Analisa Algoritma

Kontrol struktur, iteratif, rekursif

Algoritma ???

Definisi:

urutan langkah” yang tepat dan pasti dalam memecahkan suatu masalah secara logis or set instruksi yang harus diikuti oleh komputer untuk memecahkan masalah

Beberapa masalah dapat diselesaikan dengan algoritma yang bermacam” asal hasilnya sama

Setiap Bahasa Pemrograman punya + - dalam mengimplementasikan algoritma

Setiap programmer dapat mengimplementasikan algoritma dengan cara yang berbeda”

Algoritma dapat dianalisis efisiensi dan kompleksitasnya, dimana program harus berhenti dalam batas waktu yang wajar (reasonable)

Penilaian algoritma didasarkan pada:

1. Time, waktu eksekusi (paling utama),
2. Space, penggunaan memori/sumber daya
3. Kesederhanaan dan kejelasan algoritma

Analisa Algoritma....What for??

Menganalisis algoritme yang efisien untuk memecahkan suatu masalah

Bagaimana mengkarakterisasi dan mengukur kinerja suatu algoritme

Mengukur jumlah sumber daya (*time & space*) yang diperlukan oleh sebuah algoritma

Analisa Algoritma....What for??

Waktu yang diperlukan (*running time*) oleh sebuah algoritma cenderung tergantung pada jumlah input yang diproses

Algoritma tidak terikat pada platform mesin, OS, BP, kualitas kompilator atau bahkan paradigma pemrograman (mis. Procedural vs Object-Oriented)

Analisa Algoritma….How???

Bagaimana menganalisa algoritma?

Jumlah waktu yang digunakan bervariasi tergantung pada

1. *kecepatan mesin*
2. *sistem operasi (multi-tasking)*
3. *kualitas kompiler*
4. *BP yang digunakan*

Sehingga kurang memberikan gambaran yang tepat tentang algoritma

Analisa Algortima....Kesimpulan

- Analisa algoritma tidak mudah dilakukan secara pasti→ hanya diambil:
  - Kondisi rata-rata (*average case*)
  - Kondisi terburuk (*worst case*)
  - Kondisi baik (best case)
- Waktu eksekusi dipengaruhi:
  - Jenis data input
  - Jumlah data input
  - Pemilihan instruksi BP

Analisa Algortima....Kesimpulan

Faktor" yang menyulitkan analisis, disebabkan:

1. Implementasi instruksi oleh BP berbeda"
2. Ketergantungan algoritma terhadap jenis data
3. Ketidakjelasan algoritma yang diimplementasikan

Langkah" analisis algoritma:

1. Menentukan jenis/sifat data input
2. Mengidentifikasi *abstract operation* dari data input
3. Menganalisis secara matematis untuk menentukan *average case, worst case* dan *best case*

Time Analysis

Algoritma bekerja berdasarkan input yang dimasukkan user.

Setiap input memiliki ukuran. Misalnya kita hendak mengurutkan sejumlah bilangan, banyaknya bilangan yang perlu diurutkan merupakan ukuran besarnya input

Makin besar ukuran input yang dimasukkan, pada umumnya waktu proses akan semakin lama -> order of growth

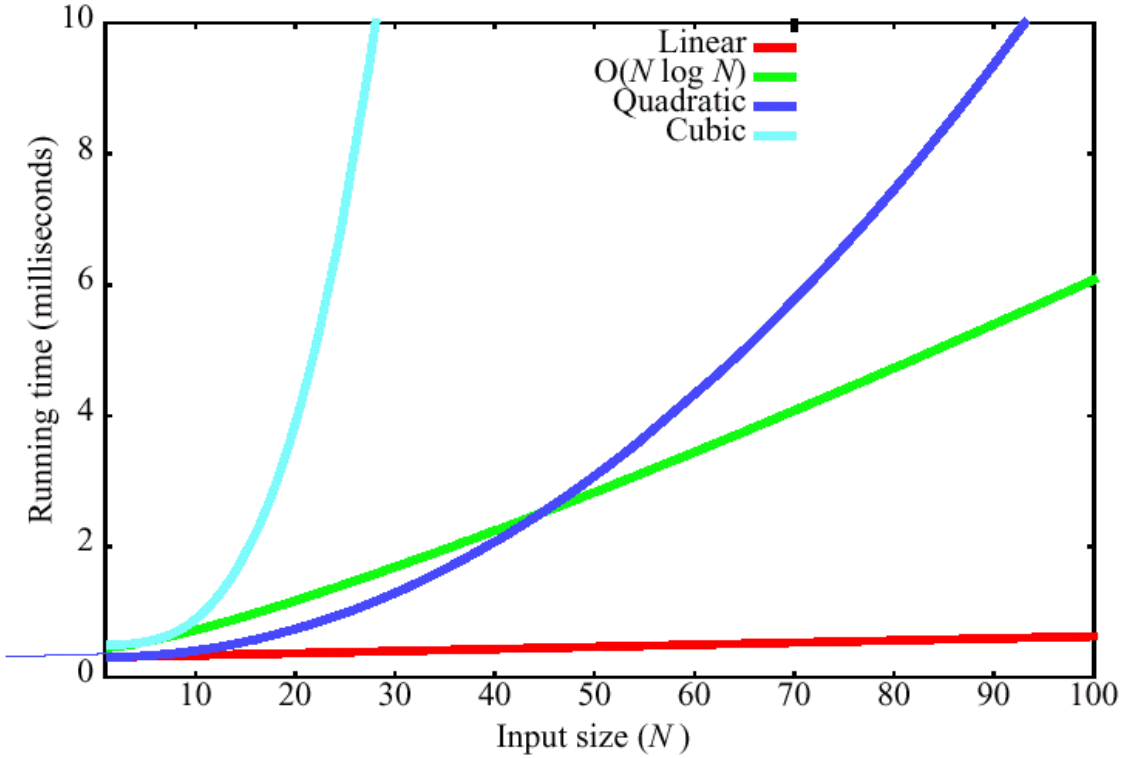
> Dengan demikian, algorithm's time complexity adalah function of the size of the input.

Tergantung pada isi input, waktu proses dapat bervariasi :

1. Keadaan terbaik (best case)
2. Keadaan rata-rata (average case)
3. Keadaan terburuk (worst case)

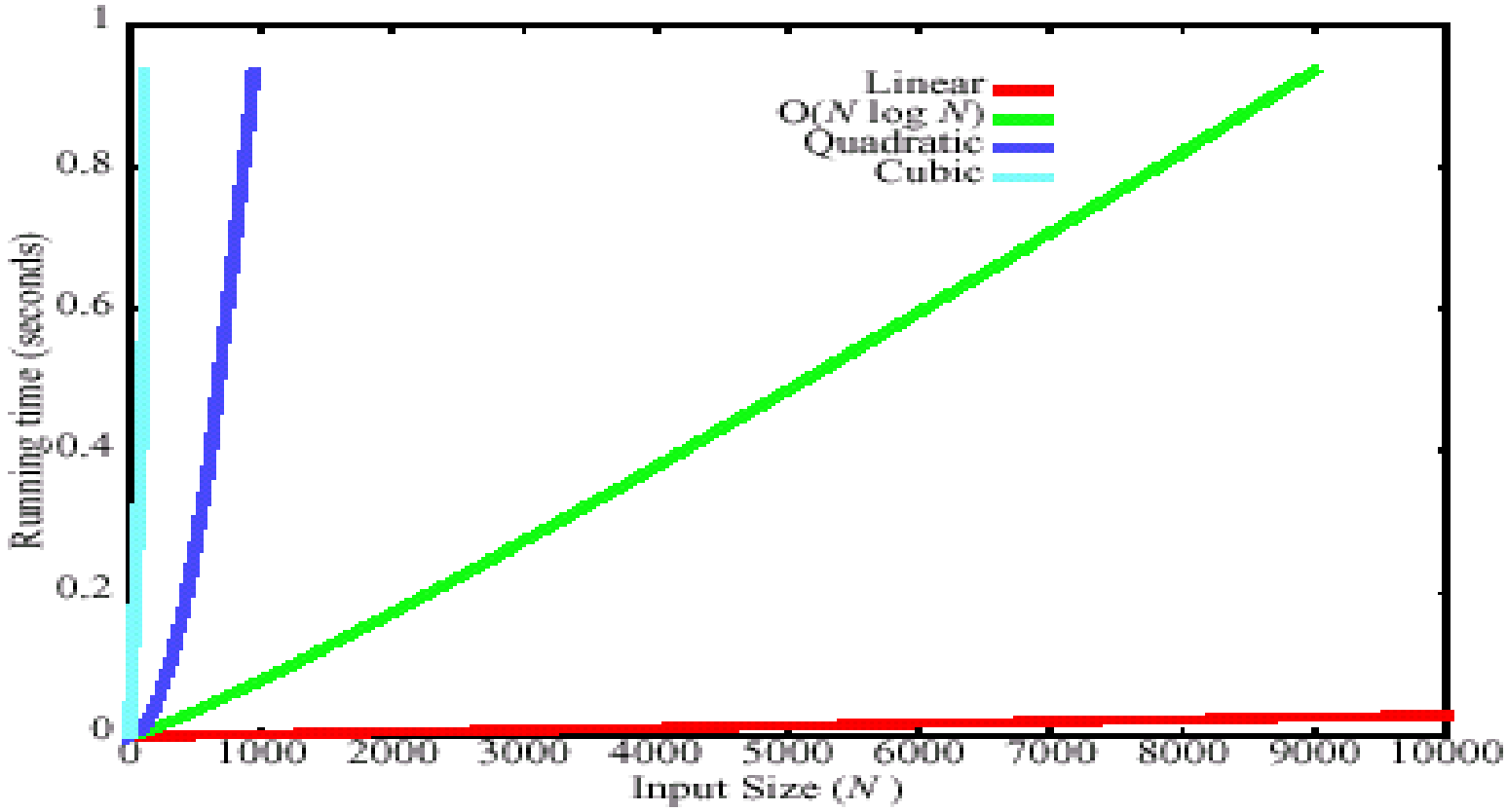
Running Time

Untuk input yang sedikit, beberapa fungsi lebih cepat dibandingkan dengan yang lain



Running Time

Untuk input yang besar, beberapa fungsi memiliki running-time sangat lambat - tidak berguna



Kompleksitas konstan

Waktu pelaksanaan
algoritma adalah tetap, tidak
bergantung pada ukuran
masukan

Contoh : printf(“hello“);

Kompleksitas linier

```
for(i=0;i<n;i++)
    printf(“hello”)
```

Berapa kali printf dieksekusi

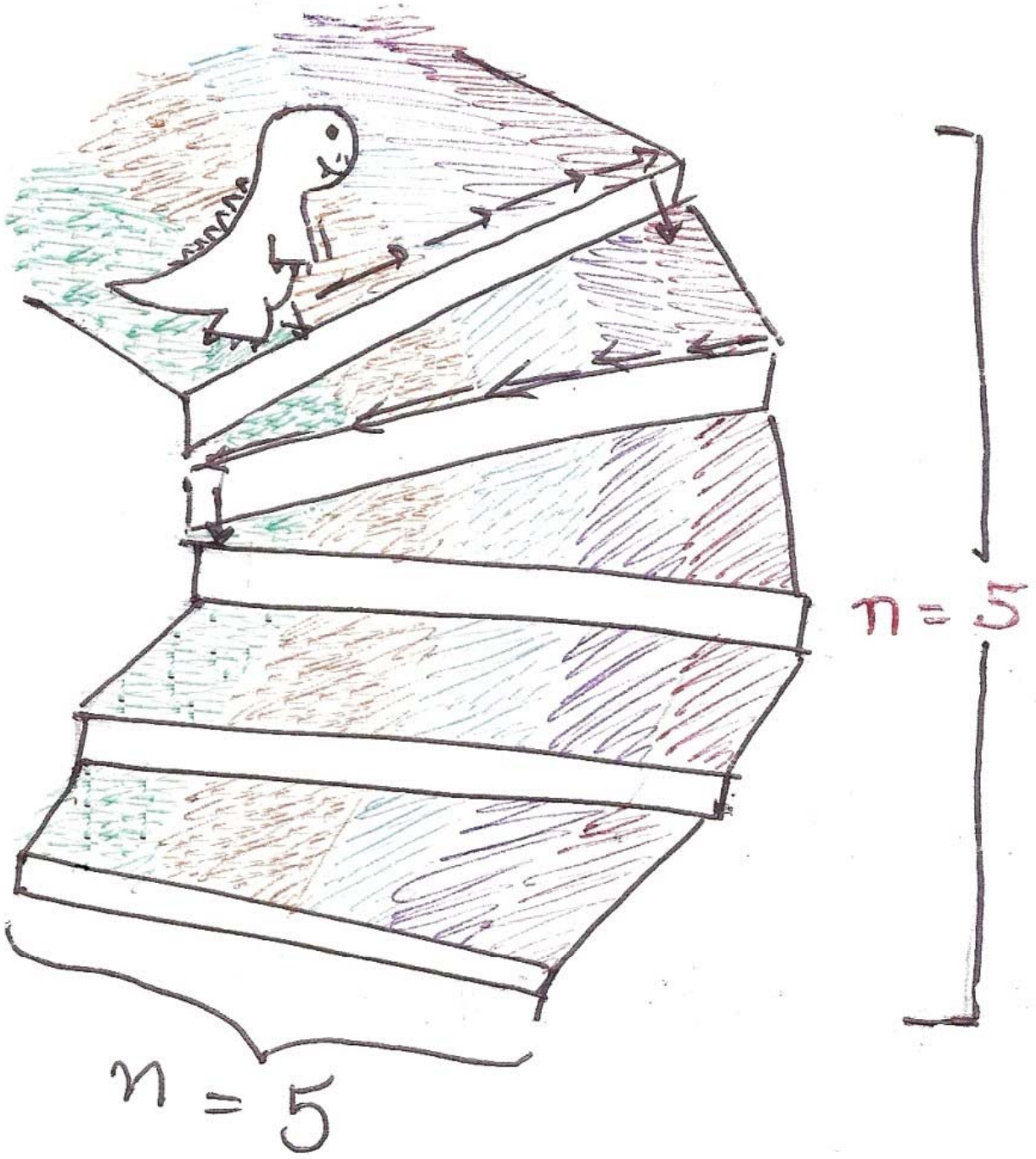
N kali

Kompleksitas kuadratik

```
for(i=0;i<n;i++)
    for (j=0; j;<n;j++)
        printf(“hello”)
```

Berapa kali printf dieksekusi

N*N kali


n = 5
n = 5

Kompleksitas kubik

```
for(i=0;i<n;i++)
    for (j=0; j;<n;j++)
       for(k=0;k<n;k++)
          printf(“hello”)
```

Berapa kali printf dieksekusi

N*N*N kali

Kompleksitas log N

```
for(i=1;i<n;i=i*10)
    { printf(“hello”);
    }
```

Berapa kali printf dieksekusi

Log N kali

Ciri kompleksitas logaritmik log N: Lebih cepat dari linier karena ada percepatan jumlah iterasi


RED ALERT! RED ALERT!
I DON'T UNDERSTAND
MemesHappen

$\log_a$

Kompleksitas N log N

```
 for(i = 1; i <= n; i++)
{
   for(i=1;i<n;i=i*10)
       { printf(“hello”);
       }
 }
```

Berapa kali printf dieksekusi

N log N kali

Notasi BIG-OH

Untuk menyatakan kompleksitas algoritma biasanya menggunakan notasi big oh

O(1) artinya algoritma konstan.

O(log n) contohnya pada full balanced Binary Search Tree

O(n) artinya algoritma linear

O(n^2) artinya algoritma quadratic

O(n^3) artinya algoritma qubic

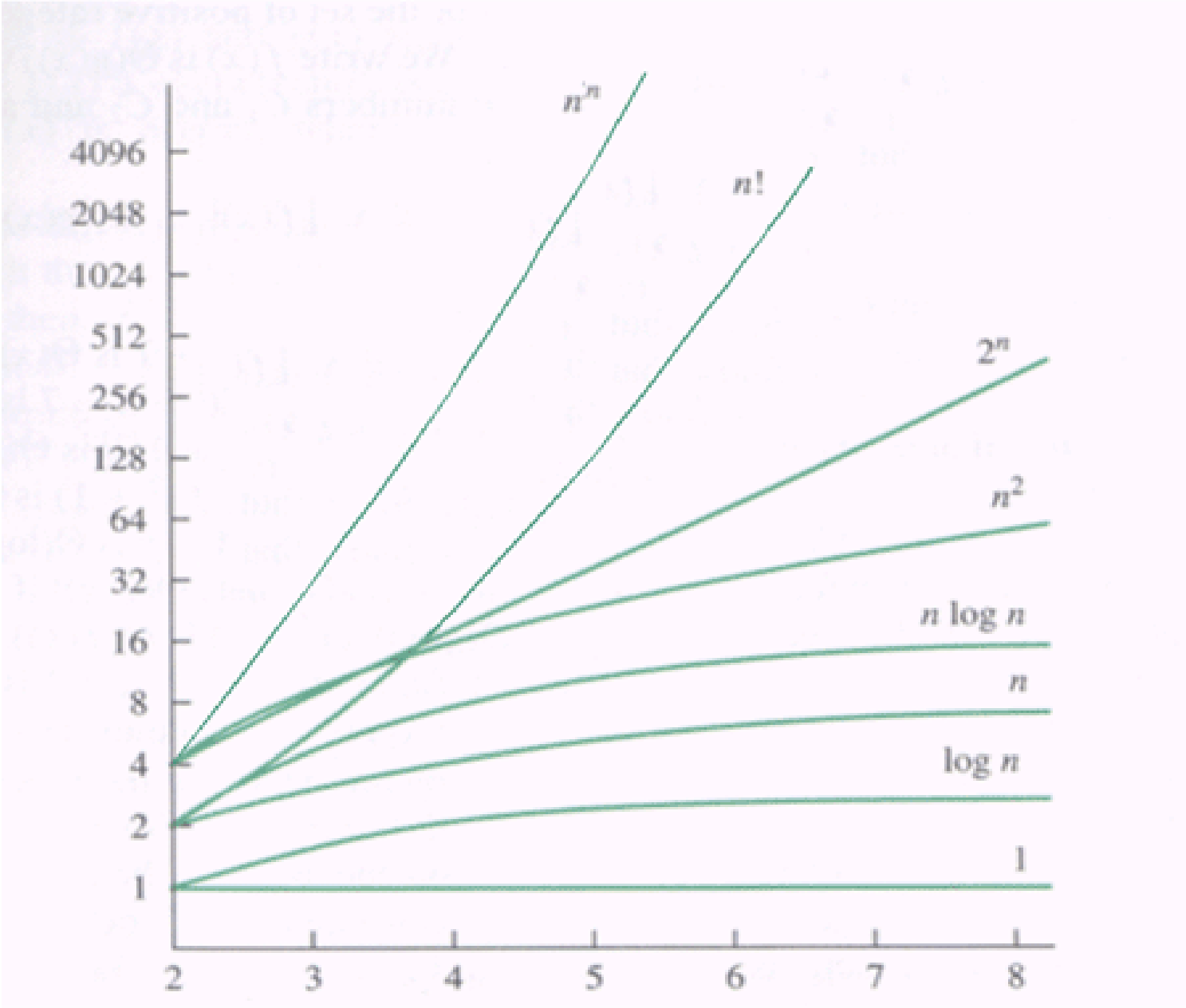
O(n^m) artinya algoritma eksponensial

Notasi Big-O bisa berisi kombinasi dari contoh di atas. Ex: O(n log n)

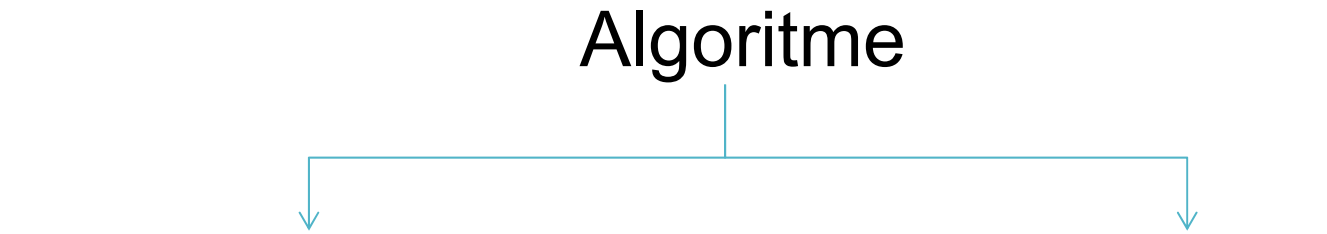
Big-Oh to Primary Sorts

- Bubble Sort = n^2
- Merge Sort = $n \log(n)$
- Quick Sort = $n \log(n)$

- Find out!
- Radix Sort = n
- Selection Sort = n^2
- Insertion Sort = n^2


n^n
$n!$
2^n
n^2
$n \log n$
n
$\log n$
1
4096
2048
1024
512
256
128
64
32
16
8
4
2
1
2
3
4
5
6
7
8

Iteratif dan rekursif




```
A(n){
for  i := 1 to n do
max(a,b)
}
```

```
A(n){
if(  )
   A(n/2)
}
```

- Iteratif: for loop, while loop, do while
- Rerkursif: calling function itself

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i;
for  i = 1 to n do
   printf(“analgor”);
}
```

```
func(n){
int i=1, s=1;
   while(s<=n)
   {
      i++;
      S=S+i;
      printf(“analgor”);
   }
}
```

Remember:

1. $\sum a^i = 1/(1-a)$ jika $0 < a < 1$ dan $n \rightarrow \infty$
2. $\sum i = n(n+1)/2 \approx \frac{1}{2} n^2$
3. $\sum i^2 = n(n+1)(2n+1)/6 \approx 1/3\ n^3$
4. $\sum i^k \approx (n^{k+1})/|k+1|,\ k \neq -1$

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i;
for(i=1,i^2 <= n, i++)
   printf("analgor");
}
```

Berapa kali perintah cetak "analgor" yang dieksekusi?

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i, j,k,n;
for(i=1, i<= n, i++)
   {
   for(j=1, j<=i, j++)
      {
       for(k=1, k<=100, k++)
         {
         printf(“analgor”);
         }
      }
    }
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

ex:

```
for(i=1;i<=n;j++)
{
   for(j=1;j<=i;j++)
}
```

The complexity of this loop will be calculated like:

for i=1

inner loop runs (1 time)

for i=2

inner loop runs (2 times)

so now we need to make the series:

1+2+3+....+n= n^2(Series Sum)

Remember:

1. $\sum a^i = 1/(1-a)$ jika $0 < a < 1$ dan $n \rightarrow \infty$
2. $\sum i = n(n+1)/2 \approx \frac{1}{2} n^2$
3. $\sum i^2 = n(n+1)(2n+1)/6 \approx 1/3\, n^3$
4. $\sum i^k \approx (n^{k+1})/|k+1|,\ k \neq -1$

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i, j,k,n;
for(i=1, i<= n, i++)
    {
    for(j=1, j<=i^2, j++)
        {
         for(k=1, k<=n/2, k++)
           {
           printf("analgor");
           }
        }
     }
}
```

Berapa kali perintah cetak "analgor" yang dieksekusi?

Remember:

1. $\sum a^i = 1/(1-a)$ jika $0 < a < 1$ dan $n \rightarrow \infty$
2. $\sum i = n(n+1)/2 \approx \frac{1}{2} n^2$
3. $\sum i^2 = n(n+1)(2n+1)/6 \approx 1/3\, n^3$
4. $\sum i^k \approx (n^{k+1})/|k+1|,\ k \neq -1$

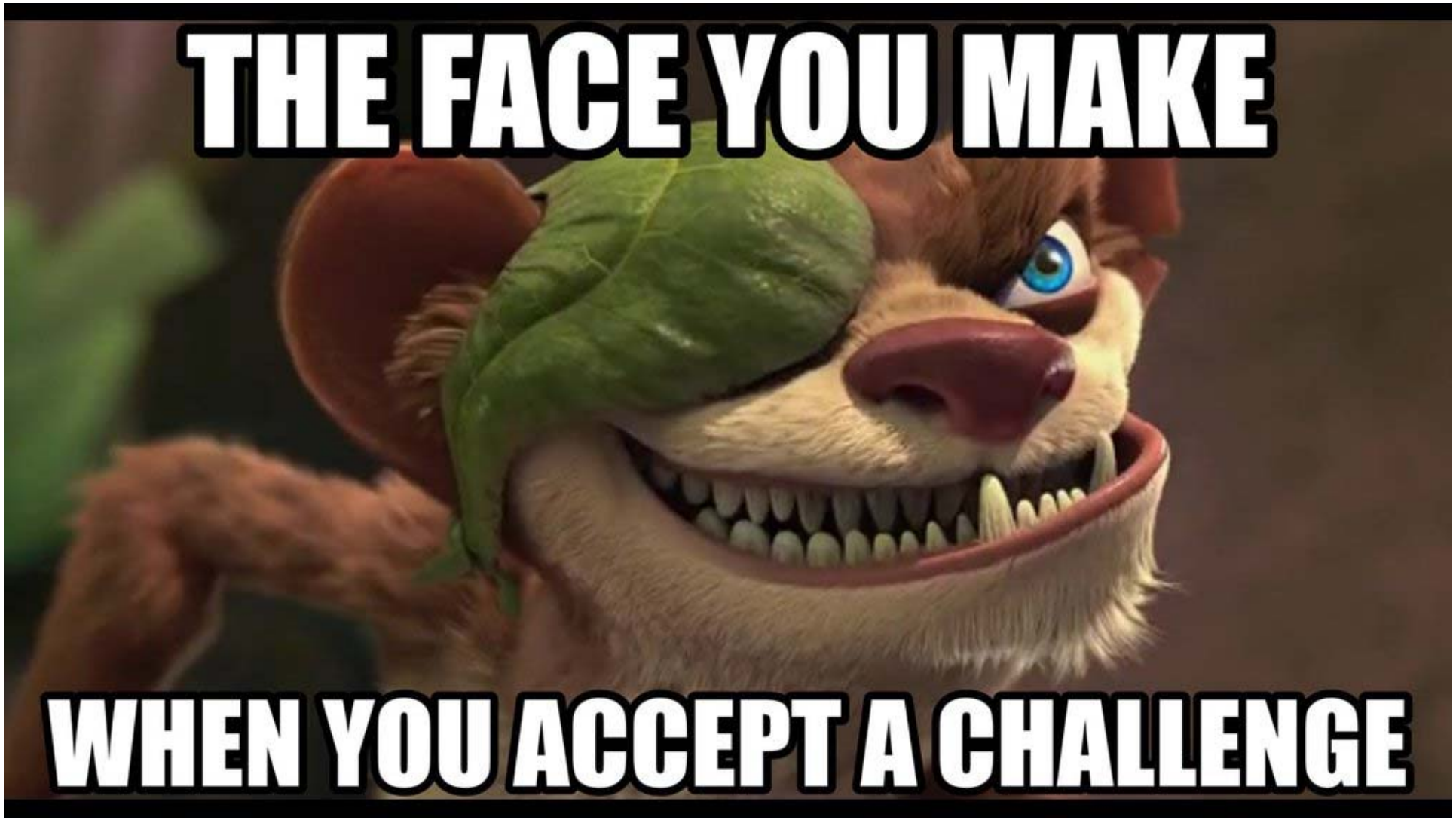
Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i;
for(i=1, i<n, i=i*2)
   printf(“analgor”);
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

More exercise?


THE FACE YOU MAKE
WHEN YOU ACCEPT A CHALLENGE

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
Asumsi n>=2

func(n)
{
  While (n>1)
   printf(“analgor”);
   n=n/2;
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

Remember

$$ {}^{a}\log b = \frac{{}^{n}\log b}{{}^{n}\log a} $$

Syarat $n > 0$ dan $n \neq 1$

Rumus Praktis Logaritma	
1.	${}^{a}\log b = x \Leftrightarrow a^{x} = b \; ; a > 0, b > 0, \text{dana} \neq 1$
2	${}^{a}\log a = 1 \text{ dan } {}^{a}\log 1 = 0$
3	${}^{a}\log b = \frac{1}{{}^{b}\log a} = \frac{{}^{n}\log b}{{}^{n}\log a} \; ; n > 0 \text{ dan } n \neq 1$
4	$a^{{}^{a}\log b} = b$
5	${}^{a}\log(b.c) = {}^{a}\log b + {}^{a}\log c$
6	${}^{a}\log \frac{b}{c} = {}^{a}\log b - {}^{a}\log c = -{}^{a}\log \frac{c}{b}$
7	${}^{a}\log b^{m} = m {}^{a}\log b \rightarrow {}^{a^{n}}\log b^{m} = \frac{m}{n} {}^{a}\log b$
8	${}^{a}\log b . {}^{b}\log c . {}^{c}\log d = {}^{a}\log d$
By : bahanbelajarsekolah.blogspot.com	

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i, j,k,n;
for(i=n/2, i<= n, i++)
   {
   for(j=1, j<=n/2, j++)
      {
       for(k=1, k<=n, k=k*2)
         {
         printf(“analgor”);
         }
      }
    }
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i, j,k,n;
for(i=n/2, i<= n, i++)
   {
   for(j=1, j<=n, j=j*2)
      {
       for(k=1, k<=n, k=k*2)
         {
         printf(“analgor”);
         }
      }
    }
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n){
int i,j;
for(i=1, i<= n, i=++)
    for(j=1, j<=n, j=j+i)
        printf(“analgor”);
}
```

Berapa kali perintah cetak “analgor” yang dieksekusi?

Iteratif dan rekursif

- Analisa struktur kontrol (iteratif)

```
func(n,k){
int n=2^(2^k);
for(i=1, i<= n, i=++)
   {
      j=2;
      while(j<=n)
       {
         j=j^2
         printf("analgor");
      }
   }
}
```

Berapa kali perintah cetak "analgor" yang dieksekusi?

References

https://www.youtube.com/watch?v=FEnwM-iDb2g