Question 1 I will combine the 3 conclusions at last. Let's begin computing:

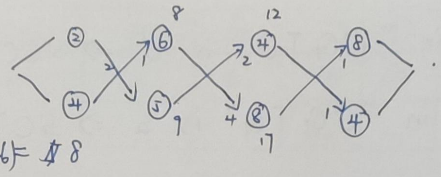
~~1)~~ Firstly, we can set $f_1[j]$, $f_2[j]$ as the minimum time to station $S_{1,j}$, $S_{2,j}$

For assembly line 1: $f_1[j] = \min(f_1[j-1] + a_{1,j},\ f_2[j-1] + t_{2,j-1} + a_{1,j})$

For assembly line 2: $f_2[j] = \min(f_2[j-1] + a_{2,j},\ f_1[j-1] + t_{1,j-1} + a_{2,j})$.

where a is the assembly time and t is the transition time.

$f_1[1] = 2$, $f_2[1] = 4$



$f_1[2] = \min(2+6,\ 4+1+6) = 8$

$f_2[2] = \min(4+5,\ 2+2+5) = 9$

$f_1[3] = \min(8+4,\ 9+2+4) = 12$

$f_2[3] = \min(9+8,\ 8+4+8) = 17$

$f_1[4] = \min(12+8,\ 17+1+8) = 20$

$f_2[4] = \min(12+1+4,\ 17+4) = 17$

we get f^* 17 and we can find it $f_2[4] \to f_1[3] \to f_1[2] \to f_1[1]$

in calculation, which means in sequence order: $S_{1,1}$, $S_{1,2}$, $S_{1,3}$, $S_{2,4}$

In all:

1).

Station	$S_{1,1}$	$S_{1,2}$	$S_{1,3}$	$S_{1,4}$
min_time	2	8	12	20

Station	$S_{2,1}$	$S_{2,2}$	$S_{2,3}$	$S_{2,4}$
min_time	4	9	17	17

2) $f^* = 17$

3) $S_{1,1}$, $S_{1,2}$, $S_{1,3}$, $S_{2,4}$

Question 2.

1) C G T G C $\Rightarrow len(P) = 5$

$\quad$ 0 1 2 3 4

A : Doesn't exist in this string $\Rightarrow S(A) = 5$

G : For the string "CGTG" The last G from left to right is

$\quad$ 0 1 2 3

in 3 $\Rightarrow$ distance $= 4-3 = 1 \Rightarrow S(G) = 1$

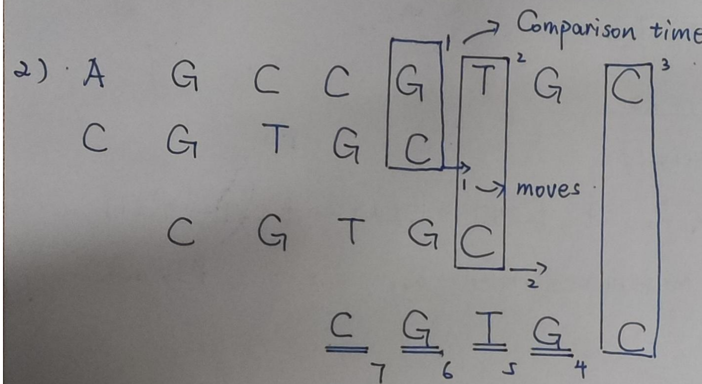
C : The rightest in "CGTG" is 0 $\Rightarrow S(C) = 4-0 = 4$

T : The rightest in "CGTG" is 2 $\Rightarrow S(T) = 4-2 = 2$.

In all : Shift Table

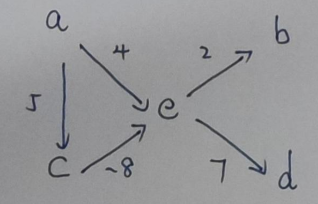
A	G	C	T
5	1	4	2

2)



$\Rightarrow$ The string comparisons are 7

Question 3



Initialization

vertices	A	B	C	D	E
cost	0	∞	∞	∞	∞
pre	—	—	—	—	—

Interation 1 : we consider the edge from sequential order in alphabet

For a :

V	A	B	C	D	E
C	0	∞	5	∞	4
P	—	—	A	—	A

For b x

For C :

V	A	B	C	D	E
C	0	∞	5	∞	-3
P	—	—	A	—	C

For d x

For e :

V	A	B	C	D	E
C	0	-1	5	4	-3
P	—	E	A	E	C

Interation : 2~5 : No updating

In all : the shortest paths from a → b is a → c → e → b -1

a → c is a → c 5

a → d is a → c → e → d 4

a → e is a → c → e -3

$\boxed{3}$ a → a is 0

Question 4.

1.

		0	1	2	3	4
			A	A	T	G
0		0	← -5	← -10	← -15	← -20
1	A	-5 ↑	↖ 2	↖ ← -3	← -8	← -13
2	G	-10 ↑	-3 ↑	↖ -3	← -8	↖ -6
3	C	-15 ↑	-8 ↑	-8 ↑	↖ -8	-11 ↑

① The path (3,4) (2,4) (1,3) (1,2) (1,1) (0,0) corresponds to:

A	A	T	G	–
A	–	–	G	C

② The path (3,4) (2,4) (1,3) (1,2) (0,1) (0,0) corresponds to:

A	A	T	G	–
–	A	–	G	C

In all, both of them are optimal global aligment

2.

		0	1	2	3	4
			A	A	T	G
0		0	0	0	0	0
1	A	0	↖2	↖2	0	0
2	G	0	0	0	0	↖2
3	C	0	0	0	0	0

Starting at highest score 2, we have three optimal aligment to 0

① (2, 4)

G
G

② (1, 2)

A
A

③ (1, 1)

A
A

Question 5

1. We define the class P consits of those decision problems that can be solved by a deterministic Turing machine in polynomial time. In math, a problem is in P if there exists an algorithm for this problem, that it runs $O(n^k)$, where k is a constant, n is input size.

According to the saying "An algorithm is efficient if its running time is bounded by a polynomial of its input size" as class P is polynomial; that they are considered efficient

2. The class NP (Nonedeterministic Polynomial) doesnot imply the solution itself can be found in polynomial time, but once the solution is given, it can be verified in polynomial time.

Example: Boolean Satisfiability Problem (you are given a Boolean formula 𝔉 and asked whether there is assignment of true and false values to make formula goes true.) Reason: you can check whether it comes true by inserting the values into the formula.

Reason of challenging: There is no solutions to find method for all NP problems, which has exponential time complexity. Also, the question of whether P equals NP is still unsolved, ~~that~~ when we meet currently.

3. NP complete problems are satisfied if they are in NP and every problem in NP can be reduced to it in polynomial time.

The significance of NPC Problems relys on that. If an efficient algorithm was discovered for even one NP-complete problem, it would mean $P = NP$, as it is also ~~so~~ seen as the boudary between problems that are feasibly solvable and hard, which also guide the development in logistics, scheduling or other subjects.

4. Polynomial-time Reduction refers to transforming one problem into another with solution corresponding and be achievable in polynomial time.

Way: We use an approximation algorithm or Heuristics that can find an approximate solution in polynomial time and solve it to maximize the original problem.