Q1.

Ayush Agarwal MT22095
Shubham Agarwal MT22124

```
1.  global size;
2.  merge(A,B,C  n):
3.      size = n
4.      merging (A,B,C,1,1,1)
5.
6.  merging(A,B,C,i,j,K)        - T(n,n)  // TC of function, used in Q2
7.      if(K <= 2*size)
9.          if(j>n)
9.              C[K] = A[i]
10.             merging(A,B,C,i+1,j,K+1)    - T(n-1,n)
11.         else if (i>n)
12.             C[K] = B[j]
13.             merging(A,B,C,i,j+1,K+1)    - T(n,n-1)
14.         else if(A[i] <= B[j])
15.             C[K] = A[i]
16.             merging(A,B,C,i+1,j,K+1)    - T(n-1,n)
17.         else
18              C[K] = B[j]
19.             merging(A,B,C,i,j+1,K+1)    - T(n,n-1)
20.
```

2. merges two sorted arrays $A(1 \cdots n)$ & $B(1 \cdots n)$ into $C(1 \cdots 2n)$
6. i, j, K iterate over $A(1 \cdots n)$, $B(1 \cdots n)$ & $C(1 \cdots 2n)$ respectively
8. B is fully traversed hence rem. elements of A being copied
11. similar to 8. for A
14. A's element smaller than B hence gets filled first, otherwise its B's element in line 18.

Q2: $T(i,j)$ - TC to merge sorted arrays $A(1\ldots j)$ and $B(1\ldots j)$ for given arrays $A(1\ldots n)$, $B(1\ldots n)$, we find $T(n,n)$

At each step of the merging function until completion, we execute either of the four conditions doing a constant amount of work and move exactly 1 element, either from A[] or B[] into C[].

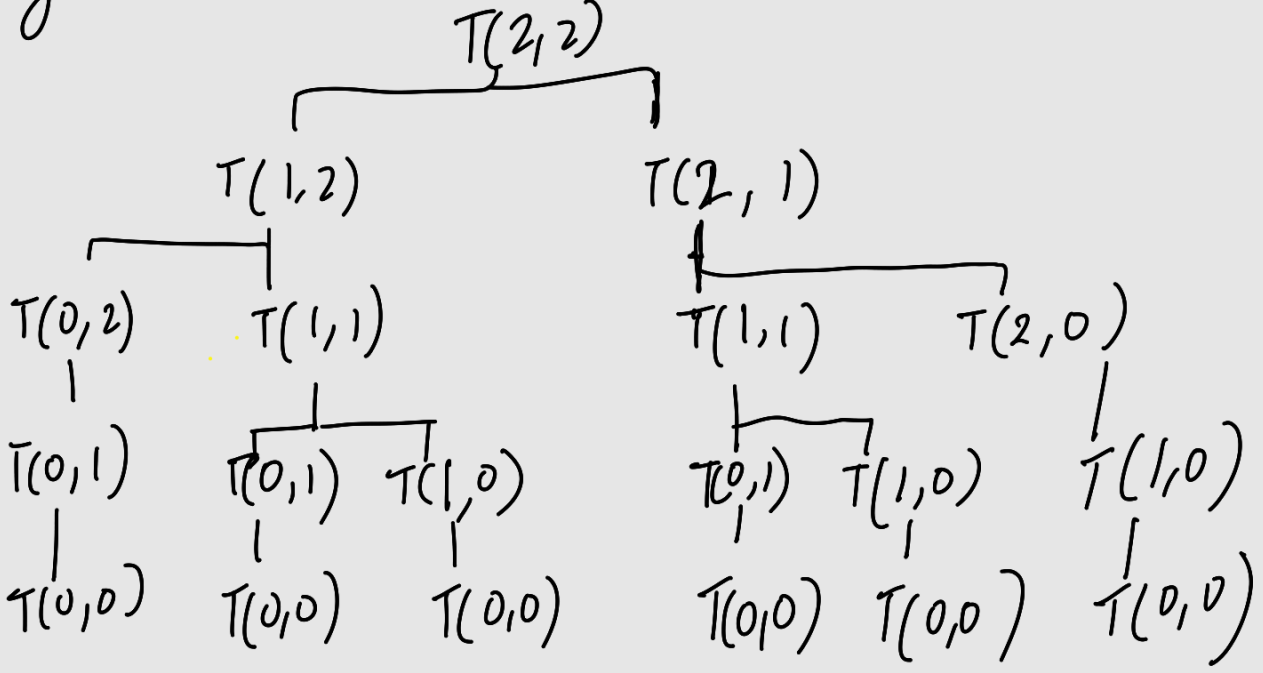
$$T(n,n) = \max \begin{cases} T(n-1,n)+C \\ T(n,n-1)+C \\ T(n-1,n)+C \\ T(n,n-1)+C \end{cases}$$

Removing redundant terms

$$= \max\left(T(n-1,n), T(n,n-1)\right) + C$$

Base case would be $T(0,0)=0$, when both arrays get fully merged.

Understanding breakdown of subproblems & their corresponding TC using recursion tree



At every level, either of the two arrays get iterated by exactly 1 element doing constant work. Consequently the TC recursion tree decrements either of the two arguments by exactly 1. Whatever way the tree follows, a general

$T(i,j)$ coming down from $T(n,n)$ will always do work = (no. of steps reduced in both arg. combined) x constant c.

$\therefore T(n,n) = T(n-p, n-q) + (p+q)\cdot c$

where p, q are no. of iterations over $A[]$ and $B[]$ respectively initially $(p+q)$ would be 1 when just 1 element correctly merged!

$(p+q) = 2$;

From here, only two mutually exclusive and exhaustive cases occur:

Case 1: When B gets completely traversed and merged in C[]
i.e $q = n$ when p is some value $0 \le p < n$

$T(n,n) = T(n-p, n-n) + (p+n)\cdot c$
$= T(n-p, 0) + (p+n)\cdot c$
$= T(n-p-1, 0) + (p+n+1)\cdot c$
⋮ from here, at each step 1st argument decreases by exactly 1 doing constant work c.
$= T(0,0) + (n-p-1)\cdot c + (p+n+1)\cdot c$
$= 0 + (n-p-1)\cdot c + (p+n+1)\cdot c$
$= 2nc$.

Case 2: when A gets completely traversed and merged in C[]
i.e. $p = n$ when q is some value $0 \le q \le n$

$T(n,n) = T(n-n, n-q) + (n+q)\cdot c$
$= T(0, n-q) + (n+q)\cdot c$
$= T(0, n-q-1) + (n+q+1)\cdot c$
⋮
$= T(0,0) + (n-q-1)\cdot c + (n+q+1)\cdot c$
$= 0 + 2nc = \underline{2nc}$.