(4.) For 2 assembly lines :

fastest_way (t, a, n, e, x)

$f_1[1] \leftarrow e_1 + a_{1,1}$

$f_2[1] \leftarrow e_2 + a_{2,1}$

for $j \leftarrow 2$ to n

if ($f_1[j-1] \leq f_2[j-1] + t_{2,j-1}$)

$f_1[j] = f_1[j-1] + a_{1,j}$

$L_1[j] \leftarrow 1$

else

$f_1[j] = f_2[j-1] + t_{2,j-1} + a_{1,j}$, $L_1[j] \leftarrow 2$

if ($f_2[j-1] \leq f_1[j-1] + t_{1,j-1}$)

$f_2[j] = f_2[j-1] + a_{2,j}$

$L_2[j] \leftarrow 1$

else

$f_2[j] = f_2[j-1] + t_{1,j-1} + a_{2,j}$

$L_2[j] \leftarrow 2$

if $f_1[n] + x_1 \leq f_2[n] + x_2$

$f^* \leftarrow f_1[n] + x_1$

$L^* \leftarrow 1$

else

$f^* \leftarrow f_2[n] + x_2$

$L^* \leftarrow 2$

Teacher's Signature

```
print (L, n)

    i ← L*
    print 'line' i 'station' n

    for j ← n to 2
        i ← Li[j]
        print 'line' i 'station' j-1

end
```

Li[j] : info. that which assembly line previous shop belongs in an optimal path

Time Complexity : 8 O(n) + 7 O(1)

Space Complexity : 4 O(n) + 5 O(1)

⇒ Improved version :

```
Opath (f, t, n, x)
  for i ← 2 to n
      if f1[i] ≤ f2[i]
          if f1[i-1] ≤ f2[i-1] + t2,j-1
              print 'line 1' 'station' (i-1)
          else
              print 'line 2' 'station' (i-1)
```

Teacher's Signature

else

if $f_1[i-1] + t_{1,i-1} \leq f_2[i-1]$

print 'line 1' 'station' $(i-1)$

else

print 'line 2' 'station' $(i-1)$

Time Complexity : $7\,O(n) + 4\,O(1)$

Space Complexity : $2\,O(n) + 2\,O(1)$

② For m assembly lines with n shops each :

fastest_way (t, a, n, e, x, m)

$O(m)$ { for k ← 1 to m
$f_k[1] \leftarrow e_k + a_{k,1}$

$O(mn) + O(mn)$ { for j ← 2 to n
for i ← 1 to m
$f_i[j] \leftarrow$ Min_OC (f, t, i, j, m, L) + a_{ij}

$O(1)$ min_exit = $f_1[n] + x_1$

$3\,O(m)$ { for i ← 2 to m
~~min_exit of 1~~
if (min_exit > $f_i[n] + x_i$)
min_exit = $f_i[n] + x_i$
$L^* = i$

Teacher's Signature

```
Min - OC (f, t, i, j, m, L)
    for k ← 1 to m
        if k != i
            M[k] ← f_k[j-1] + t_{k,j-1}
        M[k] ← f[j-1]                          } 3 O(m)

    ~~OOO ← main~~(M) min ← M[1]
    for l ← 2 to m                             } 3 O(m)
        if (M[l] < min)
            min = M[l]
            L_i[j] = l
    return min

print (L, n)
    i ← L*
    print 'line' i 'station' n                 } O(n)+2O(1)
    for j ← n to 2
        i ← L_i[j]
        print 'line' i 'station' j-1

end
```

Time complexity $\simeq O(m^2 n)$