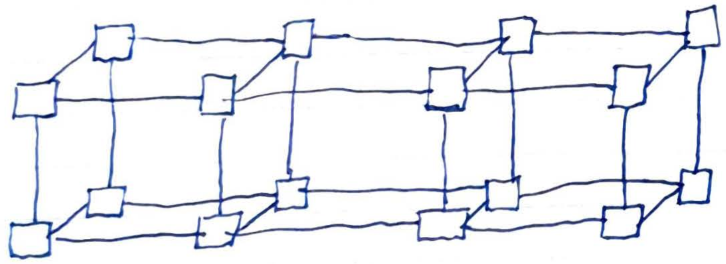
# CPSC 375
# Assignment 2

Prabhat Bhootra '20

2.1.a) $2+1+1+1+1+1+2 = 9$ nanoseconds
b) $9 \times 1000 = 9000$ nanoseconds
c) $1000+7 = 1007$ nanoseconds
d) The time taken will increase by 3 nanoseconds, if there is a level 1 cache miss. If there is a level 2 miss, then the time taken will increase by 48 nanoseconds.

2.10 a)
$$\left[\frac{10^{12}}{p \times 10^{6}} + \left(10^{9}(p-1) \times \frac{1}{10^{9}}\right)\right]$$
$$= \frac{10^{12}}{10^{9}} + \frac{10^{9}(999)}{10^{9}}$$
$$= 10^{3} + 999$$
$$= 1999 \text{ seconds}$$
b)
$$\frac{10^{12}}{10^{9}} + \frac{10^{9}(999)}{10^{3}}$$
$$= 10^{3} + 10^{6}(999)$$
$$= 999001000 \text{ seconds}$$

2.13 a)


b) For a 0-dimensional hypercube, the bisection width is $0/2 = 0$.
For a 1-dimensional hypercube, the bisection width is $2/2 = 1$.
Assuming that for a n-dimensional hypercube with p nodes, the bisection width is $p/2$.
Let us consider a n+1-dimensional hypercube. This is a combined hypercube of two n-dimensional hypercubes, each with p nodes. Therefore, the bisection width of the n+1-dimensional hypercube is

the number of edges between the two n-dimensional hypercubes, which is p. Since, the n+1 - dimensional hypercube has 2p nodes it can be seen that the bisection width is then $2p/2 = p$.

2.15 a) Core 0 will update the value of x and send a signal along the bus. If core 1 tries to execute $y=x$ before the signal is received, no value will be assigned to y, however if core 1 receives the signal before $y=x$ is executed then y will be assigned the value 5.

b) If core 0 updates the directory after core 1 executes $y=x$ then no value will be assigned to y. However, if core 0 updates the directory before core 1 then core 1 will know that core 0's cache line changed and the value of y will be 5.

c) A write-through cache could be used so that a cache line is written to main memory when it is written to the cache.

2.16 CODE AND OUTPUT IS ATTACHED TO THE END OF THE HOMEWORK

a) As n is held constant, while p increases the speedup increases while the efficiency decreases.
As p is held constant, while n increases the speedup and efficiency barely change.

b) $$\text{Parallel Efficiency} = \frac{T_{serial}}{p \cdot T_{parallel}} = \frac{T_{serial}}{p\left(T_{serial}/p + T_{overhead}\right)}$$
$$= \frac{T_{serial}}{T_{serial} + p \cdot T_{overhead}}$$

If $T_{overhead}$ grows slower than $T_{serial}$, the parallel efficiency will increase as the problem size increases, because the numerator will increase more than the denominator.
If $T_{overhead}$ grows faster than $T_{serial}$, the parallel efficiency will decrease as the problem size decreases, because the denominator will

increase faster than the numerator.

2.19

$$\text{efficiency} = \frac{n}{p \cdot (n/p + \log_2(p))}$$
$$= \frac{n}{n + p\log_2(p)}$$

If $p$ increases by a factor of $k$:

$$\frac{n}{n+p\log_2(p)} = \frac{xn}{xn + kp\log_2(kp)}$$

$$xn^2 + kpn\log_2(kp) = xn^2 + pxn\log_2(p)$$
$$xn^2 + kpn\log_2(kp) = x[n^2 + pn\log_2(p)]$$
$$kpn\log_2(kp) = x[\cancel{n^2} + pn\log_2(p) \cancel{- n^2}]$$
$$x = \frac{kpn\log_2(kp)}{pn\log_2(p)}$$
$$= \frac{k\log_2(kp)}{\log_2(p)}$$

~~x~~ $n$ should increase by $k\log_2(kp)/\log_2(p)$ to keep efficiency constant.

$$x = \frac{2\log_2 2p}{\log_2 p}$$
$$= \frac{2(\log_2 2 + \log_2 p)}{\log_2 p}$$
$$= \frac{2(1 + \log_2 p)}{\log_2 p}$$
$$= \frac{2}{\log_2 p} + 2$$

$x \neq 2$, therefore the program is not scalable.

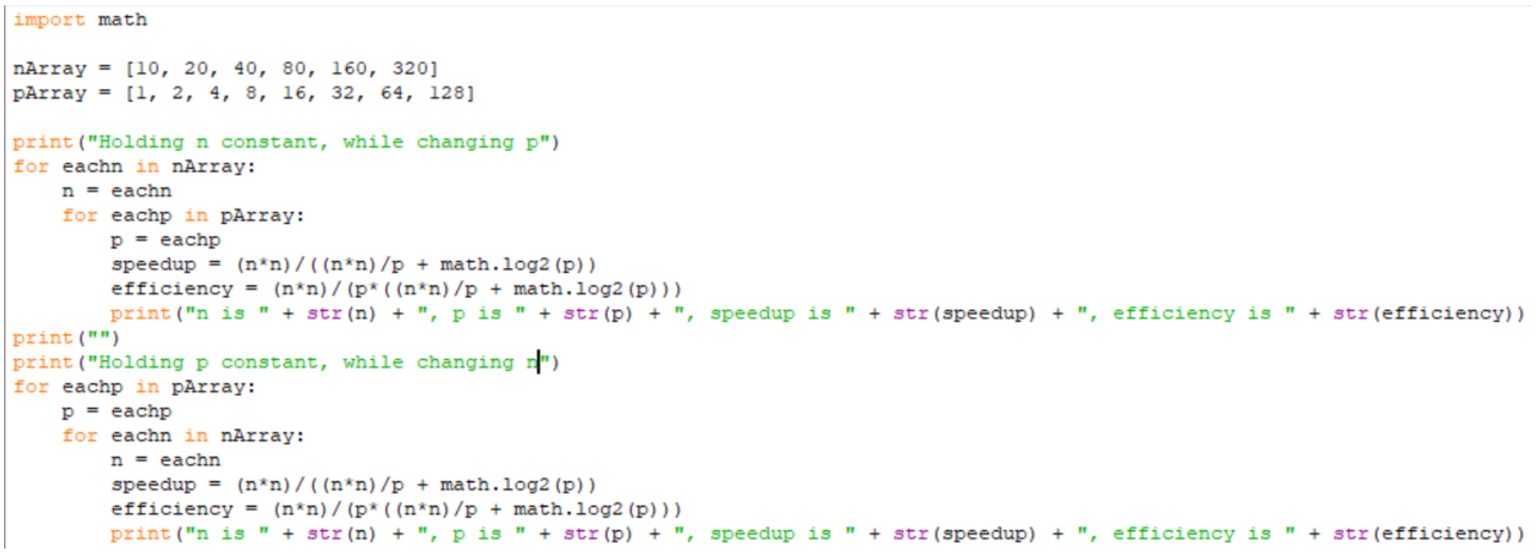
2.21 Bob should use it to time the First data set to see if there are minor differences in the original timings. Bob does not need to use it on the second data as the differences in times are more clearly visible.

~~2.23 In our application of Foster's methodology to the construction of a histogram, we essentially identified aggregate~~

2.23 This aggregation might be a problem as Find-bin will be called for certain data elements many times but the appropriate bin_count element will not be incremented at that time so this will have many extra executions of Find-bin code which is inefficient.

2.`16:

Code in python:



```python
import math

nArray = [10, 20, 40, 80, 160, 320]
pArray = [1, 2, 4, 8, 16, 32, 64, 128]

print("Holding n constant, while changing p")
for eachn in nArray:
    n = eachn
    for eachp in pArray:
        p = eachp
        speedup = (n*n)/((n*n)/p + math.log2(p))
        efficiency = (n*n)/(p*((n*n)/p + math.log2(p)))
        print("n is " + str(n) + ", p is " + str(p) + ", speedup is " + str(speedup) + ", efficiency is " + str(efficiency))
print("")
print("Holding p constant, while changing n")
for eachp in pArray:
    p = eachp
    for eachn in nArray:
        n = eachn
        speedup = (n*n)/((n*n)/p + math.log2(p))
        efficiency = (n*n)/(p*((n*n)/p + math.log2(p)))
        print("n is " + str(n) + ", p is " + str(p) + ", speedup is " + str(speedup) + ", efficiency is " + str(efficiency))
```

Output:

```
Holding n constant, while changing p

n is 10, p is 1, speedup is 1.0, efficiency is 1.0

n is 10, p is 2, speedup is 1.9607843137254901, efficiency is 0.9803921568627451

n is 10, p is 4, speedup is 3.7037037037037037, efficiency is 0.9259259259259259

n is 10, p is 8, speedup is 6.451612903225806, efficiency is 0.8064516129032258

n is 10, p is 16, speedup is 9.75609756097561, efficiency is 0.6097560975609756

n is 10, p is 32, speedup is 12.307692307692308, efficiency is 0.38461538461538464

n is 10, p is 64, speedup is 13.223140495867769, efficiency is 0.2066115702479339

n is 10, p is 128, speedup is 12.85140562248996, efficiency is 0.10040160642570281

n is 20, p is 1, speedup is 1.0, efficiency is 1.0

n is 20, p is 2, speedup is 1.9900497512437811, efficiency is 0.9950248756218906

n is 20, p is 4, speedup is 3.9215686274509802, efficiency is 0.9803921568627451

n is 20, p is 8, speedup is 7.547169811320755, efficiency is 0.9433962264150944

n is 20, p is 16, speedup is 13.793103448275861, efficiency is 0.8620689655172413

n is 20, p is 32, speedup is 22.857142857142858, efficiency is 0.7142857142857143

n is 20, p is 64, speedup is 32.6530612244898, efficiency is 0.5102040816326531

n is 20, p is 128, speedup is 39.50617283950617, efficiency is 0.30864197530864196

n is 40, p is 1, speedup is 1.0, efficiency is 1.0
```

```
n is 40, p is 2, speedup is 1.9975031210986267, efficiency is 0.9987515605493134
n is 40, p is 4, speedup is 3.9800995024875623, efficiency is 0.9950248756218906
n is 40, p is 8, speedup is 7.8817733990147785, efficiency is 0.9852216748768473
n is 40, p is 16, speedup is 15.384615384615385, efficiency is 0.9615384615384616
n is 40, p is 32, speedup is 29.09090909090909, efficiency is 0.9090909090909091
n is 40, p is 64, speedup is 51.61290322580645, efficiency is 0.8064516129032258
n is 40, p is 128, speedup is 82.05128205128206, efficiency is 0.6410256410256411
n is 80, p is 1, speedup is 1.0, efficiency is 1.0
n is 80, p is 2, speedup is 1.999375195251484, efficiency is 0.999687597625742
n is 80, p is 4, speedup is 3.9950062421972534, efficiency is 0.9987515605493134
n is 80, p is 8, speedup is 7.970112079701121, efficiency is 0.9962640099626401
n is 80, p is 16, speedup is 15.841584158415841, efficiency is 0.9900990099009901
n is 80, p is 32, speedup is 31.21951219512195, efficiency is 0.975609756097561
n is 80, p is 64, speedup is 60.37735849056604, efficiency is 0.9433962264150944
n is 80, p is 128, speedup is 112.28070175438596, efficiency is 0.8771929824561403
n is 160, p is 1, speedup is 1.0, efficiency is 1.0
n is 160, p is 2, speedup is 1.9998437622060776, efficiency is 0.9999218811030388
n is 160, p is 4, speedup is 3.998750390502968, efficiency is 0.999687597625742
n is 160, p is 8, speedup is 7.992507024664377, efficiency is 0.9990633780830471
n is 160, p is 16, speedup is 15.960099750623442, efficiency is 0.9975062344139651
n is 160, p is 32, speedup is 31.801242236024844, efficiency is 0.9937888198757764
n is 160, p is 64, speedup is 63.05418719211823, efficiency is 0.9852216748768473
n is 160, p is 128, speedup is 123.67149758454106, efficiency is 0.966183574879227
n is 320, p is 1, speedup is 1.0, efficiency is 1.0
n is 320, p is 2, speedup is 1.9999609382629246, efficiency is 0.9999804691314623
n is 320, p is 4, speedup is 3.9996875244121552, efficiency is 0.9999218811030388
n is 320, p is 8, speedup is 7.998125439350153, efficiency is 0.9997656799187691
n is 320, p is 16, speedup is 15.99000624609619, efficiency is 0.9993753903810119
n is 320, p is 32, speedup is 31.950078003120126, efficiency is 0.9984399375975039
n is 320, p is 64, speedup is 63.760896637608965, efficiency is 0.9962640099626401
n is 320, p is 128, speedup is 126.88971499380422, efficiency is 0.9913258983890955
```

Holding p constant, while changing n

```
n is 10, p is 1, speedup is 1.0, efficiency is 1.0

n is 20, p is 1, speedup is 1.0, efficiency is 1.0

n is 40, p is 1, speedup is 1.0, efficiency is 1.0

n is 80, p is 1, speedup is 1.0, efficiency is 1.0

n is 160, p is 1, speedup is 1.0, efficiency is 1.0

n is 320, p is 1, speedup is 1.0, efficiency is 1.0

n is 10, p is 2, speedup is 1.9607843137254901, efficiency is 0.9803921568627451

n is 20, p is 2, speedup is 1.9900497512437811, efficiency is 0.9950248756218906

n is 40, p is 2, speedup is 1.9975031210986267, efficiency is 0.9987515605493134

n is 80, p is 2, speedup is 1.999375195251484, efficiency is 0.999687597625742

n is 160, p is 2, speedup is 1.9998437622060776, efficiency is 0.9999218811030388

n is 320, p is 2, speedup is 1.9999609382629246, efficiency is 0.9999804691314623

n is 10, p is 4, speedup is 3.7037037037037037, efficiency is 0.9259259259259259

n is 20, p is 4, speedup is 3.9215686274509802, efficiency is 0.9803921568627451

n is 40, p is 4, speedup is 3.9800995024875623, efficiency is 0.9950248756218906

n is 80, p is 4, speedup is 3.9950062421972534, efficiency is 0.9987515605493134

n is 160, p is 4, speedup is 3.998750390502968, efficiency is 0.999687597625742

n is 320, p is 4, speedup is 3.9996875244121552, efficiency is 0.9999218811030388

n is 10, p is 8, speedup is 6.451612903225806, efficiency is 0.8064516129032258

n is 20, p is 8, speedup is 7.547169811320755, efficiency is 0.9433962264150944

n is 40, p is 8, speedup is 7.8817733990147785, efficiency is 0.9852216748768473

n is 80, p is 8, speedup is 7.970112079701121, efficiency is 0.9962640099626401

n is 160, p is 8, speedup is 7.992507024664377, efficiency is 0.9990633780830471

n is 320, p is 8, speedup is 7.998125439350153, efficiency is 0.9997656799187691

n is 10, p is 16, speedup is 9.75609756097561, efficiency is 0.6097560975609756

n is 20, p is 16, speedup is 13.793103448275861, efficiency is 0.8620689655172413

n is 40, p is 16, speedup is 15.384615384615385, efficiency is 0.9615384615384616

n is 80, p is 16, speedup is 15.841584158415841, efficiency is 0.9900990099009901

n is 160, p is 16, speedup is 15.960099750623442, efficiency is 0.9975062344139651

n is 320, p is 16, speedup is 15.99000624609619, efficiency is 0.9993753903810119

n is 10, p is 32, speedup is 12.307692307692308, efficiency is 0.38461538461538464
```

```
n is 20, p is 32, speedup is 22.857142857142858, efficiency is 0.7142857142857143

n is 40, p is 32, speedup is 29.09090909090909, efficiency is 0.9090909090909091

n is 80, p is 32, speedup is 31.21951219512195, efficiency is 0.975609756097561

n is 160, p is 32, speedup is 31.801242236024844, efficiency is 0.9937888198757764

n is 320, p is 32, speedup is 31.950078003120126, efficiency is 0.9984399375975039

n is 10, p is 64, speedup is 13.223140495867769, efficiency is 0.2066115702479339

n is 20, p is 64, speedup is 32.6530612244898, efficiency is 0.5102040816326531

n is 40, p is 64, speedup is 51.61290322580645, efficiency is 0.8064516129032258

n is 80, p is 64, speedup is 60.37735849056604, efficiency is 0.9433962264150944

n is 160, p is 64, speedup is 63.05418719211823, efficiency is 0.9852216748768473

n is 320, p is 64, speedup is 63.760896637608965, efficiency is 0.9962640099626401

n is 10, p is 128, speedup is 12.85140562248996, efficiency is 0.10040160642570281

n is 20, p is 128, speedup is 39.50617283950617, efficiency is 0.30864197530864196

n is 40, p is 128, speedup is 82.05128205128206, efficiency is 0.6410256410256411

n is 80, p is 128, speedup is 112.28070175438596, efficiency is 0.8771929824561403

n is 160, p is 128, speedup is 123.67149758454106, efficiency is 0.966183574879227

n is 320, p is 128, speedup is 126.88971499380422, efficiency is 0.9913258983890955
```