Pertemuan 10
Algoritma dan Hubungan Rekurensi

Algoritma

Algoritma

adalah suatu langkah-langkah logis untuk menyelesaikan masalah. Bedanya kalau algoritma setiap langkah difokuskan pada sistem komputer dan data.

Beberapa hal yang harus dipahami dalam mencari algoritma antara lain:

- Masalah seperti apa yang hendak diselesaikan?
- Gagasan apa yang ada pada algoritma tersebut?
- Berapa lama yang diperlukan untuk menyelesaikan masalah?
- Berapa jumlah data yang dapat ditangani oleh algoritma tersebut?

Untuk mengetahui seberapa besar kualitas suatu algoritma, dinyatakan dengan notasi-O besar (big O-notation) untuk menyatakan tingakat kekomplekan suatu algoritma.

Notasi O-besar

Notasi O-besar

Menyatakan kelompok kompleksitas suatu algoritma atau jumlah operasi yang dilakukan oleh algoritma.

Tabel Kompleksitas dengan

Notasi O-besar

Kelompok Algoritma	Nama
O(1)	Konstan
O(log n)	Logaritmik
O(n)	Linier
O(nlogn)	N log n
$O(n^2)$	Kuadratik
$O(n^3)$	Kubik
O(2n)	Eksponensial
O(n!)	Faktorial

Contoh:

Jenis Algoritma	Kompleksitas
Linier search	O(n)
Binary search	O(log n)
Bubble sort	$O(n^2)$
Selection sort	$O(n^2)$
Insertion sort	$O(n^2)$
Merge sort	O(n log n)
Quick sort	O(n log n)
Heap sort	O(n log n)

Teorema

Definisi:

T(n) = O(f(n)) (T(n) adalah O(f(n)) yang artinya T(n) berorde paling besar f(n)) bila terdapat C dan n0 sedemikian hingga:

$$T(n) \leq C(f(n))$$

Untuk $n \geq n0$, artinya jika sebuah algoritma mempunyai waktu asimtotik O(f(n)), maka jika n dibuat semakin besar waktu yang dibutuhkan tidak akan pernah melebihi suatu tetapan C dikali f(n). Jadi f(n) adalah batas atas dari T(n) untuk n yang besar. T(n) dikatakan berorde paling besar f(n).

Penjelasan masing-masing kelompok algoritma sebagai berikut:

O(1) berarti waktu untuk menjalankan adalah konstan, artinya tidak bergantung pada ukuran data masukan n. Ini berarti waktu tetap sama meskipun n menjadi dua kali semula atau lebih.

O(log n) berarti perubahan waktu akan lebih kecil dari perubahan masukan n. Algoritma yang masuk dalam kelompok ini adalah algoritma yang memecahkan persoalan besar dengan membagi masalah besar menjadi masalah yang kecil yang berukuran sama. Cont: Binary search. Di sini bilangan pokok/basis log tidak terlalu penting karena semua fungsi algoritmik meningkat 2 kali semula jika n dinaikkan menjadi n^2.

O(n) algoritma yang waktu bergantung kepada n secara linier, artinya jika n dinaikkan sebesar x kali maka waktu menjalankan algoritma itu juga naik sebesar x kali. Umumnya algoritma pencarian beruntun

O(n log n) Waktu pelaksanaan n log n terdapat pada algoritma yang memecahkan masalah menjadi beberapa bagian yang lebih kecil dimana setiap persoalan diselesaikan secara independen dan akhirnya solusi masing-masing digabung. Cont teknik bagi dan. Bila n dinaikkan menjadi 2 kali maka n log n meningkat tidak sampai dua kalinya, tetapi kalau n dinaikkan lebih dari 10 kali maka n log n naik lebih cepat dari kenaikkan n

$O = n^2$

algoritma ini akan berjalan lebih lambat dari algoritma linier maupun logaritmik, sehingga hanya praktis untuk persoalan yang berukuran kecil. Bila n dinaikkan dua kali maka waktu pelaksanaan algoritma meningkat menjadi empat kali karena prose setiap masukkannya dalam dua buah kalang bersarang.

$O = n^3$ memproses setiap masukkan dalam tiga buah kalang bersarang, misal perkalian matrik. Algoritma ini akan menjadi lebih lambat dari algoritma kuadratik meskipun n kecil atau besar. Bila n dinaikkan dua kali maka waktu pelaksanaan algoritma meningkat menjadi delapan kali.

O(2n) algoritma ini berjalan lebih lambat dari semua algoritma sebelumnya, khususnya untuk n besar. Bila n dinaikkan dua kali, waktu pelaksanaan menjadi kuadrat kali tetapi jika n= 100, maka waktu pelaksanaannya adalah 1000000 sama dengan algoritma .

O(n!) jenis ini memproses setiap masukkan dan menghubungkannya dengan n-1 masukkan lainnya, misal algoritma persoalan pedagang keliling. Bila n = 5 maka waktu pelaksanaan algoritma 120. Bila n dinaikkan dua kali, maka waktu pelaksanaan menjadi 2n!

Algoritma pencarian

Algoritma Pencarian

1. Pencarian beruntun (*sequential search*)
2. Pencarian Biner (*binary search*)

Pencarian beruntun pada array yang tidak terurut

Proses dimulai dari data pertama sampai data terakhir / data ke-n

Berikut ini algoritmanya:

```
Input ( cari)    { meminta data nilai yang akan dicari}
   I=1           { indeks array dimulai dari 1}
   while (I<n) and (A[I] ≠ cari) do
         I = I+1
  end while
  if A[I] = cari then
            output   (‘Data berada di index nomor’, I)
  else      output (‘Data tidak ditemukan’)
   endif
```


Algoritma pencarian beruntun yang sudah terurut

Pencarian beruntun pada Array yang sudah Terurut

Kasus terburuk dari pencarian beruntun kalau data tidak ditemukan atau terletak paling akhir. Kalau data sudah terurut maka akan lebih baik kalau proses pencarian dihentikan apabila nilai data yang dibandingkan sudah lebih besar dari nilai data yang dicari.

Berikut ini algoritmanya:

```
Input ( cari)    { meminta data nilai yang akan dicari}
   I=1           { indeks array dimulai dari 1}
   while (I<n) and (A[I] < cari) do
         I = I+1
  end while
  if A[I] = cari then
            output   ('Data berada di index nomor', I)
  else      output ('Data tidak ditemukan')
  endif
```


Pencarian biner

Pencarian Biner

Dilakukan untuk memperkecil jumlah operasi perbandingan yang harus dilakukan antara data yang dicari dengan data yang ada di dalam tabel, khususnya untuk data yang besar.

Prinsip dasarnya membagi 2 data sampai data ditemuakan.

Algoritmanya:

```
 Input (cari)   {meminta niali data yang akan dicari}
 Batasatas = 1    {indeks array dimulai dari 1}
Batasbawah = n
Ketemu = false
While (batasatas < batasbawah) and (not ketemu) do
Tengah = (batasatas+batasbawah)div2
If A[tengah] = cari then ketemu = true
  else if (A[tengah] <cari) then { cari dibagian kanan}
          batasatas = tengah + 1
```


else batasbawah = tengah-1 {cari dibagian kiri}
endif
endif
endwhile

Algoritma Pengurutan

if (ketemu) then
output (‘ data berada di index nomor’, tengah)
else output (‘data tidak ditemukan’)
endif

Algoritma Pengurutan

Yang akan dibahas meliputi bubble sort, selection sort, dan insertion sort

1. Bubble Sort

menggunakan prinsip kerja gelembung udara

Pengurutan dengan bubble sort

Algoritma Bubble sort

```
For I = 1 to (n-1) do
    for J = N downto (I+1) do
        if data [J] < data [J-1] then
            Bubble = data [J]
            data [J] = data [J-1]
            data [J-1] = bubble
        endif
    endfor
endfor
```

Pengurutan dengan Selection sort

2. Selection Sort

salah satu metode pengurutan berdasarkan prioritas antrian.

Algoritma Selection Sort

```
For I = 1 to (n-1) do
        for J = (I+1) to N do
            if data [J] < data [kecil] then
                kecil = J
            endif
                endfor
                sementara = data[I]
    data[I] = data[kecil]
    data[kecil] = sementara
        endfor
```

Pengurutan dengan Insertion Sort

3. Insertion Sort

Metode ini dilakukan dengan penyisipan nilai data untuk suatu aaray A yang tidak terurut ke dalam suatu tempat kosong C dan memastikan nilai data C selalu dalam keadaan terurut.

Algoritma Insertion sort

```
For I = 2 to N do
        sementara = data[I]
         j = I – 1
       while (sementara < data[J]) and (J>1) do
             data [J+1] = data [J]
              J = J – 1
         endwhile
         if sementara ≥ data[J] then
            data[J+1] = sementara
         else data [J+1] = data [J]
                 data [J] = sementara
        endif
             endfor
```

Hubungan Rekurensi

Hubungan Rekurensi

Sebuah hubungan rekurensi untuk urutan a0, a1,…adalah sebuah persamaan yang menghubungkan a0 dengan suku tertentu pendahulunya a0,a1,…,an-1. Kondisi awal untuk urutan a0,a1,… secara eksplisit memberikan nilai kepada sejumlah suku-suku tertentu dalam urutan itu.

Rekurensi adalah proses perulangan yaitu memanggil dirinya sendiri (fungsi/prosedur) dengan parameter yang berbeda, sampai pengulangan berhenti.

Cara lain menyelesaikan rekurensi adalah:

1. Memecahkan masalah yang besar menjadi dua submasalah.
2. Submasalah tersebut diselesaikan dengan cara yang sama untuk memecahkan masalah yang besar tersebut.

Permasalahan yang bisa menggunakan metode rekurensi

Permasalahan yang menggunakan metode rekuren diantaranya:

1. Faktorial
2. Fibonanci
3. Menara Hanoi

Soal-soal latihan

Soal 1 dan 2

1. Langkah-langkah logis untuk menyelesaikan masalah disebut dengan……..

   a. Algoritma
   b. Notasi O-besar
   c. Rekurensi
   d. Fungsi
   e. Relasi

2. Permasalahan pencarian biner akan optimal jika menggunakan algoritma waktu….

   a. O(1)
   b. O(n!)
   c. O(log n)
   d. O(n log n)
   e. O(n)

Soal 2 dan 3

2. Permasalahan pencarian biner akan optimal jika menggunakan algoritma waktu….

   a. O(1)
   b. O(n!)
   c. O(log n)
   d. O(n log n)
   e. O(n)

3. Jenis cara pencarian data ada…..

   a. 1 b. 2 c. 4 d. 5 e. 6

Soal 3 dan 4

3. Jenis cara pencarian data ada…..

   a. 1 b. 2 c. 4 d. 5 e. 6

4. Data 27 80 02 46 16 12 50

   jika diurutkan dengan metode seleksi maka pada langkah ke-3 diperoleh urutan……

   a. 02 80 27 46 16 12 50

   b. 02 12 27 46 16 80 50

   c. 02 12 16 46 27 80 50

   d. 02 12 16 27 46 80 50

   e. 02 12 16 27 46 80 50

Soal 4 dan 5

4. Data 27 80 02 46 16 12 50

   jika diurutkan dengan metode seleksi maka pada langkah ke-3 diperoleh urutan……

   a. 02 80 27 46 16 12 50

   b. 02 12 27 46 16 80 50

   c. 02 12 16 46 27 80 50

   d. 02 12 16 27 46 80 50

   e. 02 12 16 27 46 80 50

5. Kemampuan memanggil dirinya sendiri dengan parameter berbeda sampai pengulangan berhenti disebut……

   a. Algoritma

   b. Notasi O-besar

   c. Relasi

   d. Fungsi

   e. Rekurensi

Soal 5 dan 1

5. Kemampuan memanggil dirinya sendiri dengan parameter berbeda sampai pengulangan berhenti disebut……

   a. Algoritma
   b. Notasi O-besar
   c. Relasi
   d. Fungsi
   e. Rekurensi

1. Langkah-langkah logis untuk menyelesaikan masalah disebut dengan……..

   a. Algoritma
   b. Notasi O-besar
   c. Rekurensi
   d. Fungsi
   e. Relasi