1)
For i=0, compute() takes O(j) time for all (0 < j < m), which sums up to (m(m+1))/2
For i=1, compute() takes O(j-1) time for all (1 < j < m-1), which sums up to ((m(m+1))/2)-m
For i=2, compute() takes O(j-2) time for all (1 < j < m-2), which sums up to ((m(m+1))/2)-m-(m-1)
.
.
For i=m, compute() takes O(1).
This makes compute runs according to the triangular number series from m to 0. Summing up the triangular series gives (m(m+1)(m+2))/6
For i>m, compute() takes O(1) each, summing up to O(m)

So in total the code takes (m(m+1)(m+2))/6 + m which is $O(m^3)$
It is upper bounded by $O(2m^3)$
And lower bounded by $O(m^2)$

2) For i=1, it computes 1*2*3*4*5….*m = m!
For i=2, it computes 2*3*4*5*6….*m = m! / 1!
For i=3, it computes 3*4*5*6*7….*m = m! / 2!
For i=m, it computes m = m! / (m-1)!
In terms of i and j, B[i][j] contains the value i*(i+1)*(i+2)*...*j generally and contains 1 when j<=i.

3) For n=1000, it takes 0.07255 seconds.

4) Code file submitted

5) For n=1000, it takes 0.01196 seconds.