Penerapan Teknik *Divide and Conquer* dalam Menyelesaikan Masalah Sehari-hari

Muhammad Tito Prakasa 13519007[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]muhammadtitoprks@gmail.com

***Abstract*—Lately, in our world problem have become more complex. We, as a person who feels right away, must find a way how to solve this and how to solve the problem we face tomorrow. Fortunately, there is method known as Divide and Conquer in computer science. An universal algorithm that focusing to split a big problem to smaller problem and try to solve them one by one and combine at the end. We could try implement this to simplify real life problem that often complex.**

***Keywords*—Divide, Conquer, Recursion, Problem.**

I. PENDAHULUAN

Seiring perkembangan zaman, masalah yang kita temui tentunya juga ikut berkembang. Ambil contoh saja orang tua kita dahulu hanya memikirkan "bagaimana agar anakku bisa memiliki gizi yang baik dan kebutuhannya tercukupi?", namun hal ini sangat jauh berbeda dengan apa yang terjadi sekarang, yaitu orang tua selain memikirkan gizi dan kebutuhan tetapi juga harus memikirkan kesehatan mental, perkembangan jiwa, dll. Hal ini saya ambil hanya sebagai contoh bahwa dunia kita saat ini memang jauh lebih kompleks dari dunia kita beberapa tahun yang lalu. Setiap masalah yang muncul tidak serta-merta dengan satu-dua langkah bisa selesai, kadang diperlukan peninjauan mendalam akar masalahnya bagaimana, kronologis masalah hingga menjadi besar, atau variabea apa saja yang menyebabkan masalah ini terjadi.

Gambar 1. Ilustrasi dunia yang kompleks

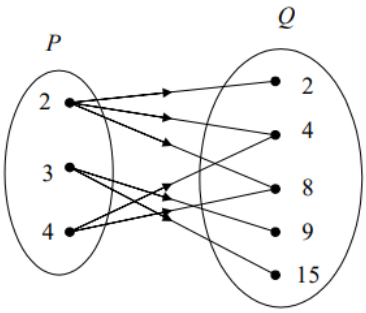
Sumber: https://duo.com/blog/simplicity-in-a-complicated-world

Lalu bagaimana kita menghadapinya? Dalam dunia *computer science* dikenal sebuah algoritma universal yaitu *divide and conquer*, sebuah algoritma yang memanfaatkan rekursivitas dari suatu masalah sehingga dapat dipecah menjadi sub-sub permasalahan yang lebih sederhana lalu disatukan kembali ketika masing-masing sub permasalahan telah diselesaikan. Sub-sub permasalahan ini tentunya lebih mudah diselesaikan daripada kita harus menyelesaikan suatu permasalahan utuh secara langsung. Dengan teknik ini tentunya dapat menjadi salah satu jawaban bagaimana kita menghadapi permasalahan yang terus berkembang seiring perkembangan zaman.

II. TEORI DASAR

A. Relasi

Relasi merupakan representative suatu hubungan antara anggota himpunan yang satu dengan anggota himpunan yang lain. Dalam relasi terdapat daerah yang dibagi menjadi dua, yaitu daerah asal (domain) dan daerah hasil (kodomain). Namun, hal ini bukan berarti tiap relasi memerlukan dua atau lebih himpunan, relasi juga dapat dilakukan dengan hanya satu himpunan.



Gambar 2. Ilustrasi relasi A habis membagi B

Sumber: Kuliah Matematika Dsikrit Materi Relasi&Fungsi

Penulisan relasi menggunakan *paired element* dengan contoh sebagai berikut:

$$a \in A, b \in B, (a,b) \in R$$
$$a \in A, c \in B, (a,c) \notin R$$

Dengan arti, terdapat sebuah relasi R yang menghubungkan beberapa elemen dari himpunan A dengan himpunan B. a merupakan elemen himpunan A,

b dan c merupakan elemen himpunan B. a memiliki relasi R dengan b tetapi a tidak memiliki relasi R dengan c.

Dalam relasi satu himpunan sering digunakan representasi relasi biner atau artinya jika suatu anggota memiliki relasi dengan anggota yang lain maka nilai relasi mereka akan bernilai 1, jika sebaliknya maka nilai relasi mereka akan bernilai 0. Selanjutnya, dalam relasi biner didefinisikan beberapa sifat, yaitu:

a. Refleksif
Relasi pada sebuah himpunan disebut refleksif jika untuk setiap elemen himpunan tersebut memiliki relasi ke elemen itu sendiri. Dalam notasi dapat dituliskan:

$$(a, a) \in R, \forall\, a \in A$$

Sehingga sebuah relasi pada suatu himpunan disebut **tidak refleksif** jika ada satu saja elemen yang tidak memiliki relasi ke elemen itu sendiri.

b. Menghantar
Relasi pada sebuah himpunan disebut menghantar jika elemen a memiliki relasi dengan elemen b, dan elemen b memiliki relasi dengan elemen c, maka elemen a juga memiliki relasi dengan elemen c. Dalam notasi dapat dituliskan:

$$(a, b) \in R, (b, c) \in R \rightarrow (a, c) \in R.\ a, b, c \in A$$

Sehingga sebuah relasi pada suatu himpunan disebut **tidak menghantar** jika ada satu saja elemen yang meng-*cancel* syarat di atas.

c. Setangkup dan Tolak-setangkup
Relasi pada sebuah himpunan disebut setangkup jika untuk setiap elemen a yang memiliki relasi dengan elemen b, maka elemen b juga memiliki relasi dengan elemen a. Dalam notasi dapat dituliskan:

$$(a, b) \in R \rightarrow (b, a) \in R$$

Sehingga sebuah relasi pada suatu himpunan disebut **tidak setangkup** jika terdapat pasangan elemen yang tidak memenuhi syarat di atas.

Selanjutnya, pengertian Tolak-setangkup mirip dengan Setangkup hanya saja elemen yang bersangkutan harus sama. Dalam notasi dapat dituliskan:

$$(a, b) \in R \rightarrow (b, a) \in R, a = b$$

Sehingga suatu relasi pada sebuah himpunan disebut **tidak tolak-setangkup** jika terdapat sebuah elemen berbeda yang saling ber-relasi.

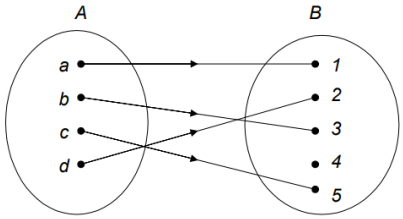
B. Fungsi
Bisa dikatakan fungsi merupakan relasi khusus, yaitu relasi yang setiap elemen dari himpunan asal dipetakan tepat satu elemen ke himpunan yang dituju. Agar penamaan lebih mudah, kita sebut himpunan asal sebagai domain dan himpunan yang dituju sebagai kodomain. Hal ini biasa dituliskan dalam syarat:

Jika $(a, b) \in f$ dan $(a, c) \in f$ maka dipastikan $b = c$

Sehingga bisa disimpulkan dalam fungsi, sebuah nilai dari domain hanya tepat dipetakan satu nilai ke kodomain.

Selanjutnya, fungsi dibedakan menjadi tiga yaitu:

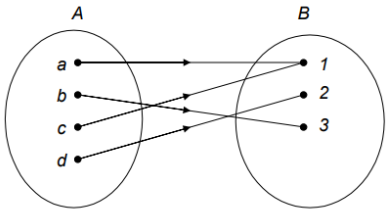
1. Fungsi Injektif
Injektif memiliki arti tidak ada elemen himpunan asal yang memiliki nilai kodomain yang sama.



Gambar 3. Ilustrasi Injective Function

Sumber: Bahan Ajar Matematika Disktit ITB

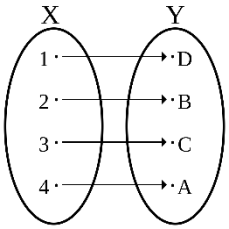
2. Fungsi Surjektif
Surjektif memiliki arti setiap nilai di kodomain merupakan nilai dari domain.



Gambar 4. Ilustrasi Surjective Function

Sumber: Bahan Ajar Matematika Diskrit ITB

3. Fungsi Bijeksi
Bijeksi memiliki arti masing-masing nilai di domain dan kodomain berkorespondensi satu ke satu. Atau bisa dibilang jika suatu fungsi merupakan fungsi injektif sekaligus fungsi surjektif, makai a merupakan fungsi bijeksi.



Gambar 5. Ilustrasi Bijective Function

Sumber: Wikipedia

C. Rekursi

Objek yang didefinisikan oleh dirinya sendiri disebut rekursif. Sedangkan proses pendefinisiannya adalah rekursi. Hal ini banyak berperan pada fungsi dalam matematika. Fungsi yang terdiri dari fungsi itu sendiri disebut fungsi rekursif. Berikut contoh fungsi Fibonacci, fungsi yang nilainya didapat dari nilai fungsi sebelumnya:

$$f(n) = f(n-1) + f(n-2), n \geq 0$$
$$f(0) = 0, f(1) = 1$$

Dapat dilihat dari contoh di atas bahwa suatu fungsi rekursif pasti memiliki dua bagian penting, yaitu basis (bawah) dan rekurens (atas). Basis berguna sebagai nilai tetap agar suatu fungsi yang mengalami rekursif akan mendapatkan nilai. Sedangkan rekurens berguna agar fungsi "terus berjalan" menuju basis.

III. *Divide and Conquer*

Divide and Conquer adalah algoritma universal yang banyak digunakan dalam dunia komputer sains untuk menyelesaikan masalah-masalah kompleks. Algoritma ini memanfaatkan rekursivitas sehingga sebuah *problem* yang terlihat rumit akan dibagi-bagi menjadi *subproblem* yang lebih sederhana. Nantinya *subproblem* inilah yang akan dipecahkan dan dari semua solusi *subproblem* akan digabungkan kembali menjadi solusi dari *problem* utama yang rumit.

Pada dasarnya terdapat tiga bagian utama dalam algoritma ini, yaitu:

A. *Divide*

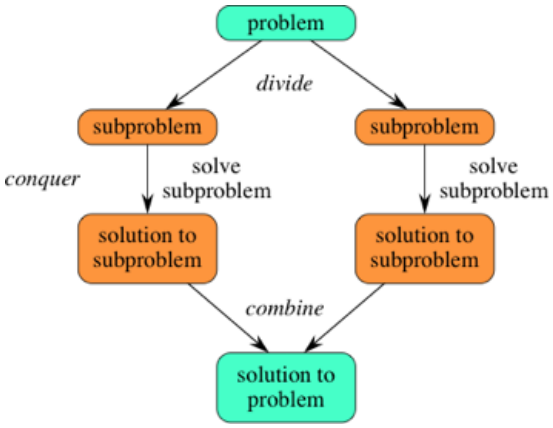
Problem utama dipisah-pisahkan menjadi beberapa *subproblem*. Pada bagian ini kita harus jeli dalam melihat apakah sebuah *subproblem* sudah cukup sederhana untuk diselesaikan atau apakah *subproblem* ini masih berkorelasi dengan *problem* utama atau tidak.

B. *Conquer*

Subproblem diselesaikan dengan cara rekursi, yaitu *Subproblem* akan diarahkan ke kasus basis dengan cara memanggil rekurens dari *Subproblem*.

C. *Combine*

Setelah semua *Subproblem* diselesaikan dan didapatkan solusinya, maka di akhir semua solusi itu digabungkan menjadi solusi dari *problem* utama.



Gambar 6. Ilustrasi garis besar divide and conquer

Sumber: https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms

IV. Penerapan *Divide and Conquer*

Dalam kehidupan kita di dunia ini, tidak lepas dari yang namanya berpikir. Menurut Eric (2004) dalam bukunya yang berjudul, "What is Thought?" mengatakan bahwa pikiran merupakan hasil olah otak kita terhadap dunia real melewati proses mental dan gagasan sehingga sebuah pikiran pasti melibatkan informasi agar dapat membentuk suatu konsep, penalaran, dan keputusan. Di zaman sekarang yang serba maju, pikiran kita dituntut untuk menyelesaikan suatu persoalan-persoalan yang rumit, seperti bagaimana agar keuntungan perusahaan meningkat 10%, membuat algoritma dalam membangun aplikasi, atau dalam me-*manage* sekelompok orang dalam dunia keorganisasian. Hal ini tentunya terasa sangat sulit kelihatannya, namun dengan pendekatan *divide and conquer* yang berakar dari rekursi maka kita akan coba menerapkannya untuk menyelesaikan masalah-masalah tersebut.

1. Penerapan dalam Organisasi

Dalam membantu menyelesaikan sebuah tugas atau *project*, di dunia organisasi kerap menggunakan metode OKR atau *Object Key Result* sebagai alat bantu mencapai *goals* yang diinginkan. Secara singkat OKR adalah metode membagi sebuah *objective* menjadi beberapa *task* sehingga jika semua *task* sudah diselesaikan, maka bisa dikatakan *objective* yang dituju sudah dicapai. Prinsipnya sangat mirip dengan prinsip *divide and conquer*, yaitu sebuah problem yang besar akan "dipecah" menjadi beberapa problem yang lebih sederhana lalu diselesaikan dan digabung solusinya di akhir.

Cara kerja OKR intinya dibagi menjadi dua bagian, yaitu *objective* dan *key results*. *Objective* merupakan *goals*

utama yang ingin dituju, sedangkan *key results*-nya adalah "batu loncatan" agar bisa mencapai *goals* tersebut. Contoh kasus:

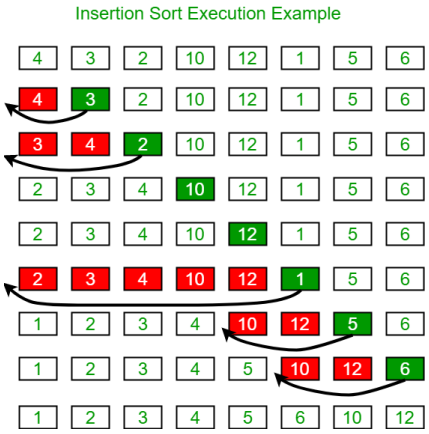
Objective: Meningkatkan produksi baju sebesar 10% dalam satu bulan

Key Results: Suplai kain 20% lebih banyak dari bulan sebelumnya, tambah lima mesin jahit dengan kondisi layak, rekrut tambahan pekerja.

Dengan analisa kasus, objektif kita adalah meningkatkan produksi sehingga hal-hal utama yang dibutuhkan untuk mencapai itu adalah suplai dan sumber daya untuk melakukan hal tersebut. Dengan peningkatan 10% penulis asumsikan dibutuhkan tambahan kain sebanyak 20% dari biasanya dan tambahan mesin jahit lima. Selanjutnya untuk melakukan hal itu maka kita memerlukan tambahan tenaga dengan asumsi merekrut lima tenaga kerja cukup. Setelah itu dari *key results* yang sudah ditentukan, kita perlu menyelesaikan *key result* tersebut dengan pendekatan rekursi atau *divide and conquer* lagi, yaitu bagi kembali hal tersebut menjadi beberapa bagian hingga cukup sederhana untuk diselesaikan oleh tim anda.

2. Penerapan dalam Algoritma

   Apa yang terpikir pertama kali saat kita ditanyakan sebuah cara bagaimana mengurutkan sekelompok angka acak menjadi terurut membesar/mengecil? Algoritma yang umum dalam mengurutkan sekelopok angka dikenal sebagai algoritma "*insertion sort"*, dengan ide secara berurut dari kiri sebuah elemen akan dibandingkan dengan semua elemen di sebelah kirinya. Jika saat dibandingkan ditemukan elemen yang nilainya lebih kecil, maka tukar posisinya. Jika sampai elemen di sebelah kirinya habis dan tidak ditemukan elemen yang nilainya lebih kecil, maka elemen tersebut dibiarkan dan cek elemen selanjutnya (sebelah kanan).

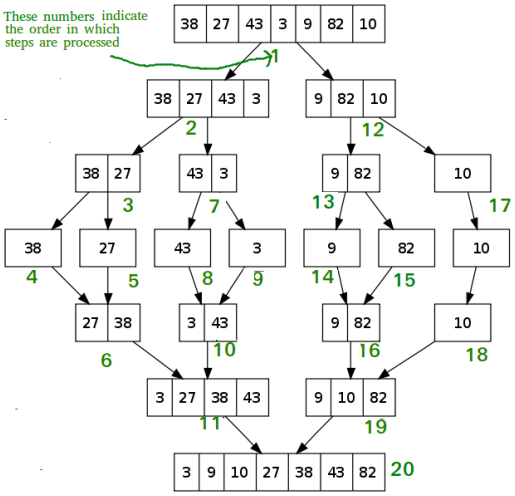


Gambar 7. Ilustrasi Insertion Sort

Sumber: https://www.geeksforgeeks.org/insertion-sort/

Algoritma ini cukup cepat untuk masukan angka yang tidak banyak karena algoritma memiliki laju pertumbuhan waktu sebesar $O(n^2)$ dengan n sebagai banyaknya angka. Maka dari itu, untuk jumlah inputan yang besar didesain algoritma yang baru yaitu *merge sort*.

Merge sort merupakan salah satu algoritma yang memanfaatkan teknik *divide and conquer* dengan ide mengurutkan, membagi terlebih dahulu kelompok angka hingga panjang kelompok angkanya satu (sebuah angka). Lalu dari masing-masing angka tersebut akan digabungkan secara rekursif sesuai terurut menaik atau menurun.



Gambar 8. Ilustrasi Merge Sort

Sumber: https://www.geeksforgeeks.org/merge-sort/

Untuk melihat perbedaan jelasnya, perhatikan analisa *pseudo-code* masing-masing algoritma sorting sebagai berikut:

Insertion Sort:

```
for i=1 to N:

    key = arr[i]
    j = i – 1

     while j >= 0 and arr[j] > key do
       arr[j + 1] = arr[j];
       j = j – 1

    arr[j + 1] = key
```

Merge Sort:

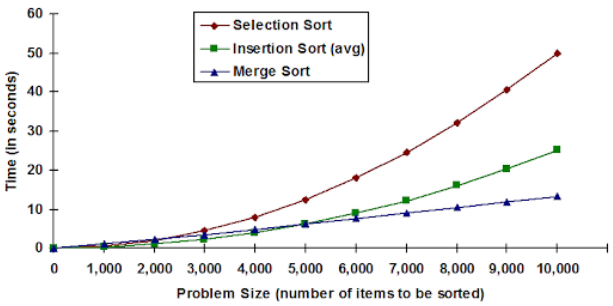
```
sort(int arr[], int l, int r)

    if  l < r then
        {Mencari titik tengah}
        m = (l + r) / 2

        sort(arr, l, m)
        sort(arr, m + 1, r)

        merge(arr, l, m, r)
```

Saat kita melakukan *insertion sort* dengan inputan misal, 8 angka maka, jumlah perbandingan yang dilakukan adalah 36 kali. Sedangkan pada *merge sort* karena dari 8 angka akan disort secara rekursif bagian setengah kiri dan bagian setengah kanan, maka secara kasar aksi yang dilakukan dalam algoritma *Merge Sort* hanya 4 kali (walaupun algoritma sort dan merge di dalam *Merge Sort* memiliki detail tersendiri).



Gambar 9. Ilustrasi Perbedaan Merge Sort dan Insertion Sort

Sumber: http://watson.latech.edu/WatsonRebootTest/ch05s4p3.html

V. KESIMPULAN

Dari beberapa contoh penerapan teknik *divide and conquer*, penulis merasa bahwa teknik ini sangat bagus apabila dijadikan sebuah *mindset* berpikir ketika sebuah masalah. Ibaratnya, ketika kita bertemu masalah maka hal yang pertama dipikirkan adalah analisa terlebih dahulu apa saja yang diperlukan untuk menyelesaikan masalah tersebut, lalu dari hasil analisa tersebut jadikan mereka menjadi sub-sub masalah yang lebih sederhana. Terakhir mulai kerjakan sub-sub masalah tersebut secara berurutan dan lengkap, maka voila! Secara tidak sadar masalah utamamu telah terselesaikan.

Contoh:

1. Mempelajari materi ujian
   Tentunya ketika ingin mempelajari suatu materi, maka kita perlu menguasai materi-materi pre-requisitenya. Jadi anggapannya, materi utamanya merupakan masalah utama dan materi pre-requisitenya merupakan sub masalah yang perlu dikerjakan terlebih dahulu.

2. Menentukan langkah efisien dalam mencuci piring
   Agar efisien dalam mencuci piring, kita dapat menganalisa terlebih dahulu benda macam apa saja yang akan dicuci, lalu urutkan dari yang terbesar ke terkecil, hitung prediksi air dan sabun yang dibutuhkan, dan terakhir jika saat dicucui dari yang terbesar ke terkecil, maka saat pembilasan dimulai dari benda yang kecil ke benda yang besar.

3. Mengatasi *habit* buruk
   Tentunya *habit* mana yang ingin diperbaiki harus diketahui terlebih dahulu, setelah itu analisa mengapa hal itu bisa sampai menjadi *habit*. Jadikan hal-hal penyebab tersebut sebagai sub masalah yang harus diselesaikan sehingga ketika semua sub masalah sudah diselesaikan, maka secara tanpa sadar *habit* buruk anda sudah berkurang atau bahkan tidak ada.

VII. UCAP SYUKUR

Pertama-tama penulis ucapkan rasa syukur kepada Tuhan Yang Maha Esa karena-Nya penulis diberikan kemampuan untuk menyelesaikan tulisan ini. Selanjutnya, penulis ingin berterima kasih kepada keluarga dan teman karena mereka penulis bisa terus bertahan hingga detik ini. Dan terakhir, saya ucapkan terima kasih kepada jajaran dosen mata kuliah Matematika Diskrit ITB, terutama ibu Fariska selaku dosen penulis, karena telah membagikan ilmunya kepada penulis dan bisa membuat tulisan tentang ilmunya, yaitu makalah ini.

REFERENCES

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/matdis20-21.htm. Diakses pada 9 Desember 2020.
[2] https://www.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/divide-and-conquer-algorithms. Diakses pada 9 Desember 2020.
[3] Baum, Eric. “What is Thought?”. MIT Press. 2004.
[4] HE, Yu-Ling, Hao Zhou. “Comparative Study of OKR and KPI”. International Conference on E-commerce and Contemporary Economic Development. 2018.
[5] https://www.geeksforgeeks.org/insertion-sort/. Diakses pada 11 Desember 2020.
[6] https://www.geeksforgeeks.org/merge-sort/. Diakses pada 11 Desember 2020.
[7] http://watson.latech.edu/WatsonRebootTest/ch05s4p3.html. Diakses pada 11 Desember 2020.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 11 Desember 2020

Muhammad Tito Prakasa - 13519007