**Name: Goutami Padmanabhan UTA ID: 1001669338**

**Section 002 Homework 1**

Q1 a) Time Complexity

**for(i = 1; i <= N; i=i+1)**

**for(k = 1; k <= i; k = 2\*k)**

**for(t = 1; t <= N; t = 2\*t)**

**printf("C");**

Let’s assume that N=2^p and lgN=Log2N

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t** | **Iteration count for t** | **total repetitions of k and t for one i** |
| 1 | 1 | 1 | 1 | 1,2,4,8,…,2^p  (or)  2^0,2^1,2^2,2^3,…,2^p | lgN | lgN |
| 2 | 2 | 1  2 | 2 | 1,2,4,8,…,2^p  1,2,4,8,…,2^p | lgN  lgN | 2 lgN |
| 3 | 3 | 1  2 | 2 | 1,2,4,8,…,2^p  1,2,4,8,…,2^p | lgN  lgN | 2 lgN |
| 4 | 4 | 1  2  4 | 3 | 1,2,4,8,…,2^p  1,2,4,8,…,2^p  1,2,4,8,…,2^p | lgN  lgN  lgN | 3 lgN |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| i | i | 1  2  4  .  .  2^j | i | 1,2,4,8,…,2^p  1,2,4,8,…,2^p  1,2,4,8,…,2^p  .  .  1,2,4,8,…,2^p | lgN  lgN  lgN  .  .  lgN | jlgN |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| N | N | 1  2  4  .  .  2^p | N | 1,2,4,8,…,2^p  1,2,4,8,…,2^p  1,2,4,8,…,2^p  .  .  1,2,4,8,…,2^p | lgN  lgN  lgN  .  .  lgN | plgN |

**Summation : T(N)= lgN+2lgN+2lgN+3lgN+…+jlgN+…+plgN**

where j and p are the number of times k loop has executed for one i.

----------------------------------------------------------------

Q1 b) Summation, Closed form, dominant term and Θ

**for(i = 1; i <= N; i=i+1)**

**for(k = 1; k <= S; k++)**

**for(t = 1; t <= i; t++)**

**printf("D");**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Iteration** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t** | **Iteration count for t** | **total repetitions due to k for k and for t loops for one i** |
| 1 | 1 | 1  2  3  .  .  S | S | 1  1  1  .  .  1 | 1  1  1  .  .  1 | S |
| 2 | 2 | 1  2  3  .  .  S | S | 1,2  1,2  1,2  .  .  1,2 | 2  2  2  .  .  2 | 2S |
| . | . | . | . | . | . | . |
| . | . | . | . | . | . | . |
| i | i | 1  2  3  .  .  S | S | 1,2,3,…,i  1,2,3,…,i  1,2,3,…,i  .  .  1,2,3,…,i | i  i  i  .  .  i | iS |
| . | . |  | . | . | . | . |
| . | . |  | . | . | . | . |
| N | N | 1  2  3  .  .  S | S | 1,2,3,…,N  1,2,3,…,N  1,2,3,…,N  .  .  1,2,3,…,N | N  N  N  .  .  N | NS |

Summation: T(N,S) = S+2S+3S+…+iS+…+NS

= S(1+2+3+…+i+…+N)

= S[N(N+1)/2] 🡪 Closed Form

= **S[(N2+N)/2]**

Dominant Term : **S and N2**

Theta : **Θ (SN2)**

----------------------------------------------------------------

Q1 c) Summation, Closed form, dominant term and Θ

**for(i = 1; i <= N; i=i+1){**

**for(k = 1; k <= N; k++)**

**for(t = 1; t <= k; t++)**

**printf("E");**

**for(k = 1; k <= M; k++)**

**for(t = 1; t <= k; t++)**

**printf("F");**

Summation table for first 4 lines of code (first 3 nested loops):

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Assumed value of N** | **Iteration** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t** | **Iteration count for t** | **total repetitions due to k for k and for t loops for one i** |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 1  2 | 1  2 | 1  2  1  2 | 2  2 | 1  1,2  1  1,2 | 1  2  1  2 | 6 |
| 3 | 1  2  3 | 1  2  3 | 1  2  3  1  2  3  1  2  3 | 3  3  3 | 1  1,2  1,2,3  1  1,2  1,2,3  1  1,2  1,2,3 | 1  2  3  1  2  3  1  2  3 | 18 |
| N | 1  2  3  .  .  N | 1 | 1  2  .  .  N  1  2  .  .  N  1  2  .  .  N  .  .  1  2  .  .  N | N  N  N  .  .  N | 1  1,2  .  .  1,2,…,N  1  1,2  .  .  1,2,…,N  1  1,2  .  .  1,2,…,N  .  .  1  1,2  .  .  1,2,…,N | 1  2  .  .  N  1  2  .  .  N  1  2  .  .  N  .  .  1  2  .  .  N | N(1+2+..+N) |

Summation : T1(N) = N(1+2+…+N)

= N[N(N+1)/2]

= N[(N2+N)/2]

= (N3+N2)/2

Summation table for the remaining lines of code (remaining nested loops):

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assumed value of N** | **Assumed value of M** | **Iteration** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t** | **Iteration count for t** | **total repetitions due to k for k and for t loops for one i** |
|  |  |  |  |  |  |  |  |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 2 | 3 | 1  2 | 1  2 | 1  2  3  1  2  3 | 3  3 | 1  1,2  1,2,3  1  1,2  1,2,3 | 1  2  3  1  2  3 | 2(1+2+3) |
| 3 | 5 | 1 | 1  2  3 | 1  2  3  4  5  1  2  3  4  5  1  2  3  4  5 | 5  5  5 | 1  1,2  1,2,3  1,2,3,4  1,2,3,4,5  1  1,2  1,2,3  1,2,3,4  1,2,3,4,5  1  1,2  1,2,3  1,2,3,4  1,2,3,4,5 | 1  2  3  4  5  1  2  3  4  5  1  2  3  4  5 | 3(1+2+3+4+5) |
| N | M | 1  2  3  .  .  N | 1  2  3  .  .  N | 1  2  .  .  M  1  2  .  .  M  1  2  .  .  M  .  .  1  2  .  .  M | M  M  M  .  .  M | 1  1,2  .  .  1,2,…,M  1  1,2  .  .  1,2,…,M  1  1,2  .  .  1,2,…,M  .  .  1  1,2  .  .  1,2,…,M | 1  2  .  .  M  1  2  .  .  M  1  2  .  .  M  .  .  1  2  .  .  M | N(1+2+3+…+M) |

Summation : T2(N,M) = N(1+2+3+…+M)

= N[M(M+1)/2]

= N[(M2+M)/2]

= (NM2+NM)/2

Summation for the entire code: T1(N)+ T2(N,M)

= [(N3+N2)/2]+[(NM2+NM)/2]

= **½(N3+N2+NM2+NM)** -> Closed Form

Dominant terms : **N3 and NM2**

Theta : **Θ (N3+NM2)**

================================================================

Q2) for(i=1; i<=N; i=i+7) ;

Let’s assume number of values of i=p

|  |  |  |
| --- | --- | --- |
| **Iteration** | **Values of i** | **Total no.of loop iterations** |
| 1 | 1 | 1 |
| 2 | 8 | 1 |
| 3 | 15 | 1 |
| 4 | 22 | 1 |
| . | . | . |
| . | . | . |
| t | i | 1 |
| . | . | . |
| . | . | . |
| p | N | 1 |

For 4th iteration, N=22, total no. of for loop iterations so far is 4 times which is 22/7=4(approx.)

For 5th iteration, N=29, total no. of for loop iterations so far is 5 times which is 29/7=5(approx.)

For pth iteration, N, total no. of for loop iterations so far is N/7 times.

Summation : T(N) = 1+1+1+…+1

= (N/7)1

= **N/7**

Dominant term : **N/7**

Theta : **Θ (N/7)**

================================================================

Q3) Theta time complexity

**i = 0;**

**while (i<=N){**

**for(t=0, k=1; k<N; t=t+1, k=2\*k)**

**printf("G");**

**i=i+t;**

**}**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Assumed value of N** | **Iteration of i** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t inside and outside for loop** | **total repetitions due to k for t loop for one i** |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 |

Let’s assume the value of **N=1**.

Since **i and t** will always be **0**, we will run into an **infinite loop** where **G will not be printed** even once.

If N>1, ‘G’ will not go into infinite loop. But we are going into an infinite loop for N=1, hence we are not discussing that.

If N>1, then below is the summation table. Let lgN=Log2N

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Assumed value of N** | **Iteration of i** | **Values of i** | **Values of k** | **Iterations count for k** | **Values of t inside for loop** | **Values of t outside for loop/inside while loop** | **total repetitions due to k for k loop and t loop for one i** |
| 2 | 1 | 0 | 1 | 1 | 0 | 1 | 3 |
| 2 | 1 | 1 | 1 | 0 | 1 |
| 3 | 2 | 1 | 2 | 0 | 1 |
| 3 | 1 | 0 | 1  2 | 1  2 | 0  1 | 2 | 4 |
| 2 | 2 | 1  2 | 1  2 | 0  1 | 2 |
| N>1 | 1  2  .  .  .  .  .  N | 0  0+t  .  .  i+t  .  .  N | 1  1,2,4,…,N-1  .  1,2,4,…,N-1  .  1,2,3,…,N-1 | lgN  lgN  .  .  lgN  .  lgN  .  lgN | 0  0,1,2,…,N-1  .  0,1,2,…,N-1  .  0,1,2,…,N-1 | 1  1,2,…,N  .  1,2,…,N  1,2,…,N  .  1,2,…,N | N  N  .  N  .  N |

Value of i depends on the value of t (i=i+t) and t depends on k (t iterates until k iterates), whereas k depends on N (k<N).

Summation: T(N) = **NlgN**

For the value of N, there is a lot of ambiguity in the values of i, k and t. Hence this code is highly susceptible to a lot of errors.

================================================================

**Q4) T(n) =** **1+31+32+33+ … +3n**

Code for the above time complexity is

**for(i = 1; i <= N; i=i\*3)**

**for (k = 1; k <= i; k++);**

Let’s assume N is a multiple of 3

For the mth iteration : Value of i=i

For the tth iteration : Value of i=N

|  |  |  |  |
| --- | --- | --- | --- |
| **Iteration of i** | **Values of i** | **Values of k** | **total repetitions due to k for k loop for one i** |
| 1 | 1 | 1 | 1=30 |
| 2 | 3 | 1,2,3 | 3=31 |
| 3 | 9 | 1,2,3,4,5,6,7,8,9 | 9=32 |
| . | . | . | . |
| . | . | . | . |
| m | i | 1,2,3,…,i | 3m-1 |
| . | . | . | . |
| . | . | . | . |
| t | N | 1,2,3,…,i,…,N | 3t-1=3n |

Summation : **T(N)= 30+31+32+33+ … +3n = 1+31+32+33+ … +3n**

Closed Form :**(3n+1-1)/2**

Dominant Term : **3n/2**

Theta : **Θ(3n/2)**

================================================================

Q5) Dominant term and Theta

**N3+500N2+NM+106**

Dominant term : **N3 and NM**

Theta : **Θ(N3+NM)**

**100N3+20N2+15M+5N**

Dominant term : **N3 and M**

Theta : **Θ(N3+M)**

================================================================

Q6) how long they take for N=10, N=100, N=300. (Also try N=1000)

- the runtime effect of replacing the ‘res = res+1’ with the ‘print…’ instruction for runtime\_increment and runtime\_print.

- how the runtime depends on the value of N for runtime\_increment.

- how much faster (i.e. for smaller values of N) the runtime\_pow becomes too slow.

**// run for N = 10, N = 100, N = 1000**

**void runtime\_increment(int N){**

**int i, k, t, res = 0;**

**for(i = 1; i <= N; i=i+1)**

**for(k = 1; k <= i; k++)**

**for(t = 1; t <= N; t++)**

**res = res + 1;**

**}**

**For N = 10**

Runtime: It’s super fast. Time taken is **less than a second** (in microseconds).

**For N = 100**

Runtime: It’s also super fast. Time taken is **less than a second** (in microseconds).

**For N = 1000**

Runtime: **Approximately 0.293 seconds**. Less than a second.

*As the value of* ***N increases****, the* ***runtime*** *of the loop also* ***increases*** *for runtime\_increment. But the rate of increase of runtime is slow when compared to increase in value of N.*

When ‘res = res+1’ is replaced with the ‘print…’, then

**// run for N = 10, N = 100, N = 300, N = 1000**

**void runtime\_print(int N){**

**int i, k, t;**

**for(i = 1; i <= N; i=i+1)**

**for(k = 1; k <= i; k++)**

**for(t = 1; t <= N; t++)**

**printf("A");**

**}**

**For N = 10**

Runtime: **Approximately 0.007 seconds**. Time taken is less than a second. A gets printed 550 times.

**For N = 100**

Runtime: **Approximately 2.5 to 3 seconds**. Time taken is just little more than 3 seconds. A gets printed 505000 times.

**For N = 300**

Runtime: **Approximately 83 to 85 seconds**. Time taken is little less than a minute and a half. A gets printed 13545000 times.

**For N = 1000**

Runtime: **Approximately 1 hour 35 mins** (5695.366 seconds). A gets printed 500500000 times.

It took too much time to give the result. But I was patient enough for the program to complete execution.

*By replacing ‘res = res+1’ with the ‘print…’, we infer that the time taken for* ‘print…’ *is slower or longer than the execution time of ‘res = res+1’.* ***The runtime of ‘res = res+1’ is much faster than runtime of ‘print…’****.*

**// run for N = 10, N = 15, N = 20, N = 25, N = 30**

**void runtime\_pow(int N){**

**int i, res = 0;**

**for(i = 1; i <= pow(2.0, (double)N); i=i+1)**

**res = res + 1;**

**}**

**For N = 10**

Runtime: It’s super fast. Time taken is **less than a second**(in few microseconds).

**For N = 15**

Runtime: It’s also fast. Time taken is less than a second. **Approximately 0.004 seconds**.

**For N = 20**

Runtime: It’s also fast. Time taken is less than a second. **Approximately 0.015 seconds**.

**For N = 25**

Runtime: It’s also fast. Time taken is less than a second. **Approximately 0.069 seconds**.

**For N = 30**

Runtime: It’s also fast. Time taken is a little more than a second. **Approximately 2 seconds**.

*As the value of* ***N increases****, the* ***runtime*** *of the loop also* ***increases*** *for runtime\_pow. But the rate of increase of runtime is fast for higher values of N.*

Example: For N=30, it took two seconds. For N=40, I had to wait for a very long time, so I stopped execution.

Q6 b) Time complexity ‘closer’ to that of the runtime\_rec

**#include <stdio.h>**

**#include <stdlib.h>**

**#include <math.h>**

**void runtime\_rec(int N, char \* str){**

**if (N==0) {**

**//printf("%s\n", str);**

**return;**

**}**

**str[N-1] = 'L';**

**runtime\_rec(N-1, str);**

**str[N-1] = 'R';**

**runtime\_rec(N-1, str);**

**}**

**int main(int argc, char\*\* argv) {**

**int N = 0;**

**char ch;**

**char str[100];**

**printf("run for: N = ");**

**scanf("%d", &N);**

**str[N] = '\0'; //to use it as a string of length N.**

**printf("runtime\_rec(%d)\n", N);**

**runtime\_rec(N, str);**

**}**

**For N=10**

Runtime: It is very fast. Time taken is less than a second. **Approximately 0.001 seconds**.

**For N=15**

Runtime: It is also fast. Time taken is less than a second. **Approximately 0.222 seconds**.

**For N=20**

Runtime: It took some time. **Approximately 7.299 seconds**.

**For N=25**

Runtime:It took a long time. **Approximately 250 seconds**.

*As the value of* ***N increases****, the* ***runtime*** *of the program also* ***increases****.*

*Moreover, we are printing the values of runtime\_rec. From question 6a, we infer that printing values(runtime\_print) take longer time than just calculating(runtime\_increment).*

Rate of increase of runtime is drastically fast for higher values of N.

Hence, **time complexity of runtime\_print is closer to that of runtime\_rec**.