**Tut 1 Q4**

Type out your Math analysis of the individual code segments.

Instead of writing if you have a tablet or copy-pasting equations made from LaTex (for those familiar), you may want to try typing using the Microsoft Equation editor (press Alt + = for Windows users, or go to Insert -> Equation). Do try it!

|  |  |
| --- | --- |
| **Code Segment** | **Your Math Analysis** |
| public static void Q1a(int n) {  int a = 0;  for(int i=0; i<n; i++)  a += Q1b(n);  System.out.println(a);  }  public static int Q1b(int n) {  int b = 0;  while(n>0) {  b += n;  n = n / 2;  }  return b;  } | The loop in A repeats from 0 to n – 1 (n times) and calls Q1b, which has a loop inside that loops log\_2 n times since n is being reduced by a factor of 2 every time, making this O(n log n) time complexity. |
| for(int i = 0; i < n; i++){  for(int k=0; k<n; k++){  /\*some sequence of O(1) steps\*/  } | Since the outside loop repeats from 0 to n – 1 and the nested loop repeats from 0 to k – 1, each loop repeats n and k times respectively, making this O(n^2) time complexity. |
| for(int x = 1; x < n; x \*= 2){  /\* some sequence of O(1) steps \*/  } | Since x is being increased by a factor of 2 until it reaches n, it takes log\_2 n times to get there and makes this O(log n) time complexity. |
| // loop 1  for(int i = 0; i < n; i++)  // loop 2  for(int j = i+1; j > i; j--)  // loop 3  for(int k = n; k > j; k--) System.out.println(“\*”); | Loop 1 runs from 0 to n (n times) and loop 3 runs from n to 0 (n times). Loop 2 runs 1 time, therefore this is O(n^2) time complexity. |