Terlihat fungsi yang menggunakan konsep rekursi lebih singkat instruksinya. Ketika nilai n lebih dari 1 makan akan mengembalikan nilai `n*faktorial(n-1)`. Faktorial(n-1) inilah instruksi untuk memanggil dirinya sendiri namun dengan parameter input n-1. Jika nilai n = 3 maka nilai n pada saat menjalankan fungsi yang pertama adalah 3. Kemudian akan masuk ke instruksi baris ke-5 karena n>1. Kemudian fungsi akan mengembalikan nilai `n*faktorial(n-1)`, berarti 3*faktorial(2). Sehingga dilakukan pemanggilan fungsi faktorial dengan nilai masukan 2. Begitu seterusnya sampai nilai n = 0 yang kemudian mengembalikan nilai 1.

Permasalahan lain yang dapat diselesaikan menggunakan konsep rekursi diantaranya perhitungan pangkat n, deret fibonaci, *divide & conquer, quick sort,* dan lainnya.

5.2 Selection Sort

Salah satu penerapan array adalah digunakan pada algoritma pengurutan. *Selection sort* merupakan algoritma pengurutan yang sederhana dan efisien dengan memilih elemen terkecil (atau terbesar) pada bagian data yang belum urut dan memindahkannya ke bagian data yang sudah diurutkan. Proses pemilihan elemen terkecil dan proses pemindahan dilakukan berulang kali sampai semua data telah urut.

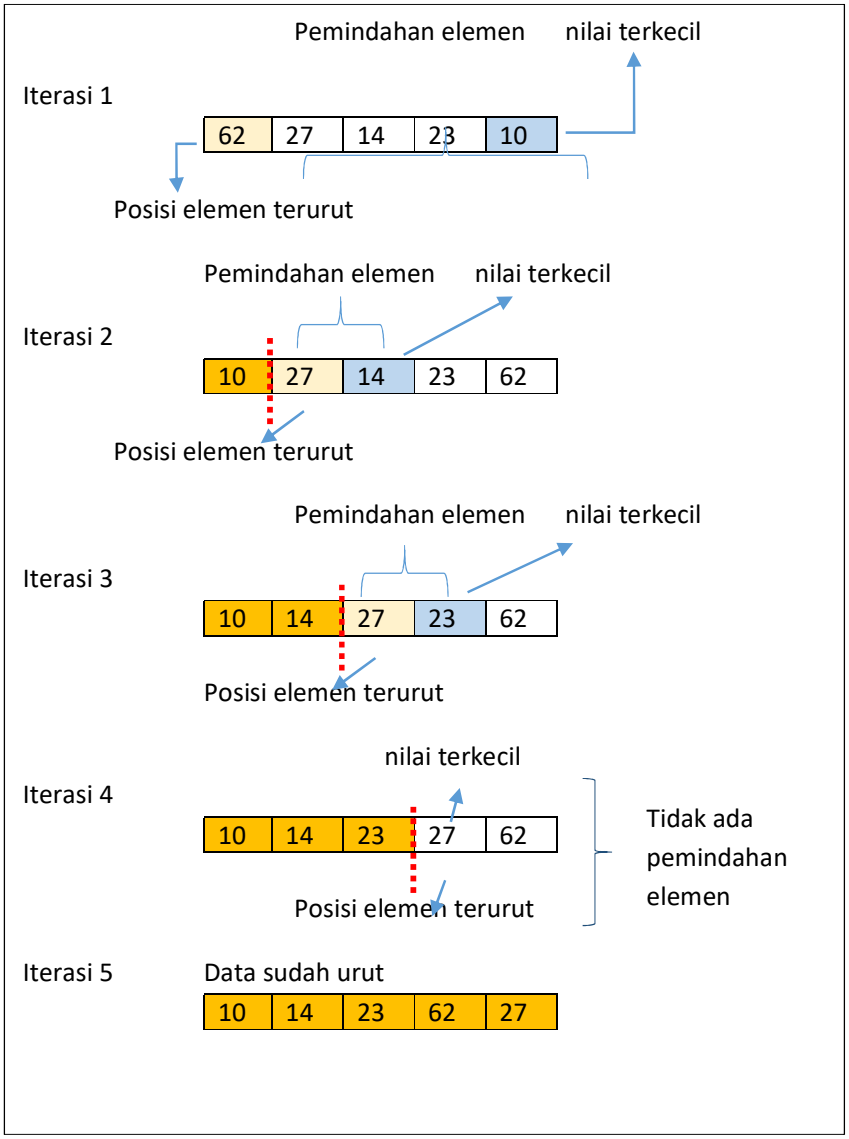
Kompleksitas waktu dari *selection sort* adalah $O(N^2)$ karena ada dua perulangan bersarang:

- Satu perulangan untuk memilih elemen Array terkecil / terbesar satu per satu = $O(N)$
- Perulangan lainnya untuk membandingkan elemen terkecil / terbesar dengan dengan setiap elemen Array lainnya = $O(N)$
- Oleh karena itu kompleksitas keseluruhan

$$= O(N) * O(N) = O(N*N) = O(N^2)$$

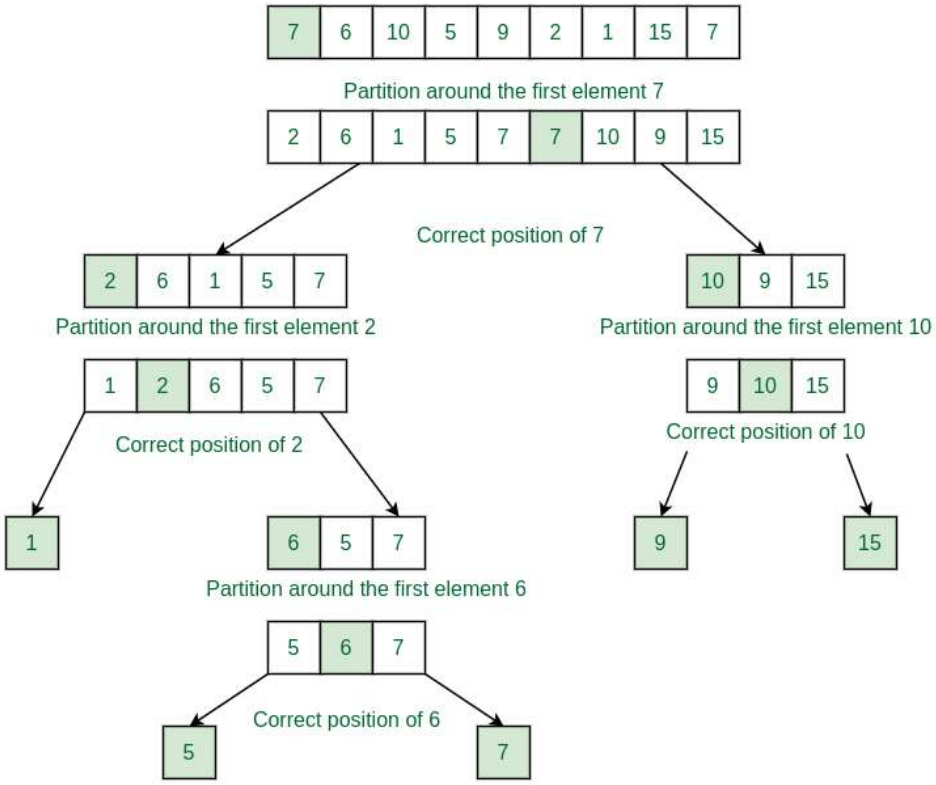
Keuntungan *selection sort* adalah sederhana dan mudah dimengerti serta bekerja dengan baik dengan kumpulan data kecil. Namun *selection sort* memiliki kelemahan yaitu kompleksitas waktu $O(n^2)$ dalam kasus terburuk dan rata-rata sehingga tidak berfungsi dengan baik pada kumpulan data besar. *selection sort* tidak mempertahankan urutan relative item dengan kunci yang sama sehingga tidak stabil.

Sebagai contoh, diketahui array A = {62, 27, 14, 23, 10}. Proses pengurutan dari kecil ke besar (*ascending*) dengan *selection sort* dapat dilihat pada ilustrasi dibawah ini.



5.3 Quick Sort

Quik sort adalah algoritma pengurutan yang menggunakan konsep rekursi. Kunci utama dari algoritma ini adalah memilih pivot dan kemudian mempartisi yaitu membagi kumpulan data menjadi 2 bagian. Satu bagian merupakan nilai yang lebih kecil dari pivot dan bagian yang lain berisi nilai yang lebih besar dari pivot. Pemilihan pivot dapat dilakukan dengan memilih elemen pertama, terakhir, tengah atau dipilih secara acak. Sebagai contoh perhatikan ilustrasi cara kerja *quick sort* dibawah ini.



Elemen pertama yaitu 7, dipilih sebagai pivot. Bandingkan angka 7 dengan elemen sebelah nya hingga menempati tempat seharusnya. Kemudian bagi data menjadi 2 bagian. Kemudian dilakukan hal yang sama pada masing-masing bagian. Hal tersebut dilakukan dengan menerapkan konsep rekursi.

Big O dari *quick sort* adalah $O(N^2)$ terjadi ketika pivot yang dipilih kurang baik, sehingga dapat dikatakan *quick sort* kurang stabil karena bergantung dengan pemilihan pivot. Kompleksitas waktu rata-ratanya adalah $O(n \log n)$. sehingga lebih efisien pada kumpulan data besar dibandingkan dengan *selection sort*.

LATIHAN 5

Untuk memperdalam pemahaman, silakan kerjakan soal-soal dibawah ini :

1. Jelaskan yang dimaksud konsep rekursi !
2. Jelaskan cara kerja *selection sort* !
3. Sebutkan kelebihan dan kekurangan *selection sort* !

4. Jelaskan cara kerja *selection sort* !
5. Sebutkan kelebihan dan kekurangan *selection sort* !

Petunjuk jawaban :

Jawaban untuk soal-soal diatas dapat ditemukan pada teori pertemuan 6 tentang pengurutan *(Sorting)*.

1. Rekursi adalah suatu fungsi yang memanggil dirinya sendiri berulang kali sampai suatu kondisi terpenuhi.
2. Cara kerja selection sort adalah sebagai berikut :
    - Langkah pertama adalah mencari nilai terkecil (ascending) dari array, kemudian menukar nilai terkecil dengan nilai elemen pertama dari array yaitu indeks ke-0.
    - Pada iterasi ke-2, dimulai dengan mencari nilai terkecil dari sisa elemen yang belum urut yaitu elemen indeks ke-1 sampai indeks ke-n, kemudian menukar nya dengan nilai indeks ke-1.
    - Pada iterasi ke-3, dimulai lagi dengan mencari nilai terkecil dari elemen indeks ke-2 sampai indeks ke-n, kemudian menukarnya dengan nilai indeks ke-2.
    - Begitu seterusnya sampai data terurut.
3. Keuntungan *selection sort* adalah sederhana dan mudah dimengerti serta bekerja dengan baik dengan kumpulan data kecil. Namun *selection sort* memiliki kelemahan yaitu kompleksitas waktu $O(n^2)$ dalam kasus terburuk dan rata-rata sehingga tidak berfungsi dengan baik pada kumpulan data besar. *selection sort* tidak mempertahankan urutan relative item dengan kunci yang sama sehingga tidak stabil.
4. Langkah pertama adalah memilih pivot. Bandingkan nilai pada pivot dengan data lainnya. Kemudian mempartisi yaitu membagi kumpulan data menjadi 2 bagian, satu bagian merupakan nilai yang lebih kecil dari pivot dan bagian yang lain berisi nilai yang lebih besar dari pivot. Kemudian dilakukan hal yang sama pada masing-masing bagian. Hal tersebut dilakukan dengan menerapkan konsep rekursi.
5. Kelemahan : Big O dari *quick sort* adalah $O(N^2)$ terjadi ketika pivot yang dipilih kurang baik, sehingga dapat dikatakan *quick sort* kurang stabil karena bergantung dengan pemilihan pivot.

   Keuntungan : Kompleksitas waktu rata-ratanya adalah $O(n \log n)$. sehingga lebih efisien pada kumpulan data besar dibandingkan dengan *selection sort*.

RANGKUMAN 5

- Rekursi adalah suatu fungsi yang memanggil dirinya sendiri berulang kali sampai suatu kondisi terpenuhi.
- *Selection sort* merupakan algoritma pengurutan yang sederhana dan efisien dengan memilih elemen terkecil (atau terbesar) pada bagian data yang belum urut dan memindahkannya ke bagian data yang sudah diurutkan.
- Kompleksitas waktu dari *selection sort* adalah $O(N^2)$ karena ada dua perulangan bersarang.
- *Quik sort* adalah algoritma pengurutan yang menggunakan konsep rekursi.
- Big O dari *quick sort* adalah $O(N^2)$ terjadi ketika pivot yang dipilih kurang baik, sehingga dapat dikatakan *quick sort* kurang stabil karena bergantung dengan pemilihan pivot.
- Kompleksitas waktu rata-rata *quick sort* adalah $O(n \log n)$. sehingga lebih efisien pada kumpulan data besar dibandingkan dengan *selection sort*.

TUGAS 5

Buatlah pseudocode dari algoritma *selection sort,* dan *quick sort* !

Petunjuk jawaban tugas 5 ada pada halaman terakhir modul. Periksalah jawaban anda, jika kurang sesuai pelajari ulang materi 5. Jika sudah cukup sesuai, silakan lanjut pada materi berikutnya.