d) Write the step count table for the function

Statement	s/e	Frequency	Total Steps
void transpose(int a[][MAX_SIZE])	0	0	0
{	0	0	0
int i,j,temp;	0	0	0
for(i=0; i<MAX_SIZE-1; i++)	1	MAX_SIZE	MAX_SIZE
for(j=i+1; j<MAX_SIZE; j++)	1	$\sum_{i=0}^{MAX\_SIZE-2} MAX\_SIZE-(i+1)$ $+(MAX\_SIZE-1)$	$\frac{MAX\_SIZE(MAX\_SIZE-1)}{2}+(MAX\_SIZE-1)$
SWAP(a[i][j],a[j][i],temp)	3	$\sum_{i=0}^{MAX\_SIZE-2} MAX\_SIZE-(i+1)$	$\frac{3MAX\_SIZE(MAX\_SIZE-1)}{2}$
}	0	0	0

(d) Total: $2MAX\_SIZE(MAX\_SIZE-1)+(2MAX\_SIZE-1)$

$= 2MAX\_SIZE^2-1$

Asymptotic Notation (O, Ω, Θ)

- Our motivation to determine step counts is to be able to compare the time complexities of two programs that compute the same function and also to predict the growth in run time as the instance characteristic change.
- Determining the exact step count (either worst case or average) of a program can prove to be an exceedingly difficult task.
- The notion of a step is inexact. Both the instructions x=y and x=y+z+(x/y)+(x*y*z-z/z) count as one step. So the exact step count is nt very useful for comparative purposes.
- For most situations, it is adequate to be able to make a statement like $c_1n^2 \leq T_p(n) \leq c_2n^2$ or $T_Q(n,m) = c_1n+c_2m$, where c_1 and c_2 are non-negative consts.
- This is so because if we have two programs with a complexity of $c_1n^2+c_2n$