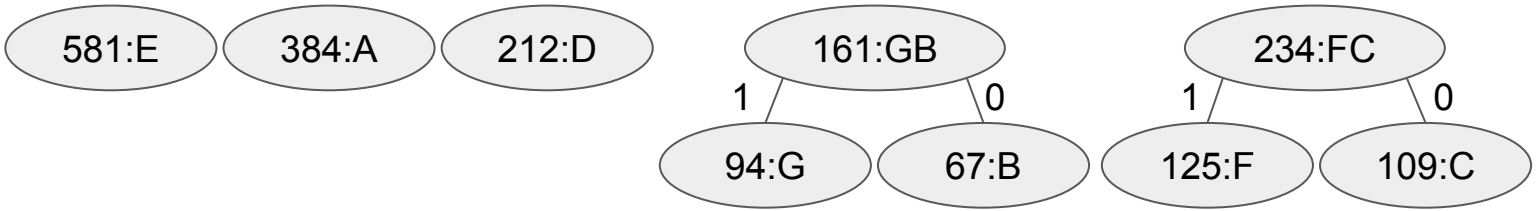
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



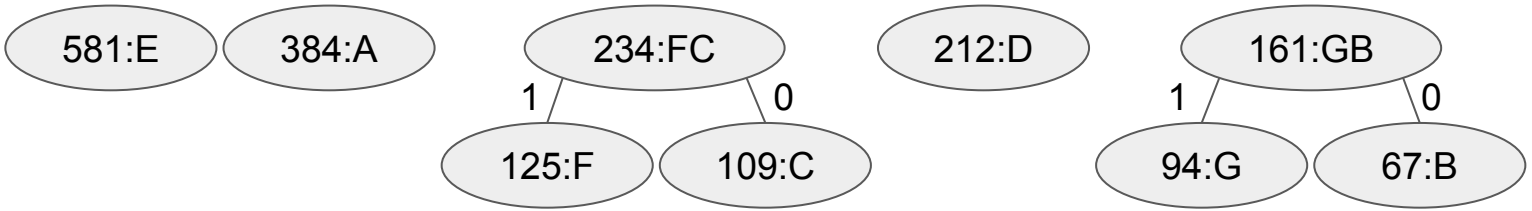
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



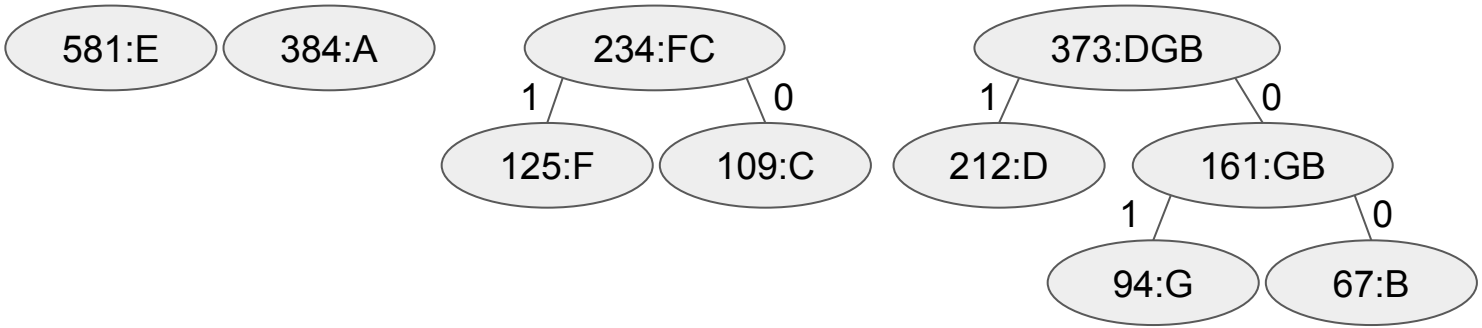
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



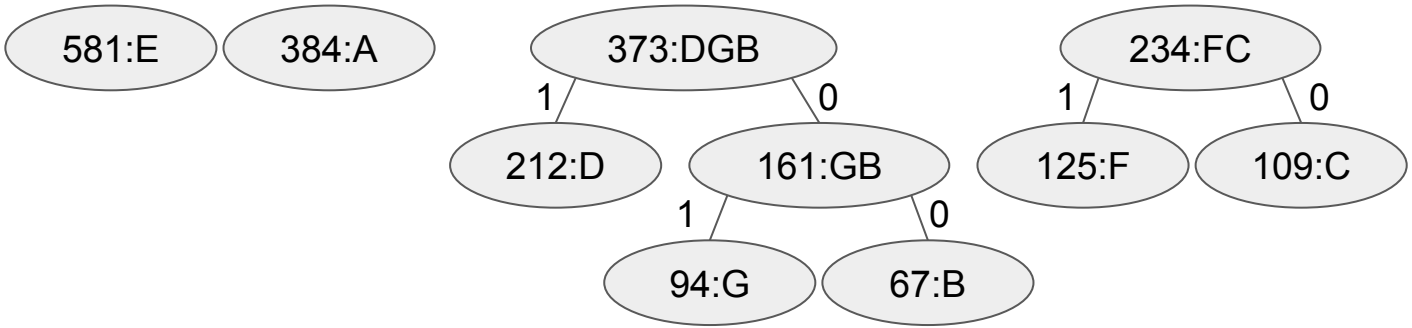
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



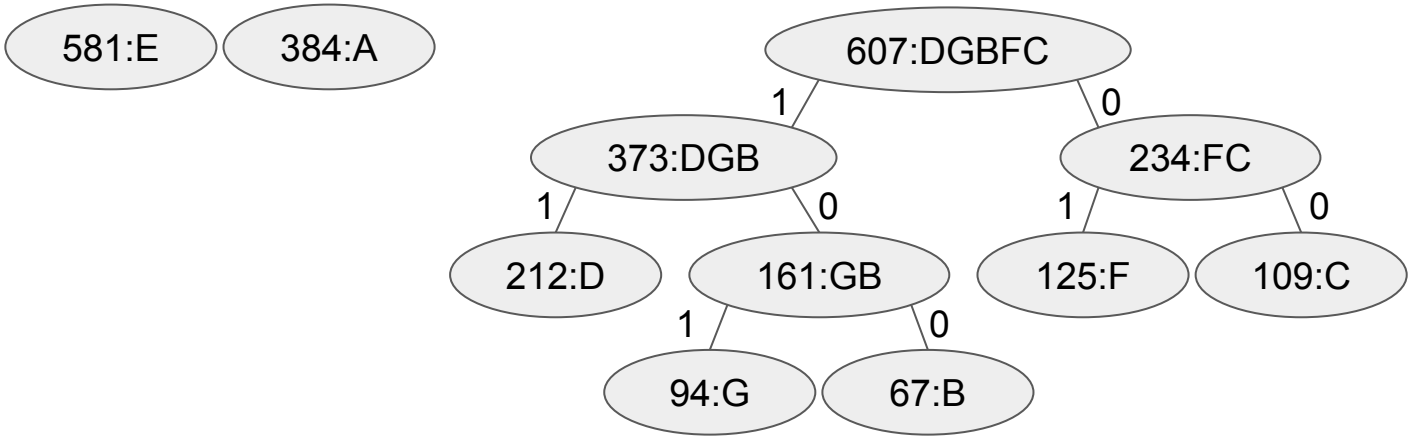
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



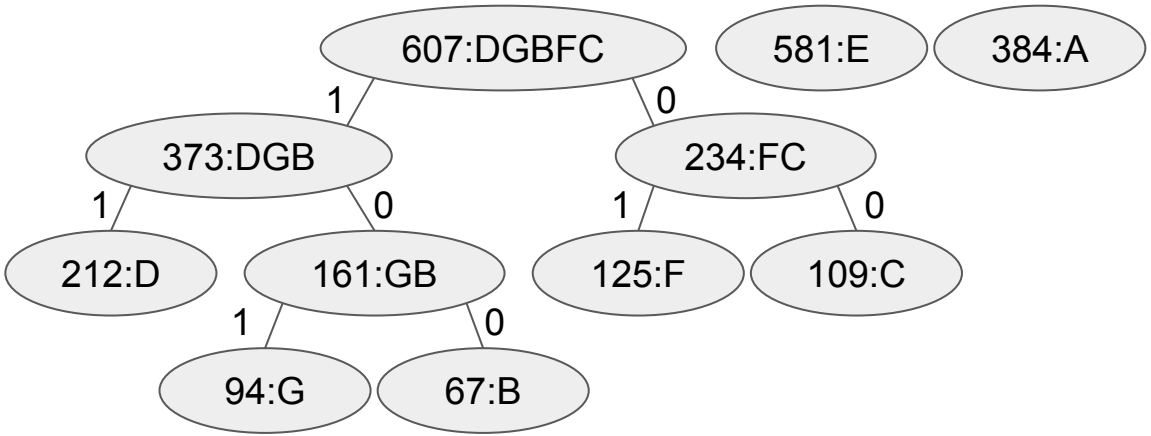
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



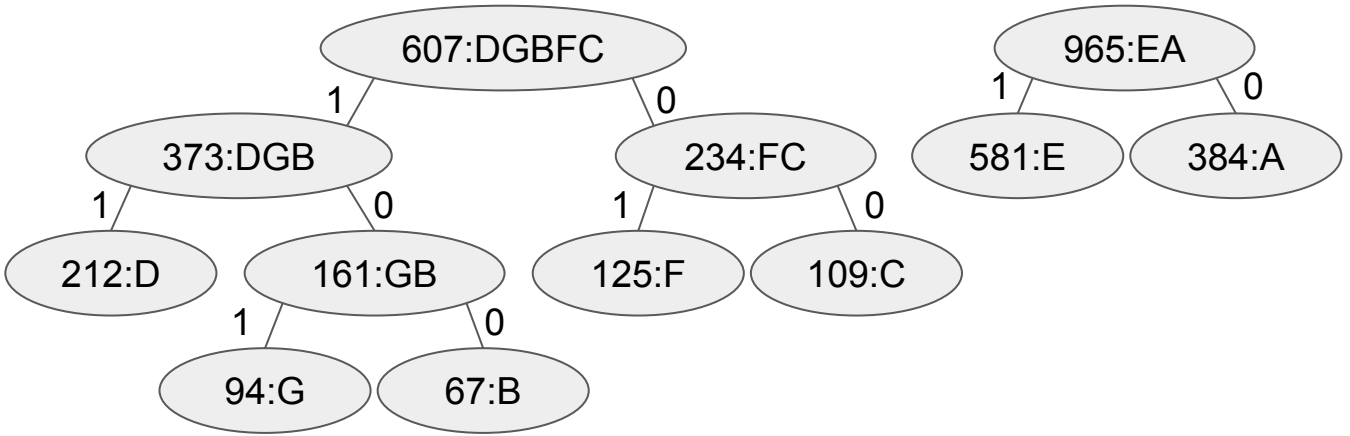
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



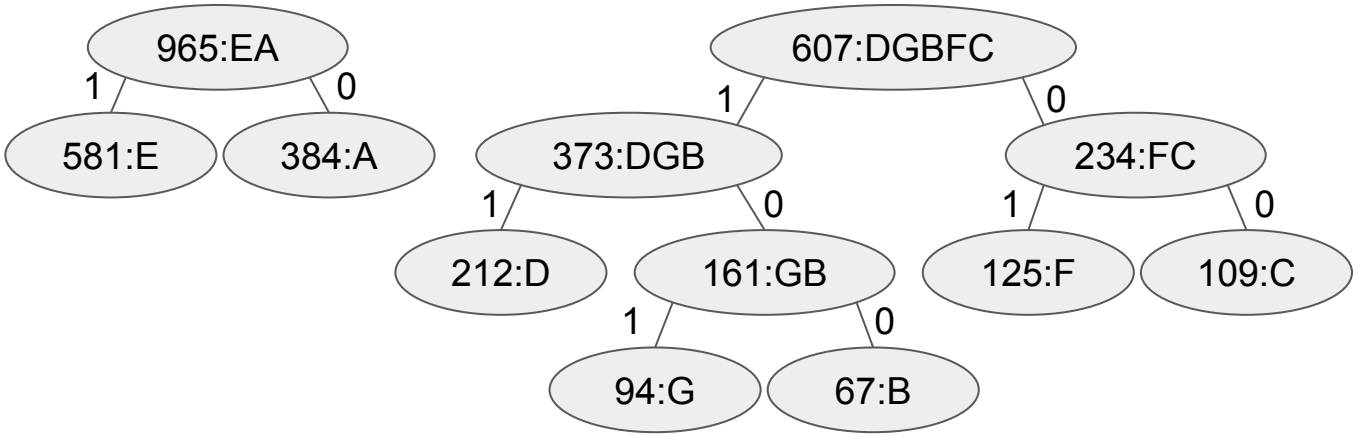
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



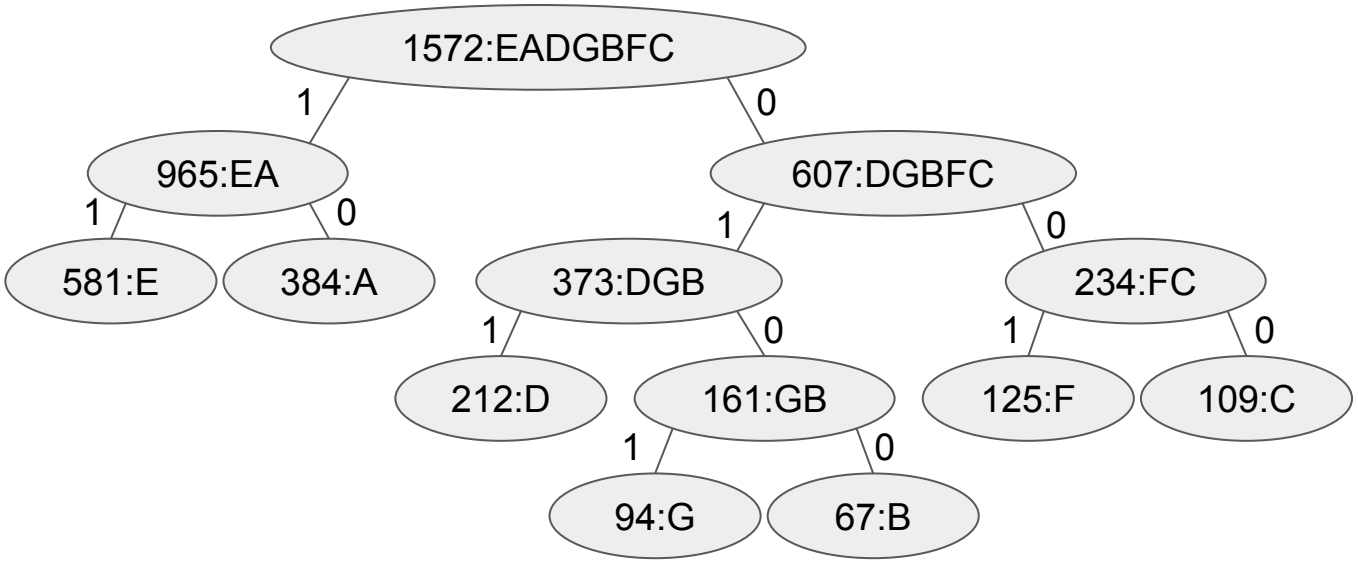
“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67



“Huffman” Encoding Example

Letter	E	A	D	F	C	G	B
Occs.	581	384	212	125	109	94	67
Bit	11	10	011	001	000	0101	0100