Name: Georges Hatem

CS 550 Homework 5

Due Date: December 12, 2021

Problem 1:

First let's state our analysis:

The number of vertices in an N-cube is as follows:

$$Number\ of\ Vertices\ in\ an\ N - cube = 2^N$$

The number of edges in an N-cube is as follows:

$$Number\ of\ Edges\ in\ an\ N - cube = (N) * (2^{N-1})$$

A tree is always a connected graph with minimum number of edges. In a tree, the number of edges is as follows:

$$Number\ of\ Edges\ in\ a\ Tree = 2^N - 1$$

From what we stated above, now we can solve parts (a) through (d):

Part A:

Using the formulas stated above (at the beginning of Problem 1):

For 1-cube, the number of vertices in 1-cube is as follows:

$$Number\ of\ Vertices\ in\ 1 - Cube = 2^1 = 2$$

The number of edges in 1-cube is as follows:

$$Number\ of\ Edges\ in\ 1 - Cube = (1) * (2^{1-1}) = (1) * (2^0) = 1$$

The number of edges in a tree for N = 1 is as follows:

$$Number\ of\ Edges\ in\ a\ Tree = 2^1 - 1 = 1$$

Therefore, we need 1 edge (the number of edges in a tree) to keep it connected.

We currently have 1 edge by calculation of edges in 1-cube above. Therefore, we can remove 0 edges and still have full connectivity (the analysis behind 0 is as follows:)

$$Number\ of\ Edges\ can\ be\ Removed = (Number\ of\ Edges\ in\ 1 - Cube) - (Number\ of\ Edges\ in\ a\ Tree)$$

So,

$$Number\ of\ Edges\ can\ be\ Removed = 1 - 1 = 0$$

Therefore, as stated above, we can remove 0 edges and still have full connectivity.

<u>Part B:</u>

Using the formulas stated above (at the beginning of Problem 1):

For 2-cube, the number of vertices in 2-cube is as follows:

$$Number\ of\ Vertices\ in\ 2 - Cube = 2^2 = 4$$

The number of edges in 2-cube is as follows:

$$Number\ of\ Edges\ in\ 2 - Cube = (2) * (2^{2-1}) = (2) * (2^1) = 4$$

The number of edges in a tree for N = 2 is as follows:

$$Number\ of\ Edges\ in\ a\ Tree = 2^2 - 1 = 3$$

Therefore, we need 3 edges (the number of edges in a tree) to keep it connected.

We currently have 4 edges by calculation of edges in 2-cube above. Therefore, we can remove 1 edge and still

have full connectivity (the analysis behind 1 is as follows:)

$$Number\ of\ Edges\ can\ be\ Removed = (Number\ of\ Edges\ in\ 2 - Cube) - (Number\ of\ Edges\ in\ a\ Tree)$$

So,

$$Number\ of\ Edges\ can\ be\ Removed = 4 - 3 = 1$$

Therefore, as stated above, we can remove 1 edge and still have full connectivity.

<u>Part C:</u>

Using the formulas stated above (at the beginning of Problem 1):

For 3-cube, the number of vertices in 3-cube is as follows:

$$Number\ of\ Vertices\ in\ 3 - Cube = 2^3 = 8$$

The number of edges in 3-cube is as follows:

$$Number\ of\ Edges\ in\ 3 - Cube = (3) * (2^{3-1}) = (3) * (2^2) = 12$$

The number of edges in a tree for N = 3 is as follows:

$$Number\ of\ Edges\ in\ a\ Tree = 2^3 - 1 = 7$$

Therefore, we need 7 edges (the number of edges in a tree) to keep it connected.

We currently have 12 edges by calculation of edges in 3-cube above. Therefore, we can remove 5 edges and still have full connectivity (the analysis behind 5 is as follows:)

$$Number\ of\ Edges\ can\ be\ Removed = (Number\ of\ Edges\ in\ 3 - Cube) - (Number\ of\ Edges\ in\ a\ Tree)$$

So,

$$Number\ of\ Edges\ can\ be\ Removed = 12 - 7 = 5$$

Therefore, as stated above, we can remove 5 edges and still have full connectivity.

<u>Part D:</u>

Using the formulas stated above (at the beginning of Problem 1):

For N-cube, the number of vertices in N-cube is as follows:

$$Number\ of\ Vertices\ in\ N-Cube = 2^N$$

The number of edges in N-cube is as follows:

$$Number\ of\ Edges\ in\ N-Cube = (N) * (2^{N-1}) = N * 2^{N-1}$$

The number of edges in a tree in function of N is as follows:

$$Number\ of\ Edges\ in\ a\ Tree = 2^N - 1$$

Therefore, we need ((2^N) – 1) edges (the number of edges in a tree) to keep it connected.

We currently have (N*(2^(N-1))) edges by calculation of edges in N-cube above. Therefore, we can remove (((2^(N-1))*(N-2)) + 1) edges and still have full connectivity (the analysis behind this is as follows:)

$$Number\ of\ Edges\ can\ be\ Removed = (Number\ of\ Edges\ in\ N-Cube) - (Number\ of\ Edges\ in\ a\ Tree)$$

So,

$$Number\ of\ Edges\ can\ be\ Removed = \left(N * (2^{N-1})\right) - (2^N - 1)$$

$$Number\ of\ Edges\ can\ be\ Removed = \left(N * (2^{N-1})\right) - 2^N + 1$$

Factoring 2^(N-1) from the first 2 terms above, we get the following:

$$Number\ of\ Edges\ can\ be\ Removed = \left((2^{N-1}) * (N-2)\right) + 1$$

Therefore, as stated above, we can remove (((2^(N-1))*(N-2)) + 1) edges and still have full connectivity.

Problem 2:

Let's first define the Arithmetic Intensity:

As stated in Section 6.10 in our zyBook for CS 550:

The Arithmetic Intensity is the ratio of floating-point operations in a program to the number of data bytes accessed by a program from main memory.

Every single-precision number takes 4 bytes. In each iteration, the code reads (4 bytes)*(2) from the main memory, which is equal to 8 bytes, and writes 4 bytes to the main memory.

The 8 bytes reading from the main memory comes from a[i] and b[i] (4 bytes from a[i] and 4 bytes from b[i]).

The 4 bytes writing to the main memory comes from c[i].

Given the analysis above, the number of data bytes accessed by a program from main memory in the code provided in Problem 2 is 8 bytes + 4 bytes = 12 bytes.

There is only one Floating-Point Operation (c[i] = a[i]*b[i]). Therefore, the number of Floating-Point Operation is 1.

Since we have all the data we need right now, we can go ahead and calculate the Arithmetic Intensity of the code provided in Problem 2.

The Arithmetic Intensity of the code in Problem 2 is as follows:

$$Arithmetic\ Intensity = \frac{(Number\ of\ FLOPS)}{Number\ of\ data\ bytes\ from\ main\ memory}$$

$$Arithmetic\ Intensity = \frac{1\ FLOP}{12\ Bytes} = 0.0833\ FLOPs/Byte$$

So, the Arithmetic Intensity of the code in Problem 2) is 0.0833 FLOPs/Byte

Problem 3:

Part A:

The Peak Single Precision Floating Point (SPFP) is as follows:

$$Peak\ SPFP = (Clock\ Rate) * (Number\ of\ SIMD\ Processors) * (SPFP\ Units)$$

$$Peak\ SPFP = (1.5\ GHz) * (16\ SIMD\ Processors) * (16\ SPFP\ Units)$$

$$Peak\ SPFP = 384\ GFLOP/sec$$

So, the Peak Single Precision Floating Point (SPFP) Operation is 384 GFLOP/sec

Part B:

Assuming each operation is an addition as Problem 3 stated, this means the following:

Each single precision operation requires 4 bytes for each of the 2 input values and 4 bytes for the output value (the result). This means that each single precision operation requires a total of 12 bytes.

Let's compute the bandwidth we get from this throughput and compare to the memory bandwidth given in this problem:

$$Bandwidth = \left(12\,\frac{Bytes}{FLOP}\right) * (Peak\ SPFP)$$

$$Bandwidth = \left(12\,\frac{Bytes}{FLOP}\right) * \left(384\,\frac{GFLOP}{Sec}\right) = 4608\ GB/Sec$$

Since 4608 GB/Sec >> 100 GB/Sec, this throughput is not sustainable, but can still be achieved in short bursts when using on-chip cache.

<u>Problem 4:</u>

To calculate the Throughput, we need to do as follows:

$$Throughput = (Active\ Thread\ Percentage) * [(Clock\ Rate) * (Number\ of\ SIMD\ Processor) * (SPFP\ Units)]$$

$$Throughput = (0.8) * [(4\ GHz) * (16\ SIMD\ Processor) * (24)]$$

$$Throughput = 1228.8\,\frac{GFLOP}{Sec}$$

So, the throughput in GFLOP/Sec for this code on this GPU is 1228.8 GFLOP/Sec.