Penerapan Program Dinamis dalam Optimalisasi Build Karakter Final Fantasy VII : Sistem Materia

Frederiko Eldad Mugiyono - 13523147[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]frederiko100405@gmail.com, 13523147@std.stei.itb.ac.id

***Abstract*—*Final Fantasy VII* adalah *role-playing game* (*RPG*) ikonik yang dikenal luas karena alur cerita epik dan sistem pengembangan karakter mendalam, termasuk Sistem *Materia*. Sistem ini memungkinkan pemain menyesuaikan kemampuan dan statistik karakter secara dinamis dengan memasang bola kristal berisi sihir, perintah, atau peningkatan statistik. Makalah ini menerapkan Program Dinamis (DP) untuk mengoptimalkan *build* karakter dalam gim tersebut. Kompleksitas Sistem *Materia*, dengan interaksi antar-*Materia*, efek sinergis, serta dampaknya pada statistik (termasuk penalti), seringkali menimbulkan tantangan optimasi. Dengan memodelkan masalah ini sebagai DP, penulis menyediakan kerangka matematis terstruktur untuk memverifikasi dan mengoptimalkan pilihan *Materia*, memungkinkan eksplorasi ruang keputusan yang efisien. Hasil analisis pada studi kasus *build* Tifa Lockhart yang berfokus pada *Strength* dan *balance* membuktikan bahwa DP mampu memetakan pilihan *Materia* secara sekuensial untuk memaksimalkan statistik tertentu, sekaligus secara otomatis memperhitungkan efek kumulatif dan penalti *Materia*. Meskipun demikian, "kutukan dimensionalitas" tetap menjadi tantangan utama untuk *build* yang lebih kompleks.**

Kata Kunci—*Final Fantasy VII*, Sistem *Materia*, Program Dinamis, Optimasi *Build* Karakter, Tifa Lockhart.

I. Pendahuluan

Final Fantasy VII merupakan salah satu *role-playing game* (RPG) ikonik yang dikenal luas karena alur cerita epik, karakter yang *memorable*, dan sistem pengembangan yang mendalam. Salah satu fitur paling inovatif dari gim ini adalah **Sistem *Materia*** atau lebih dikenal sebagai ***Materia System***. Sistem ini memungkinkan pemain untuk dapat menyesuaikan kemampuan dan statistik dari karakter mereka secara dinamis dengan memasang *Materia* – bola kristal yang mengandung sihir, *command abilities*, atau peningkatan statistik – ke dalam *slot* yang tersedia pada senjata dan zirah.

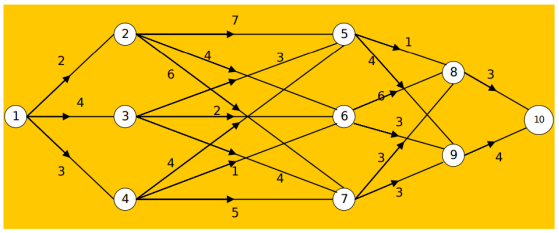
Gambar 1.1. Cover Produk *Final Fantasy VII* (Sumber:https://store.steampowered.com/app/39140/FINAL_FANTASY_VII)

Final Fantasy VII dikembangkan oleh Square (sekarang Square Enix), pertama kali dirilis pada Januari 1997. Gim ini telah menerima pujian kritis dan kesuksesan komersial yang luas, dengan perilisan ulang di berbagai *platform* dan dampak yang signifikan pada industri gim. Setiap *build* karakter yang optimal dapat dicapai berdasarkan *Materia* yang dipilih dan dikombinasikan pemain. Untuk mencapai *build* karakter yang paling efisien, metode Program Dinamis dapat digunakan untuk secara sistematis mengeksplorasi semua kemungkinan kombinasi *Materia* dan efek yang dihasilkan, sehingga memastikan bahwa konfigurasi optimal tidak terlewatkan.

II. Dasar Teori

A. Program Dinamis

Program Dinamis (*Dynamic Programming*) merupakan metode pemecahan masalah yang didasarkan pada strategi menguraikan solusi menjadi sekumpulan tahapan (*stage*). Dengan demikian, solusi dari persoalan secara keseluruhan dapat dipandang sebagai rangkaian keputusan yang saling berkaitan, dengan setiap keputusan diambil pada suatu tahap tertentu. Pendekatan ini secara khusus digunakan dalam penyelesaian persoalan-persoalan optimasi, baik maksimasi (misal, mencari keuntungan maksimum) maupun minimasi (misal, mencari lintasan terpendek).



Gambar 2.1. Persoalan Mencari Lintasan Terpendek (Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/25-Program-Dinamis-(2025)-Bagian1.pdf, halaman 4)

Penting untuk dicatat bahwa frasa "program" dalam "Program Dinamis" tidak memiliki kaitan secara langsung dengan pemrograman komputer dalam konteks bahasa pemrograman. Sebaliknya, istilah ini merujuk pada suatu "rencana" atau "prosedur" sistematis untuk menyelesaikan masalah.

Istilah "dinamis" muncul karena proses pencarian solusi melibatkan perhitungan dan penyimpanan hasil-hasil antara dalam sebuah tabel yang dapat berkembang.

B. Perbedaan dengan Algoritma Greedy

Program Dinamis dan Algoritma Greedy sama-sama digunakan untuk menyelesaikan persoalan optimasi. Namun, terdapat perbedaan mendasar dalam cara mereka mencapai solusi optimal. Algoritma Greedy bekerja dengan menghasilkan hanya satu rangkaian keputusan. Pada setiap langkah, algoritma ini membuat keputusan yang tampak paling optimal secara lokal, tanpa perlu mempertimbangkan konsekuensi jangka panjang dari keputusan tersebut. Akibatnya, keputusan lokal tersebut tidak selalu menjamin optimalisasi secara global.

Sebaliknya, Program Dinamis mempertimbangkan lebih dari satu rangkaian keputusan dengan secara sistematis untuk mengeksplorasi berbagai kemungkinan keputusan di setiap tahap. Sehingga memastikan bahwa pilihan yang dibuat mengarah kepada solusi optimal secara global. Perbedaan ini dapat diilustrasikan dengan jelas melalui Persoalan Lintasan Terpendek (Gambar 2.1) yang akan dipetakan ke dalam tabel berikut.

Algoritma	**Jalur yang dipilih**	**Biaya Kumulatif**
Greedy	1 → 2 (biaya 2)	2
	1 → 2 → 6 (biaya 4)	6
	1 → 2 → 6 → 9 (biaya 3)	9
	1 → 2 → 6 → 9 → 10 (biaya 4)	**13**
Dynamic Programming	1 → 4 (biaya 2)	2
	1 → 4 → 6 (biaya 3)	5
	1 → 4 → 6 → 9 (biaya 2)	7
	1 → 4 → 6 → 9 → 10 (biaya 4)	**11**

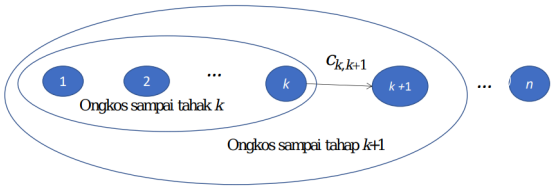
Tabel 2.1. Tabel Perbandingan Algoritma Greedy dengan *Dynamic Programming* dalam Penyelesaian Persoalan Mencari Lintasan Terpendek

Berdasarkan tabel 2.1. Algoritma Greedy mungkin mengambil jalur 1→2→6→9→10 dengan total biaya 13. Namun, pendekatan ini seringkali gagal mencapai "solusi optimal global" karena sub-masalah tidak selalu independen. Sebaliknya, Program Dinamis secara sistematis mengeksplorasi berbagai kemungkinan keputusan di setiap tahap. Ini memastikan bahwa pilihan yang dibuat mengarah pada solusi optimal secara global, seperti jalur 1→4→6→9→10 dengan total biaya 11 dalam persoalan lintasan terpendek.

C. Prinsip Optimalitas

Prinsip Optimalitas adalah fondasi teoritis yang memungkinkan Program Dinamis bekerja secara efisien. Prinsip ini menyatakan bahwa "jika solusi total optimal, maka bagian solusi sampai tahap ke-k juga optimal". Ini berarti bahwa jika kita memiliki solusi optimal untuk suatu masalah, maka setiap sub-bagian dari solusi tersebut, yang merupakan solusi untuk sub-masalah yang lebih kecil, juga harus optimal.

Prinsip ini merupakan pendorong utama struktur rekursif yang melekat pada Program Dinamis. Tanpa prinsip ini, asumsi bahwa sub-solusi optimal dapat digabungkan untuk membentuk solusi global yang optimal akan runtuh, sehingga pendekatan Program Dinamis menjadi tidak valid. Prinsip Optimalitas memungkinkan penggunaan hasil optimal dari tahap k tanpa harus kembali ke tahap awal saat bergerak dari tahap k ke tahap k+1. Dengan kata lain, untuk menentukan keputusan terbaik pada tahap saat ini, kita hanya perlu mengetahui hasil optimal dari tahap sebelumnya, tanpa perlu melacak kembali seluruh rangkaian keputusan dari awal. Ongkos pada tahap k+1 dihitung sebagai jumlah dari ongkos yang dihasilkan pada tahap k dan ongkos dari tahap k ke tahap k+1 (ck,k+1). Prinsip ini adalah justifikasi utama untuk formulasi rekursif Program Dinamis dan kebenarannya; jika suatu masalah tidak memenuhi prinsip ini, Program Dinamis tidak dapat menjamin optimalitas.



Gambar 2.2. Ilustrasi Prinsip Optimalitas (Sumber:https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/25-Program-Dinamis-(2025)-Bagian1.pdf, halaman 6)

D. Karakteristik Persoalan dengan Program Dinamis

Tidak semua persoalan optimasi dapat diselesaikan secara efektif dengan Program Dinamis. Terdapat karakteristik tertentu yang harus dipenuhi agar pendekatan ini dapat diterapkan secara optimal. Karakteristik ini berfungsi sebagai panduan untuk mengidentifikasi apakah suatu masalah cocok untuk diselesaikan dengan Program Dinamis:

1. **Pembagian Tahap**: Persoalan harus dibagi menjadi beberapa tahap (*stage*) dengan syarat setiap tahap hanya diambil satu keputusan. Sehingga memungkinkan masalah untuk dipecah secara berurutan.
2. **Status Tahapan**: Masing-masing tahap terdiri dari sejumlah status (*state*) yang berhubungan dengan tahap tersebut. Secara umum, status merupakan berbagai macam kemungkinan masukan yang ada pada suatu tahap. Status ini mendefinisikan kondisi sistem pada titik tertentu dalam proses pemecahan masalah.
3. **Transformasi Status**: Hasil dari keputusan yang diambil pada setiap tahap harus dapat ditransformasikan dari status yang bersangkutan ke status berikutnya pada tahap berikutnya. Ini menggambarkan bagaimana keputusan di satu tahap memengaruhi kondisi di tahap selanjutnya.
4. **Peningkatan Ongkos Teratur**: Ongkos (*cost*) pada suatu tahap harus meningkat secara teratur (*steadily*) dengan bertambahnya jumlah tahapan. Karakteristik ini menyiratkan bahwa Program Dinamis biasanya diterapkan pada masalah di mana biaya bersifat non-menurun atau aditif di seluruh tahapan. Jika biaya dapat menurun secara signifikan atau berfluktuasi secara tidak terduga di tahapan selanjutnya karena pilihan sebelumnya, relasi rekurens aditif sederhana mungkin tidak berlaku, berpotensi melanggar Prinsip Optimalitas. Ini menunjukkan adanya "monotonisitas" tertentu dalam akumulasi biaya yang mendukung keabsahan formulasi rekursif Program Dinamis.
5. **Ketergantungan Ongkos Tahap**: Ongkos pada suatu tahap bergantung pada ongkos tahap-tahap yang sudah berjalan dan ongkos dari tahap tersebut ke tahap berikutnya. Ini adalah dasar dari relasi rekursif, di mana solusi optimal dibangun dari sub-solusi optimal sebelumnya.
6. **Hubungan Rekursif**: Harus ada hubungan rekursif yang mengidentifikasi keputusan terbaik untuk setiap status pada tahap k yang kemudian memberikan keputusan terbaik untuk setiap status pada tahap k+1. Hubungan ini adalah inti matematis dari Program Dinamis.
7. **Penerapan Prinsip Optimalitas**: Prinsip Optimalitas harus berlaku pada persoalan tersebut. Ini adalah prasyarat fundamental yang memastikan bahwa solusi optimal global dapat dicapai dengan menggabungkan sub-solusi optimal.

E. Pendekatan Program Dinamis

Dalam pengembangan algoritma Program Dinamis, terdapat dua pendekatan utama yang dapat digunakan untuk menghitung nilai solusi optimal :

1. **Program Dinamis Maju (*Forward* atau *Up-Down*)**: Dalam pendekatan ini, perhitungan dilakukan secara berurutan dari tahap awal hingga tahap akhir. Program dinamis bergerak mulai dari tahap 1, terus maju ke tahap 2, 3, dan seterusnya sampai tahap n. Rangkaian peubah keputusan yang ditentukan adalah x1,x2,...,xn. Pendekatan ini membangun solusi dari kondisi awal menuju tujuan akhir.
2. **Program Dinamis Mundur (*Backward* atau *Bottom-Up*)**: Sebaliknya, dalam pendekatan mundur, perhitungan dilakukan dari tahap akhir menuju tahap awal. Program dinamis bergerak mulai dari tahap n, terus mundur ke tahap n−1, n−2, dan seterusnya sampai tahap 1. Rangkaian peubah keputusan yang ditentukan adalah xn,xn−1,...,x1. Pendekatan ini bekerja mundur dari tujuan akhir untuk menemukan keputusan awal yang optimal.

Meskipun kedua pendekatan ini pada akhirnya menghasilkan solusi optimal yang sama, pilihan antara pendekatan maju dan mundur seringkali memengaruhi kemudahan formulasi dan kompleksitas implementasi. Untuk beberapa masalah, mendefinisikan “status” dan “transisi” dari kondisi awal (pendekatan maju) mungkin lebih intuitif. Sementara itu, untuk masalah lain, bekerja mundur dari status tujuan (pendekatan mundur) dapat menyederhanakan relasi rekursif. Hal ini menyiratkan bahwa pemilihan pendekatan merupakan keputusan strategis bagi perancang algoritma, yang bergantung pada struktur masalah dan cara status serta transisi paling alami didefinisikan untuk formulasi rekursif yang jelas dan pengisian tabel yang efisien.

F. Langkah-langkah Pengembangan Algoritma Program Dinamis

Pengembangan algoritma Program Dinamis mengikuti serangkaian langkah sistematis untuk memastikan solusi yang optimal dan efisien. Langkah-langkah ini adalah sebagai berikut:

1. **Karakteristikkan Struktur Solusi Optimal**: Langkah pertama adalah menganalisis masalah untuk mengidentifikasi komponen-komponen utama yang dapat dipecah menjadi tahapan. Ini melibatkan pendefinisian secara jelas:

- **Tahap (*Stage*)**: Pembagian masalah menjadi langkah-langkah diskrit atau fase-fase berurutan, setiap tahap hanya satu keputusan yang diambil.
- **Peubah Keputusan (*Decision Variable*)**: Variabel-variabel yang perlu ditentukan pada setiap tahap.
- **Status (*State*)**: Berbagai kemungkinan masukan atau kondisi yang ada pada suatu tahap tertentu. Status ini mendefinisikan kondisi sistem pada titik tertentu dalam proses pemecahan masalah.
- Penting bahwa hasil keputusan pada suatu tahap dapat mentransformasikan status saat ini ke status berikutnya pada tahap selanjutnya.

2. **Definisikan secara Rekursif Nilai Solusi Optimal**: Setelah struktur masalah dikarakteristikkan, langkah selanjutnya adalah menetapkan hubungan rekursif yang mengaitkan nilai optimal dari suatu tahap dengan nilai optimal dari tahap sebelumnya. Ini biasanya melibatkan perumusan:
    - **Kasus Basis (*Base Case*)**: Kondisi awal atau kasus paling sederhana dari masalah yang solusinya diketahui secara langsung.
    - **Relasi Rekurens (*recurrence Relation*)**: Formula yang menunjukkan bagaimana nilai optimal untuk suatu status pada tahap k dapat dihitung dari nilai optimal status pada tahap k-1. Relasi ini mengidentifikasi keputusan terbaik untuk setiap status pada tahap k yang kemudian memberikan keputusan terbaik untuk setiap status pada tahap k+1.
3. **Hitung Nilai Solusi Optimal secara Maju atau Mundur**: Setelah relasi rekursif didefinisikan, nilai-nilai solusi optimal dihitung dan disimpan dalam tabel. Proses ini dapat dilakukan dengan dua pendekatan:
    - **Pendekatan Maju (*Forward*)**: Perhitungan dimulai dari tahap 1 dan berlanjut hingga tahap n, mengisi tabel secara progresif.
    - **Pendekatan Mundur (*Backward*)**: Perhitungan dimulai dari tahap n dan bergerak mundur hingga tahap 1. Penggunaan tabel ini, yang dapat berkembang seiring perhitungan, adalah alasan mengapa metode ini disebut “dinamis”.
4. **Rekonstruksi Solusi Optimal (Opsional)**: Setelah nilai solusi optimal dihitung dan tabel terisi, langkah terakhir adalah melacak kembali keputusan-keputusan yang mengarah pada solusi optimal keseluruhan. Proses rekonstruksi ini biasanya dilakukan secara mundur dari tahap akhir ke tahap awal. Sifat "opsional" dari rekonstruksi solusi menunjukkan bahwa untuk beberapa masalah, mengetahui nilai optimal (misalnya, biaya minimum atau keuntungan maksimum) saja sudah cukup, dan jalur atau urutan keputusan yang sebenarnya tidak selalu diperlukan. Hal ini menyoroti perbedaan antara menemukan nilai optimal dan menemukan jalur solusi optimal, yang terkadang dapat menyederhanakan implementasi jika hanya nilai yang dibutuhkan.

III. Mekanisme Sistem Materia dalam Gim Final Fantasy VII

Pada bagian ini, penulis akan membahas mekanisme Sistem *Materia* dalam gim *Final Fantasy VII*. Versi yang digunakan adalah versi Steam yang tersedia pada situs resmi Steam. Program dijalankan pada perangkat dengan sistem operasi Windows 10 untuk memastikan kompatibilitas dan stabilitas selama permainan.

Sistem *Materia* adalah salah satu inti kustomisasi karakter dalam *Final Fantasy VII* (*FFVII*). Mekanisme ini memungkinkan pemain untuk memodifikasi secara signifikan kemampuan, sihir, dan statistik karakter dengan memasang item berbentuk bola kristal berwarna yang disebut *Materia* ke dalam slot khusus yang tersedia pada senjata (*Weapon*) dan zirah (*Armor*) karakter. Pemahaman mendalam tentang cara kerja sistem ini sangat krusial untuk pemodelan masalah optimasi build karakter.

A. Materia (Material Orb)

Materia adalah *item* berbentuk bola kristal yang memiliki warna berbeda, menunjukkan kategori dan fungsinya:

- ***Green Materia* (*Magic Materia*)**: Memberikan akses ke berbagai sihir ofensif (misalnya, *Fire*, *Blizzard*, *Bolt*), defensif (misalnya, *Cure*, *Restore*), atau status effect (misalnya, *Poison*, *Sleep*).
- ***Blue Materia* (*Support Materia*)**: Tidak memberikan kemampuan aktif secara langsung, tetapi memodifikasi atau memperkuat efek *Materia* lain ketika dipasang pada *slot* yang terhubung (*linked slot*). Contoh umum: *All* (mengubah sihir target tunggal menjadi target banyak), *Elemental* (menambah elemen serangan/pertahanan), *Added Effect* (menambah *status effect*).
- ***Yellow Materia* (*Command Materia*)**: Menambah perintah baru yang dapat digunakan karakter dalam pertarungan (misalnya, *Steal*, *Sense*, *Manipulate*, *Enemy Skill*).
- ***Purple Materia* (*Independent Materia*)**: Memberikan bonus pasif pada statistik karakter atau efek khusus non-perintah (misalnya, *HP Plus*, *MP Plus*, *Luck Plus*, *Counter Attack*).
- ***Red Materia* (*Summon Materia*)**: Memungkinkan karakter memanggil makhluk kuat untuk menyerang musuh. *Materia* ini biasanya sangat kuat dan membutuhkan banyak *slot*.

Gambar 3.1. *Materia* pada Gim *Final Fantasy VII* (Sumber:https://static0.gamerantimages.com/wordpress/wp-content/uploads/2023/05/featured-final-fantasy-7-remake-list-of-all-materia.jpg)

B. Slot Materia pada Senjata dan Zirah

Setiap senjata dan zirah memiliki sejumlah *slot Materia*. Jumlah *slot* bervariasi antara *equipment*, dari 0 hingga 8. *Slot* ini dapat berupa:

- ***Single Slot***: Hanya dapat menampung satu *Materia*.
- ***Linked Slots* (Pasangan *Slot*)**: Dua *slot* yang terhubung secara visual. Ini esensial untuk *Blue Materia* (*Support Materia*), karena *Blue Materia* harus dipasang di salah satu *slot* yang terhubung dengan *Materia* lain yang ingin dimodifikasi di *slot* pasangannya. Misalnya, "*Fire Materia*" di satu *slot* dan "*All Materia*" di *slot* terhubungnya akan menghasilkan sihir "*Fire*" yang menyerang semua musuh.

Gambar 3.2. Sistem *Materia dalam Gim Final Fantasy VII* dokumentasi oleh penulis

C. Pengaruh Materia terhadap Statistik Karakter

Selain memberikan *skill* atau perintah, sebagian besar *Materia* juga memengaruhi statistik dasar karakter secara langsung:

- **Peningkatan Statistik**: *Materia* seperti *HP Plus*, *MP Plus*, *Luck Plus* (*Purple Materia*) secara langsung meningkatkan statistik yang bersangkutan.
- **Penurunan Statistik**: Banyak *Materia* (terutama *Magic*, *Command*, dan *Summon Materia*) memiliki efek samping berupa penurunan statistik tertentu (misalnya, *HP*, *Strength*, atau *Magic*) sebagai *trade-off* untuk kekuatan yang diberikan. Penurunan ini bisa sangat signifikan pada *Materia* dengan level tinggi.
- **Peningkatan *Growth***: Beberapa *equipment* memiliki *Materia Growth* yang berbeda (*Normal*, *Double*, *Triple*). Ini memengaruhi seberapa cepat *Materia* mendapatkan *AP* (*Ability Points*) dan naik level, namun tidak memengaruhi statistik dasar secara langsung kecuali *Materia* itu sendiri yang memberikan bonus statistik.

D. Leveling Materia

Materia mendapatkan *Ability Points* (*AP*) setelah pertarungan. Ketika *Materia* mencapai jumlah *AP* tertentu, *Materia* akan naik level dan menjadi lebih kuat (misalnya, sihir menjadi lebih kuat, *summon* memiliki lebih banyak kesempatan penggunaan), dan bahkan dapat "bereproduksi" menjadi *Materia* baru dengan kemampuan dasar yang sama. Level *Materia* juga dapat memengaruhi penurunan statistik yang diberikan.

Gambar 3.3. Sistem *Leveling Materia dalam Gim Final Fantasy VII* dokumentasi oleh penulis

IV. Pemodelan Dengan Program Dinamis untuk Optimasi *Build* Karakter

Untuk menerapkan Program Dinamis dalam mengoptimalkan *build* karakter melalui Sistem *Materia Final Fantasy VII*, masalah ini dapat dimodelkan dengan mendefinisikan *state*, tindakan, fungsi transisi, fungsi objektif, dan basis kasus (*base case*). Studi kasus spesifik dalam makalah ini adalah optimalisasi *build* karakter Tifa Lockhart.

A. Definisi Variabel dan Parameter Masalah

Dalam pemodelan Program Dinamis untuk optimasi build Materia Tifa, kita mendefinisikan variabel dan parameter sebagai berikut:

1. Definisi Keadaan (*State*)
   Sebuah keadaan **S** dalam model DP akan mewakili gambaran build karakter Tifa pada titik keputusan tertentu, menangkap semua informasi relevan yang diperlukan untuk membuat keputusan optimal

selanjutnya. Untuk mengelola ruang keadaan sambil mempertahankan detail yang diperlukan, sebuah keadaan **S** akan didefinisikan sebagai *tuple*: ($Current_{STR}$, $Current_{MAG}$, $Current_{HP}$, $Current_{MP}$, $Available_{LinkedSlots}$, $Available_{UnlinkedSlots}$, $Equipped_{MateriaConfig}$).

- $Current_{STR}$, $Current_{MAG}$, $Current_{HP}$, $Current_{MP}$: Statistik karakter turunan, dipengaruhi oleh dasar Tifa, peralatan, dan efek *Materia* (positif/negatif).
- $Available_{LinkedSlots}$, $Available_{UnlinkedSlots}$: Melacak *slot Materia* yang belum terisi; penting untuk *Materia* Pendukung.
- $Equipped_{MateriaConfig}$: *Materia* spesifik yang dipakai, termasuk level dan pemasangan *Materia* Pendukung.

Untuk studi kasus praktis, terutama yang melibatkan skenario awal-pertengahan permainan, penyederhanaan mungkin diperlukan, seperti membatasi jumlah *Materia* yang dipertimbangkan atau mengelompokkan level *Materia*. Namun, model dasar ini menyediakan kerangka kerja yang kuat untuk optimasi.

2. Definisi Aksi (Pilihan *Materia* dan *Equipment*)
   Tindakan yang dapat diambil pada setiap langkah keputusan adalah memilih dan melengkapi *Materia* atau mengganti peralatan (senjata atau zirah).
   - Memakai *Materia*: Memilih *Materia* dari inventaris dan menempatkannya ke dalam *slot* yang tersedia.
   - Memasangkan *Materia* (untuk *slot* terhubung): Jika *Materia* Pendukung Biru dipilih, tindakan ini juga mencakup pemilihan *Materia* lain untuk dipasangkan di slot terhubung yang berdekatan.
   - Mengganti Peralatan: Memilih senjata atau zirah baru dari inventaris. Peralatan baru akan mengubah jumlah dan jenis *slot Materia* yang tersedia, serta statistik dasar karakter.
3. Fungsi Transisi
   Fungsi transisi $T(S, A) = S'$ mendefinisikan bagaimana keadaan karakter **S** berubah menjadi keadaan baru **S'** setelah mengambil tindakan **A**. Ini melibatkan pembaruan statistik karakter (***Strength***, ***Magic***, ***HP***, ***MP***) berdasarkan efek Materia yang baru dilengkapi atau peralatan yang diubah, serta pembaruan jumlah slot yang tersedia.
   - **Pembaruan Statistik**: Ketika *Materia* baru dipakai, statistik karakter dihitung ulang, mempertimbangkan batas statistik 255 gim.
   - **Pembaruan Slot**: Jumlah $Available_{LinkedSlots}$ dan $Available_{UnlinkedSlots}$ akan berkurang setelah *Materia* dipakai. Jika peralatan diganti, jumlah *slot* akan diatur ulang sesuai dengan *slot* yang ditawarkan oleh peralatan baru.
4. Fungsi Objektif
   Fungsi objektif $Obj(S)$ menentukan nilai atau "kualitas" dari suatu keadaan karakter. Tujuan dari optimasi adalah untuk memaksimalkan (atau meminimalkan) nilai ini.
   - **Untuk *strength build* (Kasus Studi 1)**: Fungsi objektif akan berfokus pada memaksimalkan $Current_{STR}$ Tifa, mungkin dengan bobot tambahan untuk statistik yang mendukung *physical damage*.
   - **Untuk *balanced build* (Kasus Studi 2)**: Fungsi objektif akan menjadi kombinasi berbobot dari beberapa statistik, seperti $Current_{STR}$, $Current_{MAG}$, $Current_{HP}$, $Current_{MP}$, $Vitality$, dan $Spirit$.
5. Basis Kasus
   Kasus dasar adalah keadaan awal Tifa, sebelum *Materia* atau peralatan tambahan dipakai. Bagian ini mencakup statistik dasar Tifa pada level awal permainan, dengan peralatan awal (misalnya, *Leather Glove* dan *Bronze Bangle*) dan tanpa *Materia* yang dipakai.
6. Pendekatan Implementasi (Tabulasi/*Memoization*)
   Baik pendekatan *top-down* (*memoization*) maupun *bottom-up* (*tabulasi*) dapat digunakan.
   - ***Memoization***: Fungsi rekursif menghitung nilai optimal untuk suatu keadaan, menyimpan hasil dalam tabel untuk menghindari perhitungan ulang.
   - ***Tabulasi***: Mengisi tabel DP secara iteratif, dimulai dari kasus dasar dan membangun solusi untuk keadaan yang lebih kompleks.

B. Pengumpulan Data Materia dan Peralatan

Pengumpulan data akurat tentang *Materia* dan peralatan dalam *Final Fantasy VII* sangat penting. Data ini mencakup:

- ***Materia***: Nama, jenis, efek utama, *AP* yang dibutuhkan, dan perubahan statistik.
- **Peralatan (Senjata dan Zirah)**: Nama, statistik dasar, jumlah slot *Materia* (terhubung/tidak terhubung), dan tingkat pertumbuhan *AP Materia*.
- **Statistik Dasar Karakter**: Statistik awal Tifa dan progres statistik per level.

Data ini akan digunakan untuk menentukan parameter dalam fungsi transisi dan fungsi objektif.

C. Pembuatan Model Graf untuk Pilihan Materia (untuk Studi Kasus 2)

Untuk Studi Kasus 2, model graf akan digunakan untuk merepresentasikan ruang keputusan.

- **Simpul (*Nodes*)**: Setiap simpul mewakili keadaan karakter Tifa pada titik tertentu dalam proses *build*.

- **Sisi (*Edges*)**: Setiap sisi mewakili tindakan memakai *Materia* baru atau mengganti peralatan. Bobot sisi dapat dikaitkan dengan "biaya" atau "manfaat" tindakan.
- **Jalur (*Paths*)**: Sebuah jalur dari simpul awal ke simpul akhir mewakili urutan keputusan *Materia* dan peralatan.

Model graf ini secara efektif memetakan masalah optimasi kombinatorial ke dalam kerangka kerja yang dapat dipecahkan oleh algoritma DP.

D. Studi Kasus 1: Optimasi Build Karakter Tifa (Strength Focused, Early-Mid Game)

Optimasi *build* Tifa yang berfokus pada *strength* di fase awal-pertengahan permainan melibatkan pemilihan *Materia* dan peralatan yang secara langsung meningkatkan statistik *Strength* dan *Attack*.

1. Pemilihan Peralatan dan *Materia* Awal
   Pada awal permainan, Tifa memulai dengan *Leather Glove* (1 *slot*). Untuk *build strength focused*, senjata seperti *Metal Knuckle* (2 *slot* terhubung) atau *Mythril Claw/Motor Drive* (lebih banyak *slot*/pertumbuhan ganda) dan zirah seperti *Chain Bangle* atau *Mythril Armlet* (*slot* layak, pertumbuhan *Materia*) penting untuk memaksimalkan *Attack*.
   Materia Awal yang Relevan:
   - ***Strength Up* (*Purple Independent Materia*)**: Meningkatkan *Strength*.
   - ***HP Up* (*Purple Independent Materia*)**: Meningkatkan *HP* maksimal hingga 50% saat dikuasai.
   - ***Elemental* (*Blue Support Materia*)**: Menambahkan *elemental damage* pada serangan fisik Tifa saat dipasangkan dengan *Materia* Sihir *elemental* di senjata.
   - ***Added Cut* (*Blue Support Materia*)**: Melakukan serangan fisik tambahan setelah *Materia* yang dipasangkan digunakan.
   - ***Counter* (*Blue Support Materia*)**: Menyerang balik secara otomatis ketika diserang fisik saat dipasangkan dengan *Materia* perintah.
2. Demonstrasi Optimasi dengan Contoh Kecil (*Early-Mid Game*)
   Misalkan Tifa berada di level awal-pertengahan permainan dengan *Metal Knuckle* (2 *slot* terhubung) dan *Chain Bangle* (2 *slot* terhubung). Tujuannya adalah memaksimalkan *Strength* serangannya.
   Langkah-langkah Optimasi DP Sederhana:
   - **Definisi Keadaan Sederhana**: ($Current_{STR}$, $Available_{LinkedSlots}$, $Available_{UnlinkedSlots}$).
   - **Kasus Dasar**: Tifa dengan *Metal Knuckle*, *Chain Bangle*, 2 *slot* terhubung di senjata, 2 *slot* terhubung di zirah.
   - **Pilihan *Materia* yang Tersedia (awal-pertengahan)**: *Strength Up, HP Up, Fire, Elemental, Restore, All.*

Slot	Pilihan Materia	Efek pada *Strength*	Efek Lain	Slot Tersisa
Senjata (2 Terhubung)				
1	*Strength Up*	+5% *Strength*	-	1 terhubung
2	*Elemental + Fire*	*+Fire Damage*	-	0 terhubung
Zirah (2 Terhubung)				
1	*HP Up*	+10% *HP*	-	1 terhubung
2	*Restore + All*	Menyembuhkan semua	-	0 terhubung

Tabel 4.1. Tabel Status

   Proses Pengambilan Keputusan DP (Iteratif):
   - **Tahap 1: Memilih *Materia* untuk *Slot* Pertama Senjata**. DP memilih *Strength Up* untuk memaksimalkan *Strength* Tifa.
   - **Tahap 2: Memilih *Materia* untuk *Slot* Kedua Senjata (terhubung dengan *slot* pertama)**. DP memilih *Elemental + Fire* untuk menambahkan *elemental damage* pada serangan fisik.
   - **Tahap 3: Memilih *Materia* untuk *Slot* Pertama Zirah**. DP memilih *HP Up* untuk meningkatkan *HP* Tifa.
   - **Tahap 4: Memilih *Materia* untuk *Slot* Kedua Zirah (terhubung dengan *slot* pertama**). DP memilih *Restore + All* untuk penyembuhan area yang lebih luas.
3. Pembahasan Pemetaan dan Optimalisasi
   Pemetaan *Materia* ke dalam *slot* dan pengambilan keputusan yang berurutan adalah inti dari penerapan DP. Setiap keputusan *Materia* atau peralatan mengubah "keadaan" Tifa, dan DP mengevaluasi semua jalur keputusan yang mungkin untuk menemukan yang paling optimal.
   - **Pemetaan**: Setiap *Materia* adalah "*item*" dengan "nilai" dan "biaya". Peralatan adalah "wadah" dengan kapasitas. Ini memungkinkan masalah *Materia* diperlakukan sebagai varian masalah *knapsack*.
   - **Optimalisasi**: DP memastikan setiap keputusan *Materia* mempertimbangkan dampak jangka panjang pada *build* karakter keseluruhan, menyeimbangkan keuntungan jangka pendek dengan potensi jangka panjang (misalnya, pertumbuhan *AP* ganda).
   - **Peran Penalti Statistik**: Model DP secara otomatis mempertimbangkan penalti statistik (misalnya, *Ultima* mengurangi *Strength*) , mencegah *build* suboptimal.

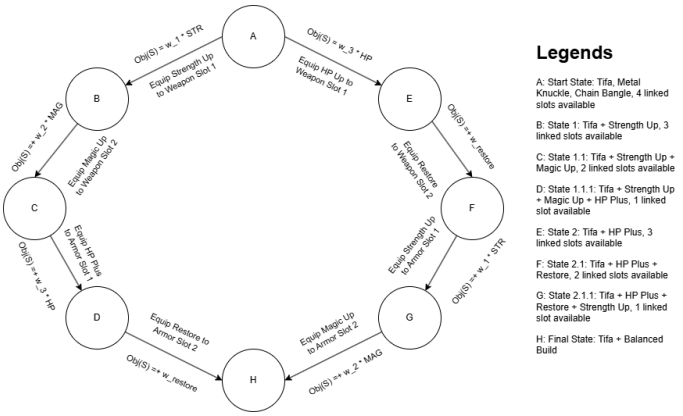
- **Keterbatasan Sumber Daya**: DP secara efisien menavigasi kendala *Materia* dan *slot* peralatan terbatas, menemukan kombinasi terbaik dari sumber daya yang ada.

E. Studi Kasus 2: Optimasi Build Karakter Tifa (Balanced Build)

Build karakter Tifa yang seimbang bertujuan mengoptimalkan beberapa statistik secara bersamaan (*Strength*, *Magic*, *HP*, *Defense*) untuk karakter serbaguna.

1. Kriteria *Balanced Build*
   Fungsi objektif akan menjadi kombinasi berbobot dari statistik berbeda (misalnya, $Obj(S) = w_1 \cdot Strength + w_2 \cdot Magic + w_3 \cdot HP + w_4 \cdot Vitality + w_5 \cdot Spirit$). *Materia* yang relevan meliputi peningkatan stat (*Strength Up*, *Magic Up*, *HP Up*), utilitas (*Restore*, *Heal*, *Barrier*), dan sinergi (*Elemental, All*).
2. Sajian Model Graf Pilihan *Materia*
   Untuk memodelkan *build* seimbang, setiap simpul mewakili konfigurasi *Materia*, dan sisi mewakili penambahan *Materia*.

   Contoh Model Graf Sederhana (untuk 4 slot terhubung):
   Misalkan Tifa berada di level awal-pertengahan permainan dengan *Metal Knuckle* (2 *slot* terhubung) dan *Chain Bangle* (2 *slot* terhubung), sama seperti Studi Kasus 1. Tujuannya adalah mencapai *build* seimbang yang meningkatkan *Strength*, *Magic*, dan *HP*.
   - **Simpul (Nodes):** Setiap simpul (S_i) mewakili keadaan karakter Tifa setelah sejumlah *Materia* dipakai, mencakup statistik saat ini dan konfigurasi *Materia*/peralatan yang dipakai.
   - **Sisi (Edges):** Setiap sisi (A_j) mewakili tindakan memakai *Materia* baru ke *slot* yang tersedia. **Bobot pada sisi ini adalah perubahan nilai fungsi objektif seimbang ($Obj(S)$) yang dihasilkan dari tindakan tersebut.** Misalnya, jika *Materia Strength Up* dipakai, bobot sisi akan mencerminkan peningkatan *Strength* yang dikalikan dengan bobot w_1 dari fungsi objektif, ditambah perubahan lain yang relevan.
   - **Jalur (Paths):** Sebuah jalur dari simpul awal (keadaan dasar Tifa) ke simpul akhir (ketika semua *slot* terisi) akan mewakili urutan keputusan *Materia* dan peralatan.



Gambar 4.1. Graf Pemodelan Studi Kasus 2
dokumentasi oleh penulis

Dalam graf ini:

- Setiap simpul adalah sebuah "keadaan" dari *build* Tifa.
- Setiap sisi adalah "tindakan" memakai *Materia.*
- Bobot pada sisi adalah perubahan nilai fungsi objektif seimbang yang dihasilkan dari tindakan tersebut. Ini dihitung berdasarkan kontribusi *Materia* terhadap statistik yang relevan, dikalikan dengan bobot w yang telah ditentukan dalam fungsi objektif ($Obj(S)$). Misalnya, jika *Materia Strength Up* dilengkapi, bobot sisi akan mencerminkan peningkatan *Strength* yang dikalikan dengan bobot w_1 dari fungsi objektif, ditambah perubahan lain yang relevan. Untuk *Materia* utilitas seperti *Restore*, bobotnya dapat berupa nilai heuristik yang ditetapkan untuk kemampuan penyembuhan.
- DP akan mencari jalur dari "*Start State*" ke "*Final State*" (ketika semua *slot* terisi) yang memaksimalkan total bobot jalur.

3. Pembahasan Optimalisasi dalam Model Graf
   Model graf memungkinkan DP secara visual dan komputasi mengeksplorasi semua kombinasi *Materia* untuk mencapai *build* seimbang.
   - **Eksplorasi Ruang Keputusan**: DP menghitung nilai optimal untuk setiap simpul, memastikan tidak ada kombinasi *Materia* yang terlewatkan.
   - **Penanganan *Trade-off* Multi-Objektif**: Dengan fungsi objektif berbobot, DP menyeimbangkan *trade-off* antara statistik berbeda.
   - **Optimalisasi Alokasi *Slot***: DP mengalokasikan *Materia* ke *slot* terhubung dan tidak terhubung secara cerdas, memprioritaskan *Materia* pendukung untuk *slot* terhubung.
   - **Fleksibilitas dan Adaptabilitas**: Model graf dapat diadaptasi untuk skenario build berbeda dengan mengubah fungsi objektif

atau kumpulan *Materia*. Tantangan utama tetap pada pengelolaan "kutukan dimensionalitas" untuk build kompleks.

V. Kesimpulan

Untuk mengoptimalkan build karakter dalam *Final Fantasy VII*, khususnya pada Sistem *Materia*, studi ini menerapkan Program Dinamis (DP) sebagai metode yang efektif. Sistem *Materia* yang kompleks, dengan interaksi berbagai jenis *Materia*, efek sinergis, dan dampaknya pada statistik karakter termasuk penalti menciptakan tantangan optimasi yang signifikan. Pendekatan intuitif seringkali tidak optimal karena mengabaikan keterkaitan yang rumit ini.

Dengan memodelkan masalah ini sebagai Program Dinamis, penulis berhasil menyediakan pendekatan matematis yang terstruktur untuk memverifikasi dan mengoptimalkan pilihan *Materia*. Pemodelan ini memungkinkan eksplorasi efisien ruang keputusan yang luas. Hasil analisis pada studi kasus *build* Tifa, yang berfokus pada kekuatan dan keseimbangan, membuktikan bahwa DP mampu memetakan pilihan *Materia* secara sekuensial untuk memaksimalkan statistik kunci, sambil secara otomatis memperhitungkan efek kumulatif dan penalti *Materia*.

Namun, "kutukan dimensionalitas" tetap menjadi tantangan untuk optimasi build yang jauh lebih kompleks atau untuk endgame. Oleh karena itu, penelitian lebih lanjut dapat mengeksplorasi metode alternatif seperti Program Dinamis Aproksimasi atau integrasi heuristik untuk optimasi yang lebih layak secara komputasi. Selain itu, penelitian dapat diperluas untuk mencakup optimasi *build* seluruh *party* atau mempertimbangkan faktor dinamis.

VI. Ucapan Terimakasih

Penulis mengucapkan terima kasih kepada Bapak Monterico Adrian, dosen pengampu mata kuliah IF2211 Strategi Algoritma pada Semester II Tahun Akademik 2024/2025 untuk Kelas K-3. Apresiasi yang mendalam juga penulis sampaikan kepada Bapak Rinaldi Munir dan Ibu Nur Ulfa Maulidevi, yang merupakan dosen-dosen pengampu mata kuliah IF2211 Strategi Algoritma. Secara khusus, terima kasih kepada Bapak Rinaldi Munir atas pengelolaan situs web https://informatika.stei.itb.ac.id/~rinaldi.munir, yang telah menyediakan beragam materi relevan dan menjadi referensi utama untuk teori-teori yang dibahas dalam tulisan ini.

References

[1] "FINAL FANTASY VII", Steam. [Online]. Available: https://store.steampowered.com/app/39140/FINAL_FANTASY_VII/. [Accessed: 23-Jun-2025].

[2] TheGamer, "Final Fantasy 7 Materia System: How Does It Work?." [Online]. Available: https://www.thegamer.com/final-fantasy-7-materia-system-how-does-it-work/. [Accessed: 24-Jun-2025].

[3] GameFAQs, "Final Fantasy VII original PC Materia system mechanics." [Online]. Available: https://gamefaqs.gamespot.com/boards/197341-final-fantasy-vii/75655250?page=1. [Accessed: 24-Jun-2025].

[4] Mrowr8d, "FF7 Materia types effects stats." [Online]. Available: https://www.mrowr8d.com/portfolio/ff/ff7/materia.php?id=support. [Accessed: 24-Jun-2025].

[5] FF7.info, "Final Fantasy VII Statistics Explained (Expert-Level Guide)." [Online]. Available: https://ff7.info/its-complicated.htm. [Accessed: 24-Jun-2025].

[6] GameFAQs, "FF7 original game Materia stat modifiers." [Online]. Available: https://gamefaqs.gamespot.com/boards/197341-final-fantasy-vii/48811703. [Accessed: 24-Jun-2025].

[7] GameFAQs, "FF7 original game Materia stat modifiers." [Online]. Available: https://gamefaqs.gamespot.com/boards/197341-final-fantasy-vii/48786394. [Accessed: 24-Jun-2025].

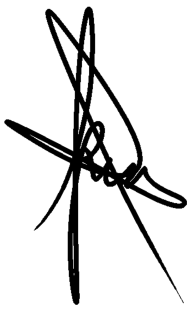
[8] R. Munir, "Program Dinamis (Dynamic Programming) - Bagian 1." [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/25-Program-Dinamis-(2025)-Bagian1.pdf. [Accessed: 23-Jun-2025].

[9] R. Munir, "Program Dinamis (Dynamic Programming) - Bagian 2." [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2024-2025/26-Program-Dinamis-(2025)-Bagian2.pdf. [Accessed: 23-Jun-2025].

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, 24 Juni 2025



Frederiko Eldad Mugiyono 13523147