CSE 2320 - Homework 1 - Solution

Total: 100 points

Topics: instruction count, time complexity for loops, good information delivery, .

Reference: 'Time Complexity for Loops'

Q1. (45 points) For each of the following pieces of code do all of the following:

- draw the table with these columns: iteration, value of i, values of k, iterations count for k, values of t, iteration count for t, total repetitions due to k fork and for t loops for one i.
- write the summation,
- write the closed form solution where required. The summations in the cheat sheet have their closed form. E.g. for $1+2+3+\ldots+(n-1)+n$ also written as $\sum_{t=1}^{n} t$, has closed form: $\frac{n(n+1)}{2}$.
- dominant term (if you could find a closed form for the summation)
- write **Θ** (Theta) (if you can)
- Use this format:

Table:

Summation: …….

Closed form solution: ……

Dominant term: …… **Θ** (…….)

a) (15 points) Closed form and Θ not required.

```
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");
```

i	Values of k	Repe-titions for k loop	Values for t	Repeti-tions for t loop (for one k)	Repetitions of the printf from loops over k and t for one value of i
1	1	1	1,2,4,8,… powers of 2 ≤N	lg(N)	lg(N)*1
2	1,2	2	1,2,4,8,… powers of 2 ≤N	lg(N)	lg(N)*2
i	1,2,4,8,… powers of 2 ≤i	lg(i)	1,2,4,8,… powers of 2 ≤N	lg(N)	lg(N)*lg(i)
N	1,2,4,8,… powers of 2 ≤N	lg(N)	1,2,4,8,… powers of 2 ≤N	lg(N)	lg(N)lg(N)

T(N) = sum of terms in rightmost column = lg(N) + 2lg(N) + … + lg(N)lg(i)+ … + lg(N)lg(N) = $\sum_{i=1}^{N} \lg(N)\lg(i) = \lg(N)\sum_{i=1}^{N} \lg(i)$

b) (15 points) Closed form, dominant term and Θ <u>required</u>.

```
for(i = 1; i <= N; i=i+1)
     for(k = 1; k <= S; k++)
          for(t = 1; t <= i; t++)
               printf("D");
```

i	Values of k	Repetitions for k loop	Values for t	Repetitions for t loop (for one k)	Repetitions of the printf from loops over k and t for one value of i
1	1,2,3,…,S	S	1	1	1S
2	1,2,3,…,S	S	1,2	2	2S
i	1,2,3,…,S	S	1,2,3,4,…,i	i	iS
N	1,2,3,…,S	S	1,2,3,4,…,N	N	NS

T(N) = sum of terms in rightmost column = S +2S +3S + … +NS = S(1+2+3+..N)= S*N*(N+1)/2

Closed form: S*N*(N+1)/2

Dominant term: N^2S

Θ(N^2S)

c) (15 points) Closed form, dominant term and Θ required.

```
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");
}
```

Because the two for k loops are sequential (not nested) and their contribution will be added (not multiplied) we will write two tables and add their results.

i	Values of k	Repeti-tions for k loop	Values for t	Repetitions for t loop (for one k)	Repetitions of the printf(E) from loops over k and t for one value of i
1	1, 2, … … … N	N	1 1,2 … 1,2,…,k … 1,2,…,N	1 2 … k … N	1+2+3+…k+…+N= N(N+1)/2
2	same as	the	above	row	N(N+1)/2
i	same as	the	above	row	N(N+1)/2
N	same as	the	above	row	N(N+1)/2

$T_1(N)$ = sum over terms in the rightmost column = N(N+1)/2 + …..+ N(N+1)/2 = N*N*(N+1)/2

The second table is identical to this one, except that K goes up to M:

i	Values of k	Repeti-tions for k loop	Values for t	Repetitions for t loop (for one k)	Repetitions of the printf(E) from loops over k and t for one value of i
1	1, 2, … … … M	M	1 1,2 … 1,2,…,k … 1,2,…,M	1 2 … k … M	1+2+3+…k+…+M= M(M+1)/2
2	same as	the	above	row	M(M+1)/2
i	same as	the	above	row	M(M+1)/2
N	same as	the	above	row	M(M+1)/2

$T_2(N)$ = sum over terms in the rightmost column = M(M+1)/2 + …..+ M(M+1)/2 = N*M*(M+1)/2

$T(N) = T_1(N) + T_2(N)$ = N*N*(N+1)/2 + N*M*(M+1)/2

Dominant terms: N^3 and NM^2

$\Theta(N^3+NM^2)$

Q1. (5 points) How many times does this **loop iterate? Give the answer as a function of N** (you can be off by +,- a constant, but not by *,/ a constant). (This is NOT a detailed instruction count, but an estimation of the count of iterations)

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

Answer: N/7

Q3. (13 points) Give the Theta time complexity and show your work in the best way you can. **5 points** will be given for how easy it is to understand your explanations. Keep in mind that a **good explanation is brief but clear and to the point** (may involve math functions/notations, the first few values of a loop variable, a table to organize the data…).

```
i = 0;
while (i<=N){
      for(t=0, k=1; k<N; t=t+1, k=2*k)
            printf("G");
      i=i+t;
}
```

→ repeats lg(N) every time

→ i increments by lg(N)every time => arithmetic progression

=> N/lg(N) repetitions of the while loop

⇨ T(N) = [N/lg(N)] * lg(N) = N

Q4. (11 points) Write code that will have the time complexity below and after that show why it has that complexity (that is, take your code and derive the time complexity for it).

$T(n) = 1 + 3^1+3^2+3^3+ \dots +3^n$

```
for(i = 0; i<=n; i++){
      for(k = 1; k<= pow(3.0, (double)i); k++)
            printf("A");
}
```

i	Values of k	Repeti-tions for k loop	Repetitions of the printf from loops over k for one value of i
0	1	1	1
1	1,2,3	3	3
2	1,2,3,4,5,6,7,8,9	3^2	3^2
i	1,2,3,4,…, 3^i	3^i	3^i
n	1,2,3,4,…, 3^i	3^n	3^n

T(N) = sum over terms in the rightmost column $=1+3+3^2+3^3+ \dots +3^n$

Q5. (6 points) Find the dominant terms and write Θ for each of the functions below. (Pay close attention.)

$N^3+ 500N^2+NM + 10^6 = \Theta(\dots\dots$**$N^3+ NM$** $\dots\dots)$

$100N^3+20N^2+15M +5N = \Theta(\dots\dots$**$N^3+M$** $\dots\dots\dots)$

Q6. (16 points) **a)** (12 points) Run the following pieces of code and see how long they take for N=10, N=100, N=300. (Also try N=1000). Pay attention to the following and BRIEFLY write your observations (e.g. for each value of N say: "it ran in less than a second", "several seconds, minutes", "I got tired of waiting after this much time",…). Remember that you are in the College of Engineering. We do not want novels, we want information to the point. Therefore make the information stand-out: use a table, bullet points, fewer words and more numbers. **The points will be given for how well the information stands-out**. (Your information stands-out if I **glance at it** and I can easily see the content.)

- 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;
}

// 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");
}

// 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;
}
```

b) (4 points) Look at the program below.
Which of the three functions above (runtime_increment, runtime_print and runtime_pow) has the time complexity 'closer' (or more similar) to that of the runtime_rec?
Note that you do not need to compute the time complexity. We did not cover that yet. Use other methods (e.g. look at the actual time it takes to execute for different values of N and see to which function from above).

Answer: Similar to `runtime _pow`.

```
#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);
}
```

Note that since this code uses the math library, if you run it on omega, you will need to link the library at compile time:

```
gcc -o rec main.c -lm
```

and then run it as usual:

```
./rec
```

Write all your answers in a pdf called hw1.pdf. **Make sure you use this exact name.**

Put your NAME and section at the top of the page.

You can write the answer on paper and scan the paper as pdf, but make sure your handwriting is neat and can be read.

- Answer what the question is asking for. For example, a question maybe asking for loop iterations as a function of input N, you will not get any point for answering detailed instruction count.
- Points will be deducted for not submitting in the proper format.