Sanjeet Nayak
21410140076

1) The condition for last interval to finish is $S_j \geq F_I$

For last intervals to start the condition will be $S_I > S_j$

So, the algorithm for last interval to start

Interval_scheduling (O, s, F, n)

Sort wrt F in non decreasing order.

```
Sol = ϕ
F = 1 , Sol = { }
For I = 2 to n
{
 if ( SI > Fj )
  {
   Sol = SolVA [I]
   I = J
  }
 Return Sol
}
```

2) Interval Scheduling (s, f, v)

```
Sol = ϕ
for (i=0 ; i<n ; i++)
  {
for (i-1, j=1 : j<=n ; j++)
  {
   Vi = Fi - Si ;
   Vf = Fj - Sj ;
   if (Vj > Vi)
   {
   Sol = Sol ∨ vj
   }
  return Sol ;
```

```
    }
  }
}
```

Time complexity : $O(n^2)$

3) a, c, f, d, e, c, g, b, h, f, a, d, c, b, e

Cache memory :- | a | b | c | d |

i) using Farthest-In-Future :-

a : 1, 11
c : 2, 6, 13
f : 3, 10
d : 4, 12
e : 5, 15
b : 8, 14
g : 7
h : 9

a	b	c	d
f	b	c	~~d~~ e
f	~~b~~	c	~~e~~ g
f	b	c	g h
f	b	c	~~h~~ a
~~f~~	b	c	a
d	b	c	~~a~~ e
d	b	c	e

∴ Total cache miss = 7

Final cache : | d | b | c | e |

ii) using Last in first out :-

Initial : | a | b | c | d |

~~d~~ ~~e~~ g ~~h~~ ~~a~~ ~~e~~

∴ Total cache miss = 8

Final cache memory : | a | b | c | e |

4. a, d, c, f, d, b, g, a, e, e, b, f, a, d, g

Initial cache memory : | a | b | c |

using optimal caching - (Recently used)

a : 1, 8, 13

b : 6, 11

c : 3, 10

d : 2, 5, 14

e : 9

f : 4, 12

g : 7, 15

a	b	c
~~a~~		
	~~b~~	
		~~c~~
~~d~~		
	~~e~~	
		~~d~~
~~g~~		
	~~f~~	
		~~g~~
~~a~~		
	~~b~~	
		~~f~~
~~d~~		
	~~e~~	
		~~a~~
~~g~~		
	~~e~~	
		~~d~~
~~e~~		

∴ Total hit = 6

Total miss = 9

Time = 0.1 × 6 + 1 × 9

= 9.6 microsec

ii) Last in first out -

Initial cache memory - | a | b | c |

Total miss = 10

Total hit : 5

Time : 5 × 0.1 + 10 × 1

= 10.5 microsec.

∴ % change = $\frac{10.5 - 9.6}{9.6} \times 100$ = 9.4 %.

Hence, optimal is better as it saves 9.4 % of the time.

5. Sort Q = {a, b, c, d, e}

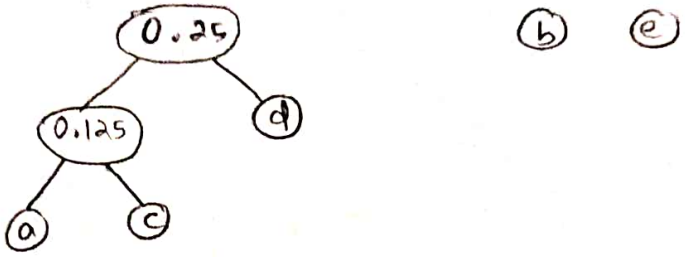
P(a) = 1/16 = 0.0625 P(b) = 1/2 , P(c) = 1/16 = 0.0625, P(d) = 1/8 = 0.125, P(e) = 1/4 = 0.25

a	b	c	d	e
0.625	0.5	0.0625	0.125	0.25

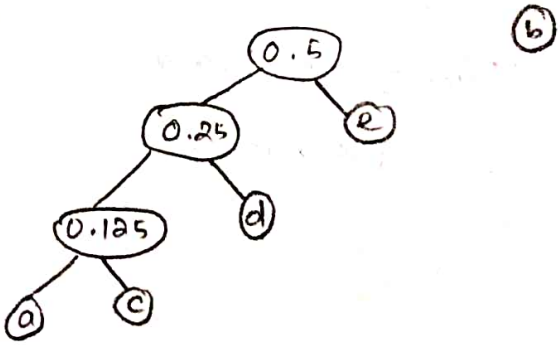
Huffman code for these letters.

Step 1:

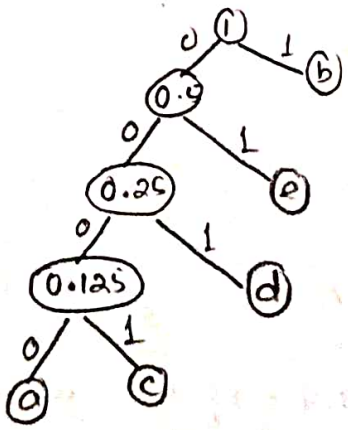
Step 2:



Step 3:



Step 4:



Huffman codes are -

a : 0000

b : 1

c : 0001

d : 001

e : 01

6. b, a, c, d, d, c, b, a, c, d, g, b, e, a, c, b.

Given cache size K = 3

It already contains : {a, c}

a : 2, 4, 8, 14

b : 1, 16

c : 3, 6, 9, 15

d : 5, 10

e : 13

f : 7, 12

g : 11

∴ The cache

a	c	

a c b

a c ~~d~~

a ~~c~~ f

a ~~d~~ f

a g ~~f~~

a g e

c g e

c b e

∴ Total no. of miss = 8

7.

Interval (i)	1	2	3	4	5	6	7	8	9	10
Starting time (Si)	0	3	3	0	2	0	4	7	6	8
Finish time (Fi)	2	4	7	1	4	4	7	9	10	10

As we got the single resource. So, the above can be solve by interval scheduling greedy approach.

Let's consider a position i which starts from position j = 2.

If the interval got schedule set assign the value of i as j,

Let's take the condition, if $S_I > F_j$ then Schedule the resources.

The greedy algorithm is

Sol = Aj

For I = 2 to n

{

if ($S_I > F_j$)

{
Sol = Sol ∪ A[i]
J = I
}
}

The Solution is - A = { A1, A2, A5, A7, A8 }

8. The optimal algo. for minimizing lateness is

Sort the Jobs wrt their deadline

Set t = 0

For J = 1 to n.

{
assign J_I [t_I, $t + t_I$)

$S_J = t_I$ $F_J = t + t_I$

$t = t + t_I$
}

Return (The set of sorted jobs)

i) Earliest deadline First

Interval	1	2	3	4	5	6	7	8	9	10
Running time	2	5	3	1	7	5	6	4	3	2
Deadline	9	14	8	15	7	11	12	16	21	22

0 J_5 $T_6 = 7$ $d_5 = 7$ J_3 $J_3 = 3$ $d_3 = 8$ J_1 $J_1 = 2$ $d_1 = 9$ J_6 $T_6 = 5$ $d_6 = 11$ J_7 $T_7 = 6$ $d_7 = 14$ J_4 J_8 $T_8 = 4$ $d_8 = 16$ J_4 $T_4 = 1$ $d_4 = 15$ J_9 $T_9 = 3$ $d_9 = 21$ J_{10} $T_{10} = 2$ $d_{10} = 22$

ii) Shortest running time -

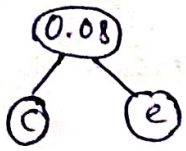
J_4	J_1	J_{10}	J_3	J_9	J_8	J_2	J_6	J_7	J_5
$T_4 = 1$	$T_1 = 2$	$T_{10} = 2$	$T_3 = 3$	$T_9 = 3$	$T_8 = 4$	$T_2 = 5$	$t = 5$	$t = 6$	$t = 7$
$d_4 = 15$	$d_2 = 2$	$d_{10} = 22$	$d_3 = 8$	$d_9 = 21$	$d_8 = 16$	$d_2 = 14$	$d = 11$	$d = 12$	$d = 7$

10) Let the Symbols be {a, b, c, d, e, f}

Symbol	a	b	c	d	e	f
Freq.	0.19	0.23	0.03	0.45	0.05	0.05

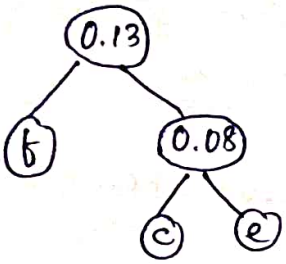
By huffman code -

Step 1 :



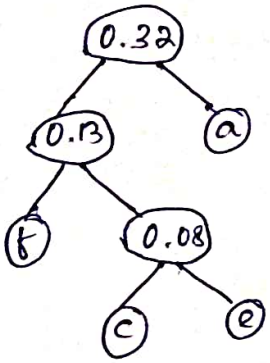
(a) (b) (d) (f)

Step 2 :

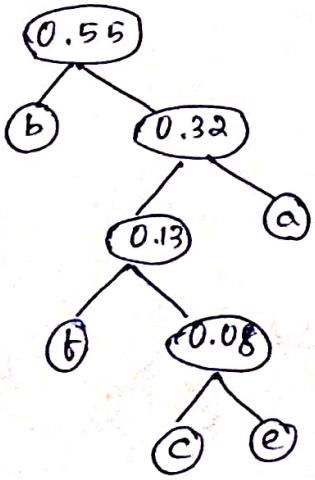


(a) (b) (d)

Step 3 :



Step 4 :



Step 5

Huffman code :-

a : 111

b : 10

c : 11011

d : 0

e : 11010

f : 1100

For 5 bit encoding, no. of bits required per number is 5.

∴ Total : (0.19 + 0.23 + 0.03 + 0.45 + 0.05 + 0.05)5

: 5

Bits required for huffman code :-

= 0.19 × 3 + 0.23 × 2 + 0.03 × 4 + 0.45 × 1 + 0.05 × 5 + 0.05 × 4

: 2.05

Here since huffman takes less no. of bits required.

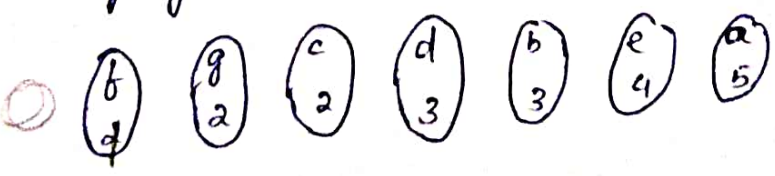
Hence huffman is the optimal solution.

II) Given String : "abbaccbaaddddufggeeea"

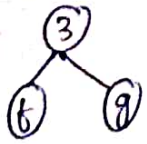
∴ Symbol a b c d e f g

Freq. 5 3 2 3 4 1 2

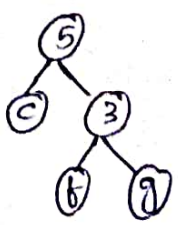
Arranging in ascending order -



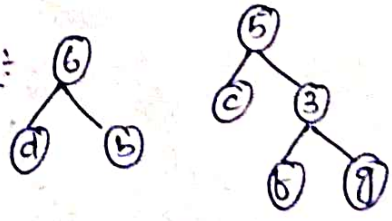
Step 1 :



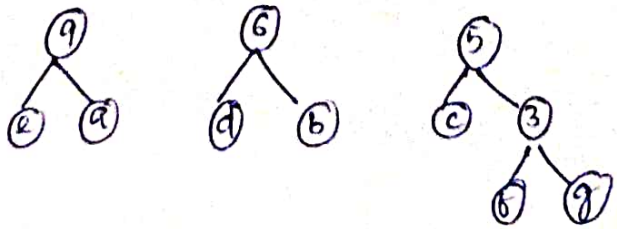
Step 2 :



Step 3 :



Step 4 :



e : 2

b : 0

g : 0

12) Yes, the given statement is true. As Spanning tree selects the 1st edge present in the min. priority tree and the cheapest edge is the 1st member at the min-priority queue. so, it is a part of the MST.

13) Yes, the given statement is true. Squaring the cost at edge will not change the order at priority as MST is constructed by using min-priority Queue. so, T must still be a minimum Spanning tree for this new instance.