Subset Sum Problem dan NP-Complete

Ros Sumiati 23513181 [1]
Program MagisterInformatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
23513181@std.stei.itb.ac.id

Abstract*— Pada bidang computer sains, Subset sum problem adalah salah satu masalah yang penting dalam teori kompleksitas dan kriptografi. Masalah utama dari Subset sum problem adalah jika terdapat n bilangan real dalam suatu himpunan dan ingin dihitung semua kombinasi yang mungkin dari himpunan bilangan tersebut., teori kompleksitas, dan puzzle. Namun sampai saat ini, masalah sum of subset termasuk ke dalam salah satu masalah yang sukar untuk diselesaikan karena termasuk dalam kelas masalah NonDeterminictic Polynomial Complete (NP-Complete atau NPC).

Kata Kunci—Subset sum problem, Undecidability, NP-Complete

I. PENDAHULUAN

The Clay Mathematics Institute (USA) menyatakan, ada tujuh problema matematika yang terkenal di dunia dan belum terpecahkan. Terangkum dalam "The Millennium Problems : The Seven Greats Unsolved Mathematical Puzzles of Our Time". Ketujuh problem matematika tersebut di antaranya :

1. The birch and Swinnerton-Dyer Conjecture,
2. The poincare Conjecture,
3. Navier-Stokes Equations,
4. P Versus NP Problem,
5. Riemann Hypothesis,
6. The Hodge Conjecture,
7. Yang-Mills Theory and The Mass Gap Hypothesis.

Didalam NP salah satu persoalan yang termasuknya adalah Subset sum problem. Masalah utama dari Sum Of Subsets adalah jika terdapat n bilangan real dan ingin dihitung semua kombinasi yang mungkin dari himpunan bilangan tersebut. Kombinasi yang diinginkan yaitu kombinasi yang jumlah seluruh elemennya sama dengan M (tertentu).

Selain memberikan hasil yang benar, efisiensi dari waktu eksekusi ataupun penggunaan memori dari algoritma adalah hal yang penting bagi sebuah algoritma. Setiap masalah yang biasa dihadapi di dunia informatika dapat diselesaikan dengan beberapa algoritma yang sudah ada (tidak hanya satu algoritma). Namun, tidak semua algoritma bisa menyelesaikan suatu masalah dengan efektif dan efisien (mangkus). Tingkat kemangkusan suatu algoritma dalam memecahkan masalah ditentukan dengan menghitung operasi khas pada algoritma tersebut lalu menyatakannya dengan notasi Big O.

Seperti layaknya hal-hal krusial lainnya pada ilmu komputer, tentunya fungsi pertumbuhan ini juga memiliki notasi matematika khusus. Penulisan fungsi pertumbuhan dilakukan dengan menggunakan notasi asmtotik. Terdapat beberapa jenis notasi asimtotik, salah satunya yaitu notasi Big-O.

Big-O dipilih karena merupakan notasi yang paling populer dan paling banyak digunakan pada kalangan peneliti ilmu komputer. Notasi Big-O digunakan untuk mengkategorikan algoritma ke dalam fungsi yang menggambarkan batas atas (*upper limit*) dari pertumbuhan sebuah fungsi ketika masukan dari fungsi tersebut bertambah banyak.

Big-O, seperti namanya, dituliskan sebagai fungsi "O" dengan nilai masukan berupa tingkat pertumbuhan dari fungsi yang dijabarkan. Misalnya, algoritma perpangkatan dengan pertumbuhan linear yang kita kembangkan pada bagian sebelumnya memiliki kelas Big-O O(n).

Fungsi Big-O	*Nama*
O(1)	Konstan
O(logn)	Logaritmik
O(n)	Linear
O(n logn)	n log n
$O(n^2)$	Kuadratik
$O(n^m)$	Polinomial
O(n!)	Faktorial

Sum of subset merupakan masalah NP-Complete sehingga sum of subset tergolong masalah yang sukar (hardest problem). Makalah ini akan membahasa lebih detail mengenail Sum subset Problem beserta NP-Complete.

II. LANDASAN TEORI

A. Subset sum problem

Masalah sum of subset adalah menentukan apakah dari suatu himpunan bilangan terdapat himpunan bagian yang jika semua anggotanya dijumlahkan akan menghasilkan M.

Misalkan banyaknya bilangan real tersebut adalah 4 (n=4) akan ditentukan lebih dahulu kombinasi-kombinasi

dari elemen-elemen bilangan tersebut. hal itu dikenal dengan istilah himpunan bagian (subsets).

Dari pohon pada Gambar 1 digambarkan bahwa setiap ruas (*edge*) diberi label sedemikian sehingga ruas dari simpul (*vertex/node*) tingkat I ke simpul tingkat I + 1 akan mewakili nilai dari x_i. Pada setiap simpul, ruang penyelesaian dibagi (dipartisi) menjadi ruang-ruang penyelesaian bagian. Ruang penyelesaian didefinisikan oleh semua jalur dari akar simpul (*node root*) ke simpul lainnya di dalam pohon tersenut. Kemungkinan jalur-jalur tersebut adalah () atau kosong, ini berarti tidak ada jalur dari akar simpul ke dirinya sendiri (1), (1,2), (1,2,3), (1,2,3,4), (1,2,4), (1,3,4), (2), (2,3), (2,3,4) dan seterusnya.

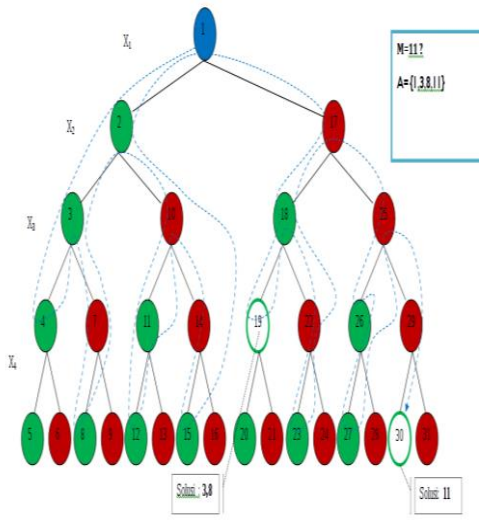
Jalur-jalur tersebut mempunyai ukuran *tuple* yang berbeda-beda, yang merupakan ruang penyelesaiannya. Secara struktur data, pencari ruang solusi di atas menggunakan queue, yang disebut juga dengan *breadth first search* (BFS)

Contoh:
Diketahui himpunan A = {1, 3, 8,11}. Apakah terdapat himpunan bagian A yang jika anggotanya dijumlahkan akan sama dengan 11?

Jawab:
ada, yaitu {3,8},{11}

lalu bagaimana komputer malakukan proses permeriksaaan bilangan M? Secara sederhana komputer akan melakukan fungsi pengecekan melalui sebuah pohon. pohon akan diturunkan sampai 4 kali. Pada kasus Subset of Sum Problem ini 3,8 dan 8,3 dianggap sama saja. Jadi hasil dari kerja untuk persoalan ini adalah:seperti pada Gambar

Mula-mula komputer akan mengecek melalui keberadaan anggota himpunan pertama pada himpunanan. Jika nilai nya masih kurang maka akan dilakukan pengecekan pada bilangan ke-i ditambah bilangan ke-i+1 dan begitu seterusnya. Jika tetap masih tidak ditemukan sampai akhir maka komputer melakukan fungsi runut balik kepada fungsi sebelumnya



B. NP-Complete

Salah satu problem adalah P versus NP problem. Yaitu persoalan mengenai P=NP. Pengertian "P" problem (Polynomial-time algorithm), yaitu sekelompok problem yang relatif mudah dipecahkan oleh algoritma dengan waktu komputer berbentuk polinomial denga meningkatnya ukuran persoalan. NP-Complete Problem (Non Polynomial-time algorithm atau Exponensial Complexity Algorithm) yaitu sekelompok problem yang sulit dan membutuhkan waktu komputer bersifat eksponensial. Tantangannya adalah membuktikan bahwa P=NP. Dengan ata lain, semua problema itu sama, yaitu semua gampang dan tak ada yang sulit.

Kebutuhan waktu algoritma yang mangkus bervariasi, mulai dari O(1), O(log log n), O(log n), O(n), O(n2), dan O(n3). Semua algoritma tersebut digolongkan "bagus" dan dikenal sebagai solusi polinomial. Hal ini karena kebutuhan waktunya secara asimptotik dibatasi oleh fungsi polinomial. Misalnya log(n) < n untuk semua n =1.

Sebaliknya, ada persoalan yang tidak terdapat solusi waktu polinomial untuk menyelesaikannya, misalnya TSP yang memiliki kompleksitas O(n!). Persoalan semacam itu digolongkan sebagai masalah"sulit".

Diluar itu, algoritmanya dinamakan *nonpolynomial-time algorithm*. Contoh: TSP, integer knapsack problem, graph coloring, sirkuit Hamilton, partition problem, bin packing, integer linear programming

Dalam membahas teori NP, dibatasi pada persoalan keputusan (decision problem) Persoalan keputusan adalah persoalan yang solusinya hanya jawaban "yes" atau "no". Contoh: 1. Diberikan sebuah integer x. Tentukan apakah elemen x terdapat di dalam tabel? 2. Diberikan sebuah integer x. Tentukan apakah x bilangan prima?

Namun, jika dapat menemukan algoritma waktu-polinom untuk jenis persoalan optimasi tersebut, maka kita juga mempunyai algoritma waktu-polinom untuk persoalan keputusan yang bersesuaian. Hal ini karena solusi persoalan optimasi menghasilkan solusi persoalan keputusan yang bersesuaian.

Non-deterministik polynomial Polynomial-time non-deterministic algorithm adalah agloritma non-deterministik dimana tahap verifikasinya adalah algoritma dengan kebutuhan waktu polinom. NP Problems adalah himpunan persoalan keputusan yang dapat diselesaikan oleh algoritma non-deterministik dalam waktu polinom. Kebanyakan persoalan keputusan adalah NP

TSDP adalah contoh persoalan NP, sebab jika diberikan sebuah terkaan string (tur), maka dibutuhkan O(n) untuk memverifikasi solusi. Integer Knapsack Decision problem dan Graph Coloring Decision Problem semuanya adalah NP. Semua persoalan P juga adalah NP, sebab tahap menerka tidak terdapat di dalam persoalan P. Karena itu, P adalah himpunan bagian dari NP. NP P NP

NP-Complete (NPC) adalah persoalan NP yang paling sulit. Sebuah persoalan X dikatakan NPC jika:
1. X termasuk ke dalam kelas NP

2. Setiap persoalan di dalam NP dapat direduksi dalam waktu polinom menjadi X Penjelasan sbb:

Untuk persoalan X di dalam NPC, pertama harus dipahami bahwa X adalah NP. Kemudian, kita seharusnya dapat mereduksi sembarang persoalan lain di dalam NP dengan transformasi sederhana menjadi instance persoalan X

Efeknya, jika transformasi ini dapat dilakukan, maka jika algoritma dalam waktu polinom ditemukan untuk X, maka semua persoalan di dalam NP dapat diselesaikan dengan mangkus. Dengan kata lain, jika X adalah NPC dan termasuk ke dalam P algoritma yang mangkus (polinom, deterministik) untuk X ditemukan juga. maka P = NP. Hal ini karena transformasi tersebut sederhana (membutuhkan waktu polinom*)*

Transformasi ini memberi sugesti bahwa jika TSDP dapat diselesaikan dengan mangkus (kebutuhan waktu dalam polinom), maka HCP juga dapat diselesaikan dengan mangkus. Lebih jauh lagi, jika HCP adalah NPC, maka TSDP juga adalah NPC. Sejauh ini, lebih dari 300 persoalan yang terbukti NP-complete NP NPC P

Masalah NP (non-deterministic polynomial) adalah masalah keputusan yang dapat diselesaikan oleh algoritma non deterministik dalam waktu polinomial. Algoritma non deterministik merupakan algoritma yang mampu menerka opsi yang tepat dari pilihan yang tersedia.

Algoritma ini dibagi menjadi dua tahap. Pada tahap pertama, algoritma ini akan menerka solusi dari persoalan yang diberikan. Tahap selanjutnya, algoritma ini akan memverifikasi apakah terkaannya itu merupakan solusi yang tepat untuk persoalan tersebut. Semua masalah sulit yang ditransformasikan menjadi masalah keputusan dan tahap verifikasi pada algoritma non deterministiknya dapat diselesaikan dalam waktu polinomial dikategorikan sebagai masalah NP.

Contoh: Masalah menentukan jumlah warna minimal yang dibutuhkan untuk mewarnai simpul-simpul pada graf agar simpul yang saling bertetangga memiliki warna yang berbeda (graph-colouring optimization problem) merupakan masalah yang tidak dapat diselesaikan dalam waktu polinomial.

Namun, jika masalah tersebut ditransformasi menjadi suatu masalah yang memeriksa apakah dengan n warna untuk graf G semua simpul yang bertetangga bisa memiliki warna yang berbeda (graph colouring decision problem), maka dapat dilakukan verifikasi dalam waktu polinomial. Oleh karena itu, masalah pewarnaan graf dikategorikan sebagai masalah NP.

Suatu masalah NP (misalnya masalah X) bisa digolongkan sebagai masalah NP-Complete jika ada masalah NP-Complete lain yang bisa ditransformasikan menjadi instansiasi dari masalah X menggunakan algoritma dalam waktu polinom. Apabila ditemukan algoritma dalam waktu polinom yang dapat memecahkan masalah NP-Complete, maka semua masalah NP dapat diselesaikan dengan mangkus. Itulah sebabnya masalahmasalah ini dinamakan “NP-Complete”.

Beberapa masalah yang dikategorikan sebagai masalah NP-Complete antara lain:

1. Partition problem, menentukan apakah mungkin membagi himpunan menjadi beberapa himpunan bagian yang disjoint dan setiap bagian mempunyai jumlah nilai yang sama.
2. Sum of subset problem, mencari himpunan bagian dari suatu himpunan bilangan yang jumlah nilainya m.
3. Clique problem, mencari himpunan bagian dari himpunan simpul di suatu graf yang semuanya terhubung.
4. Graph Coloring problem, menentukan apakah sebanyak n warna dapat digunakan untuk mewarnai simpul-simpul pada graf sehingga simpul yang bertetangga berbeda warna.
5. Vertex Cover, menentukan apakah semua anggota himpunan N simpul pada suatu graf tersambung dengan semua sisi yang ada (semua sisi dalam graf tersambung ke satu atau lebih simpul anggota N).
6. N-Puzzle, menentukan apakah N-Puzzle dapat diselesaikan dengan m langkah.
7. Knapsack problem, menentukan apakah dapat memasukkan beberapa objek ke dalam knapsack tanpa melebihi kapasitasnya tetapi profit minimum dari objek-objek tersebut sebesar P.

III. PEMBAHASAN

Tidak mungkin dibangun sebuah algoritma yang akan selalu menghasilkan nilai m yang dicari dari sembarang nilai m dan anggota himpunan oleh sebab itu maka persoalan sum subset problem tergolong dalam undecidability atau tidak bisa dibuat satu algoritma yang pasti menghasilkan nilai m yang di minta

IV. IMPLEMENTASI

Dibangunlah sebuah pseudo code untuk menyelesaikan masalah sum subset problem ini.berikut pseude code nya

```
function Hitung(int currentSum, int indeks, int sumTotalCurrent) {

   visited[indeks] = 1;

   if (currentSum + himpunan[indeks] == m) {

      for i=0 to length_indeks

         if (visited[i] == 1)


output(himpunan[i] + " ");



}else
if((currentSum+himpunan[indeks]+himpunan[indeks+1])<= m)

   Hitung(currentSum + himpunan[indeks], indeks + 1, sumTotalCurrent - himpunan[indeks]);
```

Dan berikut hasil nya yang di transformasikan kedalam bahasa pemrograman java

```
class sos {
        int m;
        int himpunan=0;
        int visited=0;
        public sos() {
                himpunan = new int[40];
                visited = new int[40];
        }
public void Hitung(int currentSum, int indeks, int
sumTotalCurrent) {
int i;
visited[indeks] = 1;

if (currentSum + himpunan[indeks] == m) {
        for (i = 0; i <= indeks; i++)
                if (visited[i] == 1)
        System.out.print(himpunan[i] + "       ");
        System.out.print(" _____ di indeks : ")


        for (i = 0; i <= indeks; i++)
                if (visited[i] == 1)
                System.out.print(i + "  ");
                System.out.println();

}
else if ((currentSum + himpunan[indeks] +
himpunan[indeks + 1]) <= m)


        Hitung(currentSum + himpunan[indeks],
indeks + 1, sumTotalCurrent
- himpunan[indeks]);


if ((currentSum + sumTotalCurrent -
himpunan[indeks] >= m)
&& (currentSum + himpunan[indeks + 1] <= m)) {

        visited[indeks] = 0;
        Hitung(currentSum, indeks + 1,
sumTotalCurrent - himpunan[indeks]);
                        }
                }
        }
```

Sehigga hasil program nya adalah seperti gambar dibawah ini

V. KESIMPULAN

Pada bidang computer sains, Subset sum problem adalah salah satu masalah yang penting dalam teori kompleksitas dan kriptografi. Masalah utama dari Subset sum problem adalah jika terdapat n bilangan real dalam suatu himpunan dan ingin dihitung semua kombinasi yang mungkin dari himpunan bilangan tersebut., teori kompleksitas, dan puzzle. Namun sampai saat ini, masalah sum of subset termasuk ke dalam salah satu masalah yang sukar untuk diselesaikan karena termasuk dalam kelas masalah NonDeterminictic Polynomial Complete (NP-Complete atau NPC)

DAFTAR PUSTAKA

[1] Hopcroft, John E at all . "Introduction to automata theory, language, and computation",Pearson Education, 2000

[2] Rinaldi Munir. "SlideTeori Komputasi IF 5110" Bandung : ITB, 2015

[3] Neil D. Jones. "Computability and Complexity From a Programming Perspective" London:The MIT Press, 1997

[4] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2013-2014-genap/Makalah2014/MakalahIF2211-2014-069.pdf diakses 2 Januari 2016

[5] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2014-2015/Makalah2015/Makalah_IF221_Strategi_Algoritma_2015_071.pdf diakses 2 Januari 2016

[6] http://www.bertzzie.com/knowledge/analisis-algoritma/index.html diakses 1 Januari 2016

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, 7 Januari 2016

Ros Sumiati 23513181