Hands-On 3

1. For the given function we have (hypothesis: execution speed of each line is equal to 1 times n, which is the number it gets executed. Essentially c = 1)  
   x=1; # 1  
   for i=1:n # n+1  
   for j=1:n # n(n+1)  
   x=x+1; # n^2  
    the runtime would be calculated with the following summation formula:

|  |  |
| --- | --- |
| n | T(n) |
| 1 | 6 |
| 2 | 14 |
| 3 | 26 |
| 4 | 42 |
| 5 | 62 |
| 6 | 86 |
| 7 | 114 |
| 8 | 146 |
| 9 | 182 |
| 10 | 222 |
| 100 | 20202 |
| 1000 | 2002002 |
| 10000 | 200020002 |



1. For this case, I pick c1 = ½ and c2 = 5, because: (1)  
     
   Αnd we derive the following notations:
2. n\_0 in our case is chosen to be equal to 2 since for any n ≥ 2, inequality (1) is valid  
   ![A graph of a function

   Description automatically generated]()  
     
   As shown in the figure above, at x=1.4 the upper bound is larger than T(n), therefore, n\_0=2 is the first integer that T(n) respects both bounds
3. In case the function changed as stated in the question:   
   x=1; # 1  
   y=1; # 1  
   for i=1:n # n+1  
   for j=1:n # n(n+1)  
   x=x+1; # n^2  
   y=i+j; # n^2  
     
   The new T(n) would be:

1. The results change as follows:

|  |  |
| --- | --- |
| n | T(n) |
| 1 | 8 |
| 2 | 19 |
| 3 | 36 |
| 4 | 59 |
| 5 | 88 |
| 6 | 123 |
| 7 | 164 |
| 8 | 211 |
| 9 | 264 |
| 10 | 323 |
| 100 | 30203 |
| 1000 | 3002003 |
| 10000 | 300020003 |

In the low n’s there is a difference, but as n gets larger the difference to the first function is negligible.