Penjelasan Soal Seleksi Asisten Lab

Anyelyra Kantata

Universitas Indonesia

21 Agustus 2024

Daftar Isi

Tujuan Program

Terdapat grup pahlawan bernama **High Cloud Quintet**, yang terdiri dari 4 pahlawan, yaitu **Jingliu**,**Jing Yuan**, **Yingxin**, dan **Dan Heng**. Dengan kekuatan keempat karakter ini secara berturut-urut, yaitu "Destruction". "Erudition", "Destruction", dan "The Hunt".

Tujuan:

- Menghitung probabilitas "High Cloud Quintet" ditargetkan
- Melakukan simulasi penyerangan monster terhadap keempat karakter

Saat pertempuran, Baiheng memberi mukjizat kepada anggota High Cloud Quintet (selain Jing Yuan) dengan rincian:

- Jingliu: 200%
- Yingxing: 500%
- Dan Heng: 300%

Menghitung Probabilitas (Aggro)

$$A_k = BA \times (1 + M)$$

$$P = \frac{A_k}{\Sigma A_p}$$

Keterangan:
A_k = Aggro dari karakter yang dihitung
BA (Base Aggro) = Aggro standar dari kekuatan karakter
M = Mukjizat
P = Probabilitas karakter tersebut ditargetkan
A_p= Sumasi Aggro 1 party

Menghitung Probabilitas (Aggro)

Base Aggro (berdasarkan jenis kekuatan):

- The Hunt: 3
- Erudition: 3
- Harmony: 4
- Nihility: 4
- Abundance: 4
- Destruction: 5
- Presevation: 6

Input & Output

Input:

1. Nama karakter
2. Banyak mukjizat

Output:

1. Probabilitas ditargetkan
2. Simulasi penyerangan monster

Definisikan Dictionary

Dibuat dua dictionary, yaitu **karakter** dan **base_aggro**.

karakter menyimpan nama karakter dan kekuatannya, lalu **base_aggro** menyimpan karakter dan besaran BA (Base Aggro) masing-masing karakter.

Sehingga, ketika dimasukkan nama karakter, program mengetahui besaran Base Aggro.

```
karakter = {
    "jingliu": "destruction",
    "jing yuan": "erudition",
    "yingxing": "destruction",
    "dan heng": "the hunt"
}

base_aggro = {
    "the hunt": 3,
    "erudition": 3,
    "harmony": 4,
    "nihility": 4,
    "abundance": 4,
    "destruction": 5,
    "preservation": 6
}
```

Langkah Penyimpanan Input

1. Siapkan dictionary yang akan menyimpan nama dan besaran mukjizat, dinamakan **mukijzat_dict**. Lalu, disiapkan juga list yang akan menyimpan nama keempat karakter yang diinput, dinamakan **list_nama**.
2. Terima input **nama**, lalu masukkan dalam list **list_nama**.
3. Terima input **mukjizat**, lalu masukkan dalam dictionary **mukijzat_dict**.
4. Untuk simulasi penyerangan monster, maka akan diminta untuk memasukkan banyaknya simulasi dalam input **simulasi**.

Probabilitas Ditargetkan

Fungsi **def cari_aggro_karakter(nama)**. Fungsi ini akan mencari besar besar *BA* berdasarkan kekuatan, yang telah didapatkan dari nama karakter. Lalu, memasukkannya pada formula yang telah diberikan.

```
def cari_aggro_karakter(nama):
    kekuatan = karakter.get(nama)
    base = base_aggro[kekuatan]
    mukjizat = mukjizat_dict.get(nama)
    aggro_karakter = base * (1 + mukjizat)
    return aggro_karakter
```

Fungsi **def prob_targeted(nama)**. Fungsi ini akan mencari mencari probabilitas suatu karakter ditargetkan.

```
def prob_targeted(nama):
    total_aggro = sum(cari_aggro_karakter(k) for k
    in list_nama)
    if total_aggro == 0:
    return 0
    return cari_aggro_karakter(nama) / total_aggro
```

Simulasi Serangan Monster

Simulasi penyerangan monster yang juga dilakukan dengan **def simulate_attack**.

1. Ketika fungsi berjalan, maka serangan akan disebar sesuai dengan probabilitas yang dimiliki oleh setiap karakter berdasarkan nama yang diambil dari list.
2. Lalu, hitung sisa serangan yang tersedia dengan mengurangi jumlah simulasi serangan dengan penjumlahan dari seluruh serangan yang telah dibagikan.
3. Jika masih ada serangan, maka serangan akan dibagikan secara random, tetapi dengan weight. Weight ini diperlukan karena pembagian acak perlu dibagikan secara proporsional sesuai probabilitas yang dimiliki oleh karakter.

```
def simulate_attack():
    attack_counts = {k: 0 for k in list_nama}

    for k in list_nama:
        attack_counts[k] = int(prob_targeted(k) * simulasi)

    allocated_attacks = sum(attack_counts.values())
    remaining_attacks = simulasi - allocated_attacks

    if remaining_attacks > 0:
        for _ in range(remaining_attacks):
            selected = random.choices(list_nama,
            weights=[prob_targeted(k) for k in list_nama],
            k=1)[0]
            attack_counts[selected] += 1

return attack_counts
```

Hasil Program

Terakhir, probabilitas ditargetkannya suatu karakter dan simulasi serangan akan ditampilkan dengan memanggil fungsi-fungsi yang telah dibuat.

```
for nama in list_nama:
    final_probs = prob_targeted(nama)
    print('The chance that {} gets targeted
    is: {:.3%}'.format(nama, final_probs))


attack_results = simulate_attack()
for nama, count in attack_results.items():
    print('The number of attacks on {}
    is: {}'.format(nama, count))
```

Input & Tujuan

Input:

- Satu list $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ yang berisi 6 bilangan bulat non negative antara 0 sampai 9
- Satu bialngan bulat positif n

Tujuan: Membuat fungsi bernama **NGAB** yang menerima list dan bilangan bulat positif. Dimana fungsi ini akan melalukan sebuah iterasi (dengan aturan) terhadap list, membuatnya menjadi tiga titik, dan membuat segitiga (apabila memungkinkan) dari ketiga titik tersebut.

Pada soal telah diberikan arahan pembuatan fungsi, maka seluruh langkah-langkah pembuatan fungsi akan sesuai dengan arahan pada soal.

Langkah Pembuatan Fungsi

Input list $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ yang diberikan akan dilakukan iterasi sebanyak n kali (yang setiap hasil iterasi akan ditampilkan). Misal list $\{y_1, y_2, y_3, y_4, y_5, y_6\}$ adalah list hasil 1 kali iterasi, berikut ini adalah aturan iterasinya:

$$y_1 = (x_1 + x_2) \mod 10$$

$$y_2 = (x_2 + x_3) \mod 10$$

$$y_3 = (x_3 + x_4) \mod 10$$

$$y_4 = (x_4 + x_5) \mod 10$$

$$y_5 = (x_5 + x_6) \mod 10$$

$$y_6 = (x_6 + x_1) \mod 10$$

```
NGAB[list_, n_] := Module[{currentList = list, nextList, i, points},

  Print["Hasil Iterasi:"]
    Print[currentList]
    For[i = 1, i <= n, i++,
     nextList = Mod[{
        currentList[[1]] + currentList[[2]],
        currentList[[2]] + currentList[[3]],
        currentList[[3]] + currentList[[4]],
        currentList[[4]] + currentList[[5]],
        currentList[[5]] + currentList[[6]],
        currentList[[6]] + currentList[[1]]
        }, 10];
     Print[nextList];
     currentList = nextList;
     ];
```

Setelah iterasi selesai, maka akan dibuat tiga titik yang diambil dari setiap 2 angka berurutan dari list angka terakhir.
Misal list terakhir adalah: $\{1, 2, 3, 4, 5, 6\}$
Maka, titik yang terbentuk adalah: {1,2}, {3,4}, {5,6}

```
titik1 = {currentList[[1]], currentList[[2]]};
titik2 = {currentList[[3]], currentList[[4]]};
titik3 = {currentList[[5]], currentList[[6]]};
titikList = {titik1, titik2, titik3};
```

Karena tujuan kita adalah untuk membuat sebuah segitiga, maka kita perlu memeriksa apakah titik-titik yang telah didapatkan dapat membentuk segitiga atau tidak.

Digunakan **Triangle Inequality** untuk memeriksa apakah segitiga dengan panjang-panjang sisi a, b, c terdapat atau tidak.

Jika $a + b > c$, $a + c > b$, dan $b + c > a$ terpenuhi. Maka, segitiga dengan panjang sisi tersebut ada.

Namun, karena yang akan diperiksa adalah dalam bentuk titik, maka akan dihitung dulu jarak antara dua titik.

Dengan formula untuk menghitung jarak antara dua titik, yaitu: Misal titik $A(x_1, y_1)$ dan $B(x_2, y_2)$. Maka, untuk menghitung jarak antara titik A dan B adalah: $AB = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$.

```
AB = Sqrt[(titik2[[1]] - titik1[[1]])^2 + (titik2[[2]] -
       titik1[[2]])^2];
BC = Sqrt[(titik3[[1]] - titik2[[1]])^2 + (titik3[[2]] -
       titik2[[2]])^2];
CA = Sqrt[(titik3[[1]] - titik1[[1]])^2 + (titik3[[2]] -
       titik1[[2]])^2];
isSegitiga = AB + BC > CA && AB + CA > BC && BC + CA > AB;
```

Jika **isSegitiga** tidak terpenuhi, maka fungsi akan berhenti dan memberikan pesan bahwa ketiga titik tersebut tidak dapat membentuk segitiga.

Namun, jika **isSegitiga** terpenuhi, maka fungsi akan melanjutkan untuk mencari besar ketiga sudut dari segitiga yang terbentuk dengan menggunakan **Aturan Cosinus**.

```
sudutA = N[ArcCos[(BC^2 + CA^2 - AB^2)/(2*BC*CA)]*180/Pi];
sudutB = N[ArcCos[(AB^2 + CA^2 - BC^2)/(2*AB*CA)]*180/Pi];
sudutC = N[ArcCos[(AB^2 + BC^2 - CA^2)/(2*AB*BC)]*180/Pi];
sudutList = Sort[{sudutA, sudutB, sudutC}];
```

Lalu, sudut-sudut tersebut akan ditampilkan secara terurut.

```
sudutList = Sort[{sudutA, sudutB, sudutC}];
```

Terakhir, akan dibuat akan ditampilkan gambar segitiga dalam bidang kartesius apabila **isSegitiga** terpenuhi.

```
If[isSegitiga, ListLinePlot[{titik1, titik2, titik3, titik1}]]
```