Brendan Morgenstern

CUS 1188

Problem Set 6

1) T(n) = 2T()+1000n

Every time the recurrence is called for a value k, the recurrence will be

T(k) = 2T()+1000k

By plugging in the value n/2 into the recurrence, the whole equation becomes

T(n) = 2[2T()+]+1000n

By expanding

T()+ ()+1000n   
If the recurrence is unfolded again with n/2 it becomes

[2T()+]+ ()+1000n

Expanding again yields

T()+ ()+ ()+1000n

The pattern becomes clear and can be written as

T()+k(1000n)

To set T(n) to 1, n must be 1 or in other words, must be n;

= n, solving for n gives k to be log2n

cancels with () to just 1

1+log2n(1000n)

T(n) O(nlogn)

2) T(n) = 7T()+

T(k) = 7T()+

7[7T()+]+

Expanding gives

T()+ ()+

By unfolding again we get

[7T() + ] + ()+

Expanding again gives

 T() + + ()+

The pattern can be expressed like so

T() +

The summation can be written according to this formula

as

()()

Similar to earlier, must be n to cancel ;

= n, solving for n gives k to be log2n

+ ()()

Because of the log rule

+ ()()   
log7 is roughly 3, so log is roughly 2; substituting these and expanding roughly equals

+ () O()

3)

|  |
| --- |
| public int recursive(int n) {  int sum=0;  for (int i=1; i<= n; i++){}  sum= sum +1;  }  if (n>1) {  return recursive(n/2);  }  else {  return 1  }  } |

The recursive function is only called once per recurrence as well as n iterations therefore

T(n) = 2T()+ n

Unfolding the recurrence gives

T(n) = 2T()++n

Unfolding again gives

2T()+++n

The pattern is simple

2T()+k

Substituting k for logn gives

nlogn O(nlogn )