COMP90056 Stream Computing and Applications
Assignment B

Theory question

Student Name: Zhuohan Xie
Student Number: 871089

(a) A simple example is a frequency vector <5, 5, -1>.

We have $U = \sum_i (i \times f_i) = 1 \times 5 + 2 \times 5 + 3 \times (-1) = 12$

$V = \sum_i (i^2 \times f_i) = 1^2 \times 5 + 2^2 \times 5 + 3^2 \times (-1) = 16$

$F_1 = \sum_i f_i = 5 + 5 - 1 = 9$

Therefore, we have $U^2 = 12^2 = 144 = 16 \times 9 = F_1 V$. It passes Ganguly's test and will be determined as 1-sparse vector by Ganguly's technique, but clearly it is not a 1-sparse, therefore, it is a false positive.

The way I got this is:
$U = \sum_i (i \times f_i)$,$V = \sum_i (i^2 \times f_i)$, $F_1 = \sum_i f_i$ and we want $U^2 = F_1 V$. That is$(\sum_i i \times f_i)^2 = \sum_i f_i \times \sum_i (i^2 \times f_i)$.
To make things simple, I just Let the index i starts from 1, so I have

$$(f_1 + 2f_2 + \cdots + nf_n)^2 = (f_1 + f_2 + \cdots + f_n) \times (f_1 + 4f_2 + \cdots + n^2 f_n)$$

So, when I set the n = 3, the left side would be
$(f_1 + 2f_2 + 3f_3)^2 = f_1^2 + 4f_1 f_2 + 6f_1 f_3 + 4f_2^2 + 12f_2 f_3 + 9f_3^2$
And the right side would be
$(f_1 + f_2 + f_3) \times (f_1 + 4f_2 + 9f_3) = f_1^2 + 5f_1 f_2 + 10f_1 f_2 + 4f_2^2 + 13f_2 f_3 + 9f_3^2$
To make these two equal, we just have

$$f_1 f_2 + 4f_1 f_3 + f_2 f_3 = 0$$
$$(4f_1 + f_2) f_3 = -f_1 f_2$$
$$f_3 = -\frac{f_1 f_2}{4f_1 + f_2}$$

Since the frequency can be negative number in this scenario, we can easily find appropriate f_1, f_2, f_3 meets the test, like I showed above. Even though it passes the test, it might still not be a 1-sparse, therefore, it can be a false positive.

(b) It's easy to show the process in the binary integer representation.
For any arbitrary subset $[a,\ b)$ of $[0,\ n)$, if we want to find the first largest dyadic intervals of the form $[k \cdot 2^j, (k+1) \cdot 2^j)$, we need to find the largest 2^j which can divide a. When a is in the binary representation, j would be the number of tailing 0s of a, that is, if a is 4, binary representation would be 100, and j would be 2, so the dyadic interval is $[1 \times 2^2, (1+1) \times 2^2) = [4, 8)$ if $b \geq 8$ and 8 could be represented as 1000. And if a is 5, binary representation, binary representation would be 101, j would be 0, so the dyadic interval is $[5 \times 2^0, (5+6) \times 2^0) = [5, 6)$ if $b \geq 6$ and 6 could be represented as 110. To conclude, when we could get the largest interval for one lower bound without thinking the upper bound would be over than range of this subset, basically in binary representation, we are adding one plus all tailing 0s of a to the a to get the upper bound of that interval.
However, there is another case, when doing so might get us over the range of subset. Let us say we have $[32, 35)$. 32 could be represented as 100000, clearly using the approach above would get us over the range, so instead of adding one plus all tailing 0s of lower bound's binary representation, we should

find the largest interval within the subset, in this case, the largest we can add is 2, which is 10 so we get $[32, 34)$ and 34 is 100010. Then, the next step is $[34, 35)$ and 35 is 100011. To analyse this, we can find that it will happen when the lower bound we have now has the same bits with the upper bound of original subset we are trying to work out the dyadic intervals and when we add one plus some zeros to this, then next step we can do is to only do the same thing with the new lower bound.

To sum up, we can find the dyadic intervals of any arbitrary subset $[a,\ b)$ of $[0,\ n)$ by adding binary representation of one plus zeros where number of zeros less than or equal to number of tailing zeros of a's binary representation.

Clearly, we don't consider n = 1, it just makes no sense here.

When b = 1, there is only one dyadic interval $[0,\ 1)$, and n should at least be 2. Therefore, $1 \leq 2\log_2 n$.

When b >1, b can be represented in binary representation by $\lfloor \log_2 b \rfloor + 1$ bits.

It takes two stages for a to achieve b. First, we add largest number as we can until a reaches the same bits as b That is to say, a at most needs to do $\lfloor \log_2 b \rfloor$ of these operations to get the same bits as b and it will at most generate at most $\lfloor \log_2 b \rfloor$ disjoint dyadic intervals. After that, when a has same bits as b, a needs to do the second operation like mentioned above, since everytime, we add a with one plus some zeros in binary representation and number of these zeros is less than the number of zeros we add last time, that is to say, for a binary representation of a, we can only turn one 0 bit into 1 when at that bit all bits at the right side are 0. And it's clear that we can only perform this $\lfloor \log_2 b \rfloor$ times and generate at most $\lfloor \log_2 b \rfloor$ disjoint dyadic intervals. And $2\lfloor \log_2 b \rfloor \leq 2\log_2 b \leq 2\log_2 n$.

Therefore, an arbitrary sub-interval of the range $[0,\ n)$ can be expressed as the union of at most $2\log_2 n$ disjoint dyadic intervals.