**CIT 285: Midterm Exam Answers**

1. a. O(n2) b. O(n2)
2. Binary Search has time complexity of O(log n). Complexity for inserting is O(n + m), where (n) is the number of elements to insert and (m) is the number of elements required to shift.
3. Since the queue has been sorted with the smallest element first, the time complexity is O(1) or constant time. We can access this element by simply using the queue.front() stl function.
4. Popping an element from the queue is done in constant time and is only performed on one element. Therefore, popping from queue has O(1) or constant time complexity.
5. Claim: “There exists a valid schedule if and only if each stream i satisfies bi <= rti”.

This is false. The ‘r’ condition follows:

(∗) For each natural number t > 0, the total number of bits you send over the

time interval from 0 to t cannot exceed rt.

Note that this constraint is only imposed for time intervals that start at

0, not for time intervals that start at any other value.

Therefore, a valid schedule exists if bn <= rtn. I’m not sure if that notation is correct so in plain English, as long as the total bits are less than or equal to r times total time intervals, and the size(b) at t=0 does not exceed r(t), a valid schedule exists. This is true for processing the entire list of streams.

1. Use a vector to store the stream bits and length intervals.

Create a struct to hold stream member properties of size(bits) and length.

Bool answer = true

Vector<stream> streams = list of streams

Sort(streams) by bit size

Constant r = link limit

Streams iterator = streams.begin()

Int totalLength = 0

int totalBits = 0;

int limit = r;

While answer is true and not at end of streams vector

totalBits = totalBits + iterator bits

totalLength = totalLength + iterator length

limit = r \* totalLength

If the totalBits are greater than the limit

Answer is false

Else

Process the next stream

Endwhile

If answer is false

Print (“Stream schedule is invalid”)

Else

Print (“Valid stream schedule”)