PENGAPLIKASIAN *DYNAMIC PROGRAMMING* UNTUK MASALAH MAXIMUM SUM SUBRECTANGLE PADA ARRAY 2-DIMENSI

Dwitika Diah Pangestuti[1)], Luke Manuel Daely[2)]
[1,2)] Telkom University
tikatikiww94@gmail.com, lukedaely@gmail.com

Abstrak

Dynamic programming adalah salah satu algoritma yang digunakan untuk mengoptimalisasi nilai dari suatu permasalahan yang sangat kompleks dengan memecah permasalahan tersebut menjadi bagian-bagian dari sebuah permasalahan. Pada dasarnya, dalam algoritma dynamic programming, pemecahan suatu masalah dapat dibagi menjadi beberapa tahapan sehingga jalan keluar/solusi dari sebuah persoalan yang ada dapat dipandang sebagai hasil optimum yang dapat dijadikan keputusan yang saling berkaitan dengan mengacu pada rangkaian keputusan. Rangkaian keputusan berguna untuk mencoba semua kemungkinan-kemungkinan dari rangkaian-rangkaian keputusan. Rangkaian keputusan yang telah dihasilkan memberikan solusi total yang optimum. Rangkaian tersebut berisikan sub-sub rangkaian optimum atau tidak dapat optimum jika konsep berpengaruh pada proses perhitungan angka sehingga tidak akan menghasilkan hasil yang optimum. Penerapan metode dynamic programming ini amat penting dalam bidang matematika untuk menghitung nilai maximum sum subrectangle. Permasalahan maximum sum dari subrectangle pada array 2-dimensi adalah permasalahan umum yang sering terjadi. Untuk menyelesaikan masalah tersebut dapat digunakan dengan metode dynamic programming. Dengan menggunakan metode dynamic programming, hasil dari perhitungan maksimumnya akan menjadi lebih optimum. Makalah ini akan membahas mengenai penggunaan algoritma dynamic programming dengan menggunakan algoritma Kadane pada penerapannya dalam pembelajaran matematika dengan contoh penggunaannya sebagai fungsi pengoptimalisasi dari algoritma Kadane sehingga memiliki hasil kompleksitas O(N^3).

Kata Kunci: *dynamic programming; subrectangle; optimisasi; algoritma; kompleksitas*

1. PENDAHULUAN

Dynamic programming adalah pendekatan dalam pemecahan suatu masalah yang menggunakan algoritma yang di desain dalam menjawab suatu pemecahan masalah, dan perhitungan yang akan digunakan harus dikembangkan agar sesuai dengan tiap-tiap situasi tertentu. Oleh sebab itu, diperlukan adanya suatu pemahaman pada struktur umum permasalahan untuk mengetahui bagaimana dan kapan suatu permasalahan dipecahkan dengan dynamic programming. Dynamic programming menggunakan teknik *bottom-up* yang mencari nilai terkecil dengan mengkombinasikan solusi-solusi dengan memperoleh jawaban untuk menyelesaikan suatu permasalahan algoritma.

Tidak adanya formulasi matematis yang standar digunakan pada dynamic programming. Array merupakan bagian dari struktur data dasar dimana memiliki indeks yang berfungsi sebagai penanda suatu obyek dengan obyek lainnya di dalam sekumpulan obyek dengan tipe yang sama. O(1) merupakan operasi pengaksesan suatu obyek ke-i yang dilakukan sangat cepat pada array. Pencarian linear dengan kompleksitas O(N)/binary search dengan kompleksitas O(log N) berfungsi sebagai pencarian nilai obyek array berdasarkan aturan tertentu jika obyek yang ada telah terurut. Dimana array dapat berisikan obyek bertipe bilangan dan berdimensi lebih dari

satu. Dimana adanya 2 perbedaan mendasar diantara dynamic programming dan linier programming. Perbedaan pertama, tidak ada algoritma yang dapat diprogramkan untuk menyelesaikan semua permasalahan yang muncul. Sebaliknya, dynamic programming adalah suatu teknik yang mengarahkan kita untuk menyelesaikan masalah yang susah menjadi tahapan dari beberapa masalah yang lebih mudah, yang kemudian di evaluasi berdasarkan tahapan. Perbedaan kedua, linier program pada waktu. Dynamic programming mempunyai kemampuan untuk mencari solusi optimum dari suatu permasalahan menjadi beberapa permasalahan dalam satuan waktu yang lebih kecil dan memecahkan tiap permasalahan tersebut dengan optimum. Pada dynamic programming menggunakan pendekatan banyak tahap (multistage) dalam mengimplementasikan ke dalam mencari sebuah solusi. Peneliti memilih pendekatan banyak tahap karena pada penerapannya digunakan untuk mengoptimisasi. Dimana menemukan solusi dengan nilai pada array 2-dimesinya minimum atau maksimum.

Menurut Bellman (1957) prinsip optimalitas merupakan serangkaian keputusan awal dengan keadaan awal yang mempunyai solusi total optimum, maka bagian solusi sampai tahap ke-k juga optimum. Kondisi ini memberikan bagian-bagian dimana saat menghitung angka-angka pada array 2-dimensi sehingga jika ada angka bernilai negatif, maka lebih mudah mendapatkan hasil yang minimum. Jika semuanya bernilai positif akan lebih mudah mencari nilai maksimum yang paling optimum. Sehingga tujuan dari penelitian ini dapat mengetahui hasil dari penerapan dynamic programming dari perhitungan angka-angka maksimum menggunakan desain algoritma yang menjadi lebih optimum. Diberikan sebuah array 2-dimensi berisikan integer, mencari subrectangle dari array tersebut yang memiliki sum terbesar. Sum pada subrectangle adalah hasil penjumlahan seluruh integer pada subrectangle. Pada masalah ini, subrectangle yang memiliki sum terbesar disebut maximum sum subrectangle.

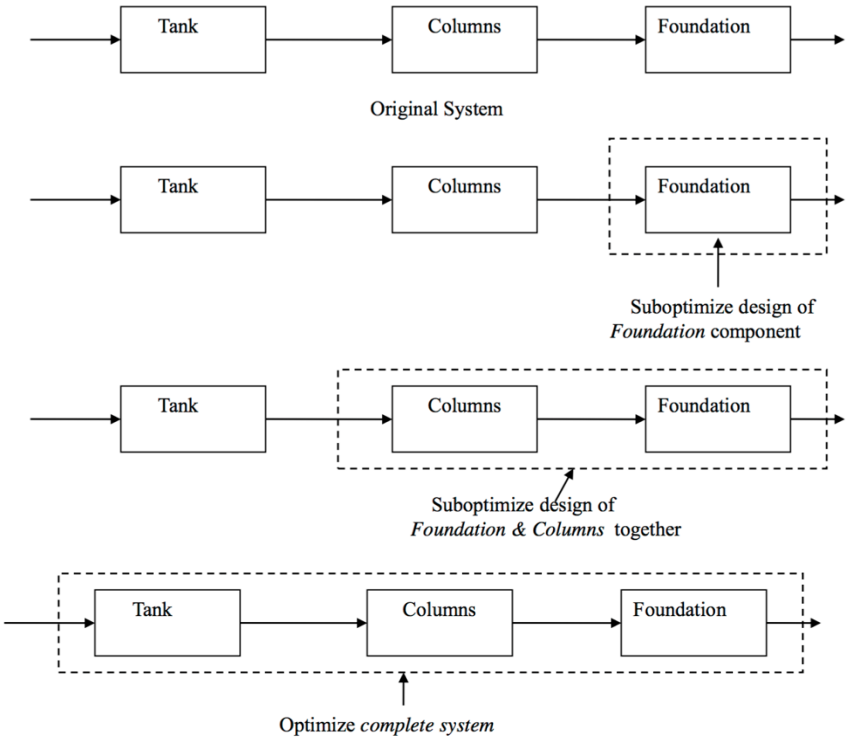
Pada dynamic programming mempunyai 4 tahapan dalam menyelesaikan sebuah permasalahan:

1. Berdasarkan aturan keputusan pada tiap-tiap tahapan. Dimana tahapan dikatakan sebagai permasalahan asli yang dipecah menjadi bagian dari sebuah permasalahan.
2. Dilakukannya pemecahan pada tahapan terakhir dari permasalahan dengan semua kondisi dan keadaan yang memungkinkan.
3. Bekerja secara mundur dari tahap terakhir dan melakukan pembagian bagian dari setiap tahapannya.
4. Pada finalnya bahwa dikatakan solusi optimum jika semua tahapan telah berhasil terpecahkan dari permasalahan yang ada.

Pada penelitian sebelumnya, untuk menjawab permasalahan maximum subarray berisi obyek yang bertipe bilangan dan mempunyai 1-dimensi. Dengan studi kasus pencarian rentang dan jumlah terbesar dari sebuah array. Sehingga pada 1-dimensi dengan adanya koneksi matematis digabungkan dengan analisis dalam penerapan algoritma memiliki keterkaitan yang kuat menjadi array 2-dimensi. Penelitian ini menggunakan array 2-dimensi dimana tersusun rangkaian-rangkaian keputusan yang optimum dibuat dengan menggunakan prinsip optimalisasi dengan menggunakan teknik *bottom-up*. Pada teknik *bottom-up* itu sendiri mempunyai tahapan-tahapannya dengan dimulai dari tahap n, terus mundur ke tahap $n - 1$, $n - 2$, dan seterusnya sampai tahap 1. Rangkaian keputusan pada penerapan dynamic programming adalah $x_n, x_{n-1}, \ldots, x_1$. Maksud dari prinsip optimalitas adalah jika solusi total optimum, maka bagian solusi sampai tahap ke-k juga optimum. Dengan prinsip optimalitas ini memberikan jaminan bahwa pengambilan dari keputusan-keputusan pada suatu tahap adalah keputusan yang baik dan sebagai solusi untuk tahap-tahap berikutnya. Subarray yang digunakan bernilai kontigu dalam permasalahan.

2. METODE PENELITIAN

Dalam pemecahan sebuah masalah maximum sum subrectangle, akan digunakan teknik dynamic programming dengan penerapan algoritma Kadane. Algoritma Kadane adalah teknik yang digunakan untuk memecahkan masalah maximum subarray problem. Untuk mengaplikasikan algoritma Kadane pada array 2-dimensi kita perlu menghitung prefix sum tiap baris pada array sehingga kita dapat memecahkan masalah maximum sum subrectangle seperti memecahkan masalah maximum subarray menggunakan algoritma Kadane. Algoritma Kadane pada 1-dimensi memiliki banyak data yang digunakan untuk mengurangi waktu kompleksitas ke O(N^3). Hasil kompleksitas O(N^3) sebagai fungsi pengoptimalisasi yang digunakan dan diimplementasi ke dalam algoritma agar menghasilkan nilai 2-dimensi yang optimum. Konsep dari proses optimalisasi dapat dilihat dalam setiap tahapan adalah sebagai berikut.



Gambar 1. Proses Optimalisasi

Ide awalnya yang diusulkan ada 2 yaitu untuk melakukan perbaikan pada kolom kiri dan kolom kanan satu per satu dan untuk menemukan jumlah maksimum baris yang bersebelahan untuk setiap pasangan kolom kiri dan kolom kanan. Pada dasarnya yaitu saat menemukan nomor baris atas dan baris bawah (yang memiliki jumlah maksimum) untuk setiap pasangan kolom kiri dan kolom kanan itu tetap. Untuk menemukan nomor baris atas dan baris bawah, hitung elemen di setiap baris dari kiri ke kanan dan simpan jumlah ini dalam array katakanlah temp[]. Jadi temp[i] menunjukkan jumlah elemen dari kiri ke kanan di baris i. Jika kita menerapkan algoritma Kadane 1-dimensi pada temp[], dan mendapatkan jumlah minimum subarray temp, jumlah maksimum ini adalah jumlah maksimum yang mungkin dengan kolom kiri dan kolom kanan sebagai batas. Batas dalam penelitian ini terdiri dari kumpulan jumlah angka yang digunakan dan adanya penyangga dari jumlah angka tersebut yang dibuat. Untuk mendapatkan jumlah keseluruhan secara menyeluruh, peneliti membandingkan jumlah yang ada dengan jumlah maksimum sejauh ini. Bahwa hasil yang disebutkan berada dalam batas kanan dan batas kiri yang berada secara

keseluruhan berada dalam jumlah maksimum.

3. HASIL PENELITIAN DAN PEMBAHASAN

Pada algoritma Kadane memiliki batas yang terdapat dalam kolom kanan dan kolom kiri. Dimana mempunyai hasil jumlah angka yang maksimum. Hasil dari angka maksimum dalam struktur data program yang tertera di bawah menunjukkan batas yang memiliki jumlah angka 1 dengan batas maksimum sum subrectangle. Dengan tipe data yang digunakan dalam struktur data dalam gambar yaitu int n. Maksud dari int adalah singkatan dari integer. Integer itu sendiri adalah salah satu tipe data pada struktur data dalam sebuah program dimana berupa bilangan bulat. Bilangan bulat merupakan suatu nilai dalam bentuk decimal maupun hexadecimal. Adanya berbagai macam tipe data numerik yang termasuk integer adalah sebagai berikut:

1. Byte: Tipe data yang bernilai integer dari -128 campai +127 dimana menempati 1 byte (8 bits) pada memori yang ada.
2. Short: Tipe data yang bernilai integer dari -32768 campai 32767 dimana menempati 2 bytes (2 x 8 bits = 16 bits) pada memori yang ada.
3. Int: Tipe data yang bernilai integer dari -2147483648 sampai 2147483647 dimana menempati 4 bytes (4 x 8 bits = 32 bits) pada memori yang ada.
4. Long: Tipe data yang bernilai integer dari -9223372036854775808 sampai 9223372036854775807 dimana menempati 8 bytes (8 x 8 bits = 64 bits) pada memori yang ada.

Struktur data pada Gambar 2 merupakan penyimpanan data yang dibuat sehingga data di dalam media penyimpanan komputer digunakan secara efisien. Struktur data pada permasalahan maksimum sum subrectangle ini adalah batas yang berada pada bagian kolom kiri dan kolom kanan pada nilai A.

```
int n, A[101][101], maxSubRect,
subRect;
```

Gambar 2. Struktur data program

Pada algoritma Kadane memiliki batas yang terdapat dalam kolom kanan dan kolom kiri yang menghasilkan hasil yang paling optimum saat memproses suatu data. Dalam penginputan data dan perhitungan prefix sum memperoleh hasil yang sum subrectangle dua persegi panjang. Saat peneliti mengurangi wilayah dalam persegi panjang dengan penambahan waktu. Sehingga adanya waktu tambahan dalam input data dan menghitung hasil prefix sum dalam proses algoritma Kadane.

```
scanf("%d", &n);
 // the dimension of input square
matrix
for (int i = 0; i < n; i++)
  for (int j = 0; j < n; j++) {
    scanf("%d", &A[i][j]);
    if (j > 0) A[i][j] += A[i][j - 1];
// only add columns of this row i
  }
```

Gambar 3. Input data dan menghitung prefix sum

Pada Gambar 3 dapat diketahui implementasi dari algoritma Kadane dimana

melakukan perhitungan maxSubRectangle dengan bertipe data integer dengan nilai 32 bit. Dengan baris dalam implementasi dari algoritma Kadane yang mempunyai dimensi lebih dari satu. Dengan penerapan 4 tahapan dalam dynamic programming di dalam algoritma Kadane memiliki solusi dengan optimisasi yang kompleks pada proses perhitungannya. Solusi yang dapat dilihat dalam gambar memiliki baris 0. Dengan hasil penginputan data tidak ada yang bernilai negatif sehingga lebih mudah dalam mendapatkan nilai maksimum yang paling optimum. Subrectangle pada sum memiliki hasil penjumlahan yang signifikan dengan jumlah keseluruhan dari integer sehingga teknik dynamic programming ini melakukan perhitungan dengan sebaik mungkin untuk dapat menghasilkan nilai angka yang maksimum.

```
maxSubRect = 0x7fffffff;
// the lowest possible value for 32-bit
integer
for (int l = 0; l < n; l++)
  for (int r = l; r < n; r++) {
    subRect = 0;
    for (int row = 0; row < n; row++) {

     // Max 1D Range Sum on columns
of this row i
      if (l > 0) subRect += A[row][r] -
A[row][l - 1];
     else       subRect += A[row][r];

      // Kadane's algorithm on rows
     if (subRect < 0) subRect = 0;
 // greedy, restart if running sum < 0
     maxSubRect = max(maxSubRect,
subRect);
    }
  }
```

Gambar 4. Implementasi algoritma Kadane untuk menghitung maxSubRectangle

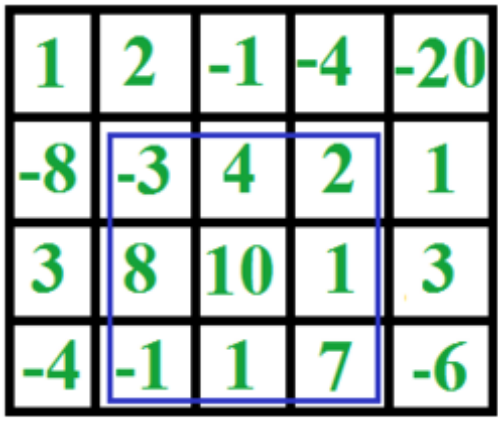
Penjelasan Algoritma di atas:

Dengan algoritma Kadane dengan penggunaan teknik prefix sum. Dimana prefix sum adalah hasil penjumlahan angka dari kolom-kolom sebelumnya. Sehingga didapatkan nilai maksimumnya. Penjumlahan kolomnya dilakukan peneliti saat melakukan pergeseran letak angka setiap ke kanan dan ke bawah pada persegi panjang.

Sebagai ilustrasi algoritma di atas perhatikan contoh berikut:
s = -, t = 0, j = 0.
Langkah:
1. (1 3 -6 5 8 -2
2. (1)| 3 -6 5 8 -2
3. (1 3)|-6 5 8 -2
4. 1 3 (-6)| 5 8 -2
5. 1 3 -6 (5)| 8 -2
6. 1 3 -6 (5 8)|-2
7. 1 3 -6 (5 8) -2|

Penjelasan:
1. Kondisi awal array, batas kiri berada pada element pertama array.
2. Kita set t=1, karena t>s, set s=1
3. Kita set t=4, karena t>s, set s=4
4. Kita set t=-3, karena t<0, set t=0 dan j=4
5. Kita set t=-3, karena t>s, set s=5
6. Kita set t=13, karena t>s, set s=13
7. Kita set t=10, karena t<s, maka tidak dilakukan apa-apa.

Kita dapatkan maximum subarray adalah [5,8]



Gambar 5. 2D Array pada perhitungan maximum subarray

Dengan menggunakan dynamic programming, kita dapat mengoptimasi algoritma yang terlihat pada Gambar 6. Dimana tersedia jumlah subarray maksimal di dalamnya. Sebagai contoh, dalam array 2D berikut, subarray jumlah maksimum disorot dengan kotak biru dan jumlah subarray ini adalah 29. Dengan keterangan pada Gambar 5 sebagai berikut.
(Top, Left) (1, 1)
(Bottom, Right) (3, 3)
Max sum is: 29
Time Complexity: O(N^3).

Penjelasan dari hasil yang dilakukan pemrosesan pemecahan setiap tahapan optimalitas yang dilakukan dengan menggunakan teknik dynamic programming menggunakan penerapan algoritma Kadane memiliki hasil yang kompleks dan bernilai maksimum dengan hasil kompleksitas membentuk array 2-dimensi pada baris top, left, bottom dan right.

4. SIMPULAN

Dalam hasil penelitian dan pembahasan penelitian ini, didapatkan kesimpulan dengan menggunakan teknik dynamic programming, peneliti mendapatkan hasil dalam mengoptimasi algoritma hingga memiliki kompleksitas O(N^3) pada array 2-dimensi. Pendekatan dalam pemecahan suatu masalah yang menggunakan algoritma Kadane memberikan pemecahan masalah yang dapat diperhitungkan dengan teknik *bottom-up* yang mencari nilai maksimum dengan hasil yang optimum dalam penerapannya.

Saran kepada peneliti dalam penelitian kedepannya yaitu pengaplikasian teknik dynamic programming dengan menggunakan algoritma Kadane pada 2-dimensi dapat dilakukan dalam pemecahan masalah yang bermacam-macam. Sehingga tantangan

peneliti yaitu mencari studi kasus yang dapat dilakukan dalam penerapan teknik dynamic programming ke depannya. Karena penerapan dynamic programming menggunakan pendekatan banyak tahap (multistage) dalam mengimplementasian ke dalam mencari sebuah solusi dari pemecahan masalah dengan lebih inovatif. Dan dapat diterapkan ke dalam kehidupan sehari-hari dari pembelajaran matematika menggunakan konsep matematika yang berbeda dengan mengkaitkan teknologi di dalamnya.

5. DAFTAR PUSTAKA

Brassard, Gilles & Bratley, Paul. (1988). *Algorithmics Theory & Practice*. Englewood Cliffs, NJ: Prentice Hall.

Wijaya, Ariyadi., Heuvel-Panhuizen, Marja van den., Doorman, Michiel., & Robitzsch, Alexander. (2014). Difficulties in Solving Context Based PISA Mathematics Tasks: An Analysis of Student's Errors. *The Mathematics Enthusiast*, **11**(3), 555 – 585.

Fathoni, M., & Triprabowo, Ari. (2012). Pencarian Rute Terpendek dengan Menggunakan Dynammic Programming. Diakses dari http://web.unair.ac.id/admin/file/f_12649_paper_dynamic_programming.pdf

Najogie, Reinhard Denis. (2012). Perbandingan Algoritma Brute Force, Divide and conquer, dan Dynamic Programming untuk Solusi Maximum Subarray Problem. Diakses dari http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2011-2012/Makalah2011/MakalahIF3051-2011-088.pdf

Dynammic Programming and Applications. Introduction and Preliminaries. Diakses dari http://content.inflibnet.ac.in/data-server/eacharya-documents/55da9cb2e41301e54e2caeb7_INFIEP_206/4218/ET/206-4218-ET-V1-S1__lecture1.pdf

Geeks for Geeks. Dynamic Programming I Set 27 (Maximum sum rectangle in a 2D matrix). Diakses dari http://www.geeksforgeeks.org/dynamic-programming-set-27-max-sum-rectangle-in-a-2d-matrix/

Dewanthi, Sinta Sih. (2015). Pengembangan Bahan Ajar Geometri Analitik Berbasis Guided Discovery untuk Memfasilitasi Berpikir Kritis. Dalam Murtadho, Ali (Eds.) *Peran Matematika dan Pendidikan Matematika Dalam Menghadapi Isu-isu Global: Prosiding Seminar Nasional Matematika dan Pendidikan Matematika*, Diselenggarakan oleh Program Studi Pendidikan Matematika, UMS, 7 Maret 2015 (hal. 187-199). Surakarta: Muhammadiyah University Press. Diakses dari https://publikasiilmiah.ums.ac.id/handle/11617/5988

Bellman, R. E. (1957). Dynamic Programming, Princeton University Press, Princeton, NJ.

Bellman, R. E. & Dreyfus, S. E. (1962). Applied Dynamic Programming. Princeton University Press Dynamic Programming, Princeton University Press, Princeton, NJ.

Urbanek, F. J. (1962). An O(log n) Algorithm for Computing The *n* th Element of the Solution of a Difference Equation, Information Processings Letters, 11(2), 66-67.

Sniedovich, Moshe. (1978). Dynamic Programming and Principles of Optimality, Journal Of Mathematical Analysis and Applications 65, 586-606.