Pencarian Makanan dan Barang Utilitas yang Optimal dalam Batasan Harga di Guild Wars 2 dengan Program Dinamis

Muhammad Rizky Ismail Faizal 13518148
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
rykzius@gmail.com

Makanan dan barang utilitas menjadi komponen pendukung yang sangat penting bagi karakter-karakter di dunia Guild Wars 2. Dibutuhkan suatu metode untuk mencari pasangan makanan dan barang utilitas apa yang optimal dalam batasan biaya dan level tertentu. Permasalahan ini tergolong permasalahan optimasi, dan karena itu dapat didekati dengan strategi program dinamis.

guild wars 2, makanan, utilitas, optimal, program dinamis

I. PENDAHULUAN

Guild Wars 2 adalah salah satu permainan dengan *genre* MMORPG yang berupa singkatan dari *Massively Multiplayer Online Role Playing Game*. Permainan dengan *genre* ini umumya berisi suatu dunia yang realistis dalam aspek hal yang dapat dilakukan. Contohnya, di Guild Wars 2 pemain dapat berburu, mengumpulkan bahan, dan membuat barang dengan sistem profesi. Profesi yang ada sangat beragam, mulai dari pembuat senjata hingga pengrajin kulit.

Salah satu profesi yang menarik adalah profesi koki. Profesi ini dapat membuat barang berjenis makanan yang dapat dikonsumsi karakter untuk meningkatkan statusnya berdasarkan suatu angka konstan. Peningkatan status ini dapat memiliki berbagai efek, contohnya peningkatan status *power* akan meningkatkan kekuatan serangan karakter dan peningkatan status *precision* akan meningkatkan peluang karakter melakukan *critical hit*. Karena efeknya yang kuat dan beragam, makanan menjadi bagian yang penting dalam Guild Wars 2.

Terdapat juga jenis barang lain yang dapat dikonsumsi oleh karakter, yaitu barang utilitas. Berbeda dengan makanan, barang utilitas dapat dibuat tidak hanya oleh satu profesi tetapi oleh berbagai profesi. Akan tetapi, meskipun barang utilitas lebih mudah dibuat, efeknya lebih kompleks dan tidak sesederhana makanan. Meskipun efek peningkatan statusnya sama, mayoritas barang utilitas ketika digunakan tidak meningkatkan status karakter sebesar suatu angka konstan seperti makanan. Barang utilitas umumnya akan meningkatkan status karakter sebesar suatu persentase dari status lainnya.

Peningkatan status yang didapat melalui penggunaan makanan dan barang utilitas dapat memiliki efek yang signifikan ketika digunakan bersamaan. Sebagai contoh, penggunaan makanan dan barang utilitas yang meningkatkan status *power* secara bersamaan akan membuat kekuatan serangan karakter meningkat drastis. Ini sangat membantu dalam aktivitas-aktivitas yang memerlukan kekuatan serangan tinggi, contohnya aktivitas *raid* dimana terdapat beberapa lawan yang harus dibunuh dalam suatu batasan waktu. Karena itu, terdapat *demand* yang besar untuk makanan dan barang utilitas yang meningkatkan status *power*. Seiring membesarnya *demand*, harga makanan dan barang utilitas juga meningkat di pasar Guild Wars 2.

Pada makalah ini akan dibuat suatu aplikasi yang dapat mencari makanan dan barang utilitas mana yang optimal untuk meningkatkan satu atau lebih status tertentu dalam suatu batasan harga.

II. LANDASAN TEORI

A. *Terminologi dalam Guild Wars 2*

Dalam Guild Wars 2, pemain bisa memiliki satu atau lebih karakter. Tiap karakter memiliki ***level*** tertentu dan *inventory* yang dapat menyimpan barang-barang tertentu yang dimiliki karakter tersebut. Tiap barang dalam inventory memiliki informasi dasar berupa nama, jenis barang, dan deskripsi mengenai barang tersebut.

Consumables adalah suatu jenis barang dalam Guild Wars 2 yang dapat dikonsumsi oleh karakter, baik sekali pakai maupun dengan pemakaian tidak terbatas. Dalam jenis barang ini juga terdapat berbagai subjenis yang terbagi berdasarkan fungsi barang. Contohnya subjenis *salvage kit* yang dapat digunakan untuk memecah barang lain menjadi bahan dasar.

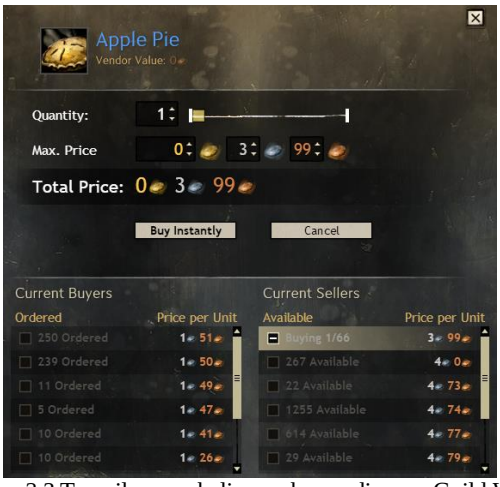
Gambar 2.1 Contoh makanan dalam Guild Wars 2

Gambar 2.2 Contoh barang utilitas dalam Guild Wars 2

Kata **makanan** yang akan banyak digunakan dalam makalah ini merujuk kepada barang berjenis *consumables* dengan subjenis *food*. Sementara itu, kata **barang utilitas** merujuk kepada barang berjenis *consumables* dengan subjenis *utility*. Makanan dan barang utilitas hanya bisa digunakan karakter yang *level*-nya kurang dari atau sama dengan ***level*** makanan atau barang utilitas yang bersangkutan.

Pada satu waktu, tiap karakter hanya dapat menikmati dampak dari satu makanan dan satu barang utilitas (jika karakter mengkonsumsi dua makanan, hanya dampak makanan terakhir yang akan aktif). Makanan dan barang utilitas yang dibahas dalam makalah ini umumnya berjenis sekali pakai. Tiap makanan dan barang utilitas juga

Ketika makanan atau barang utilitas dikonsumsi, akan terdapat efek pada karakter yang meningkatkan status sesuai **deskripsi** barang yang bersangkutan. Efek ini akan aktif selama suatu **durasi waktu** yang juga sesuai barang yang bersangkutan. Beberapa barang dapat menerapkan efek berlapis pada karakter. Contohnya jika suatu makanan menerapkan dua lapis efek peningkatan status *power* sebanyak 50 poin, secara efektif makanan meningkatkan status *power* sebanyak 100 poin.



Gambar 2.3 Tampilan pembelian makanan di pasar Guild Wars 2

Makanan dan barang utilitas, seperti barang lainnya, dapat diakuisisi dengan berbagai cara, contohnya dengan membuat barang tersebut sendiri. Cara yang dibahas pada makalah ini adalah dengan pembelian melalui pasar yang tersedia dalam permainan. Pada pasar, pemain dapat membeli barang dengan dua cara: 1. memasang pesanan dengan bayaran lebih tinggi dari pesanan lain (***buy price***); 2. membeli barang yang tersedia dengan harga terendah (***sell price***). Pembelian dengan metode pertama umumnya disebut *buy order* dan pemain harus menunggu sampai ada yang menjual barang pada harga pesanannya, sementara pembelian dengan metode kedua umumnya disebut *instant buy* karena barang langsung didapat.

Mata uang yang digunakan dalam pembelian di pasar Guild Wars 2 ada tiga jenis: copper, silver, dan gold. Tiap tingkatan senilai dengan 100 buah tingkatan dibawahnya, contohnya 23 gold adalah 2300 silver atau 230000 copper. Ini membuat konversi angka ke tulisan dapat dilakukan dengan mudah dengan modulo dan pembagian integer: 123142 dapat ditulis 12 gold 31 silver 42 copper.

B. Guild Wars 2 API

Pembuat Guild Wars 2, ArenaNet, menyediakan suatu *Application Programming Interface* (API) yang berupa antarmuka yang memungkinkan aplikasi pihak ketiga untuk mengakses data secara langsung dari server Guild Wars 2 [1]. Akses data dilakukan dengan cara membuka koneksi kepada URL sesuai dengan API yang ingin digunakan, kemudian memroses data yang dikembalikan dalam bentuk JSON.

Pada makalah ini, digunakan dua bagian API, yaitu bagian *items* untuk mendapatkan data barang dan bagian *commerce prices* untuk mendapatkan data harga tiap barang.

Bagian API *items* mengembalikan data yang dibutuhkan untuk berjalannya aplikasi, yaitu *id*, nama, jenis, gambar, level, subjenis, deskripsi, durasi, dan jumlah lapisan efek [2].

Bagian API *commerce prices* mengembalikan data yang dibutuhkan untuk berjalannya aplikasi yaitu *buy price* dan *sell price* untuk setiap barang [3].

C. JSON

JSON (JavaScript Object Notation) adalah format pertukaran data yang ringan. Sangat mudah bagi manusia untuk membaca dan menulis. Mudah bagi mesin untuk mengurai dan menghasilkan. Ini didasarkan pada subset dari Standar Bahasa Pemrograman JavaScript ECMA-262 Edisi 3 - Desember 1999. JSON adalah format teks yang sepenuhnya bebas bahasa tetapi menggunakan konvensi yang akrab bagi programmer dari keluarga bahasa C, termasuk C, C ++, C #, Java, JavaScript, Perl, Python, dan banyak lainnya. Properti ini menjadikan JSON bahasa pertukaran data yang ideal [4].

Pada makalah ini, JSON digunakan sebagai format data yang diterima dari Guild Wars 2 API untuk kemudian diproses dalam aplikasi.

D. Regular Expression

Regular expression (Regex) adalah pola pencarian yang digunakan untuk mencocokkan satu atau lebih karakter dalam sebuah string. Itu bisa cocok dengan karakter tertentu, wildcard, dan rentang karakter. Regex pada awalnya digunakan oleh utilitas Unix, seperti vi dan grep. Namun, mereka sekarang didukung oleh banyak aplikasi pengeditan kode dan pengolah kata pada berbagai platform. Regex juga dapat digunakan di sebagian besar bahasa pemrograman utama [5].

Pada makalah ini, Regex digunakan untuk mengekstrak informasi peningkatan status dari deskripsi makanan dan barang utilitas.

E. *Program Dinamis*

Program dinamis adalah metode pemecahan masalah dengan cara menguraikan solusi menjadi sekumpulan tahapan (stage) sedemikian sehingga solusi dari persoalan dapat dipandang dari serangkaian keputusan yang saling berkaitan [6].

Permasalahan yang dapat diselesaikan dengan program dinamis memiliki tiga karakteristik [6], yaitu:

1. terdapat sejumlah berhingga pilihan yang mungkin,
2. solusi pada setiap tahap dibangun dari hasil solusi tahap sebelumnya,
3. kita menggunakan persyaratan optimasi dan kendala untuk membatasi sejumlah pilihan yang harus dipertimbangkan pada suatu tahap.

Permasalahan pencarian makanan dan barang utilitas optimal dapat direpresentasikan sedemikian sehingga memenuhi tiga karakteristik di atas. Karena itu pada makalah ini digunakan pendekatan program dinamis.

III. Analisis Permasalahan

Untuk dapat mencari makanan dan barang utilitas optimal, pertama tentunya perlu didapat data seluruh makanan dan barang utilitas yang ada pada Guild Wars 2 beserta informasi harga. Karena itu akan perlu dibuat suatu bagian kode yang mengambil data dari Guild Wars 2 API kemudian memrosesnya supaya dapat diproses bagian kode program dinamis.

Setelah data didapat dan bisa diproses, permasalahan perlu direpresentasikan supaya dapat diselesaikan dengan program dinamis. Untuk ini, digunakan pendekatan yang mirip dengan pendekatan program dinamis dalam penyelesaian permasalahan *capital budgeting*.

Pada penyelesaian permasalahan *capital budgeting*, terdapat beberapa pabrik yang memiliki berbagai alternatif proposal (p) dengan biaya (c) dan keuntungan (r) masing-masing. Didapat bahwa terdapat tahap (k) sejumlah banyaknya pabrik, dan dibuat suatu fungsi (f) yang dapat menentukan pilihan proposal mana saja yang diambil pada tahap 1, 2, ..., k untuk mendapat keuntungan maksimal pada tahap k.

Dengan melihat permasalahan tersebut, dapat dirumuskan suatu struktur permasalahan yang serupa untuk pencarian makanan dan barang utilitas optimal:

1. Makanan dan barang utilitas dilihat sebagai "pabrik" yang menyediakan berbagai macam pilihan barang (p) dengan harga (c) dan kekuatan efek (r) masing-masing.
2. Dengan itu, didapat bahwa ada dua tahap (k): satu tahap untuk pencarian makanan optimal dan satu tahap untuk pencarian barang utilitas optimal.
3. Fungsi yang dibuat (f) menjadi fungsi yang dapat menentukan makanan optimal pada tahap 1 dan menentukan makanan + barang utilitas optimal pada tahap 2.

Dapat dilihat bahwa terdapat suatu permasalahan ketika ingin mencari makanan yang memberikan beberapa status: nilai mana yang diambil sebagai r? Untuk menyelesaikan permasalahan ini, digunakan rumus berikut untuk menentukan r tiap makanan atau barang utilitas (x):

$$\text{kekuatan}(\text{status}, x) = \text{angka\_di\_deskripsi}(\text{status}, x) * \text{durasi}(x) * \text{lapisan\_efek}(x)$$

$$r(\text{daftar\_status}, x) = \text{jumlah kekuatan}(\text{status}, x) \text{ untuk semua status}$$

Dengan rumus tersebut didapat nilai r untuk semua makanan dan barang utilitas yang bergantung kepada status apa saja yang dicari. Sementara untuk nilai c, dapat didapat dengan mengambil informasi *buy price* atau *sell price* sesuai metode pembelian yang ingin digunakan pengguna.

IV. Implementasi Algoritma

Dalam implementasi, digunakan bahasa pemrograman Python 3.8.2. Implementasi dibagi menjadi empat bagian, yaitu konstanta dan fungsi sederhana, pengambilan data, pengolahan data, dan program dinamis.

Cuplikan kode yang disertakan akan berupa kode yang sudah dibersihkan dari implementasi yang terlalu panjang dan tidak relevan.

A. *Konstanta dan Fungsi Sederhana*

```
# lib.py

# konstanta aplikasi
VERSION = '1.0.0'
REFRESH = False

STAT_NAMES = [
    # daftar nama status untuk tampilan aplikasi
]
STAT_LOOKUP = [
    # daftar penanda status dalam deskripsi
]

# konstanta multithreading
THREADS = []
MAX_THREADS = 7

# konstanta database
SQLITE_CREATE_ITEMS = # perintah sql
SQLITE_CREATE_CONSUMABLES = # perintah sql
SQLITE_CREATE_PRICES = # perintah sql
DATABASE_PATH = 'cheesecake.db'

# konstanta API
API_BASE_LINK = 'https://api.guildwars2.com/v2/'
ITEMS_LINK = API_BASE_LINK + 'items/'
PRICES_LINK = API_BASE_LINK + 'commerce/prices/'

# fungsi pengambilan data dari suatu url
def get_object(url):
    # ...

# fungsi penulisan jumlah uang
def moneytalk(copper):
    # ...

# fungsi membersihkan layar terminal
def clear():
    # ...
```

Gambar 4.1 Cuplikan kode lib.py

Pertama, dibuat suatu modul (lib.py) yang menyimpan konstanta dan fungsi sederhana yang digunakan di keseluruhan aplikasi.

B. *Pengambilan Data*

```python
# items.py - untuk akuisisi info barang

def item_acquirer(item_ids, ready, flag, loop, threadnumber):
    # fungsi yang dijalankan pada tiap thread untuk mengakses da
ta

def run_item_acquirers(loop, item_ids):
    # fungsi untuk menjalankan semua thread

# fungsi refresh semua database item
def refresh_items_database():
    print('Refreshing items database...')
    con = sqlite3.connect(DATABASE_PATH)
    c = con.cursor()
    with con:
        c.execute(SQLITE_CREATE_ITEMS)
        c.execute(SQLITE_CREATE_CONSUMABLES)

    print('Starting data acquisition loop...')
    gettable = 1
    loop = 1
    while (gettable > 0):
        print(f'[{loop}] Getting items data from API...')
        item_ids = get_object(ITEMS_LINK)
        if item_ids is None:
            continue
        print(f'[{loop}] Excluding already available items...')
        with con:
            available = set()
            for thing in c.execute('select id from items'):
                available.add(int(thing[0]))
            item_ids = [x for x in item_ids if x not in availabl
e]
        gettable = len(item_ids)
        if (gettable > 0):
            run_item_acquirers(loop, item_ids)
            loop += 1

    with con:
        item_count = c.execute('select count(*) from items').fet
chone()[0]
        consumables_count = c.execute('select count(*) from cons
umables').fetchone()[0]
    con.close()
    print(f'Confirmed {consumables_count} consumables out of {it
em_count} items.')
```

```python
# prices.py - untuk akuisisi info harga

# isinya relatif sama dengan items.py tetapi
# tujuannya untuk mengakuisisi info harga
```

Gambar 4.2 Cuplikan kode items.py dan prices.py

Pada bagian ini, dibuat modul yang mengurus pengambilan data dari Guild Wars 2 API bagian *items* (items.py) dan *commerce prices* (prices.py). Pengambilan data dilakukan sesuai dengan cara-cara yang tersedia dalam dokumentasi Guild Wars 2 API. Pertama, dilakukan pengaksesan terhadap URL berikut.

https://api.guildwars2.com/v2/items

Pengaksesan URL tersebut mengembalikan JSON berbentuk *array* yang berisi semua *id* barang yang ada di Guild Wars 2. Data ini disimpan untuk kemudian melakukan pengambilan informasi barang lebih detil dengan mengakses salah satu dari dua URL berikut.

https://api.guildwars2.com/v2/items/id1
https://api.guildwars2.com/v2/items?ids=id1,id2,id3,...

URL pertama dapat digunakan untuk mengakses informasi mengenai suatu barang dengan *id* berupa ‘id1’. Sementara itu, URL kedua dapat digunakan untuk mengakses informasi mengenai beberapa barang sekaligus.

Pada aplikasi ini, sebenarnya dapat digunakan URL pertama untuk mempermudah perancangan kode. Akan tetapi, pada pengujian didapat bahwa satu kali akses ke Guild Wars 2 API memakan waktu yang bervariasi antara 5-25 detik untuk mendapatkan respon. Pada pengujian juga didapat bahwa terdapat kurang lebih 52.000 barang di Guild Wars 2, yang berarti pengaksesan informasi seluruh barang akan memakan waktu paling lambat 1.300.000 detik atau sekitar 361 jam.

Menurut perhitungan, penggunaan URL kedua akan menghemat sangat banyak waktu, tetapi masih membutuhkan waktu hampir dua jam. Karena itu, digunakan sistem *multithreading* pada aplikasi untuk mempercepat pengambilan data supaya hanya memakan waktu beberapa menit.

Data yang diterima oleh aplikasi disimpan dalam bentuk database SQLite di file cheesecake.db. Ini digunakan supaya akses data pada bagian-bagian selanjutnya jauh lebih cepat dan mudah.

C. *Pengolahan Data*

```python
# bagian dari cheesecake.py

# buka akses database
con = sqlite3.connect(DATABASE_PATH)
c = con.cursor()

# bagian input
input_level = int(input('Enter character level: '))
input_budget = int(input('Enter budget (in copper coins): '))
input_instant = input('Use instant buy price? (Y/n): ').lower()
!= 'n'
for i, name in enumerate(STAT_NAMES):
    print(f'({i+1}) {name}')
input_stats = input('Enter id(s) of stat(s) to find (comma separ
ated): ')
lookup = [int(x)-
1 for x in input_stats.replace(' ', '').split(',')]

# persiapan pemrosesan
cost = [{0: 0}, {0: 0}]
result = [{0: 0}, {0: 0}]
valid = [{0}, {0}]

# bagian pemrosesan
price = 'price_sell' if input_instant else 'price_buy'
for r in c.execute('select c.id, c.description, c.duration, c.ap
ply_count, c.type, p.'+ price +' from consumables c, prices p wh
ere c.id = p.id and c.level <= ?', (input_level,)):
    id = r[0]
    score = 0
    food = r[4] == 'Food'
    for i in lookup:
        if food:
            look = STAT_LOOKUP[i]
            p = re.search(r'((?<=\+)|^|\t|\\n)[0123456789]+(?='+
look.lower()+')', r[1].lower())
        else:
            look = STAT_NAMES[i]
            p = re.search(r'(?<=gain ' + look.lower() + r' equal
 to )[0123456789]+(?=% of your)', r[1].lower())
        if p:
            score += int(p.group())
    score = score * (r[2]/1000) * r[3]
    x = 0 if food else 1
    cost[x][id] = r[5]
    result[x][id] = score

# bagian pencatatan data yang valid
for x, data in enumerate(result):
    for id, uresult in data.items():
        if uresult > 0:
            valid[x].add(id)
```

Gambar 4.3 Cuplikan kode cheesecake.py bagian pengolahan

Pada bagian ini, dibuat kerangka aplikasi utama yang pertama menanyakan kepada pengguna beberapa informasi yang diperlukan untuk pengolahan data:

1. *Level* karakter pengguna. Aplikasi hanya akan mencari makanan dan barang utilitas yang *level*-nya kurang dari atau sama dengan *level* karakter.

2. Batasan biaya (*budget*). Nilai ini akan digunakan pada bagian program dinamis untuk menjadi batasan harga pada tahap terakhir.
3. Metode pembelian yang akan digunakan. Nilai ini menentukan apakah akan digunakan *buy price* atau *sell price* dalam pengolahan nilai c.
4. Status mana saja yang dipilih. Nilai ini digunakan dalam pengolahan nilai r untuk menentukan apa saja yang dicari pada deskripsi barang.

Setelah semua informasi tersebut diinput oleh pengguna, aplikasi membuka data yang sudah disimpan pada cheesecake.db kemudian melakukan pemrosesan lebih lanjut:

1. Ambil *id* untuk dijadikan *key* pada *dictionary* c dan r bagian program dinamis.
2. Gunakan Regex untuk mengambil informasi yang relevan untuk disimpan kedalam *dictionary* r.
3. Simpan harga di *dictionary* c sesuai metode pembelian yang dipilih pengguna.
4. Tandai *id* mana saja di dictionary c dan r yang valid (memiliki r > 0 atau c = r = 0) karena barang yang memiliki harga tetapi tidak memiliki keuntungan tidak akan diperhitungkan.

Pada akhir pemrosesan di bagian ini, didapat dictionary c dan r yang berisi data harga dan keuntungan beserta set valid yang berisi daftar id yang valid di kedua dictionary tersebut untuk digunakan pada bagian program dinamis.

D. *Program Dinamis*

```
# bagian dari cheesecake.py

# fungsi program dinamis
def f(c, r, v, k, x):
    if k == 0:
        p = [([id], r[k][id]) for id in v[k] if c[k][id] <= x]
    else:
        p = []
        for id in v[k]:
            if c[k][id] > x:
                continue
            ids, previous_f = f(c, r, v, k-1, x-c[k][id])
            ids.append(id)
            p.append((ids, r[k][id] + previous_f))
    max_id = max(range(len(p)), key=lambda x: p[x][1])
    return p[max_id]
```

```
# bagian dari cheesecake.py

# pemanggilan fungsi program dinamis
ids, _ = f(cost, result, valid, 1, input_budget)
total_cost = 0

# pencetakan hasil
print('Optimal Food\n')
if (ids[0] == 0):
    print('No food.')
else:
    r = c.execute('select c.id, name, description, icon, p.' + p
rice + ', duration, level from items i, consumables c, prices p
where i.id = c.id and p.id = c.id and c.id = ?', (ids[0],)).fetc
hone()
    print(f'ID    : {r[0]}')
    print(f'Name  : {r[1]}')
    print(f'Level : {r[6]}')
    print(f'Time  : {r[5]/1000/60} minutes')
    print(f'Icon  : {r[3]}')
    print('Desc. :')
    print(r[2].replace('\\n', '\n'))
    print(f'Price : {moneytalk(r[4])} each')
    total_cost += r[4]
```

```
print('\nOptimal Utility\n')
if (ids[1] == 0):
    print('No utility.')
else:
    r = c.execute('select c.id, name, description, icon, p.' + p
rice + ', duration, level from items i, consumables c, prices p
where i.id = c.id and p.id = c.id and c.id = ?', (ids[1],)).fetc
hone()
    print(f'ID    : {r[0]}')
    print(f'Name  : {r[1]}')
    print(f'Level : {r[6]}')
    print(f'Time  : {r[5]/1000/60} minutes')
    print(f'Icon  : {r[3]}')
    print('Desc. :')
    print(r[2].replace('\\n', '\n'))
    print(f'Price : {moneytalk(r[4])} each')
    total_cost += r[4]

print('\nTotal Cost: ' + moneytalk(total_cost) + '\n')
```

Gambar 4.4 Cuplikan kode cheesecake.py bagian program dinamis

Pada bagian ini, dibuat implementasi dari fungsi program dinamis (f) yang mengambil ketiga data hasil olahan bagian sebelumnya (c, r, dan v = valid) beserta tahap (k) dan budget (x). Fungsi tersebut mengembalikan dua benda dalam satu *tuple*: daftar *id* yang dipilih dalam seluruh tahap 1, 2, ..., k dan keuntungan terbesar yang didapat.

Fungsi ini diimplementasikan sesuai penyelesaian permasalahan *capital budgeting* dengan program dinamis. Pada basis (k = 0), fungsi mengembalikan *id* dan keuntungan dari makanan yang memberikan keuntungan terbesar dengan harga dibawah x. Pada bagian rekurens (yang hanya mungkin terjadi pada k = 1), fungsi mengembalikan *id* kombinasi makanan dan barang utilitas yang memberikan keuntungan terbesar dengan harga total dibawah x.

V. HASIL DAN PENGUJIAN

A. *Pengujian Level = 50; Budget = 100 copper; Instant Buy; Status=[Power, Precision]*

```
Cheesecake 1.0.0

Enter character level: 50
```

Gambar 5.1 Pengisian level

```
Cheesecake 1.0.0

Enter budget (in copper coins): 100
```

Gambar 5.2 Pengisian *budget*

```
Cheesecake 1.0.0

Use instant buy price? (Y/n): Y
```

Gambar 5.3 Pemilihan metode pembelian

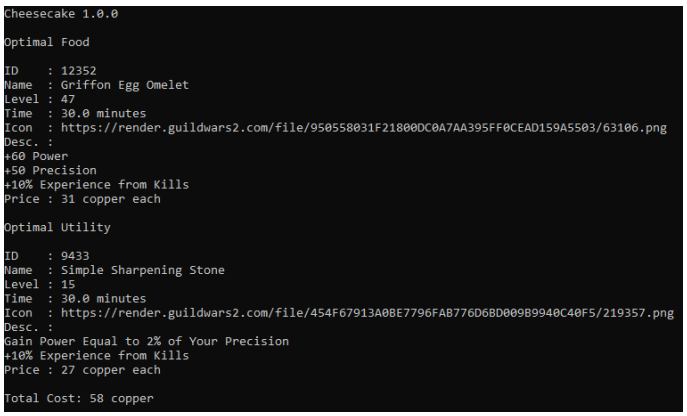
```
Cheesecake 1.0.0

Status list:
(1) Power
(2) Precision
(3) Ferocity
(4) Toughness
(5) Vitality
(6) Concentration
(7) Condition Damage
(8) Expertise
(9) Healing Power
(10) Magic Find
(11) Experience (gained)
(12) Experience (from kills)
Enter id(s) of stat(s) to find (comma separated): 1,2
```

Gambar 5.4 Pemilihan status

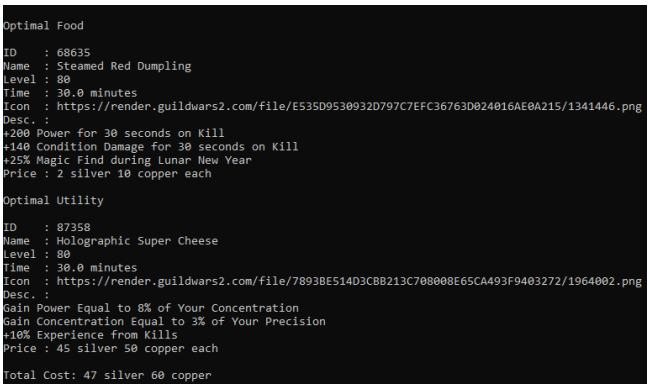
Gambar 5.5 Hasil uji A

B. *Pengujian Level = 50; Budget = 100 copper; Buy Order; Status=[Power, Precision]*



Gambar 5.6 Hasil uji B

C. *Pengujian Level = 80; Budget = 1000000 copper; Instant Buy; Status=[Power]*



Gambar 5.7 Hasil uji C

VI. KESIMPULAN

Sebagai kesimpulan, didapat bahwa pencarian makanan dan barang utilitas optimal dalam suatu batasan harga sangat memungkinkan untuk dilakukan dengan strategi program dinamis.

Berdasarkan implementasi yang dibuat, didapat bahwa implementasi program dinamis sangatlah sederhana, mudah dibuat, dan menjadi alternatif yang sangat menarik ketika lama pencarian bukanlah masalah dan diperlukan nilai yang benar-benar optimal.

Kendala yang terjadi dalam pengerjaan makalah ini mayoritas terjadi pada bagian pengumpulan data yang dipersulit oleh waktu akses data yang sangat lambat dan penyimpanan data yang perlu dipikirkan lebih lanjut.

VIDEO LINK AT YOUTUBE

https://youtu.be/pQxcQjRwVCU.

UCAPAN TERIMA KASIH

Pertama-tama penulis bersyukur atas berkat dan rahmat Allah SWT. karena dengan kehendak-Nya penulis dimudahkan dalam menyelesaikan makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada orang tua, sahabat, dan teman-teman penulis yang senantiasa memberi dukungan, motivasi, dan kasih sayang selama proses penulisan makalah ini. Penulis turut mengucapkan terima kasih kepada ke dosen Strategi Algoritma yang mengajar di kelas penulis, Ibu Masayu Leylia Khodra, yang telah memberikan pelajaran tentang Program Dinamis di mata kuliah Strategi Algoritma ini.

REFERENCES

[1] https://wiki.guildwars2.com/wiki/API:Main diakses 20:43 WIB, 4 Mei 2020

[2] https://wiki.guildwars2.com/wiki/API:2/items diakses 20:44 WIB, 4 Mei 2020.

[3] https://wiki.guildwars2.com/wiki/API:2/commerce/prices diakses 20:45 WIB, 4 Mei 2020.

[4] https://www.json.org/json-en.html diakses 20:54 WIB, 4 Mei 2020.

[5] https://techterms.com/definition/regular_expression diakses 20:59 WIB, 4 Mei 2020.

[6] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf diakses 21:01, 4 Mei 2020.

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, 29 April 2020

Muhammad Rizky Ismail Faizal