

# Aplikasi Algoritma Greedy dalam Penyelesaian *ConSumo* pada Permainan *Bully: Scholarship Edition*

Jose Galbraith Hasintongan 13519022  
Program Studi Teknik Informatika  
Sekolah Teknik Elektro dan Informatika  
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung  
galbraith.jose02@gmail.com

**Abstract**—Algoritma greedy merupakan algoritma sederhana untuk memecahkan persoalan optimasi. Algoritma greedy mengambil pilihan paling optimal di setiap langkahnya yang disebut dengan solusi optimum lokal. Dari setiap optimum lokal tersebut, diharapkan bahwa akan membawa kita ke solusi optimum global.

*Bully* merupakan *video game* yang dirilis tahun 2006. *Video game* ini adalah *video game* yang *open-world sandbox action-adventure* dan berlatar tempat di suatu sekolah berasrama dan pemain akan bermain sebagai anak remaja bernama Jimmy Hopkins. *Bully: Scholarship Edition* merupakan versi *game Bully* yang dirilis tahun 2008 yang telah di-update dan secara grafik lebih ditingkatkan, ada tambahan fitur, serta perbaikan dari *bug* di versi sebelumnya.

*ConSumo* merupakan salah satu *arcade game* atau *minigame* yang ada di dalam *video game bully* ini. *Goal* utama dari *minigame* ini adalah memakan makanan sebanyak mungkin sampai mendapat *score* lebih besar dari 1010 lbs.

Makalah ini akan membahas mengenai aplikasi algoritma greedy untuk bertahan selama mungkin di dalam *minigame ConSumo* dan mendapat *score* tertinggi (*Highscore*) yaitu 1010 lbs yang dipegang oleh *Fatty Johnson*.

**Kata kunci**— Algoritma Greedy, *Consumo*, *Bully*, *Highscore*.

## I. PENDAHULUAN

### A. *Bully* (*Video Game*)

Permainan *Bully: Scholarship Edition* merupakan versi *video game Bully* yang telah di-update. *Bully* atau yang juga dikenal sebagai *Canis Canem Edit* di wilayah PAL (Australia, Selandia Baru, Britania Raya, dan negara-negara Eropa) adalah suatu permainan *third person action-adventure video game* yang dirilis oleh Rockstar Vancouver untuk PlayStation 2 pada 17 Oktober 2006 di Amerika Serikat. *Bully: Scholarship Edition* sendiri baru dirilis pada 24 Oktober 2008 untuk PC.

### B. *Gameplay*

*Video Game Bully* merupakan *open world sandbox action-adventure game* yang berlatar di suatu sekolah berasrama bernama Bullworth Academy, sebuah sekolah asrama fiksi di

New England. Karena merupakan *game* yang *open world*, pemain bebas untuk mengeksplorasi area sekolah atau kota atau juga bisa menyelesaikan misi utama.

*Bully* memiliki banyak sekali *minigames*, ada yang untuk mendapatkan uang dan ada yang untuk meningkatkan kemampuan Jimmy. Kelas-kelas di sekolah menggunakan *Minigames* dan memiliki level kesulitan yang selalu bertambah. Kelas Inggris merupakan *minigame* dengan huruf-huruf acak dan setelah berhasil menyelesaikan suatu level, Jimmy mendapatkan kemampuan baru seperti belajar untuk meminta maaf kepada orang lain dan juga meminta maaf kepada seorang prefek serta polisi.

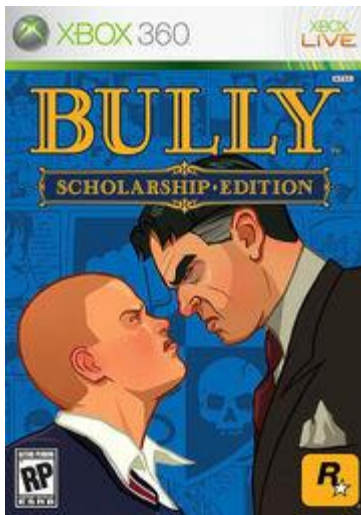
Jimmy memiliki banyak pilihan senjata yang ia bisa gunakan. Senjata yang ia gunakan sama seperti anak-anak nakal sekolah lainnya seperti, katapel, kelereng, *stink bombs*, *bottle rocket launcher*, dan *spud cannon*. Kendaraan yang bisa digunakan Jimmy dalam *game* adalah sepeda, *moped*, *go-kart*, *skateboard*, dan *lawn mower*.

### C. *Setting*

*Video game* ini berlatar tempat di Bullworth Academy yakni suatu sekolah berasrama fiksi di New England, Amerika Serikat. Jimmy didaftarkan di sekolah ini Ketika ibunya dan ayah tiri kelimanya memutuskan untuk pergi selama setahun untuk *honeymoon cruise*. *Timeline* dari *video game* ini tidak begitu jelas. *Developer* tidak hanya menggunakan *influence* dari satu periode. Mobil dan arsitektur bisa dilihat dari tahun 1970an, 1980an, atau 1990an. *Game* ini terbagi menjadi enam chapter yang bisa diselesaikan oleh pemain.

### D. *Karakter*

*Game* berfokus pada murid baru di Bullworth Academy yaitu Jimmy Hopkins. Dia disapa oleh Gary Smith, seorang yang licik dan Pete Kowalski, seorang murid yang pemalu. Demi bisa bertahan di sekolah ini, Jimmy harus bisa beradaptasi dan membuat suatu relasi dengan kelompok-kelompok disekolah ini - Bullies, Nerds, Preppies, Greasers dan Jocks – dan juga Townie Kids.



Gambar 1.1 Cover dari video game *Bully: Scholarship Edition*

(Sumber:

[https://bully.fandom.com/wiki/Bully:\\_Scholarship\\_Edition](https://bully.fandom.com/wiki/Bully:_Scholarship_Edition))



Gambar 1.2 *Cliques* atau kelompok yang ada di video game (Sumber:

[https://www.reddit.com/r/bully/comments/fzpz15/i\\_updated\\_my\\_bully\\_clique\\_chart\\_and\\_this\\_time\\_i/](https://www.reddit.com/r/bully/comments/fzpz15/i_updated_my_bully_clique_chart_and_this_time_i/))

Pada makalah ini, akan disampaikan algoritma *greedy* untuk penyelesaian salah satu *arcade game* yang bernama *ConSumo*.

## II. LANDASAN TEORI

### A. Algoritma Greedy

Algoritma *greedy* merupakan algoritma yang dipakai untuk persoalan optimasi. Persoalan optimasi artinya mencari solusi yang optimal untuk persoalan tersebut. Terdapat dua macam persoalan optimasi, yaitu:

1. Maksimasi (maximization)
2. Minimasi (minimization)

*Greedy* memiliki arti rakus atau tamak. Dengan begitu, algoritma *greedy* akan mengambil langkah yang lebih menguntungkan pada saat proses pengambilan keputusan. Prinsip dari *greedy* adalah “take what you can get now!” Dalam setiap langkahnya, algoritma *greedy* akan mengambil keputusan terbaik dalam menentukan pilihan untuk langkah selanjutnya. Keputusan terbaik pada setiap langkahnya disebut sebagai optimum lokal (*local optimum*). Pemilihan-pemilihan solusi optimum lokal tersebut diharapkan dapat mengarahkan program ke solusi optimum global

Pemetaan elemen – elemen dalam algoritma *greedy* sebagai berikut:

1. Himpunan kandidat (C): berisi kandidat yang akan dipilih pada setiap langkah (misal: simpul/sisi di dalam graf, *job*, *task*, koin, benda, karakter, dsb)
2. Himpunan solusi (S): berisi kandidat yang sudah dipilih
3. Fungsi solusi: menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (*selection function*): memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*): memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak)
6. Fungsi obyektif: memaksimalkan atau meminimumkan

Algoritma *greedy* mencari himpunan solusi (S) dari himpunan kandidat (C). Elemen-elem dalam himpunan solusi harus memenuhi syarat-syarat yang sudah dipenuhi oleh fungsi seleksi dan fungsi kelayakan.

Walaupun dari langkah-langkah yang menghasilkan solusi optimum lokal dan mengarahkan ke optimum global, optimum global tidaklah sama dengan solusi optimum (terbaik), tetapi hanyalah suboptimum atau pseudo-optimum. Hal ini dikarenakan algoritma *greedy* tidak beroperasi secara menyeluruh terhadap semua kemungkinan solusi yang ada (sebagaimana pada metode *exhaustive search*) dan terdapat beberapa fungsi seleksi yang berbeda, sehingga kita harus memilih fungsi yang tepat jika kita ingin algoritma menghasilkan solusi yang optimum.

Berikut ini merupakan skema umum dari algoritma *greedy*:

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi
Algoritma:
  S ← {} { inialisasi S dengan kosong }
```

```

while (not SOLUSI(S)) and (C ≠ { }) do
    x ← SELEKSI(C) { pilih sebuah kandidat dari C }
    C ← C - {x} { buang x dari C karena sudah dipilih }
    if LAYAK(S ∪ {x}) then { x memenuhi kelayakan
    untuk dimasukkan ke dalam himpunan solusi }
        S ← S ∪ {x} { masukkan x ke dalam
    himpunan solusi }
    endif
endwhile
{ SOLUSI(S) or C = { } }
if SOLUSI(S) then { solusi sudah lengkap }
    return S
else
    write('tidak ada solusi')
endif

```

Pada akhir setiap iterasi, solusi yang terbentuk adalah optimum lokal.

Pada akhir kalang *while-do*, diperoleh optimum global (jika ada).

Supaya lebih memahami algoritma greedy, bisa digunakan cerita seperti ini:

Bayangkan seseorang sedang melakukan pendakian untuk bisa mencapai puncak tertinggi yang mungkin dicapai oleh seorang pendaki. Orang ini sudah memiliki peta tetapi di peta tersebut ada ribuan jalan yang mungkin digunakan untuk mencapai puncak tertinggi. Orang ini terlalu malas untuk mengevaluasi seluruh kemungkinan yang ada. “*Be Greedy and short-sighted,*” tanpa menggunakan peta, akhirnya orang ini memutuskan untuk melakukan pendakian dan hanya melalui jalan yang paling miring ke atas. Algoritma greedy tidak pernah meninjau kembali pilihan yang sudah dia ambil sebelumnya, jika suatu ketika ada rintangan yang sangat sulit dilalui pendaki akibat ia tidak melihat peta (melihat kemungkinan lain yang lebih baik), maka bisa saja perjalanan tidak bisa dijalankan atau ia harus mencari jalan lain terdekat yang ia lihat.

## B. Consumo

Seperti yang sudah disebutkan sebelumnya, pada permainan *Bully* memiliki banyak *minigame* dan memiliki kelompok atau *cliques* yang memengaruhi jalan cerita atau kemampuan atau *ability* yang dimiliki oleh karakter utama kita, Jimmy Hopkins. Setiap menyelesaikan misi atau *chapter* dari *game* bisa saja ada muncul *non-storyline mission challenge* yang bisa diselesaikan untuk mendapat *save point* baru atau dalam hal ini tempat tinggal baru. Setelah pemain berhasil menyelesaikan *game* hingga *chapter* 3, muncul suatu *non-storyline challenge mission* yang bisa dikerjakan bernama “*Nerd Challenge.*” Sesuai

Namanya, misi ini merupakan misi yang diberikan oleh kelompok atau *clique Nerd.*

Jimmy Hopkins memasuki toko komik Dragon’s Wing Comics dan di sana, penjaga toko bernama Zack Owens, memberitahu Jimmy bahwa ada “*club meeting*” di *basement* toko dan Jimmy segera turun ke *basement* melalui pintu yang ada di belakang *counter* depan.

Algernon Papadopoulos, Fatty Johnson, dan Bucky Pasteur ada di *basement*. Jimmy yang baru dating mengganggu mereka dan Fatty ingin mengetahui alasan kedatangan Jimmy serta melarang Jimmy untuk ikut bergabung dengan kelompok mereka. Lain dengan Fatty, Bucky memberi kesempatan kepada Jimmy jika dan hanya jika Jimmy bisa mengalahkan *score* tertinggi dari permainan *Consumo* yang saat itu dipegang oleh Fatty sebesar 1010 lbs. Jimmy menerima tantangan tersebut. *Arcade game* ini bisa dimainkan kapan saja dalam *game*. Jika Jimmy berhasil mendapatkan skor tertinggi mengalahkan Fatty, para anggota Nerd akan kagum dan Bucky akan memberikan Jimmy *bottle rocket launcher* dan *save point* baru terbuka untuk Jimmy yaitu Dragon’s Wing Comics Store.



Gambar 2.1 Permainan ConSumo (Sumber: [https://bullyfanon.fandom.com/wiki/Nerd\\_Challenge](https://bullyfanon.fandom.com/wiki/Nerd_Challenge))







Gambar 2.2 *Basement* Dragon’s Wing Comics Store (Sumber: [https://bully.fandom.com/wiki/Nerd\\_Challenge](https://bully.fandom.com/wiki/Nerd_Challenge))

Pada permainan ConSumo, sesuai namanya, pemain akan mengontrol seorang pegulat sumo (yang menggunakan celana ungu di gambar 2.1). Objektif dari *game* adalah mengumpulkan poin sebanyak-banyaknya dengan cara memakan makanan yang *edible*. Pemain memiliki 3 kesempatan dalam sekali permainan untuk tetap bertahan dalam permainan. Jika memakan *puffer fish* atau memakan makanan busuk hingga “*weight gauge*” habis maka akan diberikan *penalty* berupa hilangnya 1

kesempatan. Pemain dapat bergerak di 8 arah kardinal, tidak dapat bergerak melewati matras – jika pemain terpantul keluar dari *screen* akibat pegulat sumo lain maka pemain akan segera *respawn* ke tengah matras tanpa *penalty*.

Ikon dari makanan segar dan busuk, ikan buntal, serta pegulat sumo lain akan bergerak lurus menyeberangi matras. Pemain harus memakan makanan segar untuk menambah berat badan. Dengan bertambahnya berat badan, “*weight gauge*” secara bertahap terisi. Makanan busuk akan mengakibatkan “*weight gauge*” berkurang.

Setiap terisi, pemain bertambah besar, membuatnya semakin sulit menghindari makanan busuk, *puffer fish*, pegulat sumo lain. Makanan-makanan dan pegulat sumo lain juga bertambah besar. Mereka mengisi atau mengosongkan *weight gauge* badan Anda dalam jumlah yang semakin besar, semakin besar jumlahnya. Pegulat sumo lain dapat menyebabkan pemain terpantul yang bisa menyebabkan pemain terpantul ke makanan busuk atau *puffer fish* atau bahkan pesumo lain lagi yang dapat memantulkan pemain lagi. *Puffer fish* menyebabkan pemain langsung kehilangan kesempatan. Kesempatan juga bisa berkurang jika memakan makanan busuk dengan *weight gauge* yang sudah mau habis.

No	Ikon	Penjelasan
1.		Pegulat sumo yang dimainkan pemain
2.	  	Makanan-makanan segar yang dapat dimakan pemain dan menyebabkan <i>weight gauge</i> dan skor bertambah. Semakin besar ukuran makanan yang dimakan maka <i>weight gauge</i> akan bertambah banyak

3.	  	Makanan-makanan busuk yang harus dihindari. Jika tidak sengaja termakan akan mengakibatkan <i>weight gauge</i> berkurang. Semakin besar ukuran makanan busuk yang dimakan maka <i>weight gauge</i> dan skor akan berkurang banyak
4.		<i>Puffer fish</i> menyebabkan kehilangan 1 kesempatan
5.		Pesumo yang dapat menyebabkan pesumo pemain memantul

Tabel 2.1 Ikon ConSumo (Sumber: [https://www.youtube.com/watch?v=3\\_08FFU7ACI](https://www.youtube.com/watch?v=3_08FFU7ACI))





Gambar 2.3 Permainan Consumo dengan ukuran yang beragam (Sumber: [https://www.youtube.com/watch?v=3\\_08FFU7ACI](https://www.youtube.com/watch?v=3_08FFU7ACI))

### III. IMPLEMENTATIONS

Dalam pengerjaan algoritma greedy ini, penulis memiliki beberapa asumsi sebagai berikut:

1. Akan dibuat seperti bot untuk pesumo pemain.
2. Peta dapat direpresentasikan sebagai suatu matriks.
3. Pergerakan pemain dapat direpresentasikan sebagai Point (menggunakan koordinat x dan y)

```

.....F.....
.....
..B...A..
.....
..S.P.S..
.....
..A....B.
.....
.....
.....

```

Gambar 3.1 Representasi Peta dengan ukuran 9 x 9

Peta tersebut memiliki keterangan sebagai berikut:

1. P : Pegulat sumo yang digunakan oleh pemain
2. B : Makanan busuk yang harus dihindari
3. F : *Puffer fish*
4. A : Makanan segar yang harus dimakan untuk menambah poin dan berat badan

Selain dari pemain, ikon bergerak lurus secara vertical atau horizontal.

Penerapan algoritma greedy untuk penyusunan strategi pada permainan ini dapat digunakan untuk tiga fungsi seleksi. Fungsi pertama adalah *Greedy by Nearest Food*. Fungsi ini akan mempertimbangkan makanan *edible* yang terdekat dari pemain. *Greedy by Safest Point* merupakan fungsi seleksi kedua. Pada fungsi ini, pemain akan mencari poin pada peta yang paling aman dari pegulat sumo lain, makanan busuk, atau *puffer fish*. Fungsi ketiga adalah *Greedy by Weight Gauge*. Fungsi ini meninjau kondisi terkini *weight gauge*, jika dalam kondisi aman (hampir terisi penuh) maka akan terus mencari makan sedangkan jika tidak, maka akan fokus menghindari makanan busuk, *puffer fish*, atau pegulat sumo lain.

Sebagai himpunan kandidat adalah seluruh item yang sedang bergerak lurus menyeberangi matras. Himpunan solusi berisi makanan *edible* yang dimakan. Fungsi seleksi seperti yang disebutkan di atas yaitu, *Greedy by Nearest Good*, *Greedy by Safest Point*, dan *Greedy by Weight*

*Gauge*. Fungsi kelayakan adalah memeriksa apakah ikon yang ada di dekat bot adalah makanan segar atau bukan. Fungsi objektif dari permainan ini adalah mengumpulkan skor sebesar-besarnya.

#### A. Greedy by Nearest Food

Fungsi seleksi pertama adalah *Greedy by Nearest Food*. Seleksi langkah yang akan dilalui berdasarkan makanan *edible* terdekat yang bisa ditempuh pesumo. Fungsi ini sangat berguna ketika kondisi di sekitar pemain benar-benar tidak ada rintangan.

Langkah penentuan keputusan optimum lokal dari strategi ini sebagai berikut. Ketika bermain dan di sekitar pemain tidak ada rintangan maka bot pesumo akan langsung memakan makan tersebut. Dari titik terakhir dia memakan makanan tersebut, bot pesumo akan membuat keputusan lagi untuk mencari makanan terdekat. Hal ini terus dilakukan. Kelemahan dari strategi ini jelas terlihat dari ketidakmampuannya dalam meninjau bahaya di sekitar bot. Strategi ini tidak terlalu kompleks karena hanya meninjau *Point* disekitar bot yang terdapat makanan *edible*.

procedure greedybynearestfood(P : Current Point)

Algoritma

{Meninjau Point yang ada di atas, bawah, kiri, kanan pesumo dengan representasi point}

POINT P1 ← Point\_PlusDelta (P,1,0)

POINT P2 ← Point\_PlusDelta (P,-1,0)

POINT P3 ← Point\_PlusDelta (P,0,-1)

POINT P4 ← Point\_PlusDelta (P,0,1)

if IsFoodExist(P1) then

    MoveTo(P1)

else if IsFoodExist(P2) then

    MoveTo(P2)

else if IsFoodExist(P3) then

    MoveTo(P3)

else if IsFoodExist(P4) then

    MoveTo(P4)

endif

#### B. Greedy by Safest Point

Fungsi seleksi kedua adalah *Greedy by Safest Point*. Seleksi langkah yang akan dilalui pesumo berdasarkan *Point* teraman.

Langkah penentuan optimum lokal dari strategi ini sebagai berikut. Ketika bermain dan di dekat bot ada ancaman, maka bot akan bergerak ke atas, bawah, depan,

atau belakang ke titik lain untuk menjauhi ancaman tersebut. Ketika sampai di titik yang aman itu, maka bot akan kembali meninjau daerah sekitar bot, jika ada ancaman lagi maka bot akan ke atas, bawah, depan, atau belakang ke titik lain menjauhi ancaman tersebut. Kelemahan dari strategi ini adalah lebih fokus menyelamatkan diri dibandingkan memakan makanan sehingga selesainya *game* akan lama dan karena meninjau titik teraman dari titik asal, tidak menjamin bahwa titik tujuan saat ia sampai di titik tersebut aman. Strategi ini juga tidak terlalu kompleks karena hanya meninjau titik berbahaya yang ada di dekat titik asal dan berjalan ke titik yang lebih aman.

```
procedure greedysafestpoint(P : CurrentPoint)
```

Algoritma

{Meninjau Point yang ada di atas, bawah, kiri, kanan pesumo dengan representasi point}

```
POINT P1 ← Point_PlusDelta (P,1,0)
```

```
POINT P2 ← Point_PlusDelta (P,-1,0)
```

```
POINT P3 ← Point_PlusDelta (P,0,-1)
```

```
POINT P4 ← Point_PlusDelta (P,0,1)
```

```
if isSafe(P1) then
```

```
    MoveTo(P1)
```

```
else if isSafe (P2) then
```

```
    MoveTo(P2)
```

```
else if isSafe (P3) then
```

```
    MoveTo(P3)
```

```
else if isSafe (P4) then
```

```
    MoveTo(P4)
```

```
endif
```

```
procedure greedysafestpoint(P : CurrentPoint)
```

Algoritma

{Meninjau Point yang ada di atas, bawah, kiri, kanan pesumo dengan representasi point}

```
POINT P1 ← Point_PlusDelta (P,1,0)
```

```
POINT P2 ← Point_PlusDelta (P,-1,0)
```

```
POINT P3 ← Point_PlusDelta (P,0,-1)
```

```
POINT P4 ← Point_PlusDelta (P,0,1)
```

```
While weightgauge > 25 do
```

```
    if IsFoodExist(P1) then
```

```
        MoveTo(P1)
```

```
    else if IsFoodExist(P2) then
```

```
        MoveTo(P2)
```

```
    else if IsFoodExist(P3) then
```

```
        MoveTo(P3)
```

```
    else if IsFoodExist(P4) then
```

```
        MoveTo(P4)
```

```
    endif
```

```
endwhile
```

```
if isSafe(P1) then
```

```
    MoveTo(P1)
```

```
else if isSafe (P2) then
```

```
    MoveTo(P2)
```

```
else if isSafe (P3) then
```

```
    MoveTo(P3)
```

```
else if isSafe (P4) then
```

```
    MoveTo(P4)
```

```
endif
```

### C. Greedy by Weight Gauge

Fungsi seleksi ketiga adalah *Greedy by Weight Gauge*. *Weight Gauge* yang merupakan bar yang bisa terisi warna merah (gambar 2.3), Jika pesumo pemain terkena makanan busuk saat bar berwarna merah sudah sedikit maka kesempatan akan berkurang satu. Oleh karena itu, jika, bar sudah mau habis maka bot akan bermain lebih 'aman' dengan menghindari ancaman yang terdekat. Jika bar merah masih terisi banyak maka bot akan mencari makanan terdekat. Dibandingkan dari strategi sebelumnya, strategi ini lebih kompleks karena selain harus meninjau *weight gauge* terkini setelah itu melakukan Tindakan berdasarkan informasi *weight gauge* tersebut. Bisa dibilang ini merupakan gabungan dari strategi sebelumnya tetapi saat *weight gauge* sisa sedikit maka bot hanya akan fokus menghindar tidak fokus mencari makan.

## IV. KESIMPULAN

Ketiga algoritma tersebut memiliki kekurangan dan kelebihan masing-masing. Strategi ketiga memang merupakan gabungan dari dua algoritma sebelumnya namun memiliki keterbatasan seperti dari kompleksitas dan ketidakmampuannya dalam meninjau keamanannya saat bergerak mencari makan.

## V. UCAPAN TERIMA KASIH

Puji Syukur saya panjatkan kepada Tuhan Yang Maha Esa karena atas rahmat dan berkat-Nya, makalah ini dapat diselesaikan tepat waktu. Penulis juga ingin mengucapkan terima kasih kepada Bapak Ir. Rila Mandala, M.Eng, selaku dosen K1 mata kuliah IF2211 Strategi Algoritma tahun akademik 2020/2021. Penulis juga mau mengucapkan terima kasih kepada Bapak Rinaldi Munir yang telah menyediakan *website* yang dapat mahasiswa gunakan sebagai referensi belajar. Penulis meminta maaf jika ada kesalahan kata dalam makalah ini. Semoga makalah ini dapat bermanfaat.

### VIDEO LINK AT YOUTUBE

<https://youtu.be/HyJI7gkJdg>

### REFERENCES

- [1] <https://www.rockstargames.com/games/bully>, tanggal akses terakhir: 11 Mei 2021, pukul 07.06
- [2] <https://bully.fandom.com/wiki/Bully>, tanggal akses terakhir: 11 Mei 2021, pukul 07.10
- [3] <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/stima20-21.htm>, tanggal akses terakhir: 11 Mei 2021, pukul 07.08

- [4] <https://www.freecodecamp.org/news/what-is-a-greedy-algorithm/>, tanggal akses terakhir: 11 Mei 2021, pukul 17.46
- [5] <https://bully.fandom.com/wiki/ConSumo>, tanggal akses terakhir: 11 Mei 2021, pukul 18.42

### 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.

Bekasi, 11 Mei 2021



Jose Galbraith Hasintongan 13519022