**Analisis Algoritma**

Laporan Praktikum 1



­­

**Dibuat oleh:**

Muhammad Iqbal Alif Fadilla

140810180020

**Universitas Padjadjaran**

**Fakultas Matematika dan Ilmu Pengetahuan**

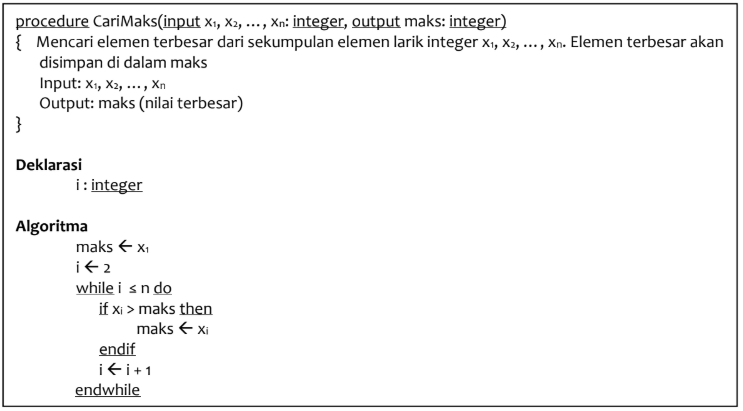
**2019**

**Worksheet 2**

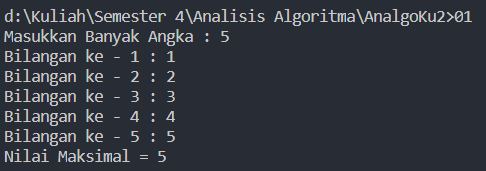
1. Studi Kasus 1 : Pencarian Nilai Maksimal

Buatlah programnya dan hitunglah kompleksitas waktu dari algoritma berikut:

**Algoritma Pencarian Nilai Maksimal**



1. #include <iostream>
2. **using** **namespace** std;
3. **int** main()
4. {
5. **int** arr[100];
6. **int** n, maks, i;
8. cout << "Masukkan Banyak Angka : ";
9. cin >> n;
10. **for** (**int** i = 0; i < n; i++)
11. {
12. cout << "Bilangan ke - " << i + 1 << " : ";
13. cin >> arr[i];
14. }
15. maks = arr[0];
16. **for** (**int** i = 0; i < n; i++)
17. {
18. **if** (arr[i] > maks)
19. maks = arr[i];
20. }
21. cout << "Output = " << maks << endl;
22. }



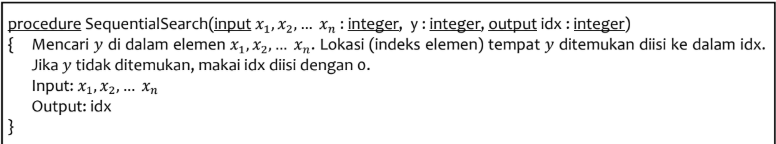
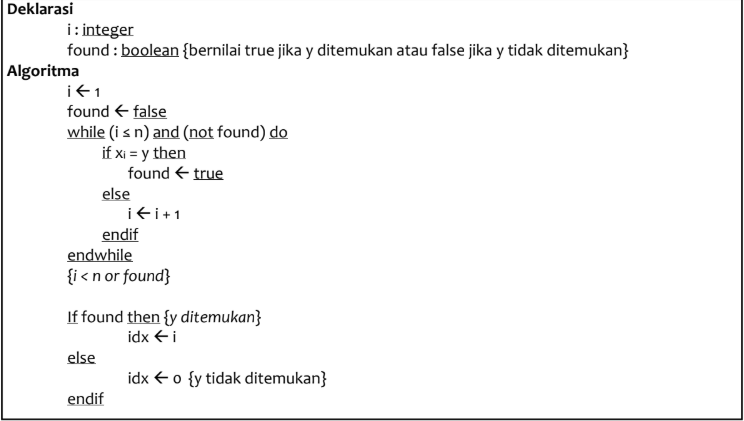
* Operasi Assignment = 1 + 1 + (n-1) + (n-1) = 2n
* Operasi Perbandingan = n-1
* Operasi Penjumlahan = n-1

Maka Tmax(n) = 4n-2

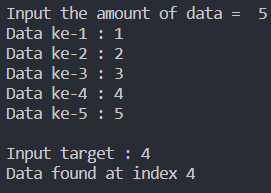
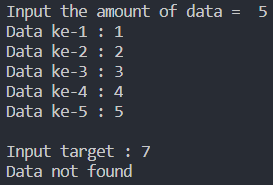
1. Studi Kasus 2 : Sequential Search

Diberikan larik bilangan bulan x1, x2, …, xn yang telah terurut menaik dan tidak ada elemen ganda.

Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma pencarian beruntun (sequential search). Algoritma sequential search berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.

1. #include <iostream>
2. **using** **namespace** std;
4. **int** main()
5. {
6. **int** n, target, arr[100], index;
7. **bool** found = **false**;
8. cout << "\nInput the amount of data =  ";
9. cin >> n;
11. **for** (**int** i = 0; i < n; i++)
12. {
13. cout << "Data ke-" << i + 1 << " : ";
14. cin >> arr[i];
15. }
17. cout << "\nInput target : ";
18. cin >> target;
20. **for** (**int** i = 0; i < n; i++)
21. {
22. **if** (arr[i] == target)
23. {
24. found = **true**;
25. index = i;
26. i = n;
27. }
28. }
29. **if** (found == **true**)
30. {
31. cout << "Data found at index " << index + 1 << endl;
32. }
33. **else**
34. {
35. cout << "Data not found" << endl;
36. }
37. **return** 0;
38. }

**Tmin(n) = 4 + 2 = 6**

* Operasi Assignment = 4
* Operasi Perbandingan = 2

**Tmax(n) = 3+n+n+1+n = 3n+4**

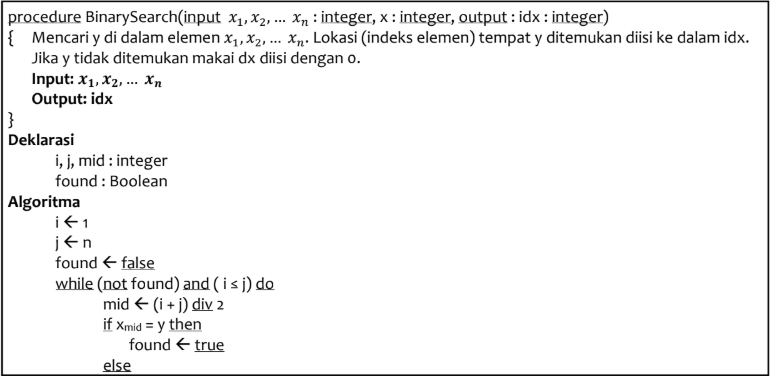
* Operasi Assignment = 1 + 1 + n + 1 = 3 + n
* Operasi Perbandingan = n + 1
* Operasi Penjumlahan = n

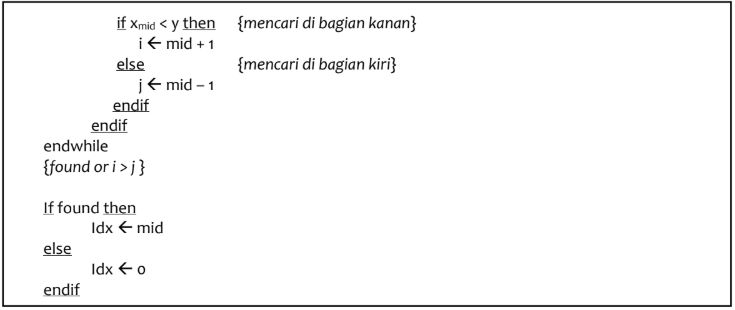
**Tavg(n) = (10+3n) / 2**

* (Tmin(n) + Tmax(n)) / 2 = (6+4+3n) / 2 = (10+3n) / 2

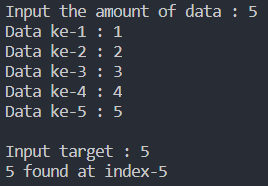
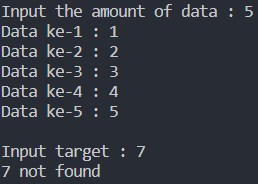
1. Studi Kasus 3 : Binary Search

Diberikan larik bilangan bulan x1, x2, ... xn yang telah terurut menaik dan tidak ada elemen ganda. Buatlah programnya dengan C++ dan hitunglah kompleksitas waktu terbaik, terburuk, dan rata-rata dari algoritma pencarian bagi dua (binary search). Algoritma binary search berikut menghasilkan indeks elemen yang bernilai sama dengan y. Jika y tidak ditemukan, indeks 0 akan dihasilkan.





1. #include <iostream>
2. **using** **namespace** std;
4. **int** main()
5. {
6. **int** n, i, arr[100], target, low, high, mid;
7. cout << "\nInput the amount of data : ";
8. cin >> n;
9. **for** (i = 0; i < n; i++)
10. {
11. cout << "Data ke-" << i + 1 << " : ";
12. cin >> arr[i];
13. }
14. cout << "\nInput target : ";
15. cin >> target;
16. low = 0;
17. high = n - 1;
18. **while** (low <= high)
19. {
20. mid = (low + high) / 2;
21. **if** (arr[mid] < target)
22. {
23. low = mid + 1;
24. }
25. **else** **if** (arr[mid] == target)
26. {
27. cout << target << " found at index-" << mid + 1 << endl;
28. **break**;
29. }
30. **else**
31. {
32. high = mid - 1;
33. }
34. mid = (low + high) / 2;
35. }
36. **if** (low > high)
37. {
38. cout << target << " not found" << endl;
39. }
40. **return** 0;
41. }

**Tmin(n) = 6 + 2 = 8**

* Operasi Assignment = 6
* Operasi Perbandingan = 2

**Tmax(n) = (log 2(n))**

Panjang array akan berubah pada setiap iterasi:

* Iterasi 1 = n
* Iterasi 2 = n/2
* Iterasi 3 = n/22
* Iterasi x = n/2k-1 ~ n/2k *(-1 diabaikan karena kecil dibanding n/2k)*

Panjang array menjadi 1.

Maka,

n/2k = 1

n = 2k

log 2(n) = log 2(2k) = k log 2(2)

k = log 2(n)

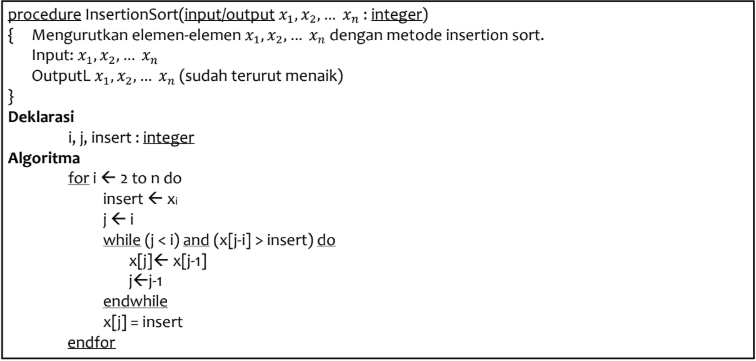
Sehingga

**Tavg(n) = (1 + log 2(n)) / 2**

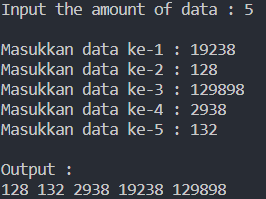
(Tmin(n) + Tmax(n)) / 2 = (1 + log 2(n)) / 2

1. Studi Kasus 4 : Insertion Sort

* Buatlah program insertion sort dengan menggunakan bahasa C++
* Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma insertion sort.
* Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.



1. #include <iostream>
2. **using** **namespace** std;
4. **int** arr[100], arr2[100], n;
6. **void** InsertionSort()
7. {
8. **int** temp, i, j;
9. **for** (i = 1; i <= n; i++)
10. {
11. temp = arr[i];
12. j = i - 1;
13. **while** (arr[j] > temp && j >= 0)
14. {
15. arr[j + 1] = arr[j];
16. j--;
17. }
18. arr[j + 1] = temp;
19. }
20. }
22. **int** main()
23. {
24. cout << "Input the amount of data : ";
25. cin >> n;
26. cout << endl;
28. **for** (**int** i = 1; i <= n; i++)
29. {
30. cout << "Masukkan data ke-" << i << " : ";
31. cin >> arr[i];
32. arr2[i] = arr[i];
33. }
35. InsertionSort();
36. cout << "\nOutput : " << endl;
38. **for** (**int** i = 1; i <= n; i++)
39. {
40. cout << arr[i] << " ";
41. }
42. **return** 0;
43. }



**Tmin(n) = 3n-3 + 4n-4 + 1 = 7n - 6**

1. Operasi Assignment: 2(n-1) + (n-1) = 3n-3
2. Operasi Perbandingan: 2\*((n-1) + (n-1)) = 2\*(2n-2) = 4n-4
3. Operasi Pertukaran: (n-1) \* n = n2-n

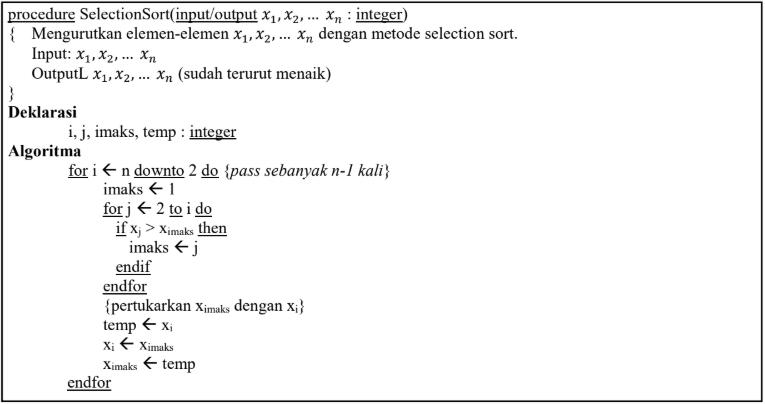
**Tmax(n) = 3n-3 + 4n-4 + n2-n = n2+6n-6**

**Tavg(n) = (n2 + 13n - 12) / 2**

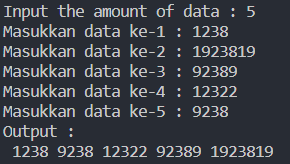
(Tmin(n) + Tmax(n)) / 2 = (7n–6 + n2+6n-6) / 2

1. Studi Kasus 5 : Selection Sort

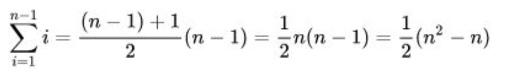
* Buatlah program selection sort dengan menggunakan bahasa C++
* Hitunglah operasi perbandingan elemen larik dan operasi pertukaran pada algoritma selection sort.
* Tentukan kompleksitas waktu terbaik, terburuk, dan rata-rata untuk algoritma insertion sort.



1. #include <iostream>
2. **using** **namespace** std;
4. **int** arr[100], arr2[100];
5. **int** n;
7. **void** swap(**int** a, **int** b)
8. {
9. **int** t;
10. t = arr[b];
11. arr[b] = arr[a];
12. arr[a] = t;
13. }
15. **void** SelectionSort()
16. {
17. **int** pos, i, j;
18. **for** (i = 1; i <= n - 1; i++)
19. {
20. pos = i;
21. **for** (j = i + 1; j <= n; j++)
22. {
23. **if** (arr[j] < arr[pos])
24. pos = j;
25. }
26. **if** (pos != i)
27. swap(pos, i);
28. }
29. }
31. **int** main()
32. {
33. cout << "\nInput the amount of data : ";
34. cin >> n;
36. **for** (**int** i = 1; i <= n; i++)
37. {
38. cout << "Masukkan data ke-" << i << " : ";
39. cin >> arr[i];
40. arr2[i] = arr[i];
41. }
43. SelectionSort();
44. cout << "Output : " << endl;
45. **for** (**int** i = 1; i <= n; i++)
46. {
47. cout << " " << arr[i];
48. }
49. **return** 0;
50. }



1. Operasi Perbandingan =



1. Operasi Pertukaran = n-1

**Tmin(n) = (4n-4) + ½(n2-n) +1 ~ n2**

**Tmax(n) = ½(n2-n) + (n-1) ~ n2**

**Tavg(n) = n2**

(Tmin(n) + Tmax(n)) / 2 = (n2+ n2) / 2