# Pseudo-random Distribution: Apakah Kamu Benar-benar Beruntung?

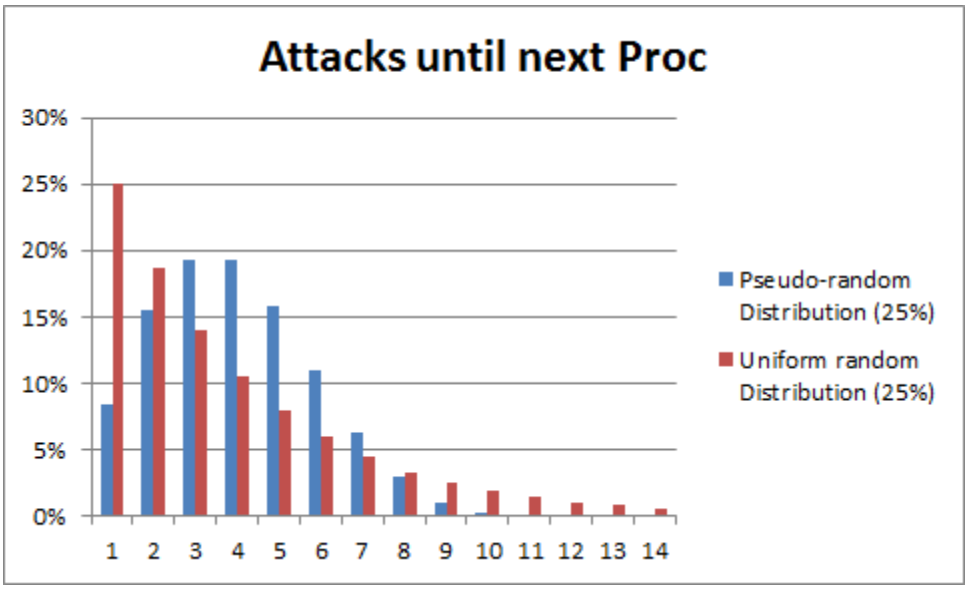
Ketika bermain *game*, tentu saja kita pernah melihat unsur 'keberuntungan' sang pemain sering dipakai. Seperti misalnya saat bermain *game online Tetris Friends* (http://www.tetrisfriends.com/), setiap kali pemain masuk ke akunnya pada hari yang berbeda, akan ditampilkan sebuah lotere berisi koin, *power ups*, maupun *diamond*. Pemain akan mengeklik lotere tersebut untuk mendapatkan tiga buah *item*. Contoh lainnya yaitu pada *game gacha* seperti *Fire Emblem Heroes* dimana untuk melakukan *summoning* sebuah *hero* diperlukan lima buah *orb*. Untuk mendapatkan *hero* yang diinginkan pun relatif tidak mudah karena *hero* yang akan didapatkan dari hasil menukar *orb* tersebut akan dipilih secara acak sehingga pemain dapat memperoleh *hero* yang belum pernah didapatkan namun juga tidak menutup kemungkinan pemain mendapatkan *hero* yang sudah pernah didapat.

Pada *game* umumnya, untuk melakukan randomisasi seperti ini digunakan distribusi seragam. Distribusi seragam (*uniform distribution*) atau distribusi acak-sebenarnya (*true-random distribution*) menggambarkan probabilitas dari kejadian-kejadian acak yang berlangsung tanpa adanya manipulasi pada peluangnya dan tidak bergantung kepada hasil keluaran sebelumnya. Hal ini berarti setiap percobaan berlangsung secara independen sehingga seorang pemain benar-benar mengandalkan keberuntungan semata.

Dalam beberapa kasus, penggunaan distribusi seragam ini tidak akan menimbulkan banyak masalah. Namun, beberapa *game developer*, khususnya *developer game* kompetitif, merasa bahwa randomisasi dengan cara ini kurang sesuai. Hal ini dikarenakan randomisasi ini tidak dapat mencerminkan keahlian seseorang dalam bermain *game* tersebut – apakah pemain tersebut memang benar cakap dalam apa yang dilakukannya atau hanya keberuntungan belaka. Untuk mengatasi hal ini, maka digunakanlah distribusi acak-semu.

Distribusi acak-semu (*pseudo-random distribution*, *pseudo* berarti semu/palsu), atau yang biasa disingkat PRD, menggunakan sistem yang seolah-olah terlihat tidak dapat diprediksi, namun pada kenyataannya menerapkan hal-hal serta kemungkinan-kemungkinan yang dapat dimanipulasi dan diprediksi. Beberapa program besar (terutama pada aplikasi permainan) di dunia seperti *Dota 2*

telah banyak menggunakan sistem PRD tersebut dalam berbagai fitur daripada menggunakan distribusi acak-sebenarnya.




Gambar 1 – Grafik yang menunjukkan perbedaan efek dari PRD dan distribusi seragam pada jumlah serangan antar aktivasi (*proc*), dengan P = 25%.

Secara umum, aktivasi (misalnya pada permainan *Dota 2*, aktivasi dapat berupa suatu *hero* dapat melakukan serangan *critical* dengan peluang tertentu yaitu *p*) beruntun dan ketidakaktifan (melakukan serangan biasa/tidak critical dengan peluang *q*) beruntun suatu kejadian (terjadi *p* beruntun atau *q* beruntun dengan p + q = 1) yang menggunakan sistem PRD akan lebih jarang terjadi. PRD bekerja dengan cara meningkatkan kemungkinan suatu aktivasi oleh suatu nilai tertentu setiap kali terjadi percobaan yang tidak mengaktivasi hingga peluang untuk aktivasi tersebut melebihi 100%. Di sisi lain, tepat ketika terjadi sebuah aktivasi, peluang untuk mendapatkan aktivasi selanjutnya secara beruntun akan menjadi lebih rendah dibandingkan dengan yang dituliskan pada deskripsi peluang aktivasi tersebut.

Contohnya, suatu serangan *critical* ditulis memiliki peluang 35%. Pada kenyataannya, peluang *critical* tersebut terjadi pada serangan pertama setelah *critical* terakhir adalah 24,324% (dibawah peluang yang dituliskan sesungguhnya yaitu 35%), peluang *critical* tersebut terjadi pada serangan

kedua setelah *critical* terakhir adalah 34,375% (masih dibawah peluang yang dituliskan sesungguhnya yaitu 35%), dan masih meningkat lagi seterusnya hingga melebihi 100%. Namun demikian, jika diperhitungkan rata-ratanya, total peluang untuk melakukan *critical* dengan perhitungan PRD tersebut akan tetap 35%. Sistem ini mengakibatkan kejadian-kejadian yang menggunakan suatu peluang akan membuat terjadinya aktivasi yang lebih konsisten daripada dengan menggunakan distribusi acak-sebenarnya yang lebih umum. Kejadian-kejadian beruntun yang biasa disebut "*good luck*" atau "*bad luck*" lebih jarang terjadi, membuat permainan atau sistem lainnya pun menjadi lebih tidak tergantung pada keberuntungan.

Terdapat berbagai cara/rumus untuk mendapatkan sebuah sistem PRD. Selama cara/rumus yang digunakan dapat menghasilkan distribusi yang cocok dan hampir sesuai/sama dengan peluang yang diinginkan, maka cara/rumus tersebut dapat digunakan & dianggap sebagai PRD. Salah satu rumus yang paling mudah dimengerti yaitu

$$P(N) = C \; x \; N$$

dengan P(N) adalah peluang terjadinya suatu aktivasi pada tes ke-N sejak terakhir kali aktivasi terjadi. Untuk setiap percobaan yang dapat menghasilkan aktivasi namun tidak, peluang dari aktivasi tersebut untuk percobaan selanjutnya akan ditambahkan oleh sebuah konstanta yaitu C. Konstanta ini jugalah yang akan menjadi peluang aktivasi awal. Misalnya untuk C = 0,2 , maka P(1) akan bernilai 0,2 , P(2) bernilai 0,4 , P(3) bernilai 0,6 , dan peluang paling maksimal yaitu P(5) bernilai 1. Nilai C tersebut dapat diperoleh dari nilai P(E) berapapun dengan menggunakan rantai Markov (*Markov Chains*).

Tabel berikut menunjukkan nilai C yang diperoleh dari beberapa nilai P(E). *Max N* adalah jumlah percobaan minimum yang akan menyebabkan P(N) = C x N menjadi lebih besar dari 1 (aktivasi yang terjamin). *Average N* adalah jumlah N yang diharapkan untuk menghasilkan sebuah aktivasi. *Most Probable N* adalah N yang memiliki peluang terbesar sebuah aktivasi terjadi (peluang terbesar jumlah percobaan antar aktivasi). *SD* merupakan standar deviasi dari PRD, sedangkan *SDt* merupakan standar deviasi dari distribusi acak-sebenarnya.

| P(T) | C | Max N | Most Probable N | Average N | SD | SDt |
|---|---|---|---|---|---|---|
| 5% | 0.00380 | 264 | 16 | 20.00 | 10.30 | 19.53 |
| 10% | 0.01475 | 68 | 8 | 10.00 | 5.06 | 9.50 |
| 15% | 0.03221 | 32 | 6 | 6.67 | 3.31 | 6.16 |
| 20% | 0.05570 | 18 | 4 | 5.00 | 2.43 | 4.48 |
| 25% | 0.08475 | 12 | 3 | 4.02 | 1.90 | 3.49 |
| 30% | 0.11895 | 9 | 3 | 3.34 | 1.54 | 2.81 |
| 35% | 0.14628 | 7 | 3 | 2.98 | 1.35 | 2.43 |
| 40% | 0.18128 | 6 | 2 | 2.65 | 1.17 | 2.10 |
| 45% | 0.21867 | 5 | 2 | 2.39 | 1.03 | 1.83 |
| 50% | 0.25701 | 4 | 2 | 2.19 | 0.91 | 1.62 |
| 55% | 0.29509 | 4 | 2 | 2.03 | 0.83 | 1.45 |
| 60% | 0.33324 | 4 | 2 | 1.89 | 0.74 | 1.30 |
| 65% | 0.38109 | 3 | 2 | 1.77 | 0.69 | 1.17 |
| 70% | 0.42448 | 3 | 2 | 1.66 | 0.63 | 1.05 |
| 75% | 0.46134 | 3 | 2 | 1.58 | 0.57 | 0.96 |
| 80% | 0.50276 | 2 | 1 | 1.50 | 0.50 | 0.87 |

Sumber : https://dota2.gamepedia.com/Random_distribution

Misalnya, seseorang pemain dapat membuka *chest* ataupun *lottery* yang memiliki kemungkinan untuk memberikan sebuah barang langka dan juga memiliki peluang untuk menghasilkan barang normal atau bahkan tidak memberikan barang apapun. Dengan menggunakan distribusi acak sesungguhnya, pemain bisa saja sudah membuka *chest* tersebut sebanyak 100 kali namun tidak berhasil sama sekali untuk memperoleh barang langka incarannya. Namun, pemain lain mungkin saja memperoleh barang langka tersebut tiga kali berturut-turut. Tidak adil, bukan?

Dengan menggunakan PRD, walaupun kemungkinan seorang pemain mendapatkan *drop item* ataupun peluang untuk melakukan *doublestrike* atau efek *skill* lain *hero* pemain tersebut kecil, namun sudah dipastikan bahwa pemain akan mendapatkan *item* atau efek *skill* yang diinginkan tersebut (walaupun butuh beberapa kali dicoba). Tentunya hal ini akan mengurangi kefrustasian pemain dalam bermain *game* tersebut.

Selain itu, pada tempat permainan *arcade*, banyak ditemukan permainan-permainan yang kelihatannya membutuhkan keahlian khusus atau bahkan tingkat keuntungan yang tinggi layaknya

permainan *jackpot*, *monster drop*, dan sebagainya. Namun, tak jarang bahwa *developer* permainan tersebut menyetel permainannya agar hanya bisa dimenangkan (menang *jackpot* atau hadiah besar) hanya sekali dalam beberapa puluh kali atau bahkan ratus kali kesempatan. Salah satunya adalah permainan *Cyclone*, dimana pada permainan tersebut seorang pemain harus menekan tombol ketika lampu yang berjalan berkedip tepat di antara dua lampu besar dihadapannya. Walaupun sang pemain telah menekan tombol pada saat yang tepat, bila pada saat itu jumlah permainan yang telah dimainkan belum memenuhi, maka lampu tetap akan berhenti di dekat *checkpoint* namun tidak pernah pas pada *checkpoint* tersebut (untuk lebih lengkap, dapat dilihat pada pranala berikut: https://youtu.be/vXBfwgwT1nQ). Hal ini tentunya akan membuat pemain permainan *arcade* menjadi *ogah-ogahan* dan lebih memilih permainan lain yang dirasa lebih memuaskan. Bila diterapkan PRD pada permainan ini, tentunya permainan ini akan terasa lebih *fair* untuk pemain dan juga tidak akan menimbulkan kerugian yang banyak pada *developer* permainan tersebut, karena pihak pemain merasa bahwa ia memiliki kesempatan untuk menang dalam permainan ini dan pihak *developer* juga memperoleh rasa kepercayaan dari pemain dan masih mendapatkan *profit* dari permainan ini.

Referensi:


https://dota2.gamepedia.com/Random_distribution

https://www.dotabuff.com/blog/2016-01-03-pseudorandom-mechanics--and-how-to-use-them-to-your-advantage

https://liquipedia.net/dota2/Pseudo_Random_Distribution

https://gaming.stackexchange.com/questions/161430/calculating-the-constant-c-in-dota-2-pseudo-random-distribution

https://www.reddit.com/r/DotA2/comments/7bj2ij/pseudo_random_in_dota_revisited/

https://youtu.be/vXBfwgwT1nQ

http://www.tetrisfriends.com


Pembagian tugas:

Cornel: Membuat definisi, penjelasan dan cara kerja PRD

Ellen: Membuat pendahuluan, penggunaan PRD, dan *publish* di GitHub Pages