# Penyelesaian *Sum of Subset Problem* dengan *Dynamic Programming*

Devina Ekawati 13513088[1]
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*[1]13513088@std.stei.itb.ac.id*

***Abstract*—Masalah *sum of subset* merupakan salah satu masalah yang cukup penting dan banyak diterapkan dalam berbagai bidang, seperti kriptografi, 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 *Non-Determinictic Polynomial Complete* (NP-*Complete* atau NPC). Meskipun termasuk masalah yang sukar untuk diselesaikan, terdapat berbagai macam cara untuk menyelesaikan masalah *sum of subset*, walaupun kompleksitas waktunya masih belum cukup mangkus. Dalam makalah ini akan dibahas penyelesaian masalah *sum of subset* dengan menggunakan program dinamis (*dynamic programming*)**

***Index Terms*—sum of subset problem, dynamic programming, NP-Complete, knapsack problem**

## I. PENDAHULUAN

*Dynamic programming* (program dinamis) adalah cara untuk menyelesaikan masalah dengan memecah masalah menjadi beberapa submasalah yang lebih sederhana. Program dinamis banyak digunakan untuk menyelesaikan masalah yang kompleks, seperti pada bidang matematika, ekonomi, *bioinformatics*, dan *computer science*. Penyelesaian masalah dengan menggunakan program dinamis memiliki kompleksitas yang lebih baik dibandingkan dengan penyelesaian masalah dengan menggunakan algoritma lain, seperti algoritma *brute force*, *greedy*, *backtracking*, dan sebagainya. Namun tidak semua masalah dapat diselesaikan dengan menggunakan program dinamis. Masalah yang dapat diselesaikan dengan menggunakan program dinamis adalah masalah yang memiliki karakteristik tertentu, salah satunya adalah masalah tersebut harus dapat dibagi menjadi beberapa subpersoalan yang lebih sederhana.

Program dinamis dapat digunakan untuk memecahkan berbagai macam masalah yang cukup kompleks, seperti mencari tur terpendek dari sebuah graf, persoalan *Travelling Salesman Person* (TSP), masalah penganggaran modal (*Capital Budgeting*), masalah *knapsack*, dan sebagainya. Selain itu, program dinamis juga dapat digunakan untuk menyelesaikan masalah *sum of subset*.

Masalah *sum of subset* merupakan salah satu masalah yang cukup menarik dan banyak digunakan dalam berbagai macam bidang, salah satunya adalah *computer science*. Selain itu, masalah *sum of subset* mejadi bagian dalam permasalahan yang lain, misalnya untuk menyelesaikan *puzzle* tertentu.

Program dinamis memiliki kompleksitas waktu yang lebih baik jika dibandingkan dengan algoritma lainnya dalam menyelesaikan masalah *sum of subset*. Namun, kompleksitas waktu penyelesaian masalah dengan menggunakan program dinamis belum cukup mangkus. Hal ini disebabkan oleh masalah *sum of subset* merupakan masalah NP-*Complete* sehingga *sum of subset* tergolong masalah yang sukar (*hardest problem*). Masalah lain yang tergolong dalam NP-*Complete* adalah masalah *travelling salesman person* (TSP), masalah *partition*, masalah *clique*, masalah pewarnaan graf, dan lain- lain.

Untuk menyelesaikan masalah *sum of subset* dengan menggunakan program dinamis, digunakan batasan, yaitu bilangan yang terdapat dalam himpunan merupakan bilangan bulat dan bilangan tersbut harus positif. Batasan ini digunakan agar karakteristik dari persoalan program dinamis dapat terpenuhi, yaitu ongkos (*cost*) pada setiap tahap terus meningkat. Jika terdapat bilangan negative di dalam himpunan bilangan tersebut, ongkos pada tahap tertentu bisa saja menurun dan hal ini menyebabkan ketidakseusaian dengan karakteristik masalah program dinamis.

## II. DASAR TEORI

Program Dinamis (*dynamic programming*) merupakan salah satu strategi atau metode untuk menyelesaikan suatu permasalahan. Istilah program dinamis muncul karena perhitungan solusi menggunakan tabel. Kata "program" pada program dinamis tidak berhubungan dengan *source* code. Yang dimaksud dengan "program" dalam program dinamis adalah perencanaan. Strategi atau metode dalam memecahkan masalah dengan menggunakan program dinamis adalah menguraikan solusi menjadi sekumpulan tahap (*stage*) sedemikian sehingga solusi dari permasalahan tersebut dapat dipandang dari serangkaian keputusan yang saling berkaitan. Penyelesaian suatu persoalan dengan program dinamis memiliki kemiripan dengan algoritma *greedy*, hanya saja pada algoritma *greedy* hanya terdapat satu solusi, sementara pada program dinamis dapat dihasilkan lebih dari satu

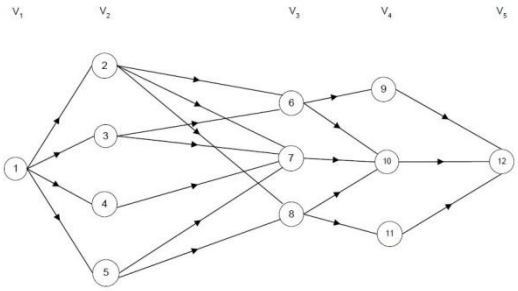
kemungkinan solusi. Karakteristik dari penyelesaian persoalan dengan menggunakan program dinamis adalah

1. Terdapat sejumlah berhingga pilihan yang mungkin.
2. Solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya.
3. Digunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Pada program dinamis, rangkaian keputusan yang optimal dibuat dengan menggunakan prinsip optimalitas. Prinsip optimalitas yang dimaksud adalah jika solusi total merupakan solusi optimal, maka bagian solusi sampai tahap ke-k juga optimal. Dengan prinsip optimalitas dapat dijamin bahwa penagmbilan keputusan pada suatu tahap adalah keputusan yang benar untuk tahap- tahap selanjutnya. Oleh karena itu, pada program dinamis, tidak perlu kembali ke tahap awal karena digunakan hasil optimal dari tahap sebelumnya.

Karakteristik persoalan program dinamis adalah

1. Persoalan dapat dibagi menjadi beberapa tahap (*stage*), yang dalam hal ini setiap tahap hanya diambil satu keputusan.
2. Masing- masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Status yang dimaksud adalah berbagai macam kemungkinan masukan yang ada pada tahap tersebut. Masing- masing tahap ini dapat digambarkan dengan menggunakan graf multitahap (*multistage graph*), yang dalam hal ini tiap simpul pada graf tersebut merepresentasikan status, sedangkan busur pada graf tersebut menyatakan tahap.




*Gambar 1 Graf Multitahap*

[2] Munir, Rinaldi. 2009. ”Diktat Kuliah IF 2251 Strategi Algoritmik”. Bandung:Program Studi Teknik Informatika STEI ITB.

3. Hasil dari keputusan yang diambil pada setiap tahap ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya.
4. Ongkos (*cost*) pada suatu tahap meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan.
5. Ongkos pada suatu tahap bergantung pada ongkos pada tahap yang sudah berjalan sebelumnya dan ongkos pada tahap tersebut.
6. Keputusan terbaik pada suatu tahap bersifat independen terhadap keputusan yang dilakukan pada tahap sebelumnya.
7. Adanya hubungan rekursif yang mengidentifikasikan keputusan terbaik untuk setiap status pada tahap k sehingga dapat diambil keputusan terbaik untuk setiap status pada tahap $k + 1$.
8. Prinsip optimalitas berlaku pada persoalan tersebut.

Terdapat dua pendekatan yang digunakan dalam program dinamis, yaitu maju (*forward* dan *up-down*) dan mundur (*backward* atau *bottom-up*). Misalkan $x_1$, $x_2$, $x_3$, …, $x_n$ yang menyatakan peubah (*variable*) keputusan yang harus dibuat masing- masing pada tahap 1, 2, 3, …, $n$. Untuk program dinamis mundur, program dinamis akan bergerak mulai dari tahap 1, 2, 3, dan seterusnya sampai tahap $n$. Runutan peubah (*variable*) keputusan adalah $x_1$, $x_2$, $x_3$, …, $xn$. Hal sebaliknya berlaku untuk program dinamis mundur. Program dinamis mundur akan bergerak mulai dari tahap $n$, $n - 1$, $n - 2$, dan seterusnya sampai tahap 1. Runutan peubah keputusan pada program dinamis mundur adalah $x_n$, $x_{n-1}$, …, $x_1$. Pada program dinamis maju, prinsip optimalitasnya adalah ongkos pada tahap $k + 1$ = (ongkos yang dihasilkan pada tahap $k$) + (ongkos dari tahap $k$ ke tahap $k + 1$). Sementara prinsip optimalitas pada program dinamis mundur adalah ongkos pada tahap $k$ = (ongkos yang dihasilkan pada tahap $k + 1$) + (ongkos dari tahap $k$+1 ke tahap $k$).

Secara umum, terdapat empat langkah yang dilakukan dalam mengembangkan algoritma program dinamis. Langkah- langkah tersebut adalah

1. Karakteristikkan struktur solusi optimal.
2. Definisikan secara rekursif nilai solusi optimal.
3. Hitung nilai solusi optimal secara maju atau mundur.
4. Konstruksi solusi optimal.

## III. SUM OF SUBSET PROBLEM

Dalam dunia *computer science*, masalah *sum of subset* merupakan masalah yang penting dalam teori kompleksitas dan kriptografi. Masalah *sum of subset* adalah sebagai berikut.

*Diberikan himpunan bilangan bulat tidak kosong dan sebuah angka yang menyatakan jumlah suatu bilangan, misalnya m. Carilah upahimpunan yang jumlahnya = m.*

Misalnya terdapat himpunan bilangan A = {-4, -1, 1, 2, 3, 8, 9} dan m = 0. Maka salah satu solusi dari masalah *sum of subset* tersebut adalah {-4, -1, 2, 3}.

Masalah *sum of subset* merupakan masalah *NP-Complete* (*Non-Deterministic Polynomial Complete*) atau biasa disebut NPC. Hal ini dikarenakan masalah *sum of subset* termasuk ke dalam kelompok kelas NP dan setiap persoalan di dalam kelas NP dapat direduksi menjadi masalah *sum of subset* dalam waktu polinom. Oleh karena itu, masalah *sum of subset* dapat dikatakan salah satu persoalan yang sukar karena belum ada persoalan NPC

yang dapat dipecahlan dalam waktu polinomial. Jika ada persoalan NPC yang dapat dipecahkan dalam waktu polinomial, maka semua persoalan di dalam NP dapat dipecahkan dalam waktu polinomial.

Masalah *sum of subset* dapat pula dikatakan sebagai masalah khusus dari persoalan *knapsack*. Hal ini berarti masalah *sum of subset* sebenarnya memiliki penyelesaian yang mirip dengan persoalan *knapsack*. Perbedaannya adalah pada persoalan *knapsack*, kita perlu mengetahui barang apa saja yang perlu diikutsertakan sambil memperhitungkan keuntungan yang diperoleh jika barang tersebut dimasukkan ke dalam knapsack agar memperoleh keuntungan maksimum dan tidak melebihi kapasitas *knapsack*, sedangkan pada masalah *sum of subset* kita hanya perlu mengetahui bilangan apa saja yang diikutsertakan ke dalam solusi agar bilangan- bilangan tersebut memiliki jumlah yang telah ditentukan sebelumnya.

Kompleksitas dari masalah *sum of subset* bergantung kepada jumlah bilangan di dalam himpunan dan presisi dari masalah tersebut. Jika jumlah bilangan di dalam himpunan cukup kecil, maka penyelesaian masalah *sub of subset* dengan algoritma *exhaustive search* sudah cukup mangkus. Algoritma *exhaustive* search tersebut adalah mencari seluruh kemungkinan penjumlahan bilangan yang mungkin dari himpunan bilangan yang diberikan lalu memeriksa setiap kemungkinan penjumlahan tersebut, apakah hasil penjumlahannya sesuai dengan jumlah bilangan yang diinginkan. Jika hasil penjumlahannya sesuai, maka penjumlahan bilangan tersebut termasuk dalam solusi, dan sebaliknya. Namun, untuk jumlah bilangan di dalam himpunan tersebut cukup besar, algoritma *exhaustive search* menjadi sangat tidak efisien karena kompleksitas waktu algoritma *exhaustive search* dalam menyelesaikan masalah *sum of subset* adalah eksponensial.

Oleh karena itu, para ahli atau *programmer* berusaha untuk menemukan algoritma yang lebih mangkus daripada *exhaustive search*. Beberapa algoritma yang dapat digunakan untuk menyelesaikan masalah *sum of subset* adalah algoritma *backtrack*, *greedy*, *dynamic programming*.

## IV. PENERAPAN ALGORITMA DYNAMIC PROGRAMMING UNTUK MENYELESAIKAN SUM OF SUBSET PROBLEM

Pada persoalan sum of subset, kita hanya tertarik pada nilai dari solusi atau ongkos dan ingin mencari solusi yang melakukan optimasi terhadap suatu nilai tertentu. Untuk mencari solusi, kita dapat membagi persoalan tersebut menjadi beberapa tahap. Oleh karena itu, persoalan sum of subset dapat diselesaikan dengan menggunakan program dinamis. Hal ini disebabkan oleh persoalan sum of subset memiliki karakteristik yang sama dengan karakteristik masalah pada program dinamis.

Langkah pertama yang perlu dilakukan untuk menyelesaikan masalah *sum of subset* dengan program dinamis adalah menentukan karakteristik dari solusi optimal. Misalkan S merupakan pilihan optimal dari himpunan angka yang telah ditentukan dan OPT(n, W) merupakan nilai dari solusi optimal, yang dalam hal ini n merupakan banyaknya bilangan dalam himpunan dan W merupakan batas jumlah bilangan. Kita perlu menentukan algoritma program dinamis untuk menghitung OPT(n, W). Untuk menghitung OPT(n, W), kita perlu menentukan nilai optimal untuk setiap tahap yang terdiri dari i bilangan pertama untuk setiap ukuran *knapsack* $0 \leq w \leq W$ sehingga nilai optimal untuk setiap tahapnya adalah OPT(i, w).

Langkah kedua yang dilakukan adalah menentukan definisi dari nilai solusi optimal secara rekursif. Untuk menghitung OPT(i, w) digunakan definisi

$$OPT(i,W) = \begin{cases} 0, jika\ i = 0\ atau\ W = 0 \\ OPT(i-1,W), jika\ w_i > W \\ max\begin{cases} OPT(i-1,W), jika\ i \notin S \\ w_i + OPT(i-1,W-w_i), jika\ i \in S \end{cases} \end{cases}$$

Langkah ketiga yang dilakukan untuk menyelesaikan masalah *sum of subset* adalah menghitung nilai dari solusi optimal. Dalam makalah ini akan digunakan program dinamis maju untuk menyelesaikan masalah *sum of subset*, yang dalam hal ini program dinamis akan bergerak dari tahap 1, 2, 3, dan seterusnya sampai tahap n.

Misalkan terdapat himpunan bilangan S = {3, 4, 12, 5, 2} dan jumlah upahimpunan = 9. Penyelesaian masalah *sum of subset* tersebut adalah sebagai berikut.

- Tahap 1
  OPT(1,W) = max (OPT(0,W), 3 + OPT(0,W-3))

| W | OPT(0, W) | 3+OPT(0 ,W-3) | Solusi Optimum | |
|---|---|---|---|---|
| | | | OPT(1 ,W) | (x1, x2, x3, x4, x5) |
| 0 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 1 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 2 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 3 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 4 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 5 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 6 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 7 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 8 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |
| 9 | 0 | 3 | 3 | (1, 0, 0, 0, 0) |

- Tahap 2
  OPT(2,W) = max (OPT(1,W), 4 + OPT(1,W-4))

| W | OPT(1, W) | 4+OPT(1 ,W-4) | Solusi Optimum | |
|---|---|---|---|---|
| | | | OPT(2 ,W) | (x1, x2, x3, x4, x5) |
| 0 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 1 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 2 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 3 | 3 | - | 3 | (1, 0, 0, 0, 0) |
| 4 | 3 | 4 | 4 | (0, 1, 0, 0, 0) |
| 5 | 3 | 4 | 4 | (0, 1, 0, 0, 0) |
| 6 | 3 | 4 | 4 | (0, 1, 0, 0, 0) |
| 7 | 3 | 4+3 = 7 | 7 | (1, 1, 0, 0, 0) |
| 8 | 3 | 4+3 = 7 | 7 | (1, 1, 0, 0, 0) |
| 9 | 3 | 4+3 = 7 | 7 | (1, 1, 0, 0, 0) |

- Tahap 3
  OPT(3,W) = max (OPT(2,W), 12 + OPT(2,W-12))

| W | OPT(2, W) | 12+OPT(2,W-12) | Solusi Optimum | |
|---|---|---|---|---|
| | | | OPT(3 ,W) | (x1, x2, x3, x4, x5) |
| 0 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 1 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 2 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 3 | 3 | - | 3 | (1, 0, 0, 0, 0) |
| 4 | 4 | - | 4 | (0, 1, 0, 0, 0) |
| 5 | 4 | - | 4 | (0, 1, 0, 0, 0) |
| 6 | 4 | - | 4 | (0, 1, 0, 0, 0) |
| 7 | 7 | - | 7 | (1, 1, 0, 0, 0) |
| 8 | 7 | - | 7 | (1, 1, 0, 0, 0) |
| 9 | 7 | - | 7 | (1, 1, 0, 0, 0) |

- Tahap 4
  OPT(4,W) = max (OPT(3,W), 5 + OPT(3,W-5))

| W | OPT(3, W) | 5+OPT(3 ,W-5) | Solusi Optimum | |
|---|---|---|---|---|
| | | | OPT(4 ,W) | (x1, x2, x3, x4, x5) |
| 0 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 1 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 2 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 3 | 3 | - | 3 | (1, 0, 0, 0, 0) |
| 4 | 4 | - | 4 | (0, 1, 0, 0, 0) |
| 5 | 4 | 5 | 5 | (0, 0, 0, 1, 0) |
| 6 | 4 | 5 | 5 | (0, 0, 0, 1, 0) |
| 7 | 7 | 5 | 7 | (1, 1, 0, 0, 0) |
| 8 | 7 | 5+3 = 8 | 8 | (1, 0, 0, 1, 0) |
| 9 | 7 | 5+4 = 9 | 9 | (0, 1, 0, 1, 0) |

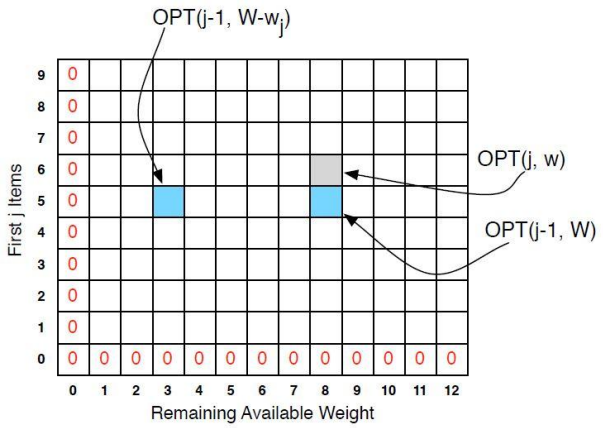
- Tahap 5
  OPT(5,W) = max (OPT(4,W), 2 + OPT(4,W-2))

| W | OPT(4, W) | 2+OPT(4 ,W-2) | Solusi Optimum | |
|---|---|---|---|---|
| | | | OPT(5 ,W) | (x1, x2, x3, x4, x5) |
| 0 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 1 | 0 | - | 0 | (0, 0, 0, 0, 0) |
| 2 | 0 | 2 | 2 | (0, 0, 0, 0, 1) |
| 3 | 3 | 2 | 3 | (1, 0, 0, 0, 0) |
| 4 | 4 | 2 | 4 | (0, 1, 0, 0, 0) |
| 5 | 5 | 2+3 = 5 | 5 | (0, 0, 0, 1, 0) Atau (1, 0, 0, 0, 1) |
| 6 | 5 | 2+4 = 6 | 6 | (0, 1, 0, 0, 1) |
| 7 | 7 | 2+5 = 7 | 7 | (0, 0, 0, 1, 1) |
| 8 | 8 | 2+5 = 7 | 8 | (1, 0, 0, 1, 0) |
| 9 | 9 | 2+7 = 9 | 9 | (0, 1, 0, 1, 0) |

Berdasarkan tahap- tahap di atas, ditemukan bahwa solusi dari permasalah *sum of subset* dengan himpunan bilangan S = {3, 4, 12, 5, 2} dan jumlah upahimpunan = 9

adalah 4 + 5.

Untuk implementasi masalah *sum of subset* dengan menggunakan program dinamis digunakan *array* dua dimensi dengan jumlah baris sebanyak jumlah bilangan pada himpunan dan jumlah kolom sebanyak batas jumlah bilangan ditambah satu (n x W+1).



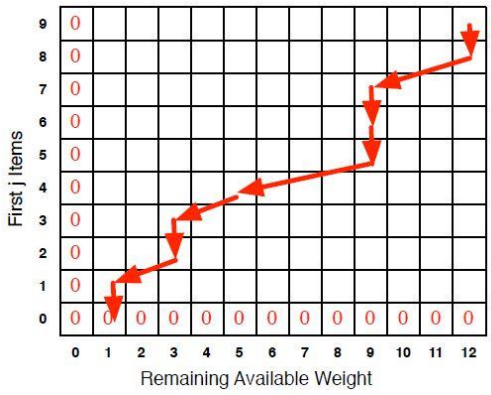

*Gambar 2 Representasi Array Untuk Menyelesaikan Masalah Sum of Subset*

[1] https://www.cs.cmu.edu/~ckingsf/class/02713-s13/lectures/lec15-subsetsum.pdf , diakses 2 Mei 2015

Pseudocode untuk menyelesaikan masalah *sum of subset* dengan program dinamis adalah sebagai berikut.

```
function subsetSumProblem → array[n][W]
  {Insialisasi Matriks baris ke-0 dengan 0}
  for r←0 to W do
    M[0, r] ← 0
  endfor
  {Insialisasi Matriks kolom ke-0 dengan 0}
  for j←1 to n do
    M[j, 0] ← 0
  endfor
  for j←1 to n do
    for r←0 to W do
        {Kasus bilangan telah melebihi W}
        if w[j]> r
          M[j,r] ← M[j-1,r]
        endif
        {Cari nilai optimum}
        M[j,r] = max(M[j-1,r], w[j] + M[j-1,W-
w[j]])
    endfor
  endfor
  return M[n, W]
```

Untuk mencari bilangan yang dipilih, kita mulai mencari dari sebelah kanan atas *array* dua dimensi menuju sebelah kiri bawah, dengan indeks baris terkecil terdapat di paling bawah *array* dan indeks kolom terkecil terdapat di paling kiri *array*.




*Gambar 3Mencari Upabilangan yang Merupakan Solusi*

[1] https://www.cs.cmu.edu/~ckingsf/class/02713-s13/lectures/lec15-subsetsum.pdf , diakses 2 Mei 2015

Kompleksitas untuk mengisi sel pada *array* dua dimensi yang digunakan adalah $O(nW)$ dan setiap sel memiliki kompleksitas $O(1)$ untuk diisi. Untuk mencari upabilangan yang merupakan solusi dari masalah *sum of subset* diperlukan waktu $O(n)$, sehingga komplesitas total dari penyelesaian masalah *sum of subset* adalah $O(nW + n)$ atau $O(nW)$.

## V. KESIMPULAN

Masalah *sum of subset* dapat diselesaikan dengan berbagai macam algoritma, salah satunya adalah dengan menggunakan program dinamis (*dynamic programming*). Program dinamis memiliki kompleksitas waktu yang lebih baik dalam menyelesaikan masalah *sum of subset* dibandingkan dengan algoritma lainnya. Kompleksitas penyelesaian masalah *sum of subset* dengan menggunakan program dinamis adalah O(nW). Meskipun penyelesaian masalah *sum of subset* memiliki kompleksitas waktu yang lebih baik, sampai saat ini belum terdapat algoritma yang cukup mangkus untuk menyelesaikan masalah *sum of subset*. Hal ini dikarenakan sampai saat ini belum ada yang berhasil menemukan cara untuk menyelesaikan masalah NP-*Complete* secara mangkus. Jika ditemukan suatu algoritma untuk menyelesaikan masalah NP-*Complete* dalam waktu yang wajar (polinomial), maka masalah *sum of subset* juga dapat diselesaikan dengan dalam waktu polinomial karena masalah *sum of subset* merupakan masalah NPC.

## VI. UCAPAN TERIMA KASIH

Pertama- tama penulis ingin mengucapkan rasa syukur kepada Tuhan Yang Maha Esa karena oleh rahmat- Nya penulis dapat menyelesaikan makalah ini. Penulis juga ingin berterima kasih kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi yang telah mengajarkan mata kuliah Strategi Algoritma, termasuk *dynamic programming* sehingga Penulis mampu untuk membuat makalah ini. Selain itu, penulis ingin mengucapkan terima kasih kepada orang tua dan rekan- rekan yang

memberikan semangat dan dorongan untuk terus belajar.

## REFERENSI


[1] https://www.cs.cmu.edu/~ckingsf/class/02713-s13/lectures/lec15-subsetsum.pdf , diakses 2 Mei 2015

[2] Munir, Rinaldi. 2009. "Diktat Kuliah IF 2251 Strategi Algoritmik". Bandung:Program Studi Teknik Informatika STEI ITB.


## 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, 4 Mei 2015

Devina Ekawati (13513088)