Penerapan Strategi Greedy dan Knapsack dalam Transfer Pemain pada Game Master League PES

Marsa Thoriq Ahmada
Program Studi Teknik Informatika
Institut Teknologi Bandung
Bandung, Indonesia
13517071@std.stei.itb.ac.id

***Abstrak*— Pada makalah ini membahas tentang algoritma greedy dan knapsack yang bisa diimplementasikan dalam dunia game khususnya game sepak bola. Dalam sepak bola terdapat transfer pemain yang merupakan strategi klub untuk membeli pemain yang merupakan anggota klub lain. Setiap klub sepak bola selalu menginginkan untuk bisa membentuk tim yang berisikan pemain pemain yang handal tetapi biasanya terdapat suatu hambatan dalam membeli pemain yaitu uang yang dimiliki suatu klub. Maka dari itu untuk permasalahan ini akan berkaitan dengan materi strategi algortima greedy dan akan dikombinasikan dengan strategi algoritma knapsack. Sehingga akan bisa menyarankan suatu klub untuk membeli beberapa pemain sesuai dengan kondisi keuangan klub pada saat itu.**

Keywords—Klub, Game, Greedy, Knapsack, PES, Master League, Transfer Pemain

I. PENDAHULUAN

Game merupakan sarana hiburan bagi banyak orang.. Hal itu dikarenakan game bisa menjadi penghibur bagi orang orang yang lelah dalam menjalankan rutinitas yang cukup melelahkan. Game sepak bola memiliki daya tarik yang kuat khususnya untuk penggemar sepak bola. Hal ini dikarenakan sepak bola merupakan olahraga yang paling diminati oleh semua kalangan dan hampir menjadi olahraga terpopuler di seluruh negara didunia. Beberapa game sepak bola sudah sangat popular dikalangan *gamer* contohnya seperti PES, dan FIFA.

Pro Evolution Soccer (PES) merupakan salah satu permainan sepak bola yang dapat dimainkan pada *device* PC, XBOX360 dan PS4. Game ini diluncurkan awalnya di jepang oleh perusahaan game Konami pada tahun 2001[1]. Pada game ini selalu dilakukan update setiap tahunnya. Pada update biasanya dilakukan beberapa perubahan seperti gameplay, atribut klub seperti stadion, pemain, logo klub, baju klub, dan lain lain. Game Pro Evolution Soccer ini memiliki banyak peminat khususnya dikawasan asia. Game ini merupakan game sepak bola paling popular di kawasan asia.

Gambar 1, https://www.google.com/search?q=gambar+pes+gameplay&safe=strict&rlz=1C1GGRV_enID774ID774&source=lnms&tbm=isch&sa=X&ved=0ahUKEwic0NnIre3hAhWLfCsKHZlHAskQ_AUIDygC#imgrc=0PDcPC1wFIw0MM:

Game Pro Evolution Soccer ini memiliki banyak sekali fitur pilihan untuk memainkan suatu permainan. Jenis jenis fitur untuk memainkan pertandingan sepak bola yaiu Play Match, Cup, League, Become a Legend, Master League. dan Fitur fitur tambahan berupa Training,, Edit, Replay. Untuk fitur memainkan pertandingan berbeda beda tergantung akan jenis fitur yang dipilih. Perbedaan dari setiap fitur fiturnya ialah sistem permainan yang berbeda beda contoh seperti cup merupakan diberlakukan sistem gugur sehingga setiap klub tidak bisa lolos ke babak selanjutnya jika kalah pada suatu pertandingan hal ini berbeda dengan fitur pertandingan

league yang setiap klub akan berhadapan satu sama lain di kedua kandang tim tersebut dan untuk sistem yang diberlakukan ialah siapa yang mengumpulkan paling banyak poin akan menjadi juara dalam league tersebut. Untuk sistem penilaian poin adalah sebagai berikut Menang : 3, Seri : 1, Kalah : 0.

Pada fitur Master League pemain game akan dihadapkan sebagai manager dari suatu klub yang bisa mengatur banyak hal seperti transfer pemain, pola latihan pemain, formasi pemain, pendapatan klub, dan, akademi pemain muda. Selain itu klub yang pemain game pilih dapat melakoni beberapa jenis pertandingan yaitu cup maupun league secara bersamaan. Perbedaan yang paling terlihat dalam Master League dengan fitur yang lainya ialah pada Master League bisa dilakukan transfer pemain baik merupakan jual,beli, pinjam maupun meminjamkan pemain ke klub lain.

Untuk setiap pemain memiliki overal rating yang merupakan nilai dari pemain tersebut. Sehingga kualitas pemain bisa dilihat dari overal rating tersebut. Tapi tingginya overal rating dibarengi dengan tingginya harga jual dari pemain tersebut selain itu hal yang memengaruhi tinggi dan rendahnya niali pemain adalah umur dari pemain itu sendiri. Hal tersebut dikarenakan pemain usia muda memiliki potensi tinggi untuk berkembang dengan menambah kemampuan. Sehingga overal rating bisa bertambah. Sebaliknya untuk pemain usia yang cukup tua overal rating malah akan berkurang. Selain itu pemain yang sudah tua (usia 35-40 tahun dalam sepak bola umur tersebut sudah termasuk usia tua) akan melakukan pensiun sehingga akan putus kontrak dengan klub. Pada permasalahan pengembangan klub untuk membuat klub lebih berkualitas dapat diatasi dengan transfer pemain. Sehingga hal ini bisa diatasi dengan greedy dan algoritma knapsack yang sangat cocok dengan permasalahan kali ini

II. DASAR TEORI

A. Algoritma Greedy

Algoritma greedy merupakan salah satu metode penyelesaian masalah yang cukup popular dalam persoalan optimasi.Yang dimaksud persoalan optimasi sendiri ialah pencarian solusi optimum dalam suatu permasalahan,

ada dua macam optimasi yaitu maksimasi dan minimasi.

a. Maksimasi

   Mengambil suatu langkah yang menyebabkan nilai yang diinginkan maksimum.

b. Minimasi

   Mengambil suatu langkah yang menyebabkan nilai yang diinginkan minimal.

Prinsip greedy adalah mengambil suatu hal permasalahan yang kecil dengan optimum dan berharap untuk mendapat niali yang optimal. Greedy membentuk solusi dengan cara langkah per langkah (*step by step*). Dengan prinsip tersebut algorima greedy selalu mengambil langkah optimal dalam semua langkah.

Algoritma *greedy* disusun oleh beberapa elemen berikut:

a. Himpunan kandidat, C.

   Himpunan kandidat berisi dengan semua elemen kandidat yang masih mungkin untuk membentuk solusi.

b. Himpunan solusi, S.

   Himpunan solusi berisi dengan beberapa elemen saja yang merupakan nilai optimum dalam suatu permasalahan kecil (Optimum lokal).

c. Fungsi seleksi

   Pemilihan kandidat yang paling memunginkan dalam mencapai solusi akhir optimal (Optimum Global). ,Karena berbentuk solusi berbentuk himpunan, solusi yang sudah dipilih tidak akan dipilih lagi pada semua langkah berikutnya.

d. Fungsi kelayakan

   Memeriksa apakah kandidat solusi yang telah terpilih sebelumnya dapat memberikan solusi yang layak , yaitu semua kandidat solusi yang sudah terbentukbersama sama tidak melanggar *constraint* yang ada pada permasalahan. Jadi kandidat yang tidak layak akan dibuang dari solusi dan tidak pernah dipertimbangkan lagi.

e. Fungsi objektif

   Fungsi yang memaksimalkan atau meminimumkan nilai solusi contohnya: meminimumkan biaya perjalanan, memaksimalkan keuntugan dalam penjualan barang.

Pada permasalahan yang diselesaikan oleh algoritma greedy tidak selalu menghasilkam jawaban paling optimal dikarenakan yang diambil secara optimal hanyalah sub permasalahan / suatu langkah saja (Optimum lokal). Jadi penyususnan elemen elemen dalam penyusunan algoritma greedy sangat menentukan apakah permasalaahan itu cocok atau tidak dalam menyelesaikan permasalahan tersebut.

```
function greedy(input C: himpunan_kandidat)→ himpunan_kandidat
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greed
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
Deklarasi
    x : kandidat
    S : himpunan_kandidat

Algoritma:
    S ← {}   { inisialisasi S dengan kosong }
    while (not SOLUSI(S)) and (C ≠ {} ) do
       x ← SELEKSI(C)          { pilih sebuah kandidat dari C}
       C ← C - {x}             { elemen himpunan kandidat berkurang sa
       if LAYAK(S ∪ {x}) then
          S ← S ∪ {x}
       endif
    endwhile
    {SOLUSI(S) or C = {} }

    if SOLUSI(S) then
       return S
    else
       write('tidak ada solusi')
    endif
```

Gambar 2, Slide Kuliah Rinaaldi Munir

B. Knapsack 0/1

Knapsack 0/1 merupakan jenis knapsack yang paling popular dimana dalam knapsack 0/1 , merupakan persoalan dimana diberikan n objek dan dengan kapsitas total maksimal k, dan setiap objek memiliki bobot (W_i) dan keuntungan (P_i), dalam kodisi diatas diharuskan mendapat keuntungan yang paling maksimal untuk ukuran kapasitas total k.

Persoalan knapsack bisa diselesaikan melaui beberapa strategi algoritma yaitu:

a. Algoritma Brute Force

Algoritma *brute force* merupakan salah satu pendekatan dalam menyelesaikan masalah dengan cara yang sederhana. Algoritma *brute force* biasanya disebut juga dengan algoritma *exhautive search* jadi pertama tama algoritma ini mencari semua kemungkinan himpunan solusi yang bisa dibentuk dan setelah itu dilakukan pengecekan apakah setiap himpunan solusi tersebut memenuhi kriteria solusi dari permasalahan. Algoritma brute force sering dipakai dikarenakan sifatnya yang sederhana dan mudah diimplementasikan.

Pada permasalahan knapsack 0/1 langkah brute force / exhautive search yang dilakuan adalah sebagai berikut:

1. Enumerasi: mencari semua kemungkinan himpuan bagian (subset) dari keseluruhan objek yang mungkin dalammembuat solusi.

2. Evaluasi: Hitung total bobot apakah masih muat dan tidak melanggar *constraint* jika tidak maka menjadi suatu kandidat solusi dengan menyimpan nilai total keuntungan

3. Pemilihan kandidat solusi: ambil total keuntungan paling banyak dari suatu kandidat solusi.

Terlihat bahwa algoritma brute force tidak melakukan langkah optimasi sama sekali terhadap enumerasi dari kemungkinan solusi. Bisa saja jika barang yang sudah diambil diawal sudah melanggar *constraint* tetap diproses hingga selesai. Sehingga solusi yang diambil disini dijamin merupakan solusi optimal . Selain itu hal ini menimbulkan suatu permasalahan waktu eksekusi yang sangat lama. Secara kompeleksitas waktu eksekusi algoritma ini adalah $O(n*2^n)$. Pada proses enumerasi memerlukan kompleksitas waktu $O(2^n)$ dikarenakan mencari semua subset kemunginannya. Dan di proses evaluasi memerlukan kompleksitas O(n) dikarenakan secara tranversal mengecek constraint dengan n merupakan jumlah barang. Untuk kompleksitas memori bisa dilakukan dengan kompleksitas O(n) dikarenakan bisa dengan secara bersamaan mengevaluasi dari enumerasi yang dibentuk.

b. Algoritma *Dynamic Programming*

Dynamic Programming merupakan salah satu algoritma terbaik dalam menyelesaikan permasalahan. Jadi pada algoritma *Dynamic Programming* adalah sebuah metode untuk memecahkan masalah dengan cara menguraikan solusi menjadi tahapan tahapan yang saling berhubungan. Hasil solusi pada suatu tahap merupakan hasil perhitungan dari solusi pada tahap sebelumnya. Algoritma *Dynamic Programing* merupakan salah satu algoritma terbaik dikarenakan untuk kompleksitas waktu sangat mangkus dibandingkan algoritma lain dengan permasalahan yang sama. Hal ini dikarenakan pada *dynamic programing* dilakukan hasil dari suatu solusi tidak perlu dihitung kembali, jika ada pada langkah yang sama seperti sebelumnya algoritma ini hanya memanggil dari memori tidak perlu melakukan perhitungan secara berulang ulang.

Dalam *dynamic programming* memiliki dua pendekatan , yaitu:

1. Pemrograman dinamis maju (up-down)
   Pemrograman dinamis maju bergerak mulai dari satu tahap lalu maju menjadi 2 tahap , dan akan terus bertambah sampai n tahap seuai peubah yang di perlukan. Untuk optimalisasi solusi, ongkos dari suatu tahap k+1 merupakan hasil tambah dari ongkos

pada tahap k ditambah ongkos dari tahap k ke k+1. Pada Pemrogrman dinamis maju dilakukan secara rekursif.

```
1: function SOLVE(x)
2:     if (x = 0) then
3:         return 0
4:     end if
5:     if computed[x] then
6:         return memo[x]
7:     end if

8:     best ← ∞
9:     for k ← 1, M do
10:        if (a_k ≤ x) then
11:            best ← min(best, SOLVE(x − a_k))
12:        end if
13:    end for
14:    computed[x] ← true
15:    memo[x] ← best
16:    return best
17: end function
```

Gambar 3, William Ghozali,Buku Pemrograman kompetitif dasar

2. Pemrograman dinamis mndur (bottom-up)
Pemrograman dinamis mundur bergerak mulai dari n tahap lalu maju menjadi n-1 tahap , dan akan terus berkurang sampai 1 tahap seuai peubah yang di perlukan. Untuk optimalisasi solusi, ongkos dari tahap k adalah ongkos pada tahap k+1 ditambah dengan ongkos dari tahap k+1 ke tahap k. Pada Pemrogrman dinamis maju dilakukan secara iteratif dan disimpan di array.

```
1: function SOLVE()
2:     f[0] ← 0

3:     for x ← 1, N do
4:         best ← ∞
5:         for k ← 1, M do
6:             if (a_k ≤ x) then
7:                 best ← min(best, f[x − a_k] + 1)
8:             end if
9:         end for
10:        f[x] ← best
11:    end for

12:    return f[N]
13: end function
```

Gambar 4, William Ghozali,Buku Pemrograman kompetitif dasar

Dari kedua pendekatan tersebut memiliki kompleksitas yang sama yaitu O(NM) dimana N merupakan jumlah barang dan M merupakan kapasitas maksimal. Tapi untuk semua kasus *dynamic programing* tidak menjamin untuk bisa diselesaikan dengan kedua pendekatan diatas melainkan hanya salah satu saja.

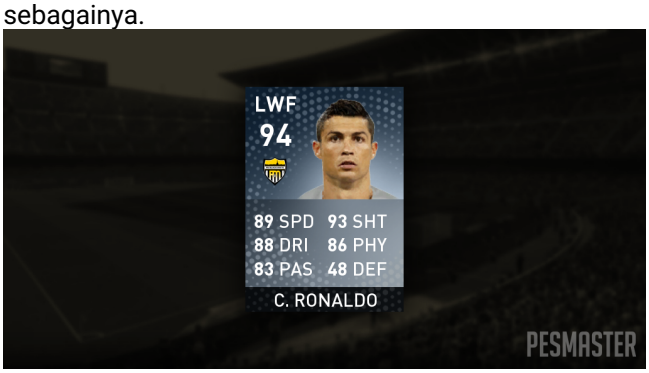
Dalam permasalahan knapsack 0/1 untuk n buah barang , maka tiap pemiliha barang merupakan sebuah tahap ke i menandakan apakah barang ke i diambil atau tidak , untuk setiap tahapan akan dicari suatu nilai yang optimumdengan membandingkan apakah barang tersebut diambil (jika masih sesuai *constraint*) atau tidak.

```
turns the maximum value that can be put in a knapsack of capacity W
napSack(int W, int wt[], int val[], int n)

Base Case
(n == 0 || W == 0)
 return 0;

If weight of the nth item is more than Knapsack capacity W, then
this item cannot be included in the optimal solution
(wt[n-1] > W)
 return knapSack(W, wt, val, n-1);

Return the maximum of two cases:
(1) nth item included
(2) not included
se return max( val[n-1] + knapSack(W-wt[n-1], wt, val, n-1),
               knapSack(W, wt, val, n-1)
             );
```

Gambar 5, https://www.geeksforgeeks.org/0-1-knapsack-problem-dp-10/

III. Implementasi Strategi pada Game

Untuk membentuk klub yang bagus bisa dilihat secara kuantitatif dengan rata rata overal rating dari setiap pemain. Jika *Overal rating* tinggi otomatis atribut dari pemain tersebut juga tinggi Dan setiap pelatih biasanya hanya memerlukan beberapa pemain tambahan saja dan juga memiliki kriteria pemilihan baik dari segi umur, skill, posisi yang ditempati dan lain sebagainya.



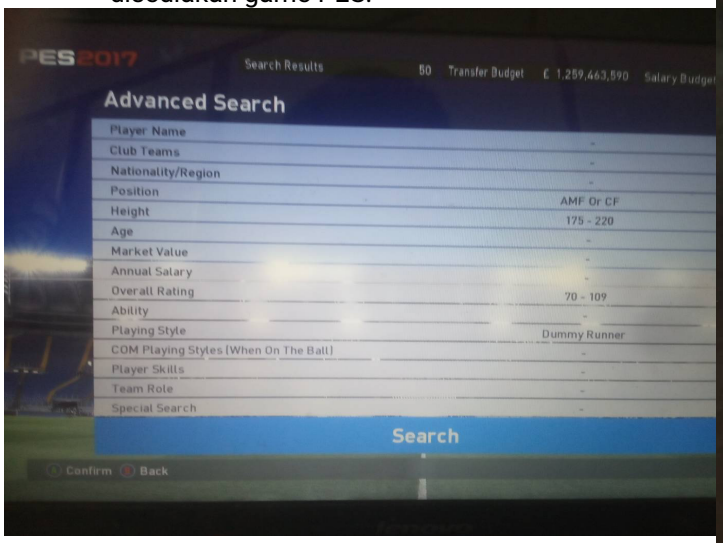
Gambar 6,https://www.pesmaster.com/c-ronaldo/pes-2019/player/4522/

Untuk kasus ini pembelian pemai yang akan disarankan adalah pembelian pemain yang akan meningkatkan rata rata overall rating dari klub dan masih sesuai dengan *budget* / anggaran belanja klub.

Untuk setiap pemain selain mempunyai overall rating juga mempunyai umur , pemain muda mempunyai potensial overall rating yang akan meningkat untuk usia berhentinya overall rating meningkat dalam PES biasanya ialah pada usia 28 tahun sehingga pada usia 28 tahun keatas overall rating dari pemain berkurang dan pada usia kurang dari 28 tahun overall rating bertambah. Untuk penambahan dan pengurangan sendiri tergantung akan seberapa sering pelatih memainkan pemain tersebut. Sehingga dalam algoritma nanti disediakan pilihan preferensi pelatih lebih sering memainkan pemain muda atau senior.

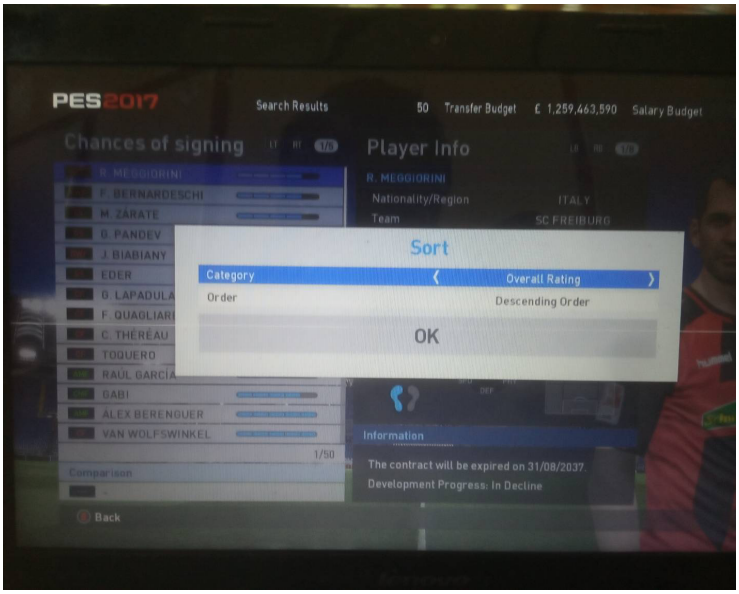
Untuk langkah greedy yang dilakukan ialah sebagai berikut:

1. Filter semua pemain yang akan dimasukan dalam pertimbangan di beli atau tidak. Filter dilkukan melalui pilihan advanced search yang sudah disediakan game PES.



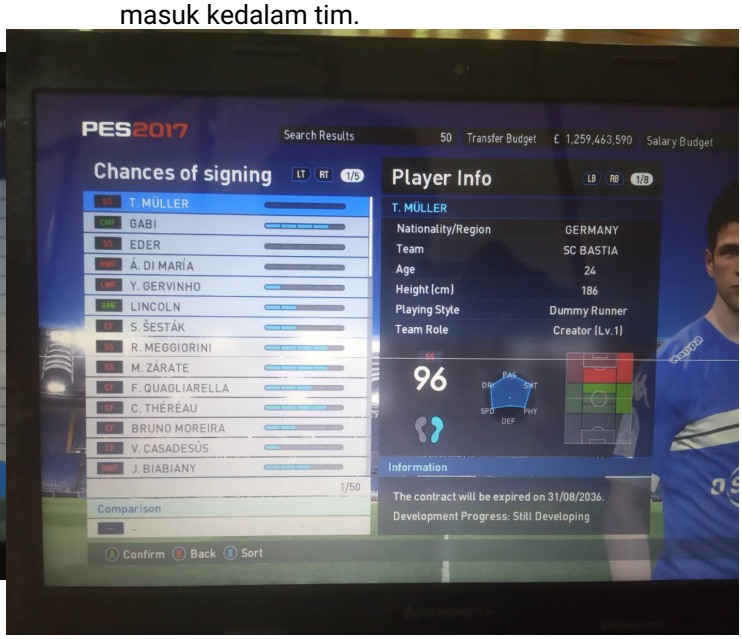
Gambar 7, dokumentasi sediri

2. Urutkan pemain berdasarkan overall rating. Untuk pengurutan sendiri juga sudah ada fitur sort dari game PES.



Gambar 8, dokumentasi sediri

3. Pilih n pemain yang masuk pertimbangan untuk masuk kedalam tim.



Gambar 9, dokumentasi sediri

Setelah terpilih n pemain pemain dimasukan kedalam knapsack 0/1 yang dengan langka adalah sebagai berikut:

1. Masukan banyaknya pemain ,anggaran , dan juga preferensi pemain
2. Sebanyak isi nama pemain, overall rating,, harga dan umur pemain. Rumus yang diberlakukan untuk bobot ialah harga dan untuk keuntunganya adalah (28-umur_pemain)*x + overall_rating. X berisi 2 jika preferensi pemain merupakan pemain muda dan X bernilai 1 jika preferensi bukan pemain muda. Karena pelatih yang

mempunyai preferensi pemai muda akan sering memainkan pemain muda sehingga rating dari pemain muda akan bertambah lebih cepat sebaliknya untuk pemain senior akan lebih jarang diberi kesempatan sehingga akan berkurang dengan lebih cepat

3. Masukan ke fungsi knapsack 0/1
4. Keluar hasil pemain yang disarankan

Berikut source codenya

```
#include <bits/stdc++.h>
using namespace std;
long tc,n,k,q,tot,x,y;
bool pr;
bool select[1005];
long bl[1005],dp[1005][1005],val[1005];
string saved[1005];
long f(long a, long b){
	cout<<val[a]<<bl[a]<<endl;
	long save;
	if(a<=0){
		return 0;
	}
	if (dp[a][b]!=-1){
		return dp[a][b];
	}
	save=f(a-1,b);
	if (b>=bl[a]){
		if (save<f(a-1,b-bl[a])+val[a]){
			select[a]=1;
			save=f(a-1,b-bl[a])+val[a];
			cout<<"cek"<<endl;
		}else{
			select[a]=0;
		}
	}
	return dp[a][b]=save;
}
int main(){
	cout<<"Budget dalam M : ";
	cin>>k;
	cout<<"Banyak pemain : ";
	cin>>n;
	cout<<"Preferensi pemain muda : 1 jika ya 0 jika tidak :";
	cin>>pr;
	for(int j=1;j<=n;j++){
		cout<<"Nama Pemain : ";
cin>>saved[j];
		cout<<"Umur : "; cin>>x;
		cout<<"Overall : "; cin>>y;
		cout<<"Harga dalam M : "; cin>>bl[j];
		if (pr){
			val[j]=(28)*2 + y;
		}else{
			val[j]=(28) + y;
		}
	}
	cout<<n<<" "<<k<<endl;
	tot=f(n,k);
	memset(dp,-1,sizeof dp);
	for(int i=1;i<=n;i++){
		if (select[i]){
			cout<<saved[i]<<endl;
		}
	}
	cout<<tot<<endl;
	return 0;
}
```

KESIMPULAN

Strategi algoritma greedy dan knapsack sangat cocok untuk digabungkan sebagai strategi dalam bermaiin game. Bahkan kedua algoritma tersebut juga bisa digunakan dalam penentuan pembelian pemain dalam suatu klub pada master league PES.

Selain dibuat stategi dalam permainan greedy dan knapsack juga bisa digunakan dalam kehidupan sehari hari. Banyak implementasi seperti pengambilan barang saat belanja dan lain lain

SARAN

Untuk pengembang lain strategi ini bisa menjadi lebih baik jika memepertimbangkan lebih banyak lagi aspek seperti kebutuhan pemain , skill yang dimiliki pemain dan lain sebagainya agar pembelian pemain lebih akurat untuk membuat tim yang bagus.

REFERENSI

[1] https://www.konami.com/wepes/2019/eu/en/

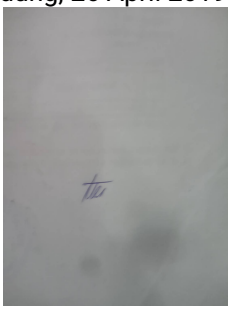
[2] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Algoritma-Greedy-(2018).pdf

[3] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2017-2018/Program-Dinamis-(2018).pdf

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, 26 April 2019



Marsa Thoriq Ahmada