Penyelesaian n query koefisien binomial dengan waktu linier dan perbandingannya dengan pendekatan segitiga pascal

Emery Fathan Zwageri-13522079
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): emeryzwageri@gmail.com

Penentuan koefisien binomial seringkali menjadi permasalahan umum dalam bidang kombinatorika dan analisis algoritma. Koefisien binomial, yang dinyatakan sebagai C(n,k), merupakan jumlah cara untuk memilih k elemen dari n elemen tanpa memperhatikan urutan. Penyelesaian langsung dari perhitungan koefisien binomial dapat dilakukan dalam waktu komputasi yang konstan, namun tantangan muncul ketika kita dihadapkan pada banyak query (n query) dalam satu waktu. Makalah ini mengusulkan sebuah metode untuk menyelesaikan n query koefisien binomial dengan waktu linier menggunakan konsep inverse modular. Teknik ini memanfaatkan sifat inverse modular untuk mempercepat perhitungan koefisien binomial dalam konteks modulo bilangan prima besar. Dengan memanfaatkan prinsip invers modular, kita dapat melakukan pra-pemrosesan dalam waktu untuk kemudian menjawab setiap query dalam waktu konstan. Hal ini dicapai melalui penggunaan rumus dasar koefisien binomial dan sifat-sifat modular arithmetic atau segitiga pascal yang memungkinkan optimasi perhitungan.

***Keywords*—Kombinatorik, inverse, query, pre-proses.**

I. INTRODUCTION

Koefisien binomial adalah konsep fundamental dalam matematika, terutama dalam kombinatorika, yang digunakan untuk menghitung jumlah cara memilih k elemen dari n elemen tanpa memperhatikan urutan. Koefisien binomial dilambangkan sebagai $C(n,k)$ dan dapat dihitung menggunakan rumus faktorial. Permasalahan ini sering muncul dalam berbagai aplikasi seperti analisis algoritma, teori probabilitas, dan pemrograman dinamis.

Namun, menghitung koefisien binomial secara langsung untuk sejumlah besar query dapat menjadi tidak efisien. Pendekatan naif dengan menghitung setiap koefisien binomial dari awal memiliki kompleksitas waktu $O(nq)$, yang tidak praktis untuk jumlah query yang besar. Oleh karena itu, diperlukan metode yang lebih efisien untuk menangani perhitungan ini.

Makalah ini mengusulkan metode untuk menyelesaikan n query koefisien binomial dengan waktu komputasi $O(n + \log MOD)$ menggunakan konsep invers modular. Pendekatan ini melibatkan pra-pemrosesan faktorial dan invers faktorial, yang memungkinkan setiap query dijawab dalam waktu konstan O(1). Hasil dari metode ini menunjukkan peningkatan signifikan dalam efisiensi komputasi dibandingkan dengan pendekatan tradisional.

II. THEORETICAL BASIS

A. Koefisien Binomial

Koefisien binomial nCk adalah jumlah cara untuk memilih k elemen dari n elemen tanpa memperhatikan urutan, yang diberikan oleh rumus:

$$(\binom{n}{k}) = n!/k!\,(n-k)!$$

di mana n!n! adalah faktorial dari n, yaitu hasil perkalian dari semua bilangan bulat positif kurang dari atau sama dengan n.

$$C(n,k) = \binom{n}{k} = \frac{n!}{k!\,(n-k)!}$$

Gambar 2.1 Koefisien Binomial
Sumber:
https://cdn.corporatefinanceinstitute.com/assets/combination2.png

B. Inverse Modular

Modular arithmetic adalah sistem aritmetika untuk bilangan bulat di mana bilangan "membungkus kembali" ke nilai awal setelah mencapai modulus tertentu. Dalam konteks ini, kita sering menggunakan modulus bilangan prima besar untuk menghindari overflow dan menjaga efisiensi komputasi.

Inverse modular dari suatu bilangan a modulo m adalah bilangan b sedemikian rupa sehingga:

$$a \cdot b \equiv 1 \ (mod\ m)$$

Invers modular dapat dihitung menggunakan Extended Euclidean Algorithm atau lebih efisien

menggunakan eksponensiasi modular ketika modulus adalah bilangan prima, berdasarkan Teorema Fermat Kecil:

$$a^{p-1} \equiv 1 \ (mod\ p)$$

$$a^{p-2} \equiv a^{-1} \ (mod\ p)$$

C. Eksponensiasi Modular

Eksponensiasi modular adalah teknik untuk menghitung xnmod mxnmodm secara efisien dengan kompleksitas waktu $O(log\ n)$. Hal ini dapat dilakukan menggunakan algoritma eksponensiasi cepat (fast exponentiation).

D. Dynamic Programming

Dynamic Programming (DP) adalah teknik pemrograman yang digunakan untuk menyelesaikan masalah kompleks dengan membaginya menjadi sub-masalah yang lebih sederhana. DP memanfaatkan sifat dari sub-masalah yang tumpang tindih dan solusi optimal dari sub-masalah yang lebih kecil untuk membangun solusi optimal untuk masalah keseluruhan.

Dalam konteks koefisien binomial, DP dapat digunakan untuk menghitung nilai (nk)(kn) menggunakan relasi rekursif berikut:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

Basis dari relasi ini adalah:

Algoritma DP untuk menghitung koefisien binomial membangun tabel dua dimensi CC di mana $C[n][k]C[n][k]$ menyimpan nilai$\binom{n}{k}$ .Kompleksitas waktu dari pendekatan ini adalah $O(n^2)$.

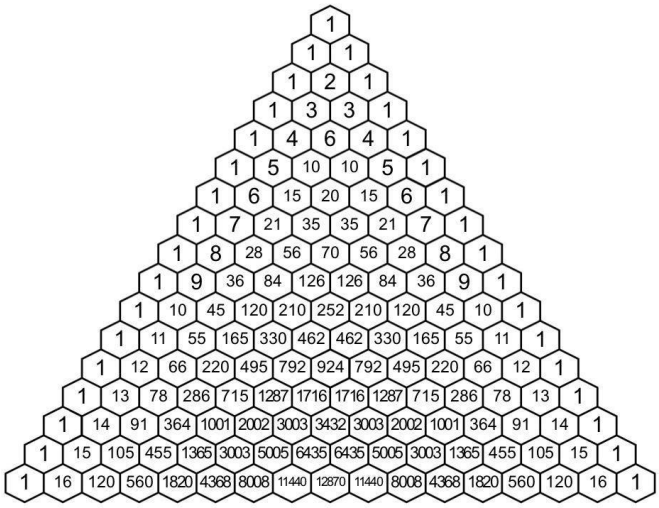
E. Segitiga Pascal

Segitiga Pascal adalah representasi segitiga dari koefisien binomial. Setiap elemen dalam segitiga adalah jumlah dari dua elemen yang berada tepat di atasnya. Struktur ini menggambarkan secara visual sifat rekursif dari koefisien binomial.

Baris ke-n dari Segitiga Pascal memberikan koefisien binomial $\binom{n}{k}$ untuk k dari 0 hingga n. Baris pertama dan terakhir selalu bernilai 1, dan elemen lainnya dihitung dengan menjumlahkan dua elemen di atasnya.

F. Bilangan Prima

Bilangan Prima adalah bilangan asli selain 1 yang hanya habis dibagi diri dia sendiri dan 1 contoh bilangan prima adalah 2, 3, 5, 7, 11.



Gambar 2.2 Ilustrasi Segitiga Pascal
Sumber:https://www.cantorsparadise.com/8-secrets-of-pascals-triangle-349cb5e46b09

III. IMPLEMENTATION

Berikut variable global yang digunakan

```cpp
#include <iostream>
#include <chrono>
#include <fstream>
using namespace std;
using namespace std::chrono;

const int MAXN = 1000;
const long long MOD = 1000000007;

long long fac[MAXN + 1];
long long inv[MAXN + 1];
```

Gambar 3.1 Variabel global
(Sumber: Dokumentasi Pribadi)

1. Teknik Inverse

Penyelesaian n query koefisien binomial dengan waktu n linier dilakukan dengan teknik prekomputasi dan menggunakan pendekatan inverse modular. Awal mula nya kita set global variabel untuk menyimpan nilai factorial dan inverse nya, lakukan prekomputasi untuk nilai faktorial dan nilai inverse nya. Lalu hitung nCr dengan rumus berikut

$$\binom{n}{r} = ((n!)(r)^{-1} mod\ MOD\ ) \times ((n-r)^{-1} mod\ MOD)$$

Dengan MOD adalah bilangan prima yang cukup besar

Untuk implementasi teknik inverse kita gunakan pendekatan berikut, gunakan fakta bahwa

$(n!)^{-1} \equiv (n!)^{-1} \times (n+1)^{-1} \times (n+1) \equiv ((n+1)!^{-1}) \times (n+1)$

Implementasi terbagi menjadi 2 yaitu pembuatan fungsi fungsi dan main program. Penyelesaian koefisien binomial dengan n query dapat diselesaikan dengan langkah langkah berikut.

A. Fungsi Eksponenensiasi

```
ll exp(ll x, ll n, ll m) {
    x %= m;
    ll res = 1;
    while (n > 0) {
        if (n % 2 == 1) { res = res * x % m; }
        x = x * x % m;
        n /= 2;
    }
    return res;
}
```

Gambar 3.2 Fungsi eksponensiasi
Sumber: Dokumentasi Pribadi

Fungsi ini mengembalikan nilai dari $x^m mod\ n$ dalam $log(p)$

B. Fungsi factorial

```
void factorial() {
    fac[0] = 1;
    for (int i = 1; i <= MAXN; i++) { fac[i] = fac[i - 1] * i % MOD; }
}
```

Gambar 3.3 Fungsi Factorial
Sumber: Dokumentasi pribadi

Fungsi ini menghitung nilai Factorial biasa.

C. Inverse Function

```
void inverses() {
    inv[MAXN] = exp(fac[MAXN], MOD - 2, MOD);
    for (int i = MAXN; i >= 1; i--) { inv[i - 1] = inv[i] * i % MOD; }
}
```

Gambar 3.4 Fungsi Inverse
Sumber: Dokumentasi pribadi

Fungsi ini melakukan prekomputasi untuk menghitung nilai inverse dari MAXN sampai 1.

D. Fungsi Choose

```
ll choose(int n, int r) { return fac[n] * inv[r] % MOD * inv[n - r] % MOD; }
```

Gambar 3.5 Fungsi Choose
(Sumber :Dokumentasi Pribadi)

Fungsi ini melakukan perhitungan akhir untuk jawaban pada query menggunakan pendekatan yang dijelaskan pada bagian sebelumnya yiatu:

$$(n!)^{-1} \equiv (n!)^{-1} \times (n+1)^{-1} \times (n+1)$$
$$\equiv ((n+1)!^{-1}) \times (n+1)$$

E. Fungsi main

```
int main() {
    ifstream infile("tc.txt");
    if (!infile.is_open()) {
        cerr << "Unable to open file for reading.\n";
        return 1;
    }

    int n;
    infile >> n;

    auto start = high_resolution_clock::now(); // Mulai menghitung waktu
    factorial(MOD);
    inverses(MOD);
    solve(infile, n);
    auto end = high_resolution_clock::now(); // Akhiri menghitung waktu
    auto duration = duration_cast<milliseconds>(end - start); // Hitung durasi dalam milidetik
    cout << "Waktu eksekusi: " << duration.count() << " milidetik\n";

    return 0;
}
```

Gambar 3.6 Fungsi main
(Sumber :Dokumentasi Pribadi)

2. Dynamic Programming

```
int main() {
    ifstream infile("tc.txt");
    if (!infile.is_open()) {
        cerr << "Unable to open file for reading.\n";
        return 1;
    }

    int n;
    infile >> n;

    auto start = high_resolution_clock::now(); // Mulai menghitung waktu
    binomial(1000, 1000, MOD);
    solve(infile, n);
    auto end = high_resolution_clock::now(); // Akhiri menghitung waktu
    auto duration = duration_cast<milliseconds>(end - start); // Hitung durasi dalam milidetik
    cout << "Waktu eksekusi: " << duration.count() << " milidetik\n";

    return 0;
}
```

Gambar 3.7 Fungsi main Segitiga Pascal
(Sumber: Dokumentasi Pribadi)

Pada solusi ini saya menggunakan segitiga pascal sebagai pendekatan. Teknik ini juga melakukan pre komputasi, tetapi teknik ini memerlukan kompleksitas kuadratik.

```cpp
/** @return nCk mod p using dynamic programming */
#include<bits/stdc++.h>
using namespace std;

int binomial(int n, int k, int p) {

    // dp[i][j] stores iCj

    vector<vector<int>> dp(n + 1, vector<int>(k + 1, 0));


    // base cases described above

    for (int i = 0; i <= n; i++) {

        /*

         * i choose 0 is always 1 since there is exactly one way

         * to choose 0 elements from a set of i elements

         * (don't choose anything)

         */

        dp[i][0] = 1;

        /*

         * i choose i is always 1 since there is exactly one way

         * to choose i elements from a set of i elements

         * (choose every element in the set)

         */

        if (i <= k) { dp[i][i] = 1; }

    }


    for (int i = 0; i <= n; i++) {

        for (int j = 1; j <= min(i, k); j++) {

            if (i != j) {  // skips over the base cases

                // uses the recurrence relation above

                dp[i][j] = (dp[i - 1][j - 1] + dp[i - 1][j]) % p;

            }

        }

    }


    return dp[n][k];  // returns nCk modulo p

}
```

Gambar 3.7 Segitiga Pascal pada perhitungan koefisien binomial

(Sumber: Dokumentasi Pribadi)

Pendekatan Dynamic Programming dalam menghitung koefisien binomial menggunakan segitiga pascal sebagai ide.

IV. ANALYSIS

Input diambil dari tc.txt

```
3
4 2
6
9 3
84
10 9
10
```

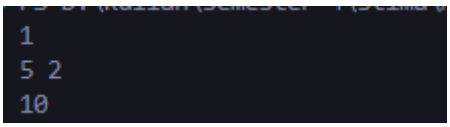
Gambar 4.1 solusi dari percobaan 1

(Sumber: Dokumentasi Pribadi)

```
PS D:\Kuliah\semester 4\stima\Makalah> ./binomial.exe
10
4 3
4
66 2
2145
22 5
26334
4 1
4
10 3
120
5 3
10
55 2
1485
55 34
810711579
44 2
946
44 6
7059052
```

Gambar 4.2 Solusi dari percobaan 2

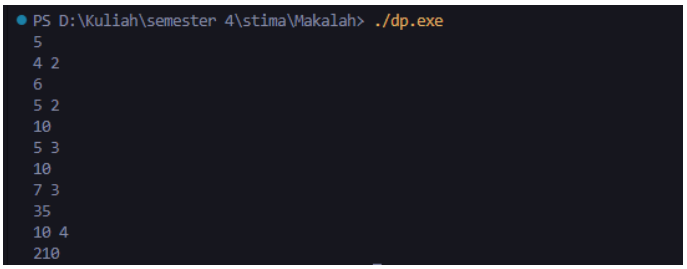
(Sumber: Dokumentasi Pribadi)



```
1
5 2
10
```

Gambar 4.3 Solusi percobaan 3 (Pascal)

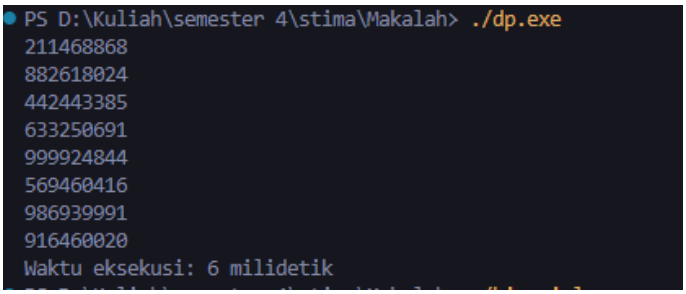
(Sumber: Dokumentasi Pribadi)



```
PS D:\Kuliah\semester 4\stima\Makalah> ./dp.exe
5
4 2
6
5 2
10
5 3
10
7 3
35
10 4
210
```

Gambar 4.4 Solusi Percobaan 4 (Pascal)

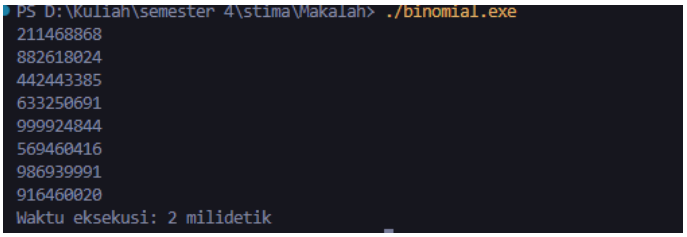
(Sumber : Dokumentasi Pribadi)



```
PS D:\Kuliah\semester 4\stima\Makalah> ./dp.exe
211468868
882618024
442443385
633250691
999924844
569460416
986939991
916460020
Waktu eksekusi: 6 milidetik
```

Gambar 4.5 Solusi Percobaan 5(Pascal)

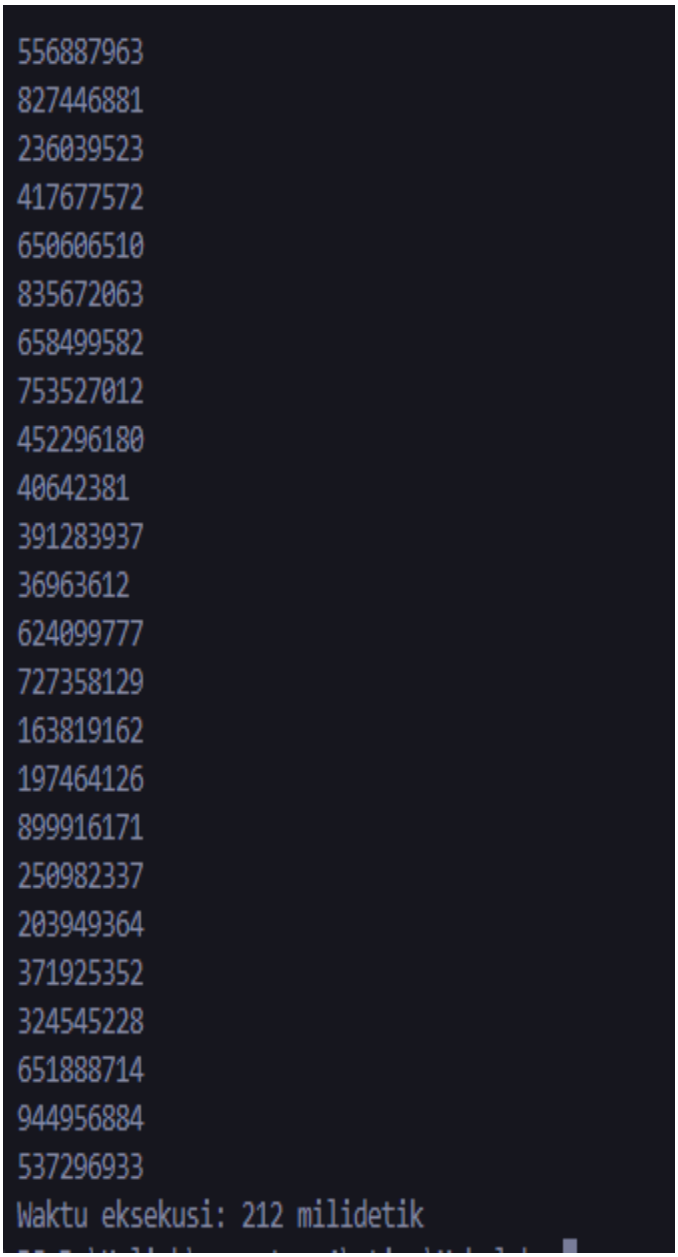
(Sumber : Private Documentation)



```
PS D:\Kuliah\semester 4\stima\Makalah> ./binomial.exe
211468868
882618024
442443385
633250691
999924844
569460416
986939991
916460020
Waktu eksekusi: 2 milidetik
```

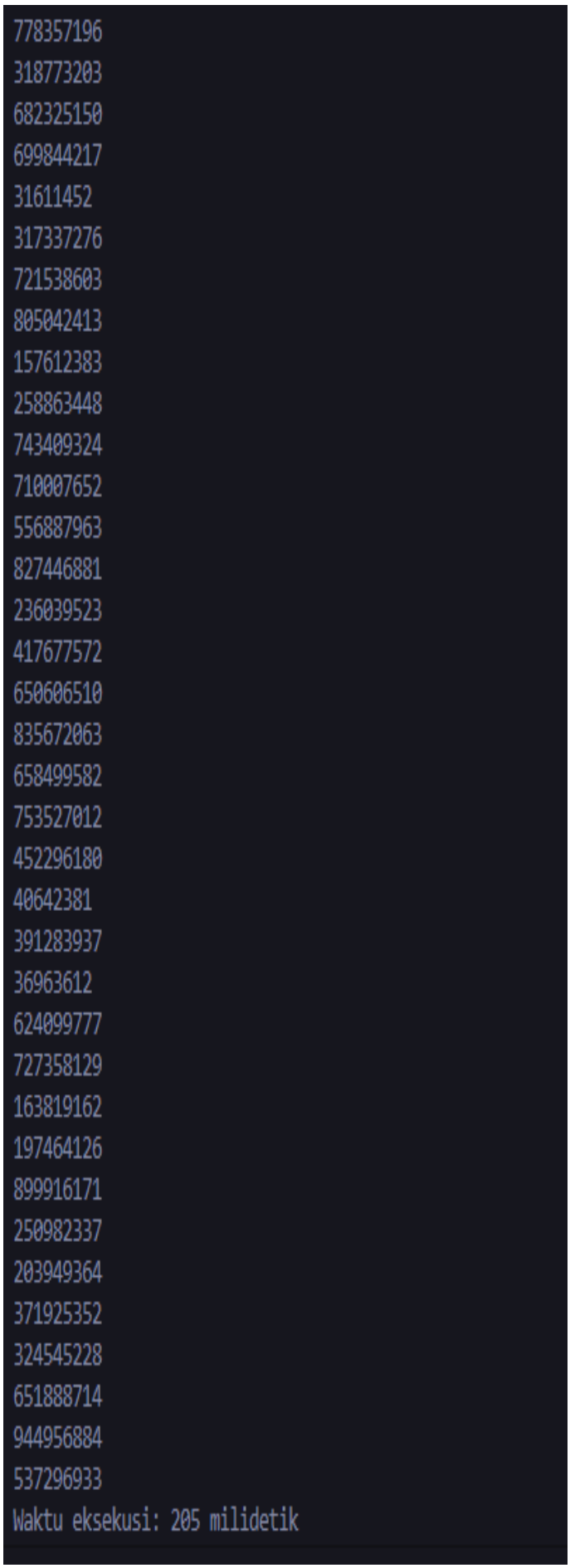
Gambar 4.6 Solusi Percobaan 5(Inverse)

(Sumber : Sumber Pribadi



Gambar 4.7 Solusi percobaan 6(Pascal)

Pada percobaan 6 dengan segitiga pascal memiliki waktu eksekusi 212 milidetik sedangkan teknik inverse modular 205 milidetik



Gambar 4.8 Solusi percobaan 6(inverse)
(Sumber: Dokumentasi Pribadi)

```
900
361 270
631 462
928 507
759 26
790 772
923 259
837 478
836 514
951 421
897 453
432 89
917 803
947 480
999 786
835 473
512 337
468 362
964 643
727 693
389 115
877 777
422 307
871 528
887 814
864 785
969 636
976 637
938 662
614 415
846 564
898 520
975 369
766 348
934 822
301 59
749 490
685 127
```

Gambar 4.9 Test Case untuk percobaan 6
(Sumber: Dokumentasi Pribadi)

```
8
361 270
631 462
928 507
759 26
790 772
923 259
837 478
836 514
```

Gambar 4.10 Test Case untuk percobaan 5
(Sumber: Dokumentasi Pribadi)

Dari percobaan percoban 5 dan 6 dapat disimpulkan bahwa penggunaan teknik Inverse dalam menyelesaikan masalah koefisien binomial dengan n query lebih baik karena sesuai perhitungan kompleksitas waktu nya, yaitu $O(n + log\ MOD)$ (linier) sedangkan dengan Dynamic Programming kompleksitas waktunya $O(n^2)$. Ditambah lagi Dengan Dynamic Programming tidak bisa menyelesaikan query sebanyak teknik inverse karena hanya menggunakan 1 array untuk memoisasi DP nya sedangkan modular inverse menggunakan array FAC dan array INV. Keduanya menggunakan Dynamic Programming.

V. EVALUASI

Berikut evaluasi kelebihan dan kekurangan menggunakan dp segitiga pascal dan teknik *inverse* modular.

Keuntungan dp pascal:
1.Pembuatan lebih mudah
2. Ide pendekatan lebih gampang
Ini karena ide dari menggunakan dp nya *straightforward* dan tidak perlu teknik *advance* seperti *inverse* modular
Kekurangan dp pascal:
1. Lebih lambat karena memiliki array 2 dimensi yang berarti kompleksitasnya $O(n^2)$.
2. Tidak bisa menerima Query yang sangat banyak karena hanya menggunakan 1 array 2 dimensi.

Kelebihan Inverse modular:
1. Lebih cepat karena hanya membutuhkan waktu linier
2. Dapat menerima input query yang lebih besar karena menggunakan 2 array yaitu INV dan FAC yang masing masing menyimpan nilai inverse dan factorial.

Kekurangan Inverse modular:
1.Lebih susah untuk diimplentasi karena memiliki pendekatan yang lebih *advance.*

VI. CONCLUSION

Penyelesaian koefisien binomial dengan n query menggunakan teknik inverse modular lebih cepat dan lebih efisien dari pada menggunakan segitiga pascal. Penggunaan teknik inverse modular juga dapat menerima input lebih query lebih banyak dari segitiga pascal. Namun, implementasi segitiga pascal lebih gampang karena pendekatannya cukup sederhana sedangkan pendekatan inverse modular lebih *advance* karena menggunakan konsep matematika yang lebih dalam. Intinya adalah keduanya sama sama bagus dan dapat menyelesaikan permasalahan koefisien binomial dengan n query dengan optimal, kembali lagi penggunaannya tergantung kebutuhan, misal jika ingin menyelesaikan dengan query besar kita bisa menggunakan teknik inverse modular, tetapi kalau ingin menyelesaikan tidak mementingkan waktu dan ingin program yang sederhana saja bisa menggunakan segitiga pascal. Selain itu kedua teknik diatas menggunakan Dynamic Programming dalam menyelesaikan masalahnya.

VII. UCAPANTERIMA KASIH

Penulis ingin mengucapkan rasa syukur kepada beberapa pihak atas kontribusi mereka untuk makalah ini. Pertama-tama, penulis ingin memanjatkan rasa syukur kepada Allah Swt. Atas bimbingannya dalam keseluruhan proses

pembelajaran dan penulisan makalah ini. Berikutnya, penulis ingin mengucapkan terima kasih kepada dosen IF2210 Strategi Algoritma Pak Rinaldi Munir, atas dukungan dan ilmu-ilmu yang diberikan. Pengetahuan dan bimbingan dari Pak Rinaldi sangat membantu dalam proses belajar penulis. Terima kasih juga ingin penulis berikan pada keluarga dan teman-teman penulis, atas dukungan mereka selama keseluruhan semester.

REFERENSI

[1] https://artofproblemsolving.com/wiki/index.php/Factorial/ Diakses pada 12 Juni 2024
[2] https://www.geeksforgeeks.org/binomial-coefficient-dp-9/ Diakses pada 11 juni 2024
[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian1.pdf Diakses pada 10 juni 2024
[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Program-Dinamis-2020-Bagian2.pdf Diakses pada 10 juni 2024
[5] https://www.cantorsparadise.com/8-secrets-of-pascals-triangle-349cb5e46b09 Diakses pada 11 juni 2024

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, 12 Juni 2024

Emery Fathan Zwageri 13522079