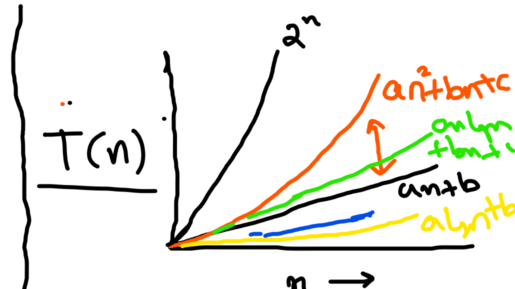
Step Count Analysis of nested loops

Sunday, 16 August 2020 12:45 PM



(we are interested in how $T(n)$ behaves for large n)

What is the total **step-cost** of the following pieces of code?

```
int i = 1, j=1, sum=0;
for(;i<=n; i++)
    for(j=1;j<=n; j++)
        sum=sum+1;
```

$\rightarrow c_1$

$\rightarrow m c_2$

$c_3(\underbrace{n+n+n+\cdots+n}_{m \text{ invoke}}) = c_3 mn$

$+ mnc_4$

$\rightarrow c_4 mn$

→ iterates n times

→ it is invoked n times

$m=n$

$T(m,n) = c_1 + m c_2 + mnc_3$

$= c_5 mn + m c_2 + c_1$

$= an^2+bn+c$

```
int i = 1, j=1, sum=0;
for(;i<=n; i++)
    for(j=1;j<=n; j=j*2)
        sum=sum+1;
```

$\rightarrow c_1$

$\rightarrow c_2 n$

$\rightarrow c_3 n\lg n$

$\rightarrow c_4 n\lg n$

invoked n time

iterates $\lg n$

$j=1,2,4,8,\ldots n$

$n, \frac{n}{2}, \frac{n}{4} + \cdots 1$

$\lg(n)$ terms

$\boxed{T(n) = an\lg n + bn + c}$

$= n\lg n$

$\underbrace{\lg n + \lg n + \cdots + \lg n}_{n}$

```
int i = n, j=1, sum=0;
for(;i>0; i=i/2)
    for(j=1;j<=n; j=j*2)
        sum=sum+1;
```

$\rightarrow c_1$

$\rightarrow c_2 \lg n$

$\rightarrow c_3 \lg^2 n$

$\rightarrow c_4 \lg^2 n$

$T(n) = c_5 \lg^2 n + c_2 \lg n + c_1$

$\lg_2$

$\lg^2 n$

Most Important Example

```
int i = n, j=1, sum=0;
for(;i>0; i=i/2)
    for(j=1;j<=i; j++)
        sum=sum+1;
```

$\rightarrow c_1$

$\rightarrow c_2 \lg n$

j=j*2

$\lg n$ invokations

$\underbrace{\lg n + \lg\left(\frac{n}{2}\right) + \lg\left(\frac{n}{4}\right)}_{\lg n \text{ terms}} + \cdots \lg(1)$

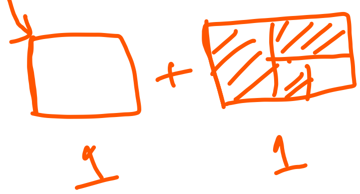
n^2

=

Worst Case Analysis

Tuesday, 18 August 2020 4:28 PM

$\lg n\left[1+\frac{1}{2}+\frac{1}{4}+\dots\right]$

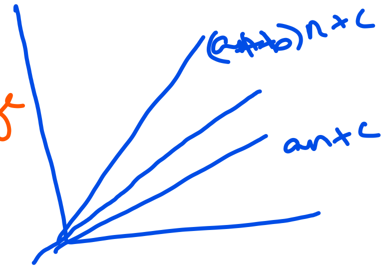


Our function belongs to the family of lines

$an+b\lg n+c$

$\underbrace{an+c}_{\text{line}} \le an+b\lg n+c \le \underbrace{(a+b)n+c}_{\text{line}}$

$\Theta(n)$



$n \le \dfrac{1+2+3+\dots n}{n} = \text{average} = n$

Push-back

$\hookrightarrow$ worst case scenario.

```
int search(int A[], int n, int key){
    int ind=0;
    while(ind<n && A[ind]!=key) ind++;
    if(ind<n)
        return ind;
    else
        return -1;
}
```

$\to c_1$

worst case scenario

key not found.

$\to c_2 n$

$\to c_3 n$

$\to c_4$

$T(n)=an+b$

$T(n)=an+b$

$T(n)=c$

Before writing $T(n)$ we decide which scenario of the program are we analyzing.

(we select: worst case scenario)

size<=cap

size<cap

$\lg\left(\frac{a}{b}\right) \stackrel{?}{=} \lg a-\lg b$

Pessimistic ✓

worst case

many times it is realistic.

$\overbrace{\lg\left(\frac{n}{1}\right)+\lg\left(\frac{n}{2}\right)+\lg\left(\frac{n}{4}\right)+\dots+\lg\left(\frac{n}{n}\right)}^{\lg n}$

$=\left[\lg n-\lg 1+\lg n-\lg 2+\lg n-\lg 4+\dots(\lg(n)-\lg n)\right]$

$=\left[\lg n-\lg 1+\lg n-\lg 2+\lg 4+\dots\lg(n)\right]$

$=\left[\lg^2 n-(\lg 1+\lg 2+\lg 4+\dots+\lg n)\right]$

$=\left[\lg^2 n-\left[0+1+2+3+\dots+\lg n\right]\right.$

Worst-case Analysis

Sunday, 16 August 2020 12:46 PM

$$= \lg^2 n - \left[\frac{\lg n(\lg n - 1)}{2}\right]$$

$$= a\lg^2 n + b\lg n + c$$

Consider the process of inserting elements in an array *in order*:

Repeat n times:

- Take an input x
- Insert x into array A at its correct place in non-decreasing order.

Here is the code:

```
void insertNumberIntoArray(int A[], int n){

    int x;
    for(int i=0; i<n; i++){
        cin>>x;
        int k=i-1;
        while(k>=0 && A[k]>x){
            A[k+1]=A[k];
            k--;
        }

        A[k+1]=x;
    }
}
```

$\boxed{2 * cap - 5}$

cap.

random

10 20

Next time

worst case Analysis

+

{Big-oh

Big-omega

Big-theta}