**WOKSHEET III**

**ANALISIS ALGORITMA**

****

**Disusun oleh :**

Hanif Dwi Prasetiyo

140810180035

**PROGRAM STUDI S1 TEKNIK INFORMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS PADJADJARAN**

**2020**

Pendahuluan

Minggu lalu kita sudah mempelajari menghitung kompleksitas waktu T(n) untuk semua operasi yang ada pada suatu algoritma. Idealnya, kita memang harus menghitung semua operasi tersebut. Namun, untuk alasan praktis, kita cukup menghitung operasi abstrak yang mendasari suatu algoritma, dan memisahkan analisisnya dari implementasi. Contoh pada algoritma searching, operasi abstrak yang mendasarinya adalah operasi perbandingan elemen x dengan elemen-elemen dalam larik. Dengan menghitung berapa perbandingan untuk tiap-tiap elemen nilai n sehingga kita dapat memperoleh efisiensi relative dari algoritma tersebut. Setelah mengetahui T(n) kita dapat menentukan kompleksitas waktu asimptotik yang dinyatakan dalam notasi Big-O, Big-Ω, Big-Θ, dan little-ω.

Setelah mengenal macam-macam kompleksitas waktu algoritma (best case, worst case, dan average case), dalam analisis algoritma kita selalu mengutamakan perhitungan worst case dengan alasan sebagai berikut:

• Worst-case running time merupakan upper bound (batas atas) dari running time untuk input apapun. Hal ini memberikan jaminan bahwa algoritma yang kita jalankan tidak akan lebih lama lagi dari worst-case

• Untuk beberapa algoritma, worst-case cukup sering terjadi. Dalam beberapa aplikasi pencarian, pencarian info yang tidak ada mungkin sering dilakukan.

• Pada kasus average-case umumnya lebih sering seperti worst-case. Contoh: misalkan kita secara random memilih angka dan mengimplementasikan insertion sort, average-case = worst-case yaitu fungsi kuadratik dari .

Perhitungan worst case (upper bound) dalam kompleksitas waktu asimptotik dapat menggunakan Big-O Notation. Perhatikan pembentukan Big-O Notation berikut!

Misalkan kita memiliki kompleksitas waktu T(n) dari sebuah algoritma sebagai berikut:

* Untuk yang besar, pertumbuhan sebanding dengan
* Suku tidak berarti jika dibandingkan dengan 2 , dan boleh diabaikan sehingga T(n) = 2 + suku-suku lainnya.
* Koefisien 2 pada 2 boleh diabaikan, sehingga T(n) = O() -> Kompleksitas Waktu Asimptotik

**DEFINISI BIG-O NOTATION**

Definisi 1. artinya berorde paling besar bila terdapat konstanta C dan sedemikian sehingga

Untuk ≥

Jika dibuat semakin besar, waktu yang dibutuhkan tidak akan melebihi konstanta C dikalikan dengan , -> adalah upper bound.

Dalam proses pembuktian Big-O, perlu dicari nilai dan nilai C sedemikan sehingga terpenuhi kondisi .

Contoh soal 1:

Tunjukan bahwa,

**Penyelesaian:**

Kita mengamati bahwa ≥ 1, maka ≤ dan 1≤ sehingga

Maka kita bisa mengambil C=9 dan =1 untuk memperlihatkan:

**BIG-O NOTATION DARI POLINOMIAL BERDERAJAT M**

Big-O Notation juga dapat ditentukan dari Polinomial n berderajat m, dengan TEOREMA 1 sebagai berikut:

Polinomial berderajat dapat digunakan untuk memperkirakan kompleksitas waktu asimptotik dengan mengabaikan suku berorde rendah

Contoh: , dinyatakan pada

**TEOREMA 1**

Bila adalah polinom berderajat m maka

Artinya kita mengambil suku paling tinggi derajatnya (“Mendominasi”) yang diartikan laju pertumbuhannya lebih cepat dibandingkan yang lainnya ketika diberikan sembarang besaran input. Besaran dominan lainnya adalah:

* Eksponensial mendominasi sembarang perpangkatan (yaitu, )
* Perpangkatan mendominasi ln (yaitu, )
* Semua logaritma tumbuh pada laju yang sama (yaitu
* log tumbuh lebih cepat daripada tetapi lebih lambat dari

Teorema lain dari Big-O Notation yang harus dihafalkan untuk membantu kita menentukan nilai Big-O dari suatu algoritma adalah:

**TEOREMA 2**

Misalkan

(b)

(c)

(d)

Berikut adalah contoh soal yang mengaplikasikan Teorema 2 dari Big-O notation:

Contoh Soal 2

Misalkan,

1. ……………………………………Teorema 2(a)(i)
2. …………………………………………………………..Teorema 2(a(ii)
3. ……………………………………………………..Teorema 2(b)

Contoh Soal 3

1. ………………………………………………………………………………..Teorema 2(c)
2. ………………………………………………………………………………………..Teorema 2(d)

Aturan Menentukan Kompleksitas Waktu Asimptotik

* Cara I

Jika kompleksitas waktu T(n) dari algoritma sudah dihitung, maka kompleksitas waktu asimptotiknya dapat langsung ditentukan dengan mengambil suku yang mendominasi fungsi T dan menghilangkan koefisiennya (sesuai TEOREMA 1)

**Contoh**:

Pada algoritma cariMax,

* Cara 2

Kita bisa langsung menggunakan notasi Big-O, dengan cara: Pengisian nilai (assignment), perbandingan, operasi aritmatika (+,-,/,\*, div, mod), read, write, pengaksesan elemen larik, memilih field tertentu dari sebuah record, dan pemanggilan function/void membutuhkan waktu O(1)

Contoh Soal 4:

Tinjau potongan algoritma berikut:

read(x) O(1)

x x + 1 O(1) + O(1)=O(1)

write(x) O(1)

Kompleksitas waktu asimptotik algoritmanya ( 1)+( 1)+( 1)=( 1)

Penjelasan:

Teorema 2(a)(i)

…………………………………

……………………………….. Teorema 2(a)(ii)

………………………………..

**DEFINISI BIG-Ω DAN BIG-Θ NOTATION**

Notasi Big-O hanya menyediakan batas atas (upper bound) untuk perhitungan kompleksitas waktu asimptotik, tetapi tidak menyediakan batas bawah (lower bound). Untuk itu, lower bound dapat ditentukan dengan Big-Ω Notation dan Big-θ Notation.

**Definisi Big-Ω Notation:**

yang artinya berorde paling kecil bila terdapat konstanta C dan sedemikian sehingga

Untuk ≥

**Definisi Big-θ Notation:**

**Contoh Soal 5:**

Tentukan Big-Ω dan Big- Θ Notation untuk

Penyelesaian:

Karena ≥2 untuk ≥ 1, dengan mengambil C=2, kita memperoleh

Karena dan , maka

**Penentuan Big-Ω dan Big- dari Polinomial Berderajat m**

Sebuah fakta yang berguna dalam menentukan orde kompleksitas adalah dari suku tertinggi di dalam polinomial berdasarkan teorema berikut:

**TEOREMA 3**

Bila adalah polinom berderajat m maka

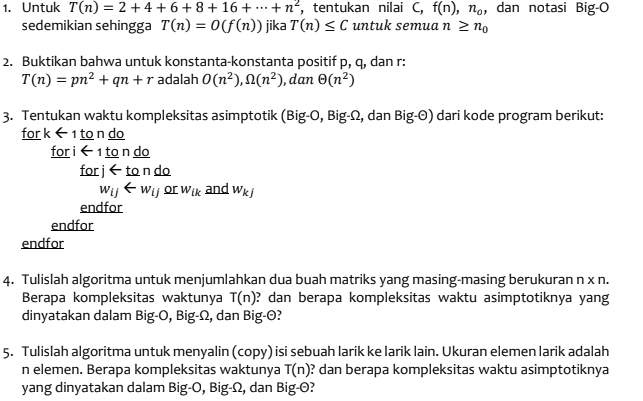
Contoh soal 6:

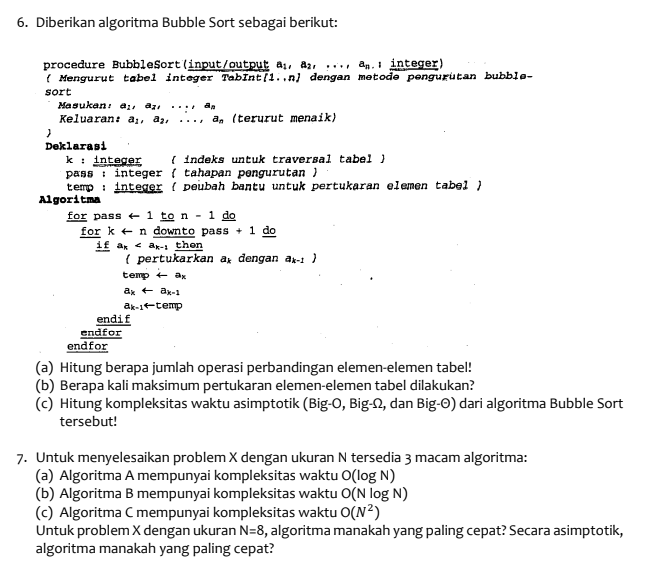
Bila

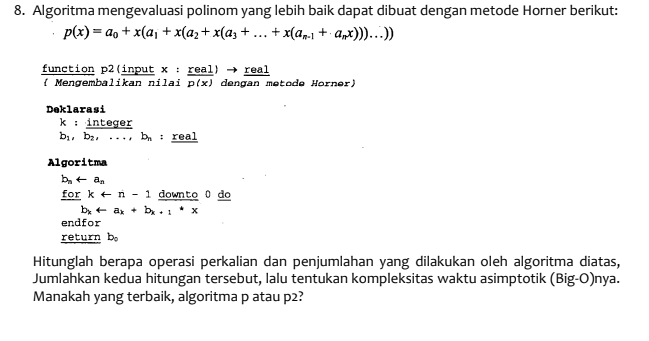
maka T(n) adalah berorde , yaitu

**Latihan Analisa**

Minggu ini kegiatan praktikum difokuskan pada latihan menganalisa, sebagian besar tidak perlu menggunakan komputer dan mengkoding program, gunakan pensil dan kertas untuk menjawab persoalan berikut!



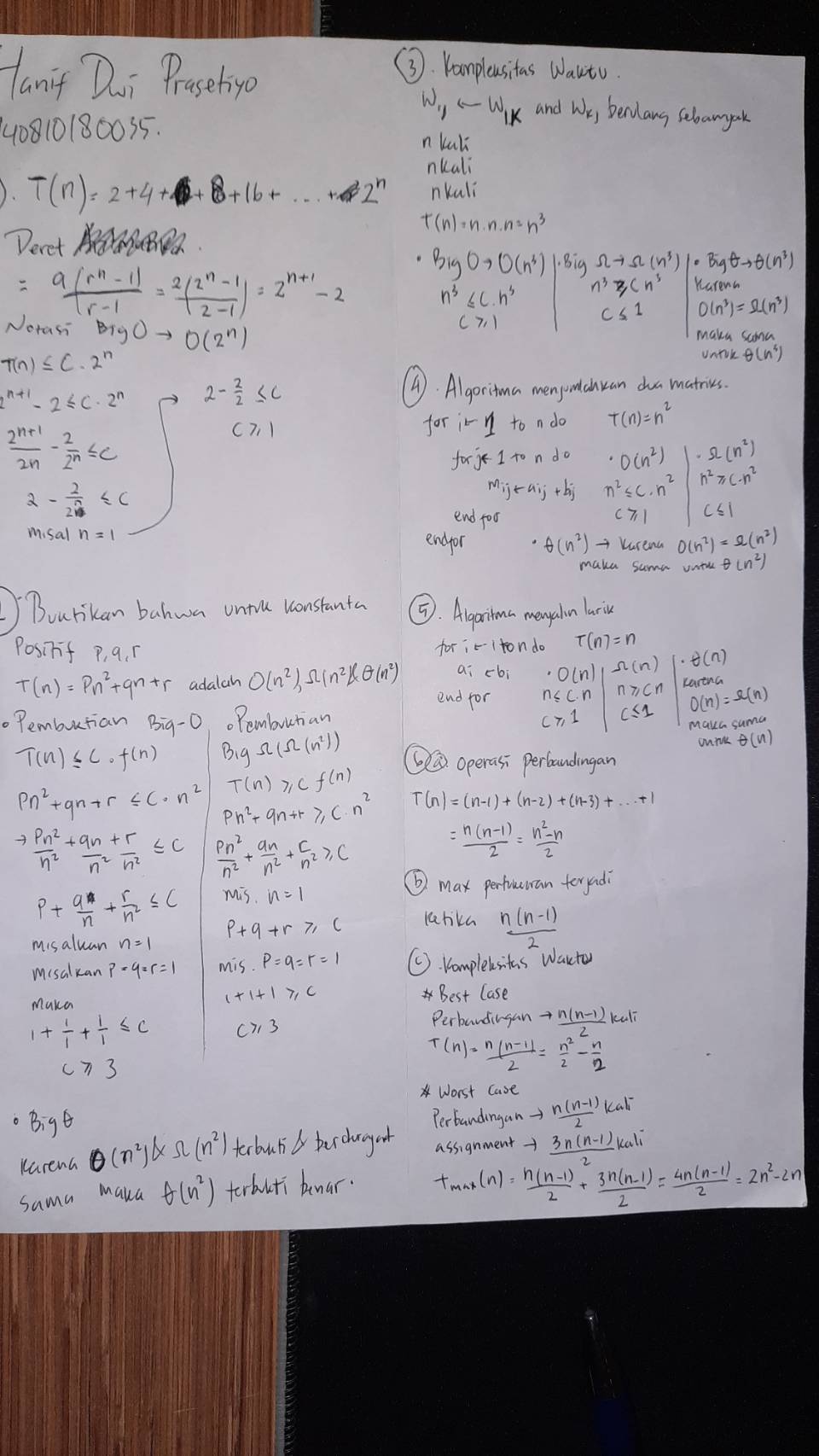


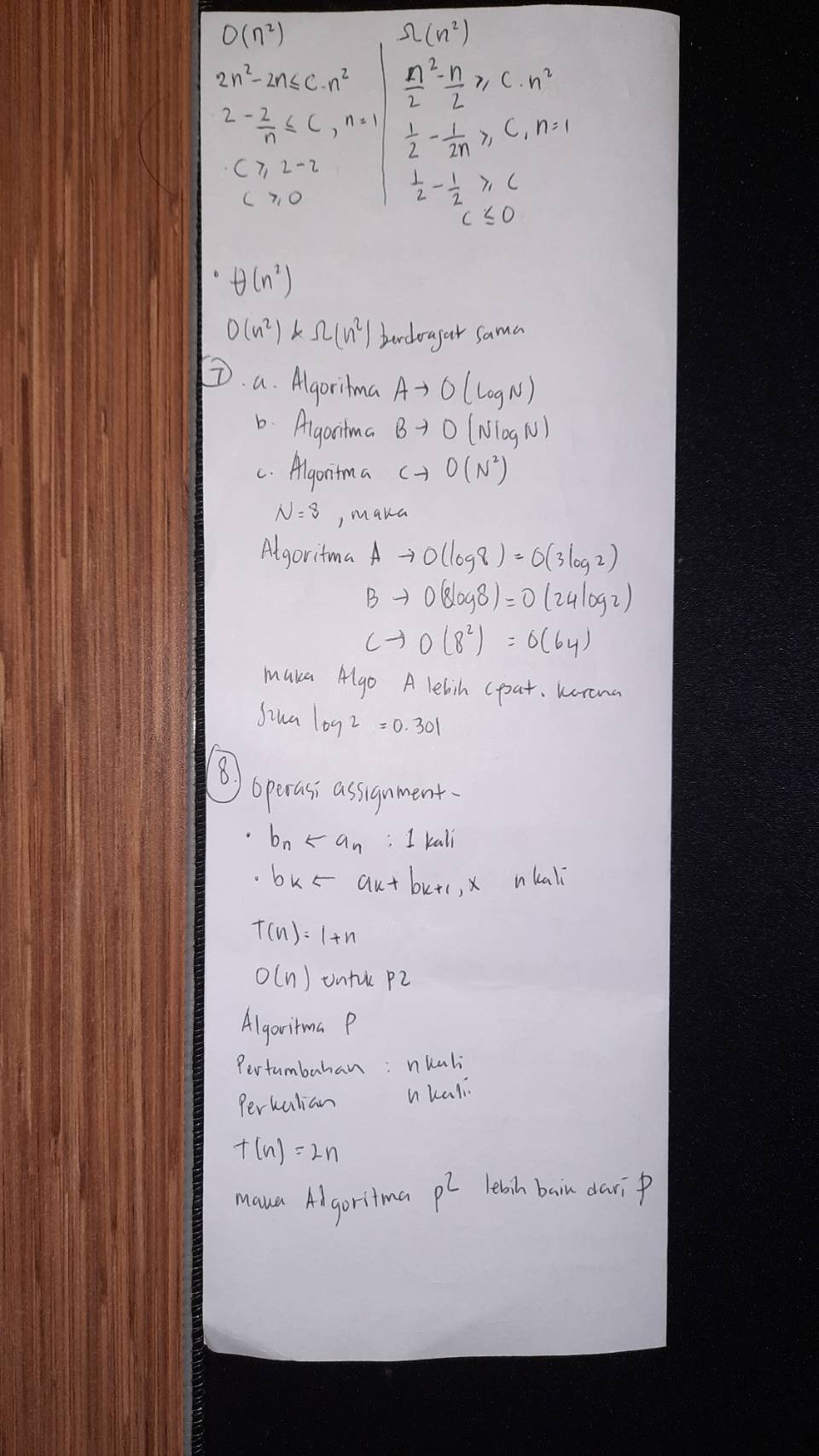


Teknik Pengumpulan

* Semua jawaban ditulis di kertas dan dikumpulkan ke asisten praktikum pada akhir praktikum

Jawaban Latihan Analisa





Penutup

* Ingat, berdasarkan Peraturan Rektor No 46 Tahun 2016 tentang Penyelenggaraan Pendidikan, mahasiswa wajib mengikuti praktikum 100%
* Apabila tidak hadir pada salah satu kegiatan praktikum segeralah minta tugas pengganti ke asisten praktikum
* Kurangnya kehadiran Anda di praktikum, memungkinkan nilai praktikum Anda tidak akan dimasukkan ke nilai mata kuliah.