LAPORAN PRAKTIKUM 2

ANALISIS ALGORITMA



DISUSUN OLEH:

NAMA : RAISSA AMINI

NPM : 140810180073

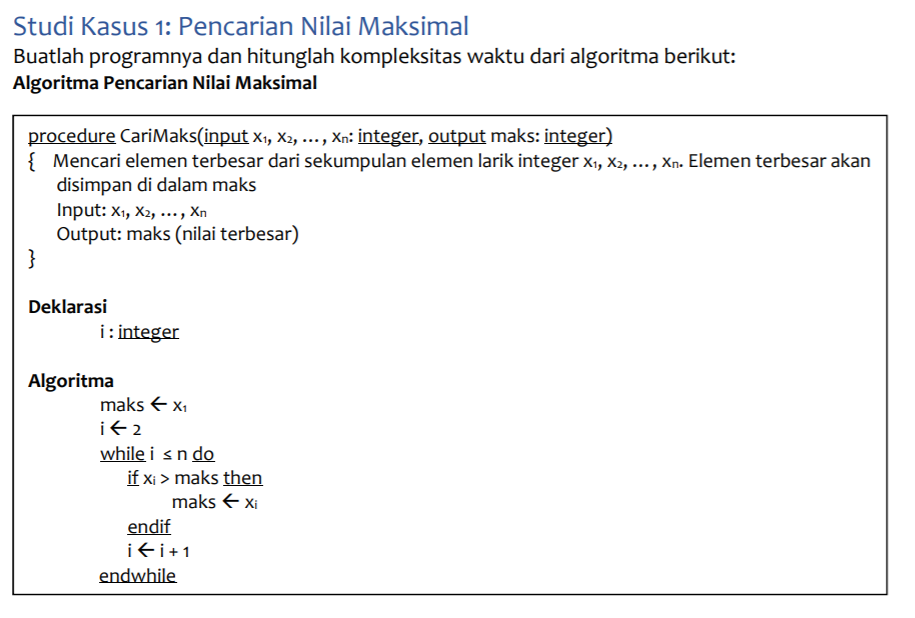
Program Studi S-1 Teknik Informatika

Departemen Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Padjadjaran

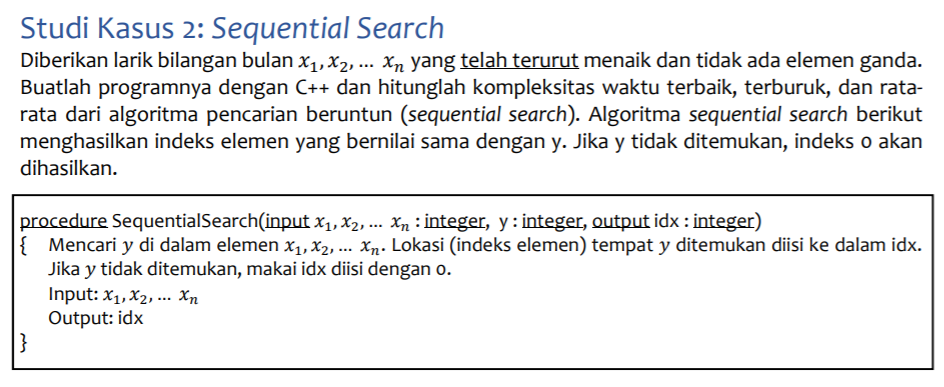
2018

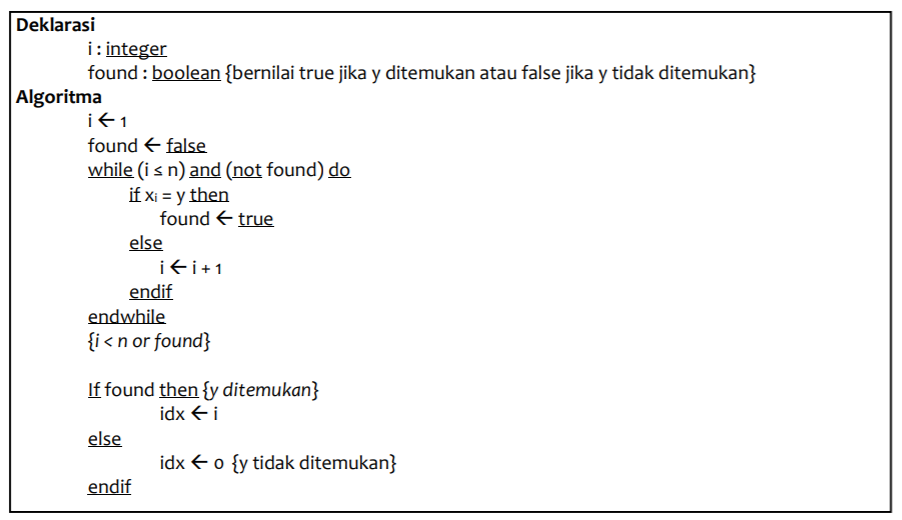


Jawaban Studi Kasus 1:

T(n) = 2(n – 2) + (n – 2) + 2

= 3 n - 4





Jawaban Studi Kasus 2 :

1. Kompleksitas waktu terbaik (best case) : ini terjadi bila a1 = x.

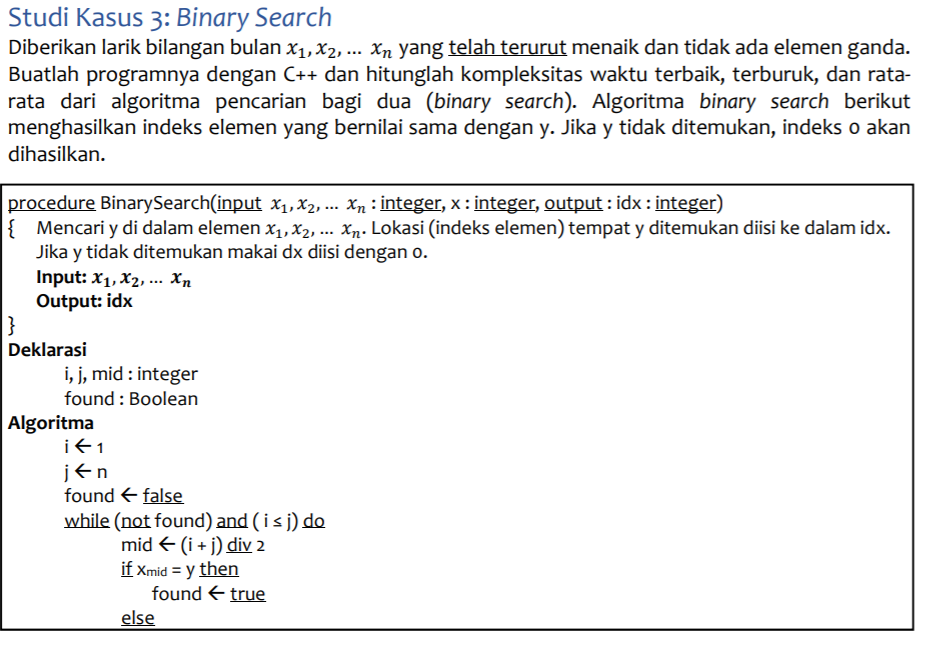
Tmin(n) = 1

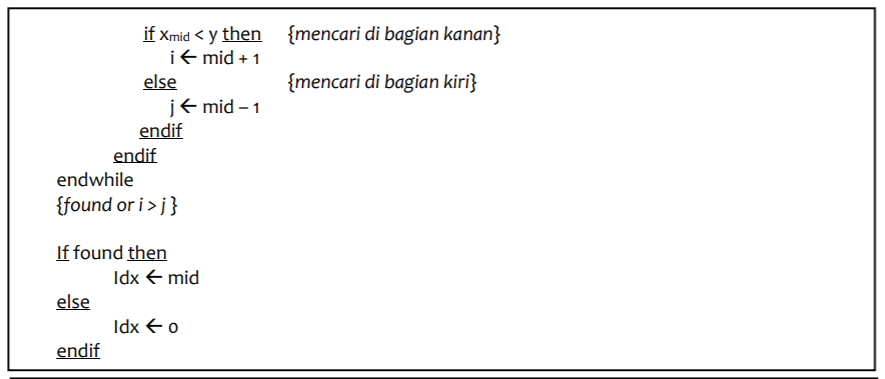
1. Kompleksitas waktu terburuk (worst case) : bila an = x atau x tidak ditemukan.

Tmax(n) = n

1. Kasus kompleksitas waktu rata-rata : Jika *x* ditemukan pada posisi ke-*j*, maka operasi perbandingan (ak = x) akan dieksekusi sebanyak *j*kali.

Tavg(n) = 





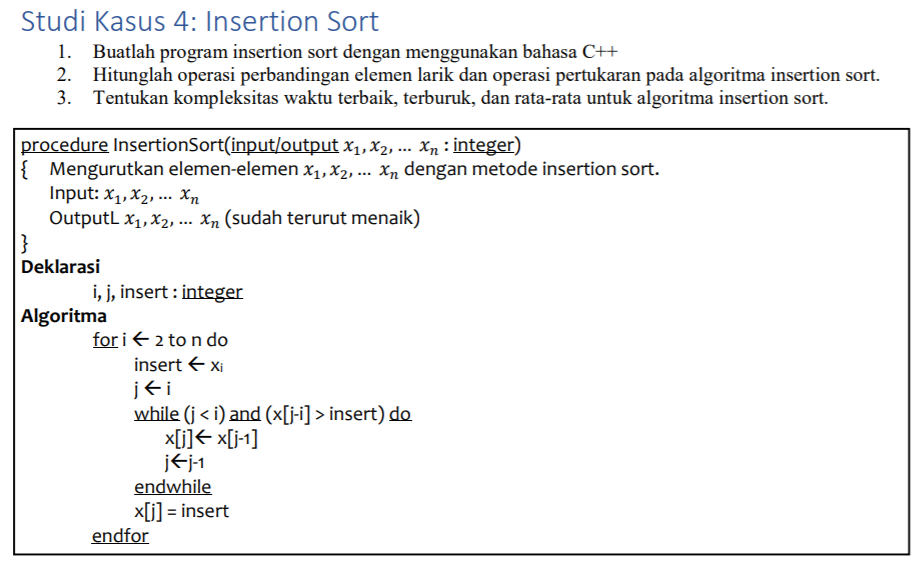
Jawaban Studi Kasus 3 :

* Kasus waktu terbaik (Best case) :

Tmin(n) = 1

* Kasus waktu rata-rata : jika terdapat index pada awal dan akhir elemen.
* Kasus waktu terburuk (Worst case) :

Tmax (n) = 2log n



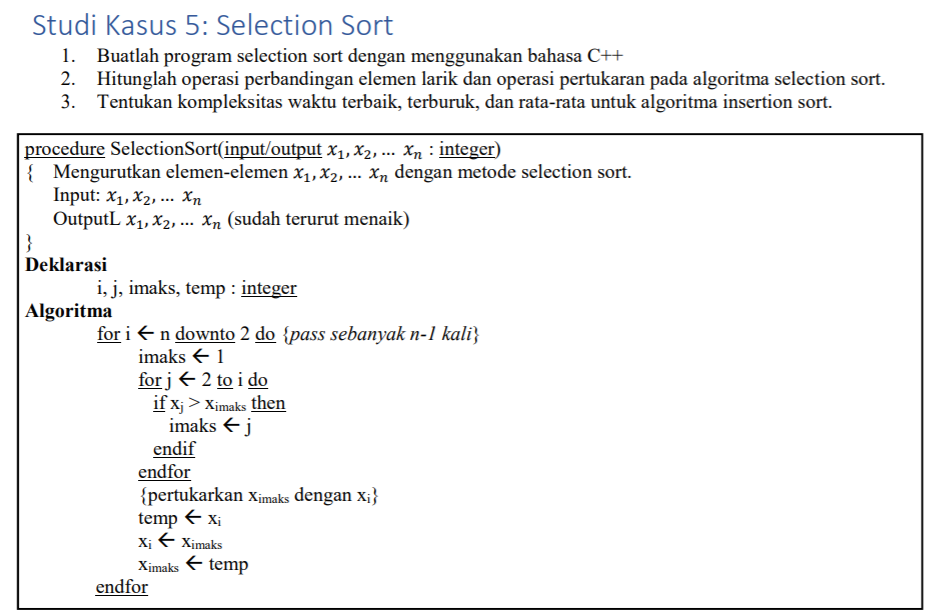
Jawaban Studi Kasus 4 :

* Kasus waktu terbaik (Best case) : jika array yang ada sudah terurut dengan benar, jadi looping tidak akan dilakukan lagi.
* Kasus waktu rata2 : jika array elemen yang ada sudah terurut setengahnya / sebagian dari seluruh elemen

Loop sementara dijalankan hanya jika i> j dan arr [i] <arr [j]. Jumlah total iterasi loop sementara (Untuk semua nilai i) sama dengan jumlah inversi.

Kompleksitas waktu keseluruhan dari jenis penyisipan adalah O (n + f (n)) di mana f (n) adalah jumlah inversi.Jika jumlah inversi adalah O(n), maka kompleksitas waktu dari jenis penyisipan adalah O(n)..

* Dalam kasus terburuk, bisa ada inversi n \* (n-1) / 2. Kasus terburuk terjadi ketika array diurutkan dalam urutan terbalik. Jadi kompleksitas waktu kasus terburuk dari jenis penyisipan adalah O (n2).



Jawaban Studi Kasus 5 :

1. Jumlah operasi perbandingan element. Untuk setiap *pass* ke-*i*,

*i*= 1 –>  jumlah perbandingan  = *n* – 1

*i* = 2 –>  jumlah perbandingan = *n* – 2

*i*= 3  –> jumlah perbandingan = *n* – 3

:

*i* = *k* –>  jumlah perbandingan = *n* – *k*

:

*i*= *n* – 1  –> jumlah perbandingan = 1

Jumlah seluruh operasi perbandingan elemen-elemen larik adalah    *T*(*n*) = (*n* – 1) + (*n* – 2) + … + 1 = 

Ini adalah kompleksitas waktu untuk kasus terbaik dan terburuk, karena algoritma Urut tidak bergantung pada  batasan apakah data masukannya sudah terurut atau acak.

1. Jumlah operasi pertukaran

Untuk setiap*i* dari 1 sampai *n* – 1, terjadi satu kali pertukaran elemen, sehingga jumlah operasi pertukaran seluruhnya   adalah  *T*(*n*) = *n* – 1.

Jadi, algoritma pengurutan maksimum membutuhkan *n*(*n* – 1 )/2 buah operasi perbandingan elemen dan *n* – 1  buah operasi pertukaran.