Aplikasi Algoritma FFD dan BFD dalam Pembuatan Fitur *Auto-Sort Inventory* untuk Permainan Resident Evil 4

Bill Clinton - 13521064
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521064@std.stei.itb.ac.id

Abstract*—Permainan Resident Evil 4 adalah permainan *Survival Horror* yang dikembangkan dan dirilis oleh Capcom pada tahun 2005. Dalam permainan ini, pemain akan bermain sebagai agen bernama Leon S. Kennedy. Pemain diberikan misi untuk menyelamatkan anak perempuan presiden Amerika Serikat, yaitu Ashley Graham, yang telah diculik oleh sekte religius di pedesaan negara Spanyol. Dalam perjalanan ini, pemain harus melawan berbagai macam monster yang telah terinfeksi parasit mematikan. Untuk bertahan hidup, pemain akan mengumpulkan dan menggunakan sumber daya, seperti senjata dan obat-obatan yang dapat disimpan dalam *inventory* sehingga dapat dibawa kemanapun. *Inventory* ini terbatas sehingga pemain harus menggunakan sumber daya yang dimilikinya secara efisien. Di samping penggunaan yang efisien, penting juga bagi pemain untuk bisa menyimpan barang dalam *inventory* dengan efisien sehingga tidak ada tempat kosong yang terbuang secara sia-sia. Dalam permainan ini, belum ada fitur *auto-sort* untuk *inventory* sehingga pemain harus mengefisiensikan penyimpanan barang dalam *inventory* secara manual. Oleh karena itu, dalam makalah ini, akan dibahas pembuatan fitur tersebut menggunakan algoritma *First Fit Decreasing* (FFD) dan *Best Fit Decreasing* (BFD).

Keywords*—Resident Evil 4, *inventory*, *First Fit Decreasing*, *Best Fit Decreasing

I. PENDAHULUAN

Pada tahun-tahun belakangan ini, permainan video atau *video game* menjadi hal yang sangat populer dan *trending* di tengah masyarakat. *Video game* adalah salah satu bentuk hiburan yang dapat dinikmati oleh berbagai kalangan, mulai dari anak-anak hingga orang dewasa. *Video game* dapat digunakan dari sebagai sarana pelepasan penat hingga sebagai sarana untuk profesi misalnya saja *game streamer*. Kategori *video game* sendiri ada berbagai macam, misalnya *action*, *adventure*, *arcade*, *board*, *casual*, *horror*, *indie*, *puzzle*, *racing*, *role playing*, *RPG*, *simulation*, *space*, *sports*, *strategy*, *survival*, *trivia*, dan *word*.

Resident Evil 4 merupakan permainan berkategori *horror* dan *survival*. Permainan ini adalah permainan *third-person shooter*. Dalam permainan ini, pemain akan bermain sebagai agen bernama Leon S. Kennedy yang diberi misi untuk menyelamatkan anak perempuan presiden Amerika Serikat, yaitu Ashley Graham. Sepanjang perjalanan, pemain harus bertahan hidup menghadapi monster-monster yang terinfeksi parasit dengan menggunakan sumber daya, misalnya senjata, peluru, hingga obat-obatan. Pemain dapat mengumpulkan sumber daya tersebut sepanjang perjalanan, misalnya dengan menghancurkan *breakable object*, seperti tong kayu atau dengan membeli dari *merchant*. Sumber daya ini nantinya akan disimpan dalam *inventory* sehingga dapat dibawa oleh pemain.

Sumber daya merupakan hal yang krusial dalam permainan ini. Tanpa sumber daya yang cukup, pemain akan kewalahan untuk menghadapi dan mengalahkan monster-monster sehingga akan sulit bertahan hidup. Oleh karena itu, pemain harus menggunakan sumber daya dalam permainan ini secara efisien. Selain itu, *inventory* yang dibawa dalam permainan ini juga terbatas sehingga pemain harus mengefisiensikan penempatan sumber daya dalam *inventory*. Dengan pengefisiensian *inventory* tersebut, pemain dapat membawa banyak sumber daya yang nantinya akan sangat berguna selama perjalanan pemain.

Dalam permainan Resident Evil 4 yang dirilis pada tahun 2005 ini, belum ada fitur untuk mengefisiensikan penempatan sumber daya dalam *inventory* secara otomatis sehingga pemain harus melakukannya secara manual. Oleh karena itu, dalam makalah ini, penulis akan membahas penerapan algoritma *First Fit Decreasing* (FFD) dan *Best Fit Decreasing* (BFD) untuk mensimulasikan fitur tersebut. Dengan makalah ini, diharapkan pembaca dapat memperoleh wawasan mengenai algoritma-algoritma yang dapat digunakan untuk mensimulasikan *inventory* dan pengefisiensiannya dalam suatu permainan.

Gambar 1: Permainan Resident Evil 4
Sumber: https://store.steampowered.com/agecheck/app/2050650/?psafe_param=1&gclid=EAIaIQobChMInqudwMi__gIVymZ9Ch0NaA0AEAAYASAAEgKfyfD_BwE

II. TEORI DASAR

A. Definisi Algoritma

Algoritma adalah sebuah prosedur langkah demi langkah yang menjelaskan serangkaian instruksi terbatas yang digunakan untuk menyelesaikan tugas tertentu. Algoritma memiliki beberapa karakteristik, yaitu *input* (masukan), *output* (keluaran), *unambiguity* (instruksi yang jelas), *finiteness* (instruksi harus *countable*/dapat terhitung), *effectiveness*, dan *language independence* (algoritma dikembangkan secara independen dari bahasa yang mendasarinya). Dengan *language independence* ini, algoritma dapat diimplementasikan dalam lebih dari satu bahasa pemrograman.

B. Algoritma Greedy

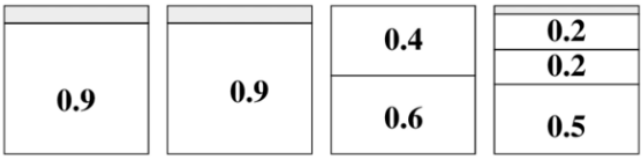
Algoritma *Greedy* adalah algoritma yang mengaplikasikan konsep *greedy* dalam penyelesaian suatu tugas tertentu. Algoritma ini sering digunakan untuk permasalahan optimasi. Permasalahan optimasi ini secara umum dapat dibagi menjadi dua, yakni permasalahan *maximization* (maksimasi) dan permasalahan *minimization* (minimasi). Dalam algoritma ini, pada setiap langkah, akan diambil pilihan terbaik yang dapat diperoleh saat itu tanpa memperhatikan konsekuensi kedepannya dengan harapan bahwa pemilihan optimum lokal pada setiap langkah ini nantinya akan berakhir dengan optimum global.

C. Masalah *Bin Packing*

Permasalahan *bin packing* adalah salah satu contoh permasalahan optimasi. Dalam permasalahan ini, diberikan barang-barang dengan ukuran yang berbeda yang harus dimasukkan ke *bin-bin* terbatas dengan kapasitas tertentu sedemikian sehingga jumlah *bin* yang digunakan minimal (secara umum dapat dituliskan sebagai n buah barang dengan ukuran $s_1, s_2, \ldots, s_n$ dengan $0 \leq s_i \leq 1$ dan $1 \leq i \leq n$). Contoh aplikasi dari permasalahan ini adalah pengisian tempat sampah daur ulang serta pemuatan truk dengan batasan berat tertentu.

D. Algoritma *First Fit Decreasing* (FFD)

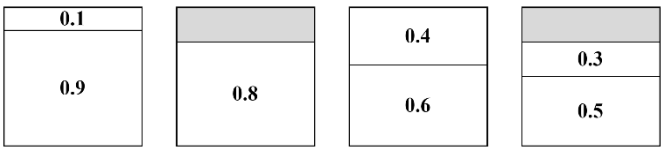
Algoritma *First Fit Decreasing* (FFD) merupakan salah satu variasi dari algoritma *Greedy*. Algoritma ini biasa digunakan untuk menyelesaikan permasalahan *bin packing*. Dalam algoritma ini, pertama, semua barang diurutkan secara menurun berdasarkan ukurannya. Setelah itu, *bin* dipindai secara berurutan dan diisi dengan barang baru yang ukurannya cukup untuk dapat masuk ke *bin* tersebut. Misalnya, jika ada barang barang yang direpresentasikan dengan himpunan $L = \{0.9, 0.2, 0.2, 0.9, 0.4, 0.6, 0.5\}$. Barang-barang tersebut diurutkan dulu secara menurun menurut ukurannya menjadi $L' = \{0.9, 0.9, 0.6, 0.5, 0.4, 0.2, 0.2\}$. Selanjutnya, bin dipindai secara berurutan dan diisi barang dengan ukuran yang cukup dari himpunan L' secara berurutan (mulai dari barang dengan ukuran 0.9) dengan catatan bahwa sebuah *bin* baru dapat dibuat hanya ketika sebuah barang tidak muat dalam *bin-bin* sebelumnya. Hasil dari penggunaan algoritma FFD tersebut ada dalam gambar berikut.



Gambar 2.1: Contoh Penggunaan Algoritma *First Fit Decreasing* (FFD)
Sumber:
https://www.researchgate.net/figure/First-Fit-and-First-Fit-Decreasing-algorithms-for-the-BPP_fig1_265974871

E. Algoritma *Best Fit Decreasing* (BFD)

Algoritma *Best Fit Decreasing* juga merupakan salah satu variasi dari algoritma *Greedy*. Algoritma ini juga biasa digunakan dalam penyelesaian permasalahan *bin packing*. Dalam algoritma ini, pertama, semua barang diurutkan secara menurun berdasarkan ukurannya. Selanjutnya, barang baru ditempatkan dalam *bin* yang membuat ruang kosong sisanya setelah penempatan tersebut minimal. Jika barang tersebut tidak muat dalam *bin* manapun, buat *bin* yang baru. Misalnya, jika ada barang barang yang direpresentasikan dengan himpunan $L = \{0.9, 0.8, 0.1, 0.5, 0.4, 0.6, 0.3\}$. Barang-barang tersebut diurutkan dulu secara menurun menurut ukurannya menjadi $L' = \{0.9, 0.8, 0.6, 0.5, 0.4, 0.3, 0.1\}$. Selanjutnya, barang dari himpunan L' dimasukkan ke bin secara berurutan (mulai dari barang dengan ukuran 0.9) sedemikian sehingga ruang kosong sisa setelah penempatan tersebut minimal. Hasil dari penggunaan algoritma FFD tersebut ada dalam gambar berikut.



Gambar 2.2: Contoh Penggunaan Algoritma *Best Fit Decreasing* (BFD)
Sumber:
Dokumen Pribadi

F. Ukuran Sumber Daya dalam Permainan Resident Evil 4

Sumber daya untuk pertahanan pemain dalam permainan Resident Evil 4 secara umum dapat dibagi menjadi beberapa golongan, yakni senjata, aksesoris senjata, peluru, dan obat-obatan. Seluruh sumber daya tersebut dapat disimpan dalam suatu *inventory* dengan ukuran yang terbatas untuk dibawa kemanapun pemain pergi.

Gambar 2.3: *Inventory* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Berikut adalah gambaran ukuran beberapa sumber daya golongan senjata dalam permainan Resident Evil 4 yang diimplementasikan dalam program yang penulis buat.

Gambar 2.4: Senjata *Broken Butterfly* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Gambar 2.5: Senjata *Incendiary Grenade* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Berikut adalah gambaran ukuran beberapa sumber daya golongan aksesoris senjata dalam permainan Resident Evil 4 yang diimplementasikan dalam program yang penulis buat.

Gambar 2.6: Aksesoris *Scope (Rifle)* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=mHUMB4SYipY

Gambar 2.7: Aksesoris *Stock (TMP)* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=mHUMB4SYipY

Berikut adalah gambaran ukuran beberapa sumber daya golongan peluru dalam permainan Resident Evil 4 yang diimplementasikan dalam program yang penulis buat.

Gambar 2.8: *Handgun Ammo* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Gambar 2.9: *Magnum Ammo* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Berikut adalah gambaran ukuran beberapa sumber daya golongan obat-obatan dalam permainan Resident Evil 4 yang diimplementasikan dalam program yang penulis buat.

Gambar 2.10: *Chicken Egg* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=mHUMB4SYipY

Gambar 2.11: *Mixed Herb* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=t8xz5Lcyn94

Gambar 2.12: *First Aid Spray* dalam Permainan Resident Evil 4
Sumber:
https://www.youtube.com/watch?v=mHUMB4SYipY

III. Aplikasi Algoritma FFD dan BFD Dalam Pembuatan Fitur Auto-Sort Inventory

Dalam pembuatan fitur *auto-sort* untuk *inventory* yang menyerupai *inventory* pada permainan Resident Evil 4 ini, penulis menggunakan bahasa pemrograman *python*. Penulis menggambarkan sumber daya dalam permainan sebagai suatu objek yang memiliki atribut ID, nama, dan ukuran, yang meliputi lebar dan tinggi. Selain itu, dalam penempatannya dalam *inventory*, sumber daya bisa diputar 90 derajat sehingga pada kelas untuk objeknya, didefinisikan *method* untuk memutar objek tersebut. Berikut adalah cuplikan implementasi untuk kelas tersebut.

```python
# Item.py
class Item:
    def __init__(self, id, name, width, height):
        self.id = id
        self.name = name
        self.width = width
        self.height = height

    def get_id(self):
        return self.id

    def get_name(self):
        return self.name

    def get_width(self):
        return self.width

    def get_height(self):
        return self.height

    def rotate_item(self):
        temp_width = self.width
        temp_height = self.height
        self.width = temp_height
        self.height = temp_width

    def get_rotated_item(self):
        return Item(self.get_id(), self.get_name(), self.get_height(), self.get_width())
```

Gambar 3.1: Kelas Item untuk Sumber Daya dalam Permainan Resident Evil 4
Sumber:
Dokumen Pribadi

Untuk *inventory*, penulis menggambarkannya sebagai suatu objek yang memiliki atribut lebar, tinggi, serta matriks yang menggambarkan isi *inventory*. Pada kelas untuk objek ini, didefinisikan *method-method*, misalnya *method* untuk menampilkan isi *inventory*, *method* untuk menghapus isi *inventory*, *method* untuk menempatkan dan menambahkan barang ke *inventory*, serta *method* untuk fitur *auto-sort inventory* melalui algoritma FFD dan BFD (beserta *method* pembantunya). Berikut adalah cuplikan implementasi untuk kelas tersebut.

```python
class Inventory:
    def __init__(self, width, height):
        self.width = width
        self.height = height
        self.matrix = [[None for j in range(width)] for i in range(height)]

    def printInventory(self):
        for i in range(self.height):
            for j in range(self.width):
                print(self.matrix[i][j] or "XX", end=' ')
            print()

    def ffd(self, items_list):
        items_list.sort(key = lambda item: item.width * item.height, reverse = True)
        for item in items_list:
            placed = False
            for i in range(self.height):
                if placed:
                    break
                for j in range(self.width):
                    if self.fits(item, i, j):
                        self.place(item, i, j)
                        placed = True
                        break
                    elif self.fits(item.get_rotated_item(), i, j):
                        item.rotate_item()
                        self.place(item, i, j)
                        placed = True
                        break
```

Gambar 3.2: Kelas Inventory untuk *Inventory* dalam Permainan Resident Evil 4 (Bagian 1)
Sumber:
Dokumen Pribadi

```python
    def bfd(self, items):
        for item in items:
            best_row, best_col, best_score = None, None, None
            for i in range(self.height):
                for j in range(self.width):
                    if self.fits(item, i, j):
                        score = self.get_score(item, i, j)
                        if best_score is None or score < best_score:
                            best_row, best_col, best_score = i, j, score
                    elif self.fits(item.get_rotated_item(), i, j):
                        score = self.get_score(item.get_rotated_item(), i, j)
                        if best_score is None or score < best_score:
                            best_row, best_col, best_score = i, j, score
                            item.rotate_item()
            if best_row is not None:
                self.place(item, best_row, best_col)

    def get_score(self, item, row, col):
        max_score = 0
        for i in range(row, min(row + item.height, self.height)):
            for j in range(col, min(col + item.width, self.width)):
                if self.matrix[i][j] is None: # Empty Place
                    max_score += 1
        return max_score

    def fits(self, item, row, col):
        for i in range(item.height):
            for j in range(item.width):
                if row + i >= self.height - 1 or col + j >= self.width - 1 or self.matrix
                [row + i][col + j] is not None:
                    return False
        return True

    def place(self, item, row, col):
        for i in range(item.height):
            for j in range(item.width):
                self.matrix[row + i][col + j] = item.id
```

Gambar 3.3: Kelas Inventory untuk *Inventory* dalam Permainan Resident Evil 4 (Bagian 2)
Sumber:
Dokumen Pribadi

```python
    def add(self, item, row, col):
        if row + item.height > self.height or col + item.width > self.width:
            print("Not Enough Space!")
            return False

        for i in range(item.height):
            for j in range(item.width):
                if self.matrix[row + i][col + j] is not None:
                    print("Space Occupied!")
                    return False

        for i in range(item.height):
            for j in range(item.width):
                self.matrix[row + i][col + j] = item.id
        return True

    def clear(self):
        self.matrix = [[None for j in range(self.width)] for i in range(self.height)]
```

Gambar 3.4: Kelas Inventory untuk *Inventory* dalam Permainan Resident Evil 4 (Bagian 3)
Sumber:
Dokumen Pribadi

Selain itu, ada beberapa objek yang penulis telah buat yang menyerupai objek sebenarnya pada permainan Resident Evil 4. Objek yang nantinya akan dimasukkan ke *inventory* ini nantinya dapat diperoleh secara acak oleh pengguna program. Implementasi objek-objek ini ada dalam *file* program utama. Berikut adalah cuplikan implementasi untuk objek-objek tersebut.

```
# Common Resident Evil 4 Items
broken_butterfly = Item("01", "Broken Butterfly", 4, 2)
tmp = Item("02", "TMP", 3, 2)
rifle = Item("03", "Rifle", 8, 1)
rifle_sa = Item("04", "Rifle (Semi Auto)", 7, 2)
striker = Item("05", "Striker", 5, 2)
black_tail = Item("06", "Blacktail", 3, 2)
incendiary_grenade = Item("07", "Incendiary Grenade", 1, 2)
hand_grenade = Item("08", "Hand Grenade", 1, 2)
flash_grenade = Item("09", "Flash Grenade", 1, 2)
handgun_ammo = Item("10", "Handgun Ammo", 2, 1)
tmp_ammo = Item("11", "TMP Ammo", 2, 1)
magnum_ammo = Item("12", "Magnum Ammo", 2, 1)
rifle_ammo = Item("13", "Rifle Ammo", 2, 1)
shotgun_shells = Item("14", "Shotgun Shells", 2, 1)
herb = Item("15", "Herb", 1, 2)
mixed_herb = Item("16", "Mixed Herb", 1, 2)
scope_r = Item("17", "Scope (Rifle)", 3, 1)
stock_tmp = Item("18", "Stock (TMP)", 2, 2)
first_aid_spray = Item("19", "First Aid Spray", 1, 2)
chicken_egg = Item("20", "Chicken Egg", 1, 1)
available_re_items = [broken_butterfly, tmp, rifle, rifle_sa, striker, black_tail,
incendiary_grenade, hand_grenade, flash_grenade, handgun_ammo, tmp_ammo, magnum_ammo,
rifle_ammo, shotgun_shells, herb, mixed_herb, scope_r, stock_tmp, first_aid_spray,
chicken_egg]
```

Gambar 3.5: Objek untuk Sumber Daya Umum dalam Permainan Resident Evil 4
Sumber:
Dokumen Pribadi

Kode lengkap program utama untuk mensimulasikan fitur *auto-sort inventory* ini dapat diakses pada tautan berikut.
Link: https://github.com/billc27/Resident-Evil-4-Inventory-Simulator

IV. Hasil Pengujian Program

Pada program utama, pengguna juga dapat mendefinisikan objek apapun yang diinginkan dan objek tersebut tetap bisa dimasukkan ke *inventory*. Dalam tampilan *inventory*, XX menandakan tempat yang kosong. Berikut adalah tampilan saat program utama dijalankan.

```
==================================o==================================
Welcome to Resident Evil 4 Inventory Simulator (with Auto-Sort Feature)
Please input the desired number of columns for the inventory: 11
Please input the desired number of rows for the inventory: 8
Do you want to get random items (y/n): y
Acquired Herb
Item ID: 15
Item size: 1 x 2
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
```

Gambar 4.1: Hasil Pengujian Program Utama (Bagian 1)
Sumber:
Dokumen Pribadi

```
Do you want to rotate the item (y/n): n
Choose the starting point for the item placement!
Row: 2
Column: 3
Item is placed successfully!
Do you want to add another item? (y/n): y
You can continue adding another item.
Do you want to get random items (y/n): y
Acquired Scope (Rifle)
Item ID: 17
Item size: 3 x 1
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX 15 XX XX XX XX XX XX XX
XX XX XX 15 XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
Do you want to rotate the item (y/n): n
Choose the starting point for the item placement!
Row: 1
Column: 2
Item is placed successfully!
```

Gambar 4.2: Hasil Pengujian Program Utama (Bagian 2)
Sumber:
Dokumen Pribadi

```
Do you want to add another item? (y/n): y
You can continue adding another item.
Do you want to get random items (y/n): n
Please input the item manually!
Enter item name: Item1
Enter item width: 3
Enter item height: 5
Item created successfully!
Item ID: 21
XX XX XX XX XX XX XX XX XX XX XX
XX XX 17 17 17 XX XX XX XX XX XX
XX XX XX 15 XX XX XX XX XX XX XX
XX XX XX 15 XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
Do you want to rotate the item (y/n): n
```

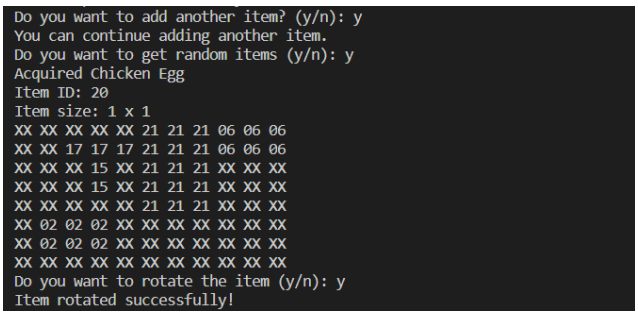
Gambar 4.3: Hasil Pengujian Program Utama (Bagian 3)
Sumber:
Dokumen Pribadi

```
Row: 1
Column: 4
Space Occupied!
Item is not placed successfully. Please try again!
Choose the starting point for the item placement!
Row: 5
Column: 1
Not Enough Space!
Item is not placed successfully. Please try again!
Choose the starting point for the item placement!
Row: 0
Column: 5
Item is placed successfully!
Do you want to add another item? (y/n): y
You can continue adding another item.
Do you want to get random items (y/n): y
Acquired TMP
Item ID: 02
Item size: 3 x 2
XX XX XX XX XX 21 21 21 XX XX XX
XX XX 17 17 17 21 21 21 XX XX XX
XX XX XX 15 XX 21 21 21 XX XX XX
XX XX XX 15 XX 21 21 21 XX XX XX
XX XX XX XX XX 21 21 21 XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
Do you want to rotate the item (y/n): n
Choose the starting point for the item placement!
Row: 5
Column: 1
Item is placed successfully!
```

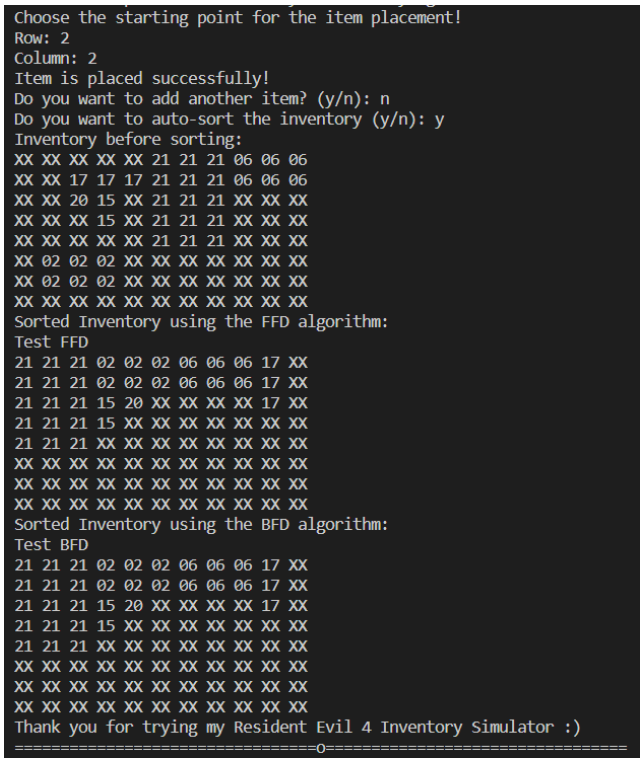
Gambar 4.4: Hasil Pengujian Program Utama (Bagian 4)
Sumber:
Dokumen Pribadi

```
Do you want to add another item? (y/n): y
You can continue adding another item.
Do you want to get random items (y/n): y
Acquired Blacktail
Item ID: 06
Item size: 3 x 2
XX XX XX XX XX 21 21 21 XX XX XX
XX XX 17 17 17 21 21 21 XX XX XX
XX XX XX 15 XX 21 21 21 XX XX XX
XX XX XX 15 XX 21 21 21 XX XX XX
XX XX XX XX XX 21 21 21 XX XX XX
XX 02 02 02 XX XX XX XX XX XX XX
XX 02 02 02 XX XX XX XX XX XX XX
XX XX XX XX XX XX XX XX XX XX XX
Do you want to rotate the item (y/n): n
Choose the starting point for the item placement!
Row: 0
Column: 7
Space Occupied!
Item is not placed successfully. Please try again!
Choose the starting point for the item placement!
Row: 0
Column: 8
Item is placed successfully!
```

Gambar 4.5: Hasil Pengujian Program Utama (Bagian 5)
Sumber:
Dokumen Pribadi



Gambar 4.6: Hasil Pengujian Program Utama (Bagian 6)
Sumber:
Dokumen Pribadi



Gambar 4.7: Hasil Pengujian Program Utama (Bagian 7)
Sumber:
Dokumen Pribadi

V. KESIMPULAN

Resident Evil 4 adalah salah satu permainan video dengan genre *horror survival*. Permainan ini merupakan permainan *third-person shooter*. Dalam permainan ini, pemain yang berperan sebagai agen bernama Leon S. Kennedy, diberi tugas untuk menyelamatkan anak perempuan presiden Amerika Serikat, yaitu Ashley Graham. Dalam perjalanannya, pemain harus bertahan hidup melawan monster-monster yang terinfeksi parasit dengan menggunakan sumber daya yang dapat disimpan dalam suatu *inventory*. Untuk meningkatkan kemungkinan bertahan hidup, selain mengefisiensikan penggunaan sumber daya, pemain juga harus mengefisiensikan penempatan sumber daya dalam *inventory*. Dalam hal ini, algoritma *First Fit Decreasing* (FFD) dan *Best Fit Decreasing* (BFD) dapat digunakan untuk membuat fitur *auto-sort* untuk membantu pemain mengefisiensikan penempatan tersebut. Dengan algoritma-algoritma tersebut, pemain dapat memiliki peluang yang lebih besar untuk bisa bertahan hidup sehingga misi bisa berhasil diselesaikan.

VI. UCAPAN TERIMA KASIH

Puji syukur kepada Tuhan Yang Maha Esa karena atas berkat dan rahmat-Nya, penulis bisa menyelesaikan makalah dengan judul "Aplikasi Algoritma FFD dan BFD dalam Pembuatan Fitur *Auto-Sort Inventory* untuk Permainan Resident Evil 4" ini. Penulis juga ingin mengucapkan terima kasih kepada Dr. Nur Ulfa Maulidevi, S.T., M.Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma. Selain itu, penulis juga ingin mengucapkan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. atas *slide* kuliah IF2211 Strategi Algoritma yang disediakan di internet yang menjadi salah satu referensi yang penulis gunakan dalam penulisan makalah ini. Tidak lupa, penulis juga ingin mengucapkan terima kasih kepada pihak luar yang menyediakan referensi berupa informasi serta gambar yang dapat penulis gunakan dalam penulisan makalah ini. Terakhir, penulis ingin mengucapkan terima kasih kepada pihak Capcom karena telah menyediakan permainan video yang sangat unik dan menarik. Penulis berharap makalah ini bisa bermanfaat bagi orang-orang yang tertarik dalam pengembangan suatu permainan yang memiliki fitur *inventory*. Akhir kata, penulis memohon maaf jika terdapat kesalahan-kesalahan dalam penulisan makalah ini dan penulis berterima kasih kepada seluruh orang yang telah membaca makalah ini.

LAMPIRAN

Berikut adalah tautan untuk *source code* lengkap dari program yang penulis buat.
Link: https://github.com/billc27/Resident-Evil-4-Inventory-Simulator

REFERENSI

[1] B. Clinton, "Resident Evil 4 Inventory Simulator," [Online]. Available: https://github.com/billc27/Resident-Evil-4-Inventory-Simulator. [Diakses pada 19/05/2023 23.20].

[2] Capcom, "Resident Evil 4, " Steam Store Webpage, [Online]. Available: https://store.steampowered.com/agecheck/app/2050650/?psafe_param=1&gclid=EAIaIQobChMInqudwMi__gIVymZ9Ch0NaA0AEAAYASAAEgKfyfD_BwE. [Diakses pada 11/05/2023 19.46].

[3] coolkarmact, "Resident Evil 4 Inventory ASMR," YouTube, [Online]. Available: https://www.youtube.com/watch?v=t8xz5Lcyn94. [Diakses pada 12/05/2023 08.14].

[4] M. T. Goodrich "Bin Packing," University of California, Irvine, [Online]. Available: https://www.ics.uci.edu/~goodrich/teach/cs165/notes/BinPacking.pdf. [Diakses pada 11/05/2023 23.09].

[5] P. Ongkunaruk, "First Fit and First Fit Decreasing algorithms for the BPP," ResearchGate, [Online]. Available: https://www.researchgate.net/figure/First-Fit-and-First-Fit-Decreasing-algorithms-for-the-BPP_fig1_265974871. [Diakses pada 11/05/2023 23.21].

[6] Reynik, "Resident Evil 4 sorting inventory ASMR," YouTube, [Online]. Available: https://www.youtube.com/watch?v=mHUMB4SYipY. [Diakses pada 12/05/2023 09.02].
[7] R. Munir, "Algoritma Greedy (Bagian 1)," [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. [Diakses pada 11/05/2023 21.30].
[8] Simplilearn, "What Is An Algorithm? Characteristics, Types, and How to write it," [Online]. Available: https://www.simplilearn.com/tutorials/data-structure-tutorial/what-is-an-algorithm. [Diakses pada 11/05/2023 21.14].

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, 20 Mei 2023

Bill Clinton 13521064