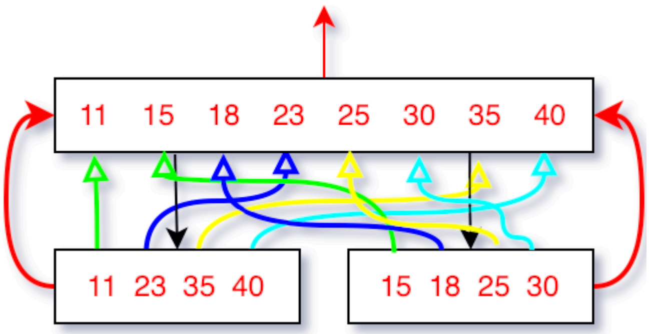
COMP108 Data Structures and Algorithms
Week 11 Tutorial Exercises

1. (a) Suppose we are given the following numbers: 40, 23, 11, 35, 15, 30, 18, 25. If we apply merge sort to sort these numbers into ascending order, the final merge step would be to merge the two sorted sequences:

$$(11, 23, 35, 40) \text{ and } (15, 18, 25, 30)$$

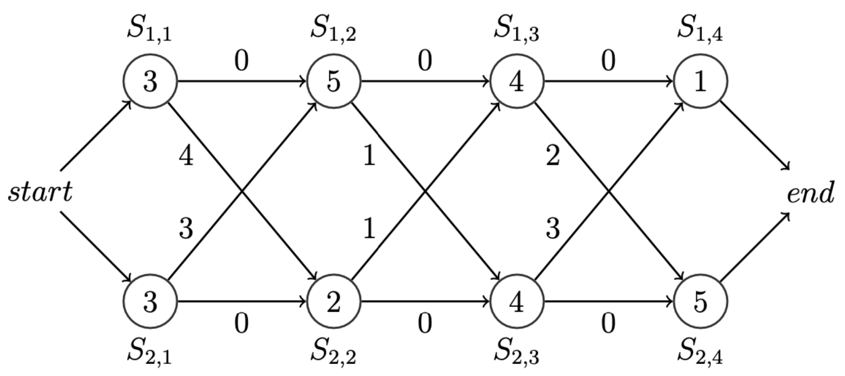
Draw figures to show this final merge step. Use two pointers to point to the two sorted sequences. Show step by step how the pointers move to obtain the final merged sequence.



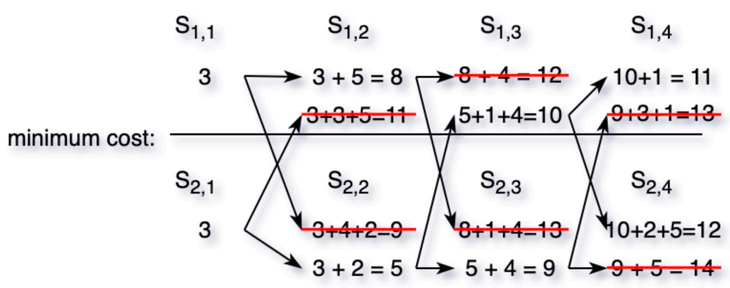
(b) Merge sort makes use of an algorithm to merge two sorted sequences. Assume that the given two sequence are already in **descending order**. Modify the pseudo code Merge() in the lecture notes to merge two descending ordered sequences into one descending ordered sequence.

```
Algorithm Merge(B[1..p], C[1..q], A[1..(p+q)])
        i <- 1, j <- 1, k <- 1
        while i <= p AND j <= q do
        begin
                if B[i] >= C[j] then
                        A[k] <- B[i],   i <- i + 1
                else
                        A[k] <- C[j],   j <- j + 1
                k <- k + 1
        end
        if i == p + 1 then
                copy C[j..q] to A[k..(p+q)]
        else
                copy B[i..q] to A[k..(p+q)]
```

2. Suppose there are two assembly lines each with 4 stations, $S_{i,j}$. The assembly time is given in the circle representing the station and the transfer time is given next to the arrow from one station to another.



(a) Using dynamic programming, fill in the table of the minimum time $f_i[j]$ needed to get through station $S_{i,j}$. You should also show **all** the intermediate steps in computing these values, e.g., in computing $f_1[2]$, you need to specify that $f_1[2] = \min\{\_\_, \_\_\}$.



j	$f_1[j]$	$f_2[j]$
1	3	3
2	8	5
3	10	9
4	11	12

(b) What is the minimum time f^* needed to get through the assembly line?

$$f^* = \min\left\{f_1[n], f_2[n]\right\}\Big|_4 = 11$$

(c) Which stations should be chosen to achieve the minimum time?

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

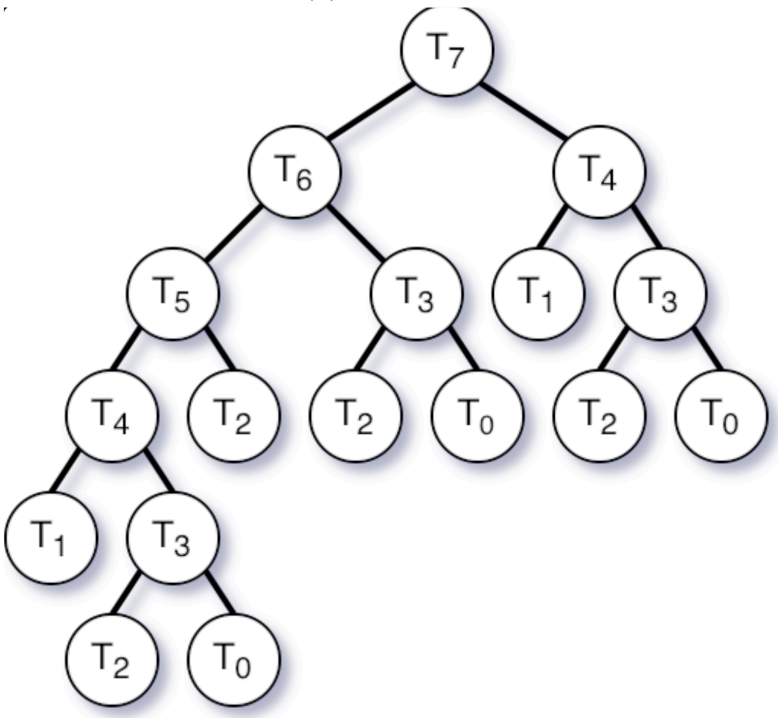
3. Consider the following recurrence.

$$T(n) = \begin{cases} 1 & \text{if } n == 0 \text{ or } n == 1 \\ 2 & \text{if } n == 2 \\ T(n-1) + T(n-3) & \text{if } n > 2 \end{cases}$$

(a) Design and write a pseudo code for a recursive procedure to compute $T(n)$.

```
Algorithm T(n)
        if n > 2 then
                return T(n - 1) + T(n - 1)
        else
                if n == 0 OR n == 1 then
                        return 1
                if n == 2 then
                        return 2
```

(b) Draw the execution tree for $T(7)$.



(c) Using the concept of dynamic programming, rewrite your recursive procedure into a non-recursive one.

```
Algorithm T(n)
        set v[0] <- 1, v[1] <- 1
        set v[2] <- 2
        for i <- 3 to n do
                v[i] <- v[i – 1] + v[i – 3]
        return v[n]
```