```
(* Adam Beck *)
(* Problem 1 *)

(* Create 3 different sized random arrays *)
small = RandomInteger[{0, 1}, 50];
medium = RandomInteger[{0, 1}, 500];
large = RandomInteger[{0, 1}, 5000];

(* Equipartition of small *)
(* 0 and 1 *)
(* Get the sum of the digits divided by length, that is the frequency of 1's *)
smallfrq1 = N[Sum[small[[i]], {i, 1, Length[small]}]] / Length[small]
```

0.42

```
smallfrq0 = 1 - smallfrq1
```

0.58

```
(* 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)
```

smallfrq00 = N[$\sum_{k=1}^{\text{Length[small]}-1}$ ((1 - small[[k]]) * (1 - small[[k + 1]])) / (Length[small] - 1)]

0.306122

```
NumberForm[smallfrq01, 16]
(* Useful for testing a printout of a certain number of digits *)
```

smallfrq01 = N[$\sum_{k=1}^{\text{Length[small]}-1}$ ((1 - small[[k]]) * (small[[k + 1]])) / (Length[small] - 1)]

0.265306

smallfrq10 = N[$\sum_{k=1}^{\text{Length[small]}-1}$ ((small[[k]]) * (1 - small[[k + 1]])) / (Length[small] - 1)]

0.285714

```
smallfrq11 = 1 - smallfrq00 - smallfrq01 - smallfrq10
```

0.142857

```
(* 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Function created in class. Gather the
 frequencies of all the 3 pairs of binary digits *)
```

smallfrq000 = N[$\sum_{k=1}^{\text{Length[small]}-2}$ ((1 - small[[k]]) * (1 - small[[k + 1]]) * (1 - small[[k + 2]])) / (Length[small] - 2)]

0.0833333

$$smallfrq001 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(1 - small[[k]]\right) * \left(1 - small[[k+1]]\right) * \left(small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.145833

$$smallfrq010 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(1 - small[[k]]\right) * \left(small[[k+1]]\right) * \left(1 - small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.145833

$$smallfrq011 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(1 - small[[k]]\right) * \left(small[[k+1]]\right) * \left(small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.125

$$smallfrq100 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(small[[k]]\right) * \left(1 - small[[k+1]]\right) * \left(1 - small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.166667

$$smallfrq101 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(small[[k]]\right) * \left(1 - small[[k+1]]\right) * \left(small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.125

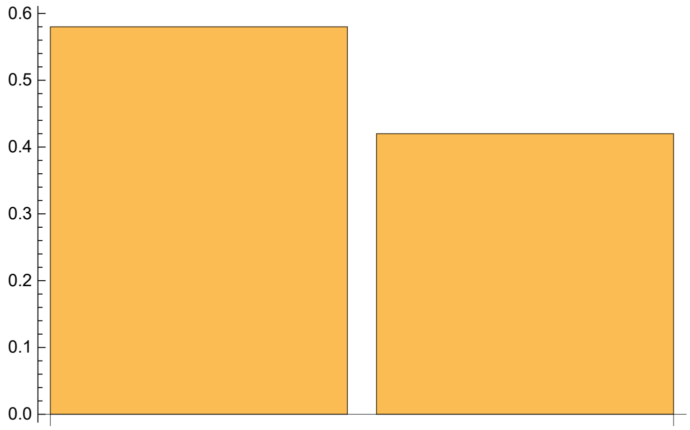
$$smallfrq110 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(small[[k]]\right) * \left(small[[k+1]]\right) * \left(1 - small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.145833

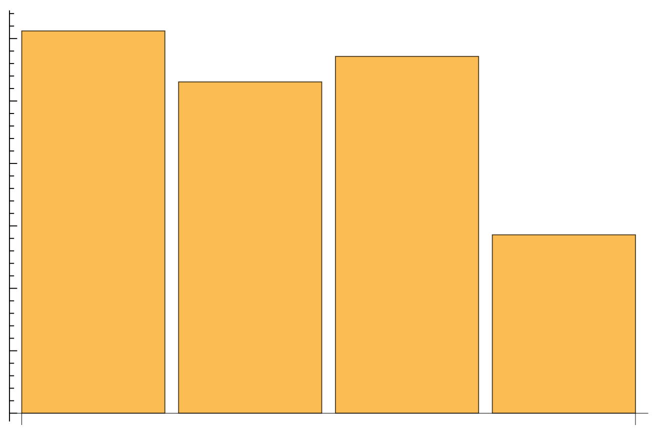
$$smallfrq111 = N\left[\sum_{k=1}^{Length[small]-2} \left(\left(small[[k]]\right) * \left(small[[k+1]]\right) * \left(small[[k+2]]\right)\right) / \left(Length[small] - 2\right)\right]$$

0.0625

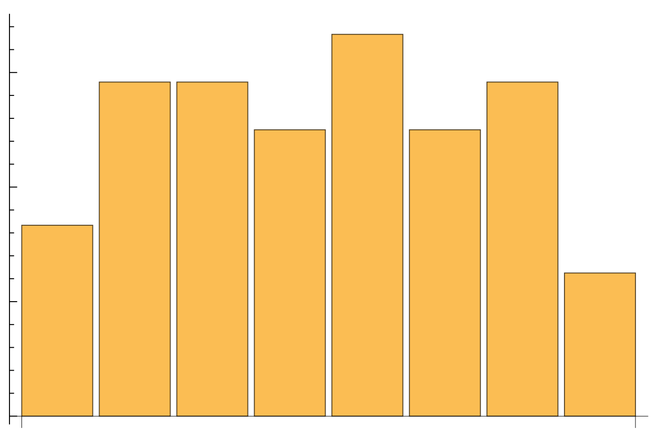
```
(* Graphing small *)
BarChart[{smallfrq0, smallfrq1}]
```




```
(* This does not look random. It is not centered at .5,
but it was a small data set so that is expected *)

BarChart[{smallfrq00, smallfrq01, smallfrq10, smallfrq11}]
```



```
(* This does not look random. It is not centered at .25,
but it was a small data set so that is expected *)
BarChart[{smallfrq000, smallfrq001, smallfrq010,
  smallfrq011, smallfrq100, smallfrq101, smallfrq110, smallfrq111}]
```



```
(* This does not look random. It is not centered at 1/8,
but it was a small data set so that is expected *)

(* In conclusion so far: the law of large numbers should
   help us out and get a more centralized data set for equipartitions *)

(* Equipartition of medium *)
(* 0 and 1 *)

(* Sum up the digits divided by length, that is the frequency for the 1's *)
mediumfrq1 = N[Sum[medium[[i]], {i, 1, Length[medium]}]] / Length[medium]
```

0.476

```
mediumfrq0 = 1 - mediumfrq1
```

0.524

```
(* 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)
mediumfrq00 =
```

$$N\left[\sum_{k=1}^{\text{Length[medium]}-1} \left(\left(1 - \text{medium[[k]]}\right) * \left(1 - \text{medium[[k+1]]}\right)\right) / \left(\text{Length[medium]} - 1\right)\right]$$

0.268537

$$\text{mediumfrq01} = N\left[\sum_{k=1}^{\text{Length[medium]}-1} \left(\left(1 - \text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right)\right) / \left(\text{Length[medium]} - 1\right)\right]$$

0.256513

$$\text{mediumfrq11} = N\left[\sum_{k=1}^{\text{Length[medium]}-1} \left(\left(\text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right)\right) / \left(\text{Length[medium]} - 1\right)\right]$$

0.218437

```
mediumfrq10 = 1 - mediumfrq00 - mediumfrq01 - mediumfrq11
```

0.256513

```
(* 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Function created in class. Gather the
 frequencies of all the 3 pairs of binary digits *)
mediumfrq000 =
```

$$N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(1 - \text{medium[[k]]}\right) * \left(1 - \text{medium[[k+1]]}\right) * \left(1 - \text{medium[[k+2]]}\right)\right) / \left(\text{Length[medium]} - 2\right)\right]$$

0.138554

$$\text{mediumfrq001} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(1-\text{medium[[k]]}\right) * \left(1-\text{medium[[k+1]]}\right) * \left(\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.130522

$$\text{mediumfrq010} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(1-\text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right) * \left(1-\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.148594

$$\text{mediumfrq011} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(1-\text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right) * \left(\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.108434

$$\text{mediumfrq100} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(\text{medium[[k]]}\right) * \left(1-\text{medium[[k+1]]}\right) * \left(1-\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.130522

$$\text{mediumfrq101} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(\text{medium[[k]]}\right) * \left(1-\text{medium[[k+1]]}\right) * \left(\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.126506

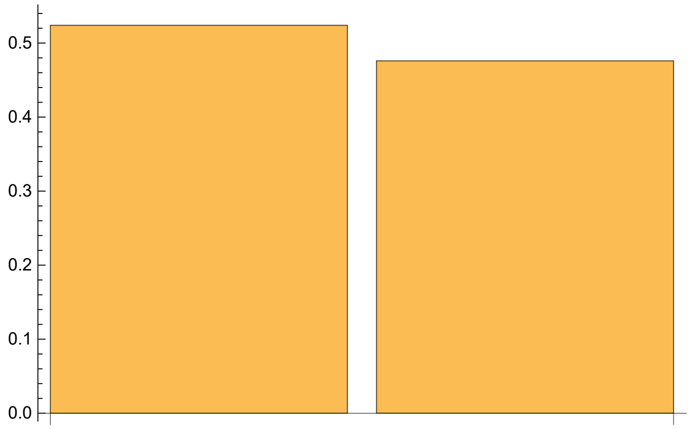
$$\text{mediumfrq110} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(\text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right) * \left(1-\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.108434

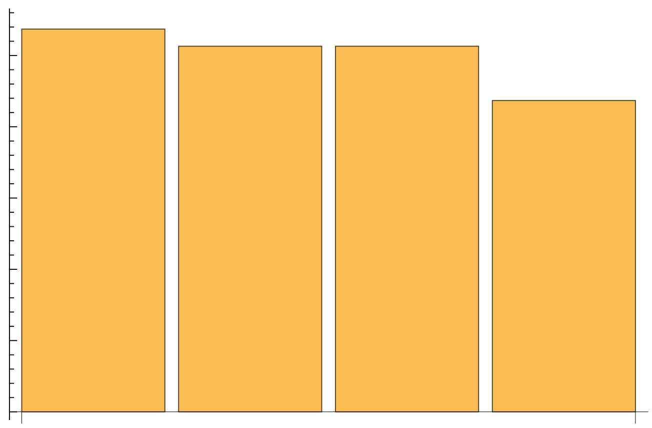
$$\text{mediumfrq111} = N\left[\sum_{k=1}^{\text{Length[medium]}-2} \left(\left(\text{medium[[k]]}\right) * \left(\text{medium[[k+1]]}\right) * \left(\text{medium[[k+2]]}\right)\right) \Big/ \left(\text{Length[medium]}-2\right)\right]$$

0.108434

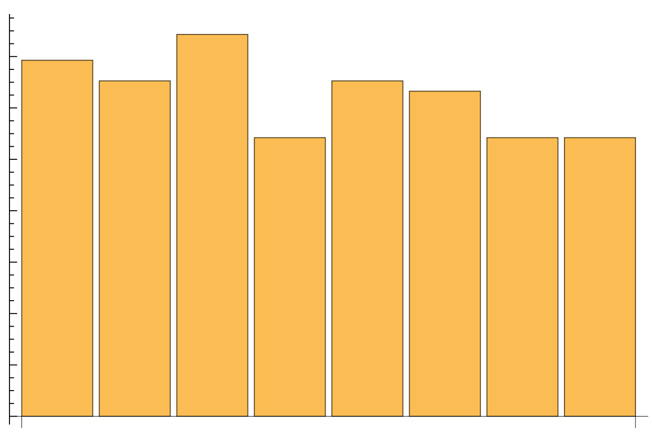
```
(* Graphing medium *)
BarChart[{mediumfrq0, mediumfrq1}]
```




```
(* This is gettting closer to .5 and this was
 the expected outcome due to the law of large numbers *)
BarChart[{mediumfrq00, mediumfrq01, mediumfrq10, mediumfrq11}]
```



```
(* This is gettting closer to .25 and this was
 the expected outcome due to the law of large numbers *)
BarChart[{mediumfrq000, mediumfrq001, mediumfrq010, mediumfrq011,
  mediumfrq100, mediumfrq101, mediumfrq110, mediumfrq111}]
```



```
(* This is gettting closer to 1/8 and this was
 the expected outcome due to the law of large numbers *)

(*In general, these graphs for the medium lengthed
 array were expected. They are more uniform around their
 expected value than the small array graphs. *)

(* Equipartition of large *)
(* 0 and 1 *)
(* Sum up the digits divided by length to find the frequency of the 1's *)
largefrq1 = N[Sum[large[[i]], {i, 1, Length[large]}]] / Length[large]
```

0.5022

```
largefrq0 = 1 - largefrq1
```

0.4978

```
(* 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)
largefrq01 =
 N[Sum[(1 - large[[i]]) * (large[[i + 1]]), {i, 1, Length[large] - 1}]] / Length[large - 1]
```

0.2552

```
largefrq00 =
 N[Sum[(1 - large[[i]]) * (1 - large[[i + 1]]), {i, 1, Length[large] - 1}]] / Length[large - 1]
```

0.2424

```
largefrq11 =
 N[Sum[(large[[i]]) * (large[[i + 1]]), {i, 1, Length[large] - 1}]] / Length[large - 1]
```

0.247

```
largefrq10 =
 N[Sum[(large[[i]]) * (1 - large[[i + 1]]), {i, 1, Length[large] - 1}]] / Length[large - 1]
```

0.2552

```
(* 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Function created in class. Gather the
 frequencies of all the 3 pairs of binary digits *)
```

$$\text{largefrq000} = \text{N}\left[\sum_{k=1}^{\text{Length[large]}-2} \left(\left(1-\text{large[[k]]}\right) * \left(1-\text{large[[k+1]]}\right) * \left(1-\text{large[[k+2]]}\right)\right) \Big/ \left(\text{Length[large]}-2\right)\right]$$

0.117847

$$\text{largefrq001} = \text{N}\left[\sum_{k=1}^{\text{Length[large]}-2} \left(\left(1-\text{large[[k]]}\right) * \left(1-\text{large[[k+1]]}\right) * \left(\text{large[[k+2]]}\right)\right) \Big/ \left(\text{Length[large]}-2\right)\right]$$

0.12465

```
largefrq010 = N[Sum[((1 - large[[k]]) * (large[[k + 1]]) * (1 - large[[k + 2]])), {k, 1, Length[large] - 2}] /
    (Length[large] - 2)]
```

0.132853

```
largefrq011 =
  N[Sum[((1 - large[[k]]) * (large[[k + 1]]) * (large[[k + 2]])), {k, 1, Length[large] - 2}] / (Length[large] - 2)]
```

0.122449

```
largefrq100 = N[Sum[((large[[k]]) * (1 - large[[k + 1]]) * (1 - large[[k + 2]])), {k, 1, Length[large] - 2}] /
    (Length[large] - 2)]
```

0.12445

```
largefrq101 =
  N[Sum[((large[[k]]) * (1 - large[[k + 1]]) * (large[[k + 2]])), {k, 1, Length[large] - 2}] / (Length[large] - 2)]
```

0.130652

```
largefrq110 =
  N[Sum[((large[[k]]) * (large[[k + 1]]) * (1 - large[[k + 2]])), {k, 1, Length[large] - 2}] / (Length[large] - 2)]
```

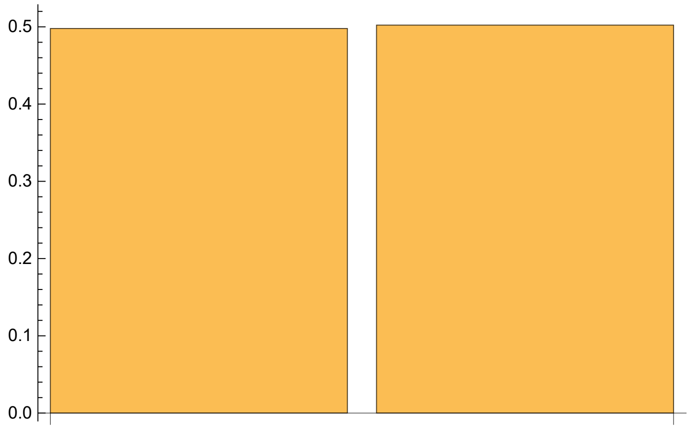
0.122449

```
largefrq111 =
  N[Sum[((large[[k]]) * (large[[k + 1]]) * (large[[k + 2]])), {k, 1, Length[large] - 2}] / (Length[large] - 2)]
```

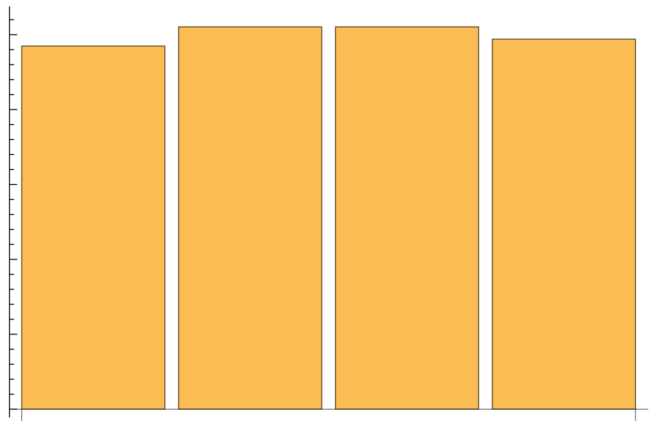
0.12465

```
(* Graphing large *)
```

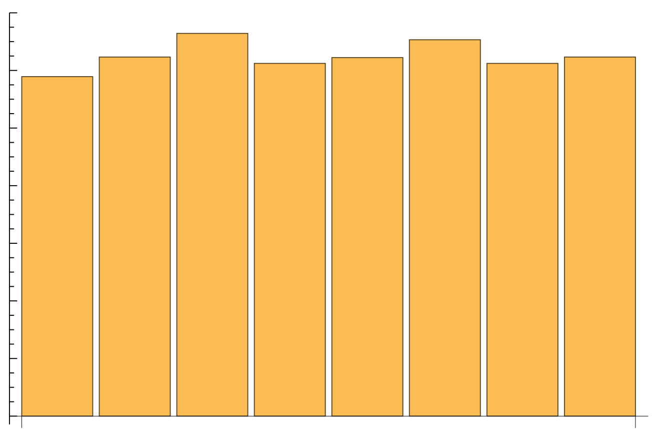
```
BarChart[{largefrq0, largefrq1}]
```




```
(* This graph is very equal around .5
 and was expected due to the law of large numbers *)
BarChart[{largefrq00, largefrq01, largefrq10, largefrq11}]
```



```
(* This graph is very equal around .25
 and was expected due to the law of large numbers *)
BarChart[{largefrq000, largefrq001, largefrq010, largefrq011,
  largefrq100, largefrq101, largefrq110, largefrq111}]
```



```
(* This graph is very equal around 1/8
 and was expected due to the law of large numbers *)

(* These graphs are all more equal compared
 to the small and medium lengthed array graphs. *)

(* Champernowne String *)
(* small *)

a = {}; (* The string *)


(* Building the champernowne string.*)

(* Gets the each number 1-50 in a binary list,
loops though each list to append to a single list, a *)
i = 1;
While[i < 51,
 x = IntegerDigits[i, 2]; (* Gets the binary digits of i, where i = 1,2,3, etc. *)
 For[k = 1, k ≤ Length[x], k++, AppendTo[a, x[[k]]]];
  (* Loop through these binary digits, add them one by one to "a" *)
 i++]


(* Now, "a" should contain the champernowne digits *)

a

{1, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1,
 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0,
 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1,
 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0,
 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1,
 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 1, 0}

(* 0 and 1 *)
(* Sum up the digits and divide by length to find the frequency of 1's *)
champernowne0 = N[Sum[(a[[i]]), {i, 1, Length[a]}]] / Length[a]

0.559671

champernowne1 = 1 - champernowne0

0.440329
```

```
(* 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)

champernowne00 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)

0.177686

champernowne01 = N[Sum[(1 - a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)

0.260331

champernowne10 = N[Sum[(a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)

0.264463

champernowne11 = N[Sum[(a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)

0.297521

(* 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Find the frequency of a three-digit binary pair, this function was created in class.
  Do this for each pair *)

champernowne000 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] /
  (Length[a] - 2)

0.06639

champernowne001 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)

0.112033

champernowne010 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)

0.120332

champernowne011 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)

0.141079

champernowne100 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)

0.112033

champernowne101 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)

0.149378
```

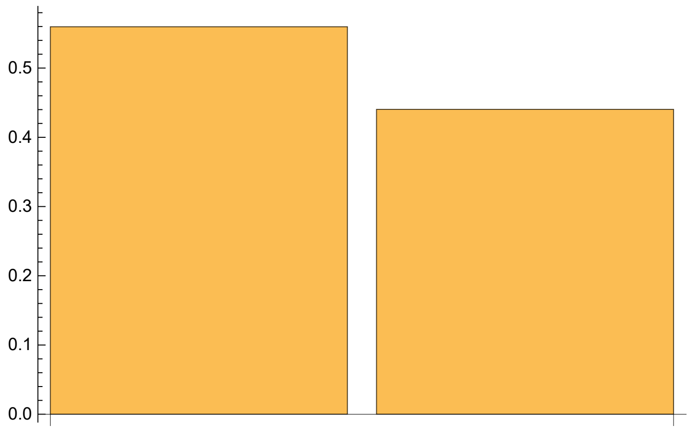
```
champernowne110 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
```

```
0.145228
```

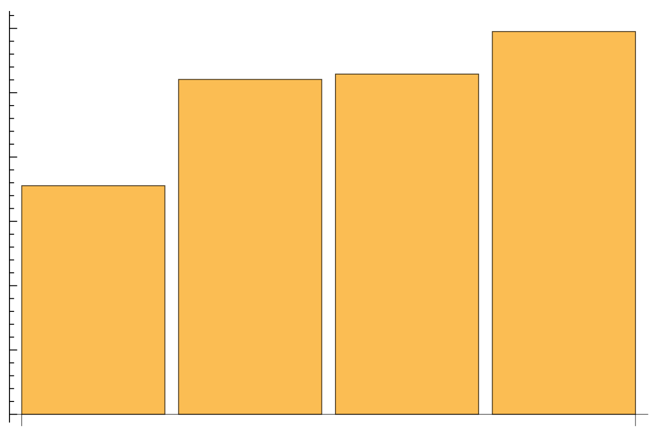
```
champernowne111 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
```

```
0.153527
```

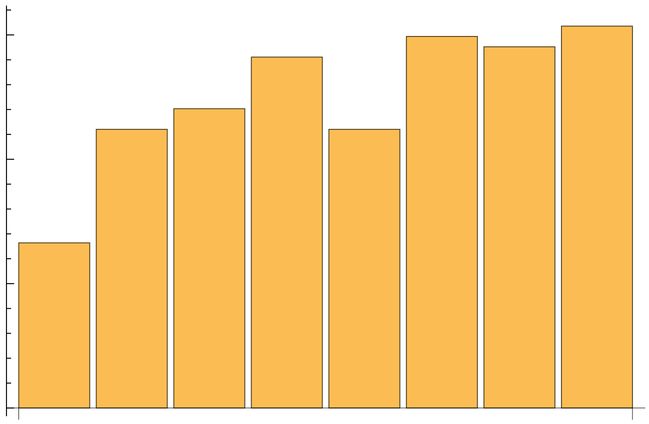
```
(* Graph 0 and 1 *)
BarChart[{champernowne0, champernowne1}]
```




```
(* This is not centered at .5, which is expected because it was a small data set *)
(* Graph 00, 01, 10, 11 *)
BarChart[{champernowne00, champernowne01, champernowne10, champernowne11}]
```



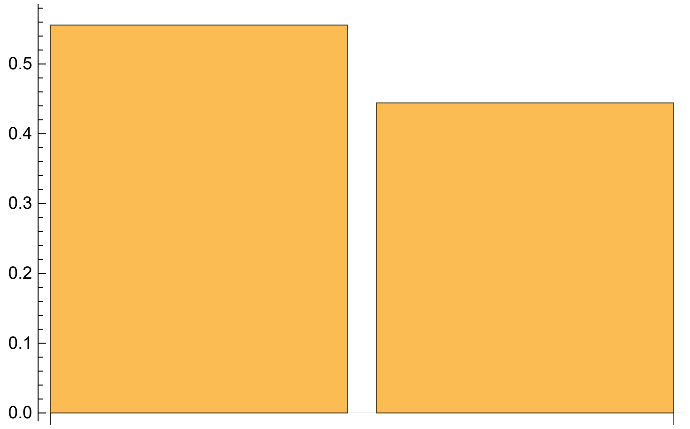
```
(* This is not centered at .25, which is expected because it was a small data set *)
(* Graph 000, 001, 010, 011, 100, 101, 110, 111 *)
BarChart[{champernowne000, champernowne001, champernowne010, champernowne011,
  champernowne100, champernowne101, champernowne110, champernowne111}]
```



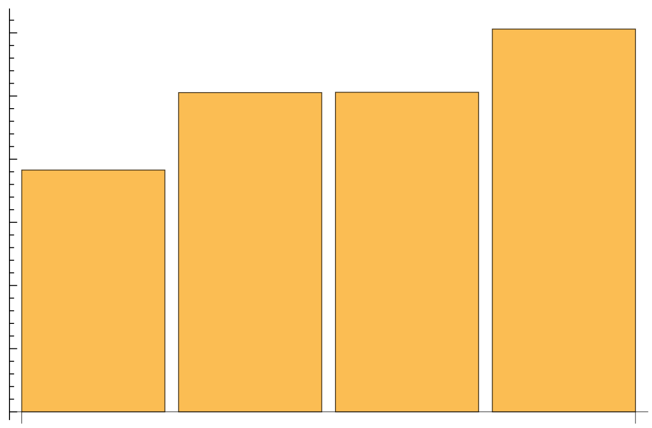
```
(* This is not centered at 1/8, which is expected because it was a small data set *)
(* Conclusions so far: The law of large number should help us
   get a more uniform graph when we graph larger champernowe lists *)
(* Champernowne medium *)


(* Append more digits to a, now a contains 500 digits of champernowne *)

i = 51;
While[i < 501,
 x = IntegerDigits[i, 2];
 For[k = 1, k ≤ Length[x], k++, AppendTo[a, x[[k]]]];
 (* Append the digits 51 through 500 to the string,
 now the string contains 1 through 500 *)
 i++]

(* Frequency 0 and 1 *)


(* Sum up the digits and divide by length to find the frequency of 1's *)

champernowne0 = N[Sum[(a[[i]]), {i, 1, Length[a]}]] / Length[a]

0.555778

champernowne1 = 1 - champernowne0

0.444222

(* Frequency 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)
champernowne00 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
0.191394
```

```
champernowne01 = N[Sum[(1 - a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
0.25269

champernowne10 = N[Sum[(a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
0.25294

champernowne11 = N[Sum[(a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
0.302977

(* Frequency 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Find the frequency of a three-digit binary pair, this function was created in class.
  Do this for each pair *)
champernowne000 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] /
  (Length[a] - 2)
0.0800801

champernowne001 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.111111

champernowne010 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.111612

champernowne011 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.141141

champernowne100 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.111361

champernowne101 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.141642

champernowne110 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.141391

champernowne111 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.161662
```

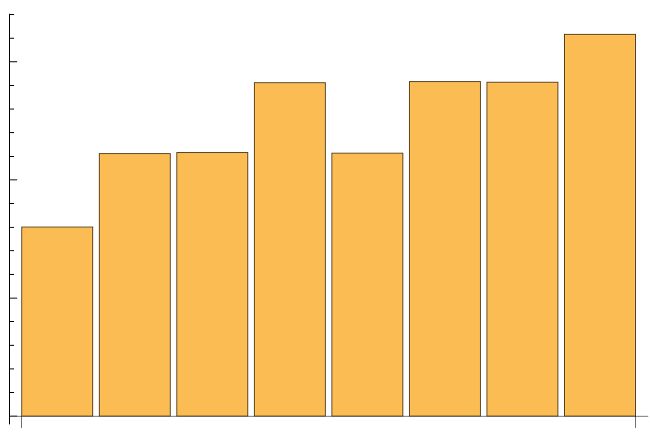
```
(* Graphing medium champernowne *)
BarChart[{champernowne0, champernowne1}]
```




```
(* This is not any closer to .5 than the small
 graph. This was not expected via the law of large numbers *)
BarChart[{champernowne00, champernowne01, champernowne10, champernowne11}]
```



```
(* This is slightly more centered compared to the small graph trial,
but it is not as centered as I expected.
  This was not expected via the law of large numbers *)
BarChart[{champernowne000, champernowne001, champernowne010, champernowne011,
  champernowne100, champernowne101, champernowne110, champernowne111}]
```



```
(* This is not any closer to 1/8 than the small
 graph. This was not expected via the law of large numbers *)
(* In conclusion so far: We will need a much larger
   champernowe list to generate a more equal equipartition plot *)

(* Champernowne Large *)

(* I will make a champernowne list of 8196 digits *)
large = {} (* I'm starting a blank list, called large *)
i = 1;
While[i < 8197, (* Loop until 8187, which is an arbitrarily large number. *)
 x = IntegerDigits[i, 2]; (* Get the integer digits of i, where i = 1,2,3, etc. *)
 For[k = 1, k ≤ Length[x], k++, AppendTo[large, x[[k]]]];
 (* Append these digits to the "large" list. *)
 i++]
```

```
{}
```

```
(* To confirm that a large amount of compernowne digits are in this list,
and no weird overflow or
 error occured, the length of this list should be very large.  *)
Length[large]
```

```
98 375
```

```
(* Frequencies for 0 and 1 *)
(* Sum up the digits and divide by length to find the frequency of 1's *)
champernowne0 = N[Sum[(a[[i]]), {i, 1, Length[a]}]] / Length[a]
```

```
0.538197
```

```
champernowne1 = 1 - champernowne0
```

```
0.461803
```

```
(* Frequency 00, 01, 10, 11 *)
(* Find the frequency of a two-digit binary pair, this function was created in class.
  Do this for each pair *)
champernowne00 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
```

```
0.212003
```

```
champernowne01 = N[Sum[(1 - a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
```

```
0.249802
```

```
champernowne10 = N[Sum[(a[[i]]) * (1 - a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
```

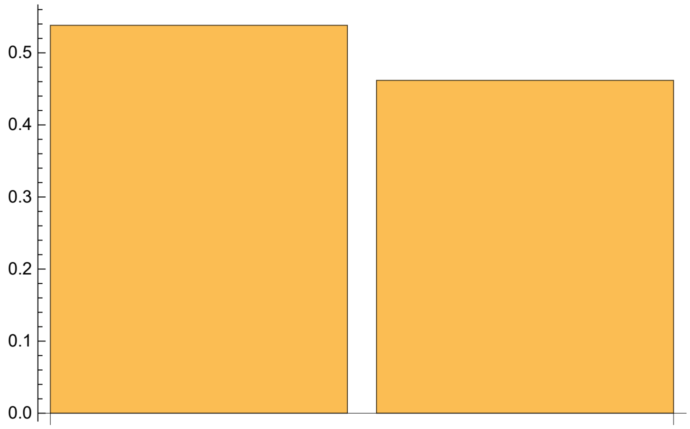
```
0.249802
```

```
champernowne11 = N[Sum[(a[[i]]) * (a[[i + 1]]), {i, 1, Length[a] - 1}]] / (Length[a] - 1)
```

```
0.288393
```

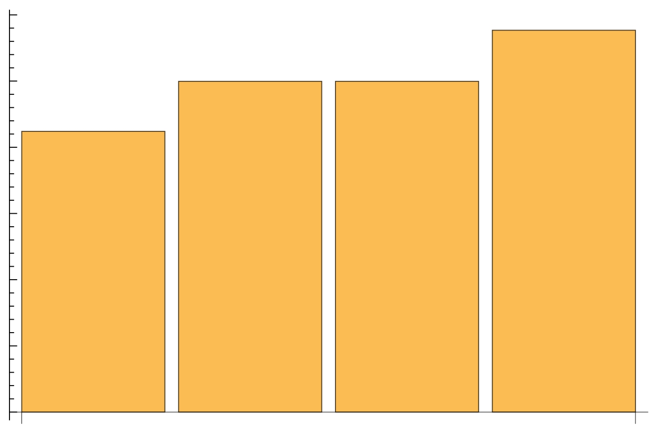
```
(* Frequency 000, 001, 010, 011, 100, 101, 110, 111 *)
(* Find the frequency of a three-digit binary pair, this function was created in class.
  Do this for each pair *)
champernowne000 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] /
  (Length[a] - 2)
0.0960299

champernowne001 =
 N[Sum[(1 - a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.115974

champernowne010 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.115532

champernowne011 =
 N[Sum[(1 - a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.13427

champernowne100 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.115974

champernowne101 =
 N[Sum[(a[[i]]) * (1 - a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.133829

champernowne110 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (1 - a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.13427

champernowne111 =
 N[Sum[(a[[i]]) * (a[[i + 1]]) * (a[[i + 2]]), {i, 1, Length[a] - 2}]] / (Length[a] - 2)
0.154121
```

```
(* Grahping champernowne large *)
BarChart[{champernowne0, champernowne1}]
```

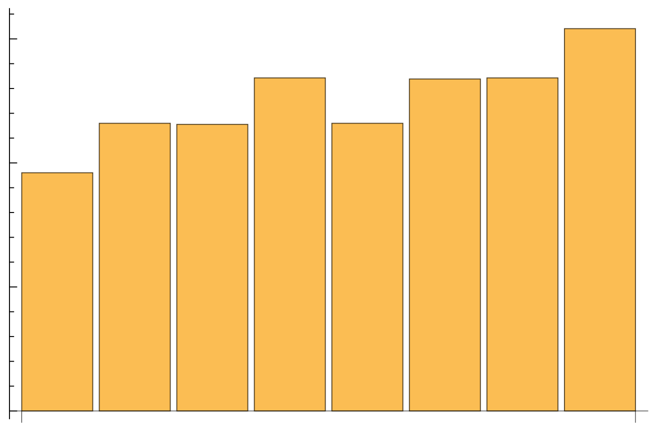



```
(* This is slightly better than the medium graph of
 champernowne equipartitions. It's not ideal, but it shows that
 we are making progress using the law of large numbers to
 back our intuition that a larger list will create a more
 equal equiparition *)
```

```
BarChart[{champernowne00, champernowne01, champernowne10, champernowne11}]
```



```
(* This is noticibly better than the medium graph of
 champernowne equipartitions. It's not ideal, but it shows that
 we are making progress using the law of large numbers to
 back our intuition that a larger list will create a more
 equal equiparition *)
BarChart[{champernowne000, champernowne001, champernowne010, champernowne011,
  champernowne100, champernowne101, champernowne110, champernowne111}]
```



```
(* This is slightly better than the medium graph of
 champernowne equipartitions. It's not ideal, but it shows that
 we are making progress using the law of large numbers to
 back our intuition that a larger list will create a more
 equal equiparition *)

(* Conclusion: In mathematica,
a random binary string satisfies the equiparition property of levels 1,
2, and 3. In the champernowne string,
the law of large numbers tells us that we are getting closer to an
 equiparition property being satisfied for the string. Our data showed this,
as the property became more apparent as we increased our data size. However,
the graphs did not show enough evidence to conclude, in my opinion,
that the property holds for the sizes that we tested. If all of the bars in
 the bar charts were almost equal in magnitude, I would conclude that the
 property holds. Even though the bars got more similar as the size increased,
they still ended up being several percentages off from each other.
```