– Sadhika Huria –

CMPT 225 Fall 2024 Assignment 3

– 301 599 274 –

Question 1 : Cartesian Product

Cost Function:

Outside loop - 1 operation

Outer loop comparison - n + 1 operations

Outer loop - 3 operations (times n)

Inner loop comparison - n + 1 operations (times n)

Inner loop - 3 operations (times n^2)

$= 3n^2 + n(n+1) + 3n + (n+1) + 1$

$= 3n^2 + n^2 + n + 3n + n + 1 + 1$

$= 4n^{2,} + 5n + 2$

Barometer operations:

Comparisons: while (i < n) {

while (j < n) {

Increments: i++;

j++;

Time Complexity : $O(n^2)$

Question 2 : Triangle

Cost Function:

Outside loop - 1 operation

First outer loop - 5n + 1

First inner loop - n(3i).

Average- n/2

$3n^2/2$

Second outer loop - 5n + 1

Second inner loop - n(3i)

Average - n/2

$3n^2/2$

$= (3n^2/2) + (3n^2/2) + 5n + 1 + 5n + 1 + 1$

$= 3n^2 + 10n + 3$

Barometer operations:

Comparisons: while (i < x) {

while (j <= i) {

while (i > 0) {

while (j <= i) {

Increments: i++;

j++;

i- -;

j++;

Time Complexity : $O(n^2)$

Question 3 : Matrix Self Multiply

Cost Function:

Outside loop - 3 operations

Outer loop comparison - rows + 1

Outer Loop - 2 operations * rows

Middle loop comparison - (columns + 1) * rows

Middle loop - 5 columns * rows

Inner loop comparison - (rows + 1) * columns * rows

Inner loop - 4 * rows * columns * rows

Rows (n) = columns, throughout the code

$= 4n^3 + n^3 + n^2 + 5n^2 + n^2 + n + 2n + n + 1 + 3$

$= 5n^3 + 7n^2 + 4n + 4$

Barometer operations:

Comparisons: `while ( r < rows ){`

`while ( c < columns ) {`

`while ( iNext = 0 ) {`

Increments : `iNext++;`

`c++;`

`r++;`

Operation Executed the most:

```
next += m[rcIndex(r, iNext, columns)]  * m[rcIndex(iNext, c, columns)];
```

Time Complexity : $O(n^3)$

Question 4 : selection sort

Cost Function:

Function is called n-1

First comparison 1 operation

Outside loop - 5 operations

While loop comparison - (n - i)

While loop - 3 operations * (n - i - 1)

Average n/2

Base case comparison - 1

Because of how the function is called at the end. It can be treated like a loop to count.

$= 6 (n-1) + (n-1)(n/2) + 3(n-1) (n/2) + 1$

$= 6n - 6 + n^2/2 - n/2 + 3n^2/2 - 3n/2 + 1$

$= 2n^2 + 4n - 5$

Barometer Operations:

Comparison: if (i < n -1) {

while (next < n) {

Increment: next++;

Function calls: ssort(arr, n, i + 1);

Time Complexity : $O(n^2)$

Question 5 : Pattern

Cost Function:

Function has 2 calls. Each one being called log2(n) times.

Since each call has 2 other calls. This can be looked at as a tree.

2^(log2(n)) calls: n calls or log2(n) + log2(n) = n

Each call:

Outside loop: 6 operations, including comparisons

Loop : 3*n operations

= 3n+ 6 operations

Total operations:

= n * (3n + 6)

= $3n^2$ + 6n

Barometer Operations:

Comparisons: while (ast < n) {

Function call: pattern(n/2 , i);

pattern(n/2 , i);

Time Complexity : $O(n^2)$

Question 6 : linear search

Cost Function:

If target not found, then n operations, however if the target is found (worst case to be at the last index), the return value of that recurvice call being 0 will invoke another call (2 calls in total); causing the function to act as a binary recursive tree. In the worst case, the functions are called 2^n times.

Each call:

Returning Comparisons - 2 operations

If target not found - 1 operation

If target found - 1 operation

Operations other than recursive calls - 4 which can be written as 2^2

Each call is making two more calls

Cost function:

$C(n) = 2^2 + 2(C(n-1))$

$= 2^2 + 2(2^2 + 2\,C(n-2))$

$= 2^2 + 2^3 + 2^2\,C(n-2)$

$= 2^2 + 2^3 + 2^2\,(2^2 + 2\,C(n-3))$

$= 2^2 + 2^3 + 2^4 + 2^3\,C(n-3)$

...

$= 2^2 + 2^3 + ... + 2^{n+1} + 2^n C(0)$

If len is 0, one operation. C(0) is 1

$= 2^0 + 2^1 + 2^2 + 2^3 + ... + 2^{n+1} + 2^n - 1 - 2$

Geometric series + 2^n

$= ((1- 2^{n+1}) / (1-2)) + 2^n - 1- 2$

$= 2^{n+1} - 1 + 2^n - 1 - 2$

$= 2^{n+2} - 4$

Barometer Operations:

Comparisons: if (len == 0) return -1;

if (arr[0] == target) return 0;

Function calls: lsearch(arr+1, len-1, target)

Time Complexity : $O(2^n)$

Question 7 : pow

Cost Function:

Loop runs log2(n) times

Outside loop - 1 operation

Loop comparison - log2(n) + 1

Inside loop - 4 * log2(n) operations

= 5 log2(n) + 2

Barometer Operations:

Comparison: while (exp > 0) {

Decrement: exp >>= 1;

Time Complexity : $O(\log_2(n))$