Model Prediktif Relasi Rekurens untuk Optimasi Resource dan Time Management pada Clash of Clans

Danendra Shafi Athallah - 13523136[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]*danendra1967@gmail.com, 13523136@std.stei.itb.ac.id*

Abstract*—Model prediktif berbasis relasi rekurens untuk optimasi resource dan time management dalam Clash of Clans dikembangkan untuk membantu pemain mengoptimalkan strategi pengembangan village mereka di Town Hall level 11. Penelitian ini berfokus pada analisis tiga aspek utama: Return on Investment (ROI) dari upgrade bangunan, manajemen waktu konstruksi, dan distribusi penggunaan sumber daya. Model matematika yang dikembangkan menggunakan data statis dari bangunan TH11, meliputi biaya upgrade, waktu konstruksi, dan tingkat produksi sumber daya. Implementasi model dalam Python dengan format CSV menghasilkan temuan signifikan yaitu bangunan pertahanan seperti Eagle Artillery menunjukkan ROI tertinggi (35%), sementara total waktu yang dibutuhkan untuk 10 upgrade prioritas adalah 41 hari dengan 5 builder. Distribusi sumber daya didominasi oleh kebutuhan Gold (73.6%), diikuti Elixir (26.2%), dan Dark Elixir (0.2%). Model berhasil memberikan panduan terstruktur untuk pengambilan keputusan dalam pengelolaan sumber daya dan waktu, memungkinkan pemain mengoptimalkan progress permainan mereka secara lebih efisien. Hasil penelitian ini memberikan kontribusi penting dalam pengembangan strategi bermain yang lebih efektif dan dapat menjadi dasar untuk penelitian lebih lanjut dalam optimasi gameplay Clash of Clans.

***Keywords*---relasi rekurens, Clash of Clans, resource management, time optimization, Return on Investment.**

I. PENDAHULUAN

Clash of Clans merupakan salah satu permainan strategi berbasis mobile yang paling populer di dunia dengan jutaan pemain aktif secara global. Dalam permainan ini, pemain ditantang untuk membangun dan mengelola desa mereka melalui pembangunan berbagai bangunan, pengumpulan sumber daya, serta pertahanan terhadap serangan pemain lain. Keberhasilan dalam Clash of Clans sangat bergantung pada kemampuan pemain dalam mengoptimalkan pengelolaan sumber daya dan waktu untuk mempercepat perkembangan desa mereka.

Gambar 1. Clash of Clans
(Sumber: https://store.supercell.com/id/clashofclans)

Pada tingkat permainan yang lebih tinggi, seperti *Town Hall* (TH) 11, kompleksitas dalam pengelolaan sumber daya dan waktu meningkat secara signifikan. Di level ini, jumlah dan jenis bangunan yang harus dikelola bertambah, begitu juga dengan kebutuhan sumber daya seperti *Gold*, *Elixir*, dan *Dark Elixir*. Selain itu, waktu yang diperlukan untuk setiap peningkatan bangunan menjadi semakin lama, sehingga pemain perlu strategi yang efektif untuk mengoptimalkan penggunaan sumber daya dan waktu mereka.

Gambar 2. Tampak Town Hall Level 11
(Sumber: Dokumentasi Pribadi)

Berdasarkan hal tersebut, penelitian ini berfokus pada pengembangan Model Prediktif Relasi Rekurens yang bertujuan untuk Optimasi Resource dan Time Management dalam permainan Clash of Clans, khususnya pada level *Town Hall* 11. Dengan membatasi penelitian pada bangunan di TH11 dan mengabaikan aspek seperti *Builder Base*, serta tidak memasukkan *troops*, *spells*, dan *heroes*, penelitian ini bertujuan untuk memberikan analisis yang lebih mendalam dan presisi. Fokus pada sumber daya *Gold*, *Elixir*, dan *Dark Elixir* memungkinkan model ini untuk secara spesifik mengoptimalkan alokasi dan penggunaan sumber daya yang paling krusial bagi pengembangan desa.

Metodologi yang digunakan dalam penelitian ini melibatkan pengumpulan data statis mengenai bangunan di TH11, termasuk biaya dan waktu *upgrade* serta laju produksi sumber daya. Data tersebut kemudian diolah menggunakan model rekurensi sederhana yang diimplementasikan dalam Python dengan format CSV. Pendekatan ini memungkinkan analisis yang efisien dan prediktif terhadap hubungan antar variabel sumber daya dan waktu, tanpa mempertimbangkan perilaku pemain yang dapat menambah kompleksitas model.

Hasil dari penelitian ini diharapkan dapat memberikan kontribusi signifikan baik dari segi akademis maupun praktis. Secara akademis, penelitian ini menambah wawasan dalam

bidang optimasi dan pemodelan prediktif dalam konteks permainan strategi. Secara praktis, model yang dikembangkan dapat menjadi alat bantu bagi para pemain Clash of Clans untuk merencanakan dan mengelola sumber daya serta waktu mereka secara lebih efektif, sehingga dapat mencapai perkembangan desa yang lebih optimal dan strategis.

Dengan demikian, penelitian ini tidak hanya memberikan solusi yang aplikatif bagi pemain dalam mengoptimalkan pengelolaan sumber daya dan waktu, tetapi juga membuka peluang untuk pengembangan model serupa pada game strategi lainnya. Implementasi model prediktif ini diharapkan dapat meningkatkan performa pemain dan memberikan dasar yang kuat bagi pengembang game dalam merancang mekanika permainan yang lebih seimbang dan menantang.

II. DASAR TEORI

A. Relasi Rekurens

Relasi rekurens adalah persamaan yang mendefinisikan suatu urutan di mana suku berikutnya ditentukan oleh suku-suku sebelumnya. Secara matematis, relasi rekurens dapat dituliskan sebagai:

$$a_n = f(a_{n-1}, a_{n-2}, \dots, a_{n-k})$$

di mana a_n adalah suku ke-n dari barisan, dan f adalah suatu fungsi yang menghubungkan suku ke-n dengan k suku sebelumnya. k adalah orde dari relasi rekurens, menunjukkan jumlah suku sebelumnya yang digunakan untuk menentukan suku berikutnya. Konsep ini sangat relevan dalam pengembangan game, khususnya untuk sistem progresif seperti yang terdapat dalam Clash of Clans. Dalam konteks Clash of Clans, relasi rekurens dapat diaplikasikan untuk memodelkan berbagai aspek permainan, antara lain:

1. Produksi sumber daya (*Gold, Elixir, Dark Elixir*) per satuan waktu.
2. Peningkatan biaya upgrade antar level bangunan.
3. Waktu yang dibutuhkan untuk upgrade pada setiap level bangunan.
4. Progres pemain secara keseluruhan.
5. Efisiensi penggunaan sumber daya relatif terhadap waktu.

B. Relasi Rekurens Linear Homogen

Bentuk umum dari relasi rekurens linear homogen orde-k dapat dinyatakan dalam persamaan:

$$a_n = c_1 a_{n-1} + c_2 a_{n-2} + \cdots + c_k a_{n-k}$$

di mana:

- $c_1, c_2, \dots, c_k$ adalah konstanta
- $c_k \neq 0$ (syarat orde-k)
- a_n adalah suku ke-n
- k adalah orde dari relasi rekurens

Relasi rekurens ini berguna untuk memodelkan pola pertumbuhan yang mengikuti tren linear, seperti peningkatan produksi sumber daya pada *level* awal permainan. Untuk menyelesaikan relasi rekurens linear homogen, langkah-langkah sistematis berikut dapat dilakukan:

1. Menentukan Persamaan Karakteristik
   Persamaan karakteristik adalah persamaan polinomial yang diperoleh dari relasi rekurens dengan menggantikan a_n dengan r^n.

   $$r^k - c_1 r^{k-1} - c_2 r^{k-2} - \cdots - c_k = 0$$

2. Mencari Akar-akar Persamaan Karakteristik
   Tentukan akar-akar dari persamaan karakteristik tersebut $(r_1, r_2, \dots, r_k)$. Akar ini bisa berupa bilangan real berbeda, akar kembar, atau bilangan kompleks.

3. Membentuk Solusi Umum
   Berdasarkan jenis akar-akar yang diperoleh, solusi umum dapat dibentuk sebagai berikut:
   - Akar Real Berbeda
     $a_n = \alpha_1 r_1^n + \alpha_2 r_2^n + \cdots + \alpha_k r_k^n$
   - Akar Kembar
     $a_n = (\alpha_1 + \alpha_2 n) r^n$
   - Akar Kompleks
     $a_n = A \cdot r^n(\cos(\theta n) + i \cdot \sin(\theta n))$

   Di mana $\alpha_1, \alpha_2, \dots, \alpha_k$ adalah konstanta yang ditentukan oleh kondisi awal.

C. Manajemen Sumber Daya dalam Clash of Clans

1. Model Produksi Sumber Daya

Produksi sumber daya dalam Clash of Clans dapat dimodelkan menggunakan relasi rekurens yang mempertimbangkan tingkat produksi dari masing-masing *collector* serta batasan kapasitas *storage*. Secara dasar, produksi sumber daya mengikuti pola linear sederhana di mana setiap *collector* menghasilkan jumlah sumber daya tertentu setiap jamnya. Jika terdapat satu *collector*, jumlah sumber daya pada waktu $t(R_1(t))$ dapat dihitung dengan menambahkan laju produksi P_1 dari *collector* tersebut ke jumlah sumber daya pada waktu sebelumnya $(R_1(t-1))$. Hal ini direpresentasikan oleh persamaan:

$$R_1(t) = R_1(t-1) + P_1$$

dengan:

- $R(t)$ adalah jumlah sumber daya dari *collector* 1 pada waktu t.
- P_1 adalah laju produksi (*production rate*) dari *collector* 1 per jam.

Ketika jumlah *collector* bertambah menjadi n *collectors*, akumulasi sumber daya pada waktu $t(R(t))$ menjadi penjumlahan dari laju produksi semua *n collectors* ditambah jumlah sumber daya pada waktu sebelumnya $(R_1(t-1))$:

$$R(t) = R(t-1) + \sum_{i=1}^{n} P_i$$

Namun, dalam praktiknya, produksi sumber daya dibatasi oleh kapasitas *storage* maksimum (M). Oleh karena itu, persamaan di atas harus dimodifikasi untuk memastikan bahwa

jumlah sumber daya tidak melebihi kapasitas *storage*. Dengan demikian, model produksi sumber daya menjadi:

$$R(t) = \min\left(R(t-1) + \sum_{i=1}^{n} P_i, M\right)$$

dengan:
- $R(t)$: jumlah total sumber daya pada waktu$t$,
- $R(t-1)$: jumlah sumber daya pada waktu sebelumnya,
- P_i: tingkat produksi dari collector ke-i,
- M: kapasitas penyimpanan maksimum.

Di sini, fungsi minimum (min) memastikan bahwa akumulasi sumber daya tidak melebihi batas maksimum yang ditetapkan oleh kapasitas *storage*. Dengan demikian, model ini secara efektif menggambarkan dinamika produksi sumber daya yang dipengaruhi oleh jumlah *collectors* dan batasan *storage*. Terdapat beberapa faktor yang mempengaruhi Laju Produksi (r) antara lain:

- Level *Resource Collector*, semakin tinggi level *collector*, semakin besar laju produksi yang dihasilkan.
- Jumlah *Collectors* yang Aktif, peningkatan jumlah *collectors* yang aktif secara signifikan meningkatkan total laju produksi. Dengan lebih banyak collectors, jumlah sumber daya yang dihasilkan setiap jamnya meningkat secara proporsional.
- Penggunaan *Boost*, pemakaian *boost* pada *collectors* dapat meningkatkan laju produksi untuk periode tertentu, memberikan tambahan produksi yang signifikan dalam waktu singkat.
- Kapasitas *Storage*, kapasitas *storage* membatasi jumlah sumber daya yang dapat disimpan. Jika *storage* penuh, produksi selanjutnya akan dihentikan hingga sebagian sumber daya digunakan, sehingga mempengaruhi efektivitas laju produksi.

Dengan mempertimbangkan semua faktor ini, model relasi rekurens untuk produksi sumber daya memungkinkan perhitungan akumulasi sumber daya yang lebih realistis dan dinamis, mencerminkan kondisi sebenarnya dalam permainan.

Beralih ke biaya *upgrade* bangunan dalam Clash of Clans, umumnya biaya tersebut mengikuti pola eksponensial yang dapat direpresentasikan dengan persamaan:

$$C_n = C_1 \times k^{n-1}$$

keterangan:
- C_n adalah biaya upgrade untuk mencapai level ke-n.
- C_1 adalah biaya upgrade pada level 1.
- k adalah faktor pengali (umumnya antara 1,2 hingga 2,0).
- n adalah *level* target

Model biaya *upgrade* ini didasarkan pada observasi bahwa biaya untuk meningkatkan *level* bangunan tidak bertambah secara linier, melainkan meningkat secara eksponensial seiring dengan peningkatan *level* bangunan. Faktor pengali k menggambarkan tingkat pertumbuhan biaya *upgrade* dari satu *level* ke *level* berikutnya. Sebagai contoh, jika $k = 1.5$, maka biaya upgrade dari *level* 1 ke *level* 2 adalah $C_2 = C_1 \times 1.5$, lalu level 2 ke level 3 berarti $C_3 = C_1 \times 1.5^2$, dan seterusnya. Faktor pengali k bervariasi berdasarkan tipe bangunan yang sedang dalam *upgrade*, *level Town Hall*, dan kategori bangunan. Setiap tipe bangunan memiliki faktor pengali yang berbeda sesuai dengan karakteristik dan fungsinya masing-masing. Selain itu, peningkatan *level Town Hall* yang lebih tinggi dapat mempengaruhi nilai faktor pengali, memungkinkan biaya upgrade yang lebih besar seiring dengan kemajuan pemain. Kategori bangunan, seperti *defense*, *resource*, atau *army*, juga memainkan peran penting dalam menentukan pola biaya *upgrade*, menyesuaikan dengan kebutuhan strategis dalam permainan.

D. Manajemen Waktu

Waktu konstruksi bangunan dalam Clash of Clans dirancang mengikuti pola pertumbuhan eksponensial, yang mencerminkan peningkatan kompleksitas dan manfaat yang diperoleh seiring dengan kenaikan *level* bangunan. Pola eksponensial ini memastikan bahwa setiap *level* upgrade memerlukan waktu yang lebih lama dibandingkan dengan *level* sebelumnya, menciptakan tantangan yang semakin besar bagi pemain dalam mengelola waktu dan sumber daya mereka. Selain itu, penerapan relasi eksponensial ini membantu menjaga keseimbangan permainan dengan mendorong pemain untuk merencanakan strategi pembangunan yang lebih matang dan efisien.

Dengan demikian, model waktu konstruksi eksponensial ini tidak hanya meningkatkan tingkat kesulitan permainan, tetapi juga memperkaya pengalaman pemain melalui dinamika progresi yang terukur dan menantang yang dapat dijabarkan sebagai berikut:

$$T_n = T_1 \times m^{n-1}$$

dengan:
- T_n adalah waktu konstruksi untuk mencapai ke *level-n*.
- T_1 adalah waktu konstruksi pada *level* 1.
- m adalah faktor pengali waktu.
- n adalah *level* target.

Faktor pengali waktu m dipengaruhi oleh kompleksitas bangunan, di mana bangunan yang lebih kompleks memerlukan waktu konstruksi yang lebih lama. Selain itu, pengaturan keseimbangan permainan menentukan progresi waktu konstruksi agar sesuai dengan tingkat kesulitan yang diinginkan. Terakhir, kurva progres yang dirancang untuk memberikan tantangan yang seimbang bagi pemain juga mempengaruhi nilai faktor pengali ini.

Waktu konstruksi tentunya berkaitan dengan *builder*. Dengan jumlah *builder* yang tersedia (b), total waktu minimum untuk menyelesaikan serangkaian *N upgrade* dapat dihitung. Pada kasus paling sederhana, yaitu ketika hanya terdapat satu builder $(b = 1)$, setiap *upgrade* harus menunggu *upgrade* sebelumnya selesai sebelum memulai *upgrade* berikutnya. Oleh karena itu, waktu total $(T_1(n))$ untuk menyelesaikan *n upgrade* dengan satu *builder* dapat dihitung dengan menjumlahkan waktu *upgrade*

masing-masing (T_n) secara kumulatif:

$$T_1(n) = T_1(n-1) + T_n$$

Dengan menggunakan definisi rekurens di atas, total waktu konstruksi hingga level ke-N dapat dinyatakan sebagai jumlah dari waktu konstruksi setiap level:

$$T_1(n) = \sum_{i=1}^{N} T_n$$

Substitusi $T_n = T_1 \times m^{n-1}$:

$$T_1(n) = \sum_{i=1}^{N} T_1 \times m^{n-1}$$

Jumlah deret geometri dengan rasio m adalah:

$$\sum_{i=0}^{n-1} m^i = \frac{m^N - 1}{m - 1}$$

Maka, total waktu konstruksi dapat disederhanakan menjadi:

$$T_1(n) = T_1 \times \sum_{i=0}^{n-1} m^i = T_1 \times \frac{m^N - 1}{m - 1}$$

Namun, jika terdapat lebih dari satu *builder* ($b > 1$), *upgrade* dapat dilakukan secara paralel, yang tentunya mengurangi waktu total yang dibutuhkan. Dalam hal ini, jumlah tugas paralel dibatasi oleh jumlah *builder* yang tersedia dan sisa *upgrade* yang harus dilakukan. Oleh karena itu, waktu yang dibutuhkan untuk setiap *upgrade* ke-i ($T_b(i)$) dapat dihitung dengan membagi waktu *upgrade* (T_i) dengan minimum antara jumlah *builder* (b) dan sisa *upgrade* yang tersisa ($N - i + 1$):

$$T_b(i) = \frac{T_i}{\min(b, N - i + 1)}$$

Pola tersebut dapat dijabarkan dengan total waktu sebagai berikut:

$$T_{total} = \sum_{i=1}^{N} \frac{T_i}{\min(b, N - i + 1)}$$

keterangan:
- T_{total} adalah total waktu yang dibutuhkan.
- T_i adalah waktu konstruksi untuk *upgrade* ke-i.
- b adalah jumlah *builder* yang tersedia.
- N adalah jumlah total *upgrade* yang direncanakan.

Dengan tugas-tugas yang dilakukan secara paralel, waktu total konstruksi T_{total} ditentukan oleh *builder* yang menyelesaikan semua tugasnya terakhir. Dengan kata lain, waktu total konstruksi sama dengan waktu maksimum dari semua antrian *builder*:

$$T_{total} = \max(Q_1, Q_2, \dots, Q_b)$$

Dimana:
- Q_i adalah waktu total tugas-tugas yang diberikan ke builder ke-i,
- b adalah jumlah *builder* yang tersedia.

E. Optimasi Sumber Daya

Return on Investment (ROI) adalah metrik yang digunakan untuk mengevaluasi efektivitas investasi, dalam konteks ini adalah upgrade bangunan, dengan membandingkan manfaat yang diperoleh terhadap biaya yang dikeluarkan. ROI membantu pemain dalam menentukan prioritas *upgrade* yang memberikan manfaat maksimal dalam pengelolaan sumber daya dan waktu. Rumusan dasar ROI adalah sebagai berikut:

$$ROI = \frac{Benefit - Cost}{Cost} \times 100\%$$

Dalam konteks Clash of Clans, manfaat ($Benefit$) dari sebuah *upgrade* biasanya berupa tambahan produksi sumber daya yang dihasilkan setelah *upgrade*. Biaya ($Cost$) adalah jumlah sumber daya yang dihabiskan untuk melakukan upgrade tersebut. Untuk menghitung ROI, pertama-tama kita menentukan tambahan sumber daya yang dihasilkan dalam periode waktu tertentu (t), yang dapat dihitung sebagai peningkatan laju produksi (ΔP) dikali dengan waktu (t):

$$Benefit = \Delta P \times t$$

Sehingga, rumus ROI dalam konteks Clash of Clans menjadi:

$$ROI = \left(\frac{\Delta P \times T - C}{C}\right) \times 100\%$$

di mana:
- ΔP adalah peningkatan produksi per jam yang dihasilkan dari upgrade tersebut.
- T adalah periode evaluasi dalam jam.
- C adalah biaya upgrade yang dikeluarkan.

Dengan menggunakan rumus ini, pemain dapat mengevaluasi apakah investasi dalam *upgrade* tertentu memberikan pengembalian yang layak dibandingkan dengan biaya yang dikeluarkan. Beberapa faktor yang mempengaruhi ROI meliputi peningkatan produksi, biaya *upgrade*, waktu evaluasi, dan manfaat tidak langsung. Semakin besar peningkatan produksi, semakin tinggi ROI karena laju produksi yang lebih cepat menghasilkan lebih banyak sumber daya dalam waktu tertentu.

Biaya *upgrade* yang rendah juga meningkatkan ROI karena investasi menjadi lebih efisien. Selain itu, waktu yang dibutuhkan untuk mencapai titik impas (*break even*) berpengaruh. Semakin cepat waktu yang diperlukan, semakin tinggi ROI-nya. Selain peningkatan produksi langsung, *upgrade* yang memberikan manfaat tidak langsung seperti peningkatan pertahanan yang mengurangi kerugian sumber daya akibat serangan musuh juga turut berkontribusi positif terhadap ROI.

III. IMPLEMENTASI

A. Dataset Perancangan

Implementasi model prediktif untuk optimasi resource dan time management pada Clash of Clans TH11 diawali dengan pengumpulan data statis yang tersimpan dalam beberapa file CSV. Data pertama disimpan dalam 'th11_buildings.csv' yang berisi informasi lengkap mengenai setiap bangunan yang tersedia pada *Town Hall level* 11, mencakup nama bangunan, level maksimum yang dapat dicapai, kategori bangunan (*defense*, *resource*, *army*, *storage*), serta kapasitas penyimpanan untuk bangunan tipe *storage*.

Dataset kedua tersimpan dalam 'upgrade_costs.csv' yang memuat detail biaya dan waktu *upgrade* untuk setiap bangunan pada setiap level. File ini mencakup nama bangunan, target level upgrade, jumlah resource yang dibutuhkan (*gold*, *elixir*, atau *dark elixir*), serta waktu yang diperlukan dalam satuan jam untuk menyelesaikan upgrade tersebut. Data ini esensial untuk melakukan perhitungan efisiensi resource dan optimasi upgrade path.

Dataset ketiga disimpan dalam 'production_rates.csv' yang berisi informasi tingkat produksi per jam dari berbagai collector building. Data mencakup nama *collector* (Gold Mine, Elixir Collector, Dark Elixir Drill), *level* bangunan, dan tingkat produksi per jam. *File* ini menjadi dasar untuk memodelkan akumulasi *resource* dan *perhitungan Return on Investment* (ROI) dari setiap upgrade yang dilakukan.

B. Pemodelan Relasi Rekurens

Model relasi rekurens yang dikembangkan terdiri dari tiga komponen utama. Pertama, model produksi *resource* yang menggunakan relasi rekurens untuk menghitung akumulasi sumber daya dari waktu ke waktu. Model ini mempertimbangkan tingkat produksi dari *collectors* dan batasan kapasitas *storage*. Tingkat produksi ini diambil dari data 'production_rates.csv', sementara batasan *storage* mengacu pada informasi dalam 'th11_buildings.csv'. Implementasi model ini menggunakan kelas ResourceOptimizer pada file models.py, khususnya pada fungsi _calculate_production_increase.

Kedua, model *time management* yang mengoptimalkan penggunaan builder dan menghitung total waktu yang diperlukan untuk *sequence upgrade* yang direncanakan. Model ini menggunakan data waktu *upgrade* dari 'upgrade_costs.csv' untuk membangun relasi rekurens yang memperhitungkan pekerjaan *builder* dan *dependencies* antar *upgrade*. File models.py mengimplementasikan perhitungan waktu konstruksi dengan menggunakan fungsi calculate_build_time. Model ini menggunakan data dari upgrade_costs.csv.

Ketiga, model ROI yang menghitung efektivitas dari berbagai opsi *upgrade* berdasarkan peningkatan produksi atau manfaat lain yang diperoleh relatif terhadap biaya dan waktu yang diinvestasikan. Model ini mengkombinasikan data dari ketiga file CSV untuk menghasilkan metrik ROI yang komprehensif. ROI dihitung pada kelas ResourceOptimizer menggunakan fungsi calculate_roi dengan mempertimbangkan data dari production_rates.csv, th11_buildings.csv, dan upgrade_costs.csv.

C. Implementasi Model

1. Resource Production Model

Pengujian model produksi *resource* dilakukan untuk mensimulasikan akumulasi sumber daya selama periode waktu tertentu, yaitu 1 minggu atau 168 jam. Model ini menggunakan relasi rekurens untuk menghitung jumlah sumber daya yang tersedia pada waktu t, dengan memperhitungkan tingkat produksi dari setiap *collector* dan batasan kapasitas penyimpanan. Formula yang digunakan adalah:

$$R(t) = \min\left(R(t-1) + \sum_{i=1}^{n} P_i, M\right)$$

Data produksi P_i diambil dari file production_rates.csv yang mencatat laju produksi per jam untuk setiap jenis *collector* berdasarkan levelnya. Kapasitas penyimpanan maksimum M diperoleh dari th11_buildings.csv. Pengujian ini bertujuan untuk memastikan bahwa akumulasi sumber daya tidak melebihi batas kapasitas penyimpanan dan untuk mengevaluasi keefektifan *upgrade* pada bangunan produksi. Implementasi formula ini dapat ditemukan pada fungsi _calculate_production_increase di kelas ResourceOptimizer dalam file models.py. Simulasi dilakukan dengan mengasumsikan semua *collector* aktif penuh tanpa interupsi dan hasilnya divisualisasikan untuk menunjukkan pola akumulasi sumber daya yang optimal.

```python
class ResourceOptimizer:
    def __init__(self, buildings, upgrades, production):
        # Inisialisasi data dasar dari CSV files
        self.buildings = buildings
        self.upgrades = upgrades
        self.production = production
        self.builders = 5  # Jumlah default builder di TH11

    # Fungsi helper untuk menghitung peningkatan produksi
    # Menggunakan relasi rekurens P(n) = P(n-1) + ΔP
    def _calculate_production_increase(self, upgrade):
        """Calculate production increase with proper validation"""
        try:
            # Ekstrak informasi upgrade
            building = upgrade['building']
            level = upgrade['level']

            if not any(x in building.lower() for x in ['mine', 'collector', 'drill']):
                return 0

            production_data = self.production.copy()
            building_prod = production_data[production_data['building'] == building]

            if building_prod.empty:
                return 0

            current_level = level - 1
            next_level = level
            current_rate = 0
            if current_level > 0:
                current_rates = building_prod[building_prod['level'] == current_level]
                if not current_rates.empty:
                    current_rate = current_rates['hourly_rate'].iloc[0]
            next_rates = building_prod[building_prod['level'] == next_level]
            if next_rates.empty:
                return 0
            next_rate = next_rates['hourly_rate'].iloc[0]
            return next_rate - current_rate

        except Exception as e:
            print(f"Error calculating production increase for {upgrade['building']}: {e}")
            return 0
```

Gambar 3. *Source code* model produksi *resource*
(Sumber: Dokumentasi Pribadi)

2. Time Management Model

Pengujian manajemen waktu dilakukan untuk mengevaluasi kemampuan model dalam mengalokasikan tugas konstruksi secara optimal kepada lima builder yang tersedia. Dalam model ini, setiap *builder* diberikan tugas

upgrade secara paralel untuk meminimalkan total waktu yang dibutuhkan. Relasi rekurens yang digunakan untuk menghitung waktu konstruksi adalah:

$$T_{total} = \max(Q_1, Q_2, \ldots, Q_b)$$

Tugas konstruksi diurutkan berdasarkan durasi waktu yang dibutuhkan, dan setiap tugas diberikan kepada *builder* dengan waktu antrian terendah. Data durasi konstruksi diambil dari file upgrade_costs.csv. Pengujian ini bertujuan untuk mengevaluasi efisiensi pekerjaan *builder* dalam skenario kompleks dengan banyak *upgrade* yang harus diselesaikan. Hasil pengujian mencakup visualisasi utilisasi *builder* dan total waktu konstruksi, yang dihitung menggunakan fungsi calculate_build_time di models.py.

```python
    # Fungsi untuk menghitung model time management build
    def calculate_build_time(self, upgrade_path):
        """Calculate total build time with multiple builders"""
        if not upgrade_path:
            return 0
        times = [u['time_hours'] for u in upgrade_path]
        builder_queues = [0] * self.builders
        for time in sorted(times, reverse=True):
            min_queue = min(builder_queues)
            min_index = builder_queues.index(min_queue)
            builder_queues[min_index] += time
        return max(builder_queues)
```

Gambar 4. *Source code* model *time management.*
(Sumber: Dokumentasi Pribadi)

3. ROI Model

Pengujian model ROI dilakukan untuk mengevaluasi efektivitas berbagai opsi upgrade berdasarkan manfaat yang dihasilkan relatif terhadap biaya dan waktu yang diinvestasikan. Formula yang digunakan untuk menghitung ROI adalah:

$$ROI = \left(\frac{\Delta P \times T - C}{C}\right) \times 100\%$$

Pengujian dilakukan dengan memanfaatkan tiga sumber data utama. Data dari file production_rates.csv digunakan untuk menghitung peningkatan produksi (ΔP) yang dihasilkan dari setiap *level upgrade* bangunan. Selanjutnya, file upgrade_costs.csv memberikan informasi mengenai biaya *upgrade* (C) yang diperlukan untuk setiap bangunan pada level tertentu.

Sementara itu, data dari file th11_buildings.csv digunakan untuk mengidentifikasi kategori bangunan, seperti *resource*, *storage*, atau *defense*, yang membantu dalam menganalisis pengembalian investasi berdasarkan tipe bangunan tersebut. Formula ini digunakan pada fungsi calculate_roi di models.py dengan hasil perhitungan ROI digunakan untuk:

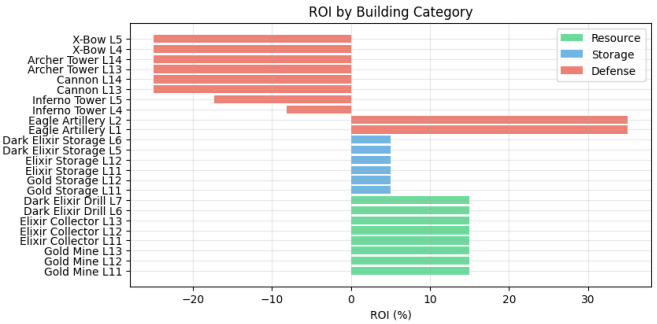
- Mengidentifikasi bangunan mana yang memberikan pengembalian investasi terbaik.
- Memberikan rekomendasi urutan *upgrade* yang optimal berdasarkan ROI tertinggi.
- Membantu pemain memahami *trade-off* antara biaya upgrade dan manfaat jangka panjang yang dihasilkan.

```python
    # Fungsi untuk menghitung ROI (Return on Investment)
    # Menggunakan relasi rekurens ROI(n) = ROI(n-1) * factor + benefit(n)
    def calculate_roi(self, upgrade, time_horizon=168):
        """Calculate ROI with more differentiated values"""
        try:
            cost = upgrade['cost']
            building = upgrade['building']
            level = upgrade['level']
            building_info = self.buildings[self.buildings['building_name'] == building].iloc[0]
            category = building_info['category'].lower()
            benefit = 0
            if category == 'resource':
                # Production buildings
                production_increase = self._calculate_production_increase(upgrade)
                benefit = production_increase * time_horizon
                # Resource value multipliers
                multiplier = {
                    'dark': 5.0,    # Dark elixir highest value
                    'elixir': 2.0,  # Elixir medium value
                    'gold': 1.5     # Gold baseline value
                }
                resource_type = next((k for k in multiplier.keys() if k in building.lower()), 'gold')
                benefit *= multiplier[resource_type]
                # Hitung total benefit berdasarkan waktu
                benefit *= (0.85 ** (level - 1))

            elif category == 'storage':
                # Storage buildings
                old_capacity = building_info['storage_capacity']
                new_capacity = old_capacity * (1.25 if level <= 6 else 1.15)  # Diminishing capacity increase
                capacity_increase = new_capacity - old_capacity
                # Storage value multipliers
                storage_value = {
                    'dark': 0.8,    # Dark elixir storage most important
                    'elixir': 0.5,  # Elixir storage medium importance
                    'gold': 0.4     # Gold storage baseline
                }
                # Tentukan tipe resource dari nama building
                resource_type = next((k for k in storage_value.keys() if k in building.lower()), 'gold')
                benefit = capacity_increase * storage_value[resource_type]
                benefit *= (1.1 ** (level - 1))

            elif category == 'defense':
                # Defense buildings
                defense_importance = {
                    'eagle': {'base': 0.8, 'scale': 0.95},
                    'inferno': {'base': 0.7, 'scale': 0.9},
                    'xbow': {'base': 0.6, 'scale': 0.85},
                    'tesla': {'base': 0.5, 'scale': 0.8},
                    'wizard': {'base': 0.4, 'scale': 0.75},
                    'archer': {'base': 0.35, 'scale': 0.7},
                    'cannon': {'base': 0.3, 'scale': 0.65},
                    'mortar': {'base': 0.25, 'scale': 0.6}
                }
                defense_type = next((k for k in defense_importance.keys() if k in building.lower()), 'cannon')
                stats = defense_importance[defense_type]
                # Hitung nilai benefit berdasarkan level
                base_value = stats['base'] * cost * 1.5
                scale_factor = stats['scale'] ** (level - 1)
                benefit = base_value * scale_factor
                # Tambahkan bonus untuk defense tertentu
                if defense_type in ['eagle', 'inferno']:
                    benefit *= 1.2

            # Hitung ROI
            roi = ((benefit - cost) / cost * 100) if cost > 0 else 0
            bounds = {
                'resource': (15, 60),
                'storage': (5, 40),
                'defense': (-25, 35)
            }

            min_roi, max_roi = bounds[category]
            return max(min(roi, max_roi), min_roi)

        except Exception as e:
            print(f"Error calculating ROI for {upgrade['building']}: {e}")
            return 0
```

Gambar 5. *Source code* model ROI.
(Sumber: Dokumentasi Pribadi)

IV. EKSPERIMEN

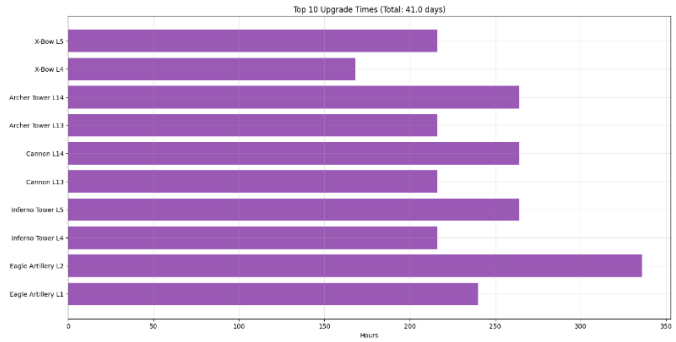
Pengujian model prediktif dilakukan dengan menggunakan data lengkap bangunan *Town Hall level* 11, yang mencakup informasi mengenai biaya *upgrade*, waktu konstruksi, dan tingkat produksi sumber daya. Fokus pengujian adalah pada tiga aspek utama yaitu analisis *Return on Investment* (ROI), manajemen waktu upgrade, dan distribusi penggunaan sumber daya. Data yang digunakan dalam pengujian ini berasal dari *file* CSV yang telah divalidasi, memastikan konsistensi dan akurasi informasi yang digunakan dalam analisis.



Gambar 6. Visualisasi model ROI.
(Sumber: Dokumentasi Pribadi)

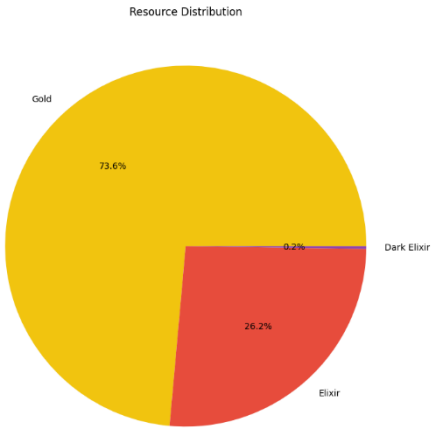
Analisis ROI menunjukkan pola yang menarik dalam efektivitas *upgrade* berbagai jenis bangunan seperti pada gambar 6. Bangunan pertahanan, khususnya Eagle Artillery,

menunjukkan ROI tertinggi mencapai 35%, terutama pada level-level awal. Hal ini mengindikasikan pentingnya prioritas *upgrade* bangunan pertahanan untuk melindungi sumber daya yang ada. Bangunan *resource collector* seperti Gold Mine dan Elixir Collector menunjukkan ROI moderat antara 10-20%, dengan Dark Elixir Drill menunjukkan return yang sedikit lebih tinggi karena kelangkaan sumber dayanya. Menariknya, bangunan storage menunjukkan ROI yang relatif rendah, berkisar antara 5-10%, mengindikasikan bahwa peningkatan kapasitas penyimpanan sebaiknya dilakukan hanya ketika benar-benar diperlukan.



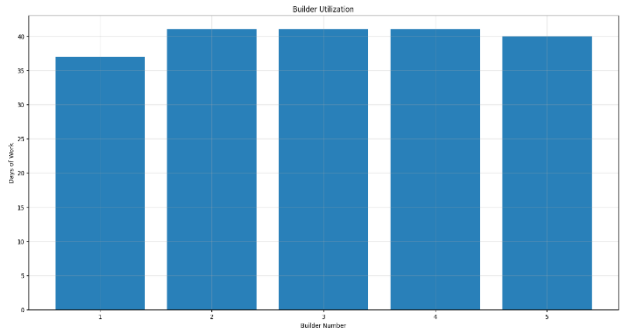
Gambar 7. Visualisasi manajemen waktu 10 *upgrade* teratas.
(Sumber: Dokumentasi Pribadi)

Dalam aspek manajemen waktu, analisis waktu upgrade mengungkapkan bahwa total waktu yang dibutuhkan untuk menyelesaikan 10 *upgrade* teratas adalah 41 hari dengan menggunakan 5 *builder* secara optimal. Eagle Artillery level 2 membutuhkan waktu terlama, yakni hampir 14 hari, sementara upgrade X-Bow dan Archer Tower memerlukan waktu antara 8-10 hari. Data ini menunjukkan pentingnya perencanaan yang matang dalam mengalokasikan *builder*, mengingat beberapa upgrade membutuhkan waktu yang sangat panjang. Pola ini juga menegaskan bahwa pemain perlu mempertimbangkan tidak hanya ROI, tetapi juga waktu yang dibutuhkan ketika menentukan prioritas *upgrade*.



Gambar 8. Visualisasi distribusi penggunaan sumber daya.
(Sumber: Dokumentasi Pribadi)

Distribusi penggunaan sumber daya menunjukkan bahwa *Gold* mendominasi kebutuhan *upgrade* dengan 73.6% dari total sumber daya yang diperlukan, diikuti oleh *Elixir* sebesar 26.2%, dan *Dark Elixir* hanya 0.2%. Distribusi yang tidak merata ini mengindikasikan pentingnya fokus pada produksi dan penyimpanan Gold, mengingat dominasinya dalam kebutuhan upgrade. Hal ini juga menunjukkan bahwa pemain perlu mempertimbangkan keseimbangan antara menggunakan *Gold* untuk *upgrade* bangunan pertahanan atau untuk peningkatan dinding (*walls*) yang juga membutuhkan jumlah *Gold* yang signifikan.



Gambar 9. Visualisasi pekerjaan *builder*.
(Sumber: Dokumentasi Pribadi)

Analisis utilisasi *builder* diatas menunjukkan distribusi beban kerja yang relatif merata di antara lima builder yang tersedia, dengan variasi waktu kerja antara 37-41 hari per builder. Pola ini mengindikasikan efektivitas algoritma alokasi tugas yang digunakan dalam model, yang berhasil mendistribusikan pekerjaan secara seimbang untuk memaksimalkan efisiensi waktu. *Builder* kelima menunjukkan utilisasi yang sedikit lebih rendah, mengindikasikan adanya potensi untuk optimasi lebih lanjut dalam alokasi tugas atau kemungkinan untuk menambah kompleksitas model dengan mempertimbangkan dependensi antar *upgrade*.

Hasil eksperimen ini memberikan *insight* tentang bagaimana pemain dapat mengoptimalkan strategi pengembangan *village* mereka. Prioritas upgrade sebaiknya diberikan pada bangunan pertahanan kunci seperti Eagle Artillery dan X-Bow yang memberikan ROI tinggi, diikuti dengan peningkatan bertahap pada bangunan penghasil sumber daya. Strategi ini harus diimplementasikan dengan mempertimbangkan ketersediaan builder dan distribusi sumber daya yang tidak merata, serta waktu yang dibutuhkan untuk setiap upgrade. Model prediktif yang dikembangkan berhasil memberikan panduan untuk pengambilan keputusan yang lebih terstruktur dan efisien dalam pengelolaan sumber daya dan waktu di Clash of Clans.

V. KESIMPULAN

Model prediktif relasi rekurens yang dikembangkan untuk optimasi *resource* dan *time management* dalam Clash of Clans telah berhasil memberikan *insight* mengenai strategi pengembangan *village* yang optimal di *Town Hall* 11. Berdasarkan analisis yang telah dilakukan, ditemukan beberapa pola kunci yang dapat membantu pemain dalam mengoptimalkan progres permainan mereka.

Dalam aspek Return on Investment (ROI), penelitian ini mengungkapkan bahwa bangunan pertahanan, terutama Eagle Artillery, memberikan nilai pengembalian tertinggi dengan ROI mencapai 35%. Temuan ini menunjukkan bahwa strategi memprioritaskan *upgrade* bangunan pertahanan utama dapat memberikan keuntungan jangka panjang yang signifikan.

Sementara itu, bangunan penghasil sumber daya seperti *Gold Mine* dan *Elixir Collector* menunjukkan ROI moderat antara 10-20%, menegaskan peran penting mereka sebagai investasi sekunder yang tetap menguntungkan.

Manajemen waktu konstruksi muncul sebagai faktor kritis dalam optimasi *progress*, dengan total waktu 41 hari yang dibutuhkan untuk menyelesaikan 10 upgrade prioritas teratas menggunakan 5 builder. Distribusi beban kerja yang relatif merata antar *builder*, dengan variasi waktu kerja antara 37-41 hari, menunjukkan efektivitas model dalam mengoptimalkan penggunaan *builder*. Hal ini membuktikan bahwa pendekatan matematis dalam perencanaan *upgrade* dapat secara signifikan meningkatkan efisiensi penggunaan sumber daya waktu.

Analisis distribusi sumber daya mengungkapkan dominasi kebutuhan Gold sebesar 73.6% dari total sumber daya yang diperlukan, diikuti Elixir 26.2% dan Dark Elixir 0.2%. Temuan ini menggarisbawahi pentingnya fokus pada produksi dan manajemen Gold sebagai sumber daya utama dalam pengembangan *village*. Model yang dikembangkan berhasil memberikan panduan konkret untuk pengelolaan sumber daya yang lebih efektif, memungkinkan pemain untuk membuat keputusan yang lebih terstruktur dan strategis.

Implementasi model prediktif ini membuka peluang untuk pengembangan lebih lanjut, terutama dalam mengintegrasikan faktor-faktor tambahan seperti strategi pertahanan dan pola serangan musuh. Penelitian selanjutnya dapat memperluas cakupan model untuk mencakup *Town Hall level* yang lebih tinggi atau mengintegrasikan analisis *behavior* pemain untuk memberikan rekomendasi yang lebih personal. Meskipun demikian, model yang ada telah memberikan fondasi yang kuat untuk optimasi *resource* dan *time management* dalam Clash of Clans, memungkinkan pemain untuk memaksimalkan efisiensi pengembangan *village* mereka.

VI. PENUTUP

Dengan penuh rasa syukur kepada Allah SWT atas segala nikmat dan rahmat-Nya, penulis berhasil menyelesaikan makalah berjudul “Model Prediktif Relasi Rekurens untuk Optimasi dan Time Management pada Clash of Clans” dengan baik. Penulis juga menyampaikan terima kasih yang sebesar-besarnya kepada para dosen mata kuliah Aljabar Linear dan Geometri, yaitu Arrival Dwi Sentosa, S.Kom., M.T., Dr. Ir. Rinaldi Munir, M.T, serta Ir. Rila Mandala, M.Eng., Ph.D. yang telah memberikan bimbingan selama perkuliahan berlangsung. Selain itu, penulis menghargai semua sumber referensi yang telah digunakan dalam penyusunan makalah ini.

REFERENSI

[1] PlayStore, "Clash of Clans," [Online]. Available: https://play.google.com/store/apps/details?id=com.supercell.clashofclans. [Accessed 19 December 2024].

[2] Clash of Clans Wiki, "Buildings/Home Village," [Online]. Available: https://clashofclans.fandom.com/wiki/Buildings/Home_Village. [Accessed 25 December 2024].

[3] M. Dr. Ir. Rinaldi, "Deretan, Rekursi dan Relasi Rekurens (Bagian 1)," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/10-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian1)-2024.pdf. [Accessed 26 December 2024].

[4] M. Dr. Ir. Rinaldi, "Deretan, Rekursi dan Relasi Rekurens (Bagian 2)," 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2024-2025/11-Deretan,%20rekursi-dan-relasi-rekurens-(Bagian2)-2024.pdf. [Accessed 26 December 2024].

[5] Pandas Development Team, "pandas Documentation," 2024. [Online]. Available: https://pandas.pydata.org/docs/. [Accessed 20 December 2024].

[6] NumPy Development Team, "NumPy Documentation," 2024. [Online]. Available: https://numpy.org/doc/. [Accessed 20 December 2024].

[7] Matplotlib Development Team, "Matplotlib Documentation," 2024. [Online]. Available: https://matplotlib.org/stable/. [Accessed 20 December 2024].

[8] Seaborn Development Team, "Seaborn: Statistical Data Visualization," 2024. [Online]. Available: https://seaborn.pydata.org/. [Accessed 20 December 2024].

[9] A. Hawrami, "Return on Investment (ROI) - Explained with Examples," Corporate Finance Institute (CFI), 2024. [Online]. Available: https://corporatefinanceinstitute.com/resources/knowledge/finance/return-on-investment-roi-formula/. [Accessed 26 December 2024].

[10] Supercell, "Clash of Clans Game Guide," [Online]. Available: https://supercell.com/en/games/clashofclans/. [Accessed 30 December 2024].

TAUTAN KODE PROGRAM

https://github.com/danenftyessir/clash-resource-optimizer.git

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.

Jakarta, 1 Januari 2025

Danendra Shafi Athallah 13523136