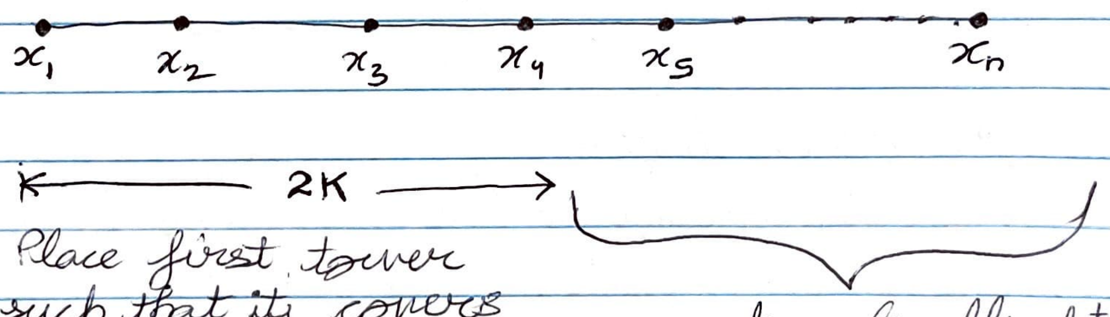
Name: Ashutosh Chauhan
UIN: 232009024

CSCE 629 Analysis Of Algorithms
Homework 2.

1.) ~~Maximize~~ Minimize the number of base stations such that all the n houses are covered inside the range.

- **Main Idea:**
  The objective is to minimize the number of base stations such that it cover all n houses. We will be using greedy approach here. We will add a tower to the solution such that it covers maximum number of possible houses from the start. Then we take remaining number of houses as new problem and move forward similarly in greedy way.



Place first tower such that it covers the maximum number of currently uncovered houses.

make a locally optimal choice at each step

- **Pseudo Code :**

```
function(house, range):
    count = 0                                           - O(1)
    i = 0                                               - O(1)
    while i < len(house)                                - O(n)
        count += 1
        while house[i] <= house[i]+2*range
            i += 1
            if i == len(house):
                break.
    return count.
```

- **Time Complexity :** $O(n)+O(1) = O(n)$

Since we are moving from left to right and calculating count of minimum base station in single loop. The complexity of above solution is $O(n)$.

- **Correctness :**

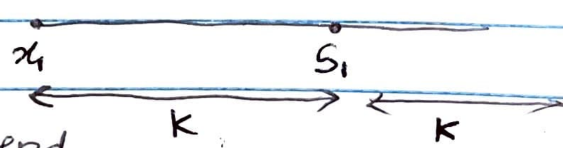
The greedy algorithm used is right because it meet the greedy choice property. The greedy choice property states that a globally optimal solution can be found by making a locally optimal choice at each step. At every step we are adding a base station such that it covers maximum local houses. We are adding a tower for a distance of 2* Range ensuring that maximum possible no. of houses are covered. By making this choice at each step, the algorithm ensures that the number of towers used is minimized. This can be proven by contradiction. Let say our solution is not optimal. Then there should be an algorithm that produces fewer base stations. However, this contradicts the fact that the algorithm's choice of base station at each step is such that it covers the maximum number.

We can show this,

Let the sol^n of our algorithm be C and place first station at $S_1 = x_1 + \left(\frac{2K}{2}\right)$.

Let us assume there is other optimal solution C' where first station is at $S_1' = x_1 + K'$

If S_1' will be before S_1, it will cover less houses and miss house towards end



as compared to S_1. If S_1' will be after S_1, it will miss few starting houses covered by S_1 range

∴ S_1' should be same as S_1 for covering max range.

2:) What is an optimal Huffman code for the following set of freq.

a : 1 ; b = 1 ; c : 2 ; d : 3 ; e : 5 ; f : 8 ; g : 13 ; h : 21

Generalize when freq are n fibonacci number.

- Main Idea:

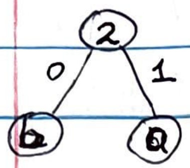
~~Co~~ Constructing the prefix-free code using greedy algorithm. The algo uses a min-priority queue Q, keyed on the freq attribute, to identify the two least-freq object and merges them. The result of merging is a new object whose freq is sum of prev two, and this continues.

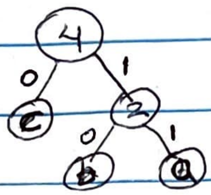
- Pseudo Code:

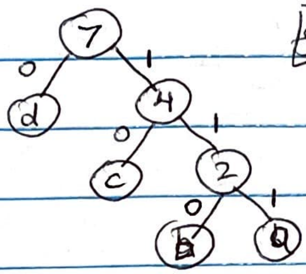
```
function (n).
    add n elements to Q
    for i = 1 to n-1 :
        create new node c
            remove
        a = min in Q
            remove
        b = min in Q.
        c.left = a
        c.right = b
        c.freq = a.freq.+ b.freq.
        Insert c in Q.
    return Q (only one element will remain)
```

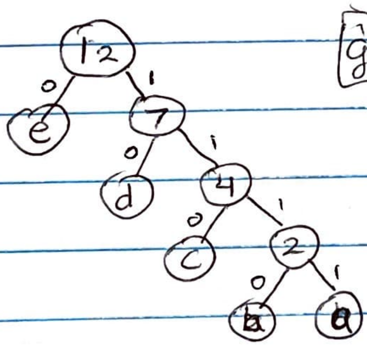
⇒ a:1 b:1 c:2 d:3 e:5 f:8 g:13 h:21

✓ ✓

⇒  c:2 d:3 e:5 f:8 g:13 h:21

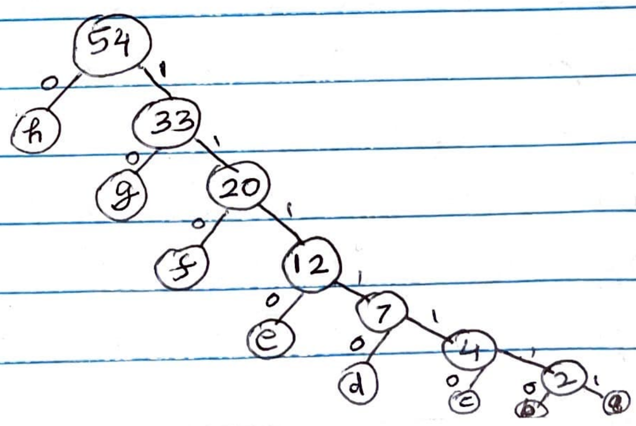
⇒ d:3  e:5 f:8 g:13 h:21

⇒ e:5  f:8 g:13 h:21

⇒ f:8  g:13 h:21

⇒ Similarly

⋮

⇒ final 

∴ From above binary tree ∴
Optimal Huffman code is as below ∴

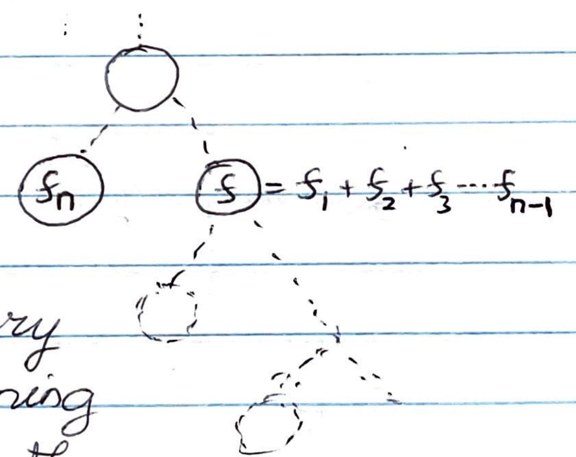
h:21	g:13	f:8	e:5	d:3	c:2	b:1	a:1
0	10	110	1110	11110	111110	1111110	11111110

⇒ Generalizing for n terms :-
from above table we could observe that code for i^{th} term of n terms is as follows ∴

$$C_i = \begin{cases} 1^{(n-1)\text{ times}} & ; i = 1 \\ 1^{(n-i)\text{ times}}0 & ; i > 1 \end{cases}$$

where $i = 1$ is the smallest freq term and $i = n$ is the highest freq term.

- <u>Correctness</u>:



While solving the huffman code for 8 fibbonaci numbers we observed that at every step we are combining two frequencies, first is the fibbonaci number at that step let say f_n and second is the freqvency of sum of all the earlier number i.e $f = f_1 + f_2 + \cdots + f_{n-1}$

So we need to proof that if the freq of node at any step is greater than the freq of its child nodes then our solution will be an optimal solution because the high freq elements will be up in tree and hence will have shorter code length. If we will prove that at every step the two lowest frequencies are combined, we could show that we have optimised code.
We will use induction for this.

⇒ for ~~n~~ = 1., we are combining least freq element a & b. so we know it is true.

⇒ Let us assume it is true for n=K

⇒ We have to show it is true for n = K+1
we will combine f_{K+2} and $(f_1 + f_2 + \cdots + f_{K+1})$
i.e we have to show

$$f_1 + f_2 + f_3 \cdots + f_{K+1} < f_{K+3} \quad -①$$

for K=1, $f_1 + f_2 < f_4$, ⇒ True. -②
(a:1) (b:1) (d:3)

Assume it is true for K = x.

$$(f_1 + f_2 + f_3 \cdots + f_x) + f_{(x+1)} < f_{x+3} \quad -③$$

Now for $K = x+1$

$$f_1 + f_2 + \dots + f_{x+1} + f_{x+2} < f_{x+3} + f_{x+2} \quad -④$$

↳ adding f_{x+2} to both sides in eqn ③

from fibbonaci series definition.

$$f_{x+3} + f_{x+2} = f_{x+4} \quad -⑤$$

using eqn ④ & ⑤

$$f_1 + f_2 + \dots f_{x+2} < f_{x+4}$$

Keeping $x = k-1$, the above exp become

$$f_1 + f_2 + \dots f_{k+1} < f_{k+3}$$

Hence proved.

As stated earlier this will proof that we are taking least frequencies at every step and the tree will more towards right side only as $(f_1 + \dots f_{k+1})$ will be always smaller than f_{k+3}. Due to this we can generalize the code of n^{th} term of fibbo series easily. Also all the bigger frequencies will be above and will have smaller code length compared to lower freq term which will ~~reduce~~ minimize our overall cost also.