Q1.

a) Let $f(i,j)$ be the minimum time to reach station j on assembly line i, where $1 \le i \le k$ and $1 \le j \le n$

b)

$$f(i,j) = \min(f(i, j-1) + a_{i,j},$$
$$\min(f(k, j-1) + t_{k,i} + a_{i,j}) \text{ for all } k \ne i)$$
$$\text{for } j > 1$$

with base cases:

$$f(i,1) = e_i + a_{i,1} \quad \text{for all } 1 \le i \le k.$$

The final answer will be:

$$\min(f(i,n) + x_i) \text{ for all } 1 \le i \le k$$

c) Algorithm:

Input: k assembly lines, n station per line, entry times $e[1 \ldots k]$, exit times $x[1 \ldots k]$, station times $a[1 \ldots k, 1 \ldots n]$, transfer times $t[1 \ldots k, 1 \ldots k]$

Initialize: Let $f[1..k, 0 \ldots n]$ be a new array, set $f[i,1] = e[i] + a[i,1]$ for all i from 1 to k.

```
for j = 2 to n:
    for i = 1 to k:
        f[i,j] = min(f[i,j-1] + a[i,j],
                     min(f[k,j-1] + t[k,i] + a[i,j] for all k≠i))
minTime = min(f[i,n] + x[i] for all i from 1 to k)
Return: minTime
```

d) The time complexity is $O(k^2 n)$, since there are $O(kn)$ subproblems and each takes $O(k)$ time to solve using the recurrence relation.

Q2

a) Let $f(i,j)$ be the minimum cost of a trip by canoe from trading post i to trading post j. using up to each intermediate trading post between i and j.

b) The recurrence relation for the subproblems can be expressed as:

$$f(i,j) = \min(C[i,j], \min(f(i,k) + f(k,j)) \text{ for } i<k\leq j)$$

c) Algorithm:

Input: The number of trading posts n, and the cost matrix C.

Initialize:

```
for i=1 to n do
    for j=1 to i-1 do
        f[i,j] = C[i,j]
for d=2 to n-1 do
    for i=n to d+1 do
        j = i-d
        for k = i-1 to j+1 do
            f[i,j] = min( f[i,j], f[i,k] + f[k,j] )
```

Return: $f[n,1]$ as the minimum cost from trading post n to 1.

d) The time complexity is $O(n^3)$, as there are $O(n^2)$ subproblems. And for each subproblem, we iterate over $O(n)$ possible intermediate trading posts.

However, if we define one dimension state $f(i)$, representing the cost from i to 1. We only need to enumerate the previous transfer k and time complexity is $O(n)$, There are total n states that need to be calculated, so the total complexity could be $O(n^2)$

Q3.

a) Let $LCS(i,j)$ be the length of the longest common subsequence between the prefixes $a[1 \ldots i]$ and $b[1 \ldots j]$

b) $LCS(i,0) = 0$ for all i
$LCS(0,j) = 0$ for all j.

c) For $i > 0$ and $j > 0$:

$$LCS(i,j) = \begin{cases} LCS(i-1, j-1) + 1 & \text{if } a[i] = b[j] \\ \max(LCS(i-1,j), \\ \quad LCS(i, j-1)) & \text{if } a[i] \neq a[j] \end{cases}$$

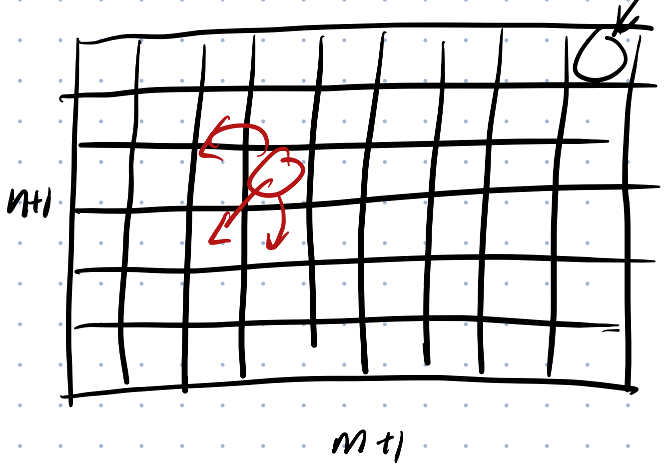
d) Algorithm:
Input: Strings a and b of lengths n and m respectively.
Initialize: Create a 2D array LCS of size $(n+1) \times (m+1)$

```
for i=0 to n do
    LCS[i,0] = 0
for j=0 to m do
    LCS[0,j] = 0
for i=1 to n do
    for j=1 to m do
        if a[i]=b[j] then
            LCS[i,j] = LCS[i-1, j-1] + 1
        else
            LCS[i,j] = max(LCS[i-1, j], LCS[i, j-1])
lcs_length = LCS[n, m]
if lcs_length = 0 then
    return "No common subsequence found".
```

```
else
    lcs = ''
    i = n
    j = m
    while i>0 and j>0 do
        if a[i] = b[j] then
            lcs = a[i] + lcs
            i = i-1
            j = j-1
        else if LCS[i-1, j] > LCS[i, j-1] then
            i = i-1.
        else
            j = j-1
    return lcs
```

e) Time complexity is $O(mn)$

value of optimal solution



We fill this table of size $(n+1)\times(m+1)$ and each entry takes $O(1)$ time to compute.