Valor Esperado em um Jogo de Dardos

Gabriel Gonçalves Rocha
21/1063102
Maria Alice Bernardo da Costa Silva
21/1063210
Pedro Henrique dos Santos Ferreira
21/1063229
Vinícius Mendes Martins
21/1063265

Resumo – Este relatório teve como objetivo calcular, na linguagem C, a esperança da pontuação em um jogo de dados, além de fazer o cálculo da pontuação que um jogador obteria a partir do lançamento de N dardos aleatórios (sendo N o expoente de um número em base 10 e possuindo 7 como seu valor máximo) através do Método de Monte Carlo. Com esse fim, a equipe se reuniu semanalmente, na plataforma Discord, para discutir quais ferramentas e funções seriam usadas na construção do código. Além das reuniões entre a equipe, foram realizadas reuniões com o professor responsável pelo trabalho.

Palavras-chave – Dardos; Esperança; Estimativa; Método de Monte-Carlo; Programação; Trabalho Acadêmico.

I. INTRODUÇÃO

O presente relatório apresenta de forma estruturada e concisa o trabalho final da matéria de Algoritmo e Programação de Computadores, lecionada pelo professor Dr. Edson Alves da Costa Júnior na Universidade de Brasília.

O propósito deste documento é esclarecer como foi desenvolvido o código que calcula, por meio do método de Monte Carlo, a pontuação obtida em um jogo de dardos, além de realizar o cálculo da esperança da pontuação do jogo citado anteriormente. O conteúdo foi dividido e está organizado nos tópicos Metodologia, Resultados, Considerações finais, bem como as fontes de informação.

II. METODOLOGIA

A equipe iniciou o trabalho buscando informações sobre o método de Monte Carlo e como sua implementação poderia ser feita na linguagem C, a informação foi encontrada em [1]. O grupo decidiu também pesquisar as funções presentes na biblioteca cmath [2] e verificar quais poderiam ser úteis, se colocadas no trabalho.

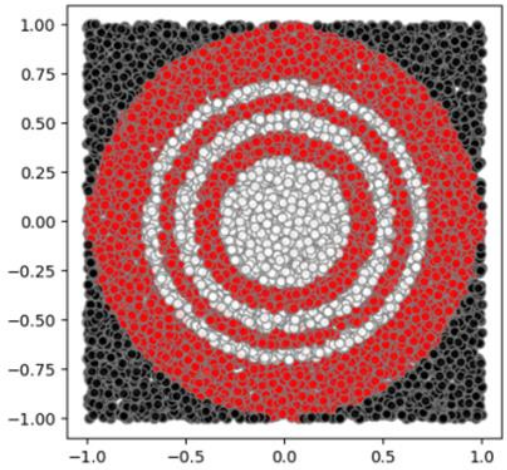
As funções do código foram separadas em diferentes arquivos, são elas: distancia.c, esperança.c, main.c e randNumber.c. Para uma maior organização do relatório, a descrição da elaboração do código foi dividida em etapas e será apresentada em tópicos.

Gerando números aleatórios

A equipe utilizou a função Srand, presente na biblioteca stdlib.h, para gerar números aleatórios a partir de uma semente, denominada Time(NULL) e presente na biblioteca time.h [3], para retornar valores aleatórios de acordo com o tempo.

Pode-se observar a variação dos valores produzidos pelo algoritmo gerador de números aleatórios, para obtenção da posição dos dardos, no Gráfico 1. Os valores foram restringidos de forma que tivessem 0 e 1 como menor e maior valor, respectivamente, para os eixos X e Y, conforme exigido nos critérios do trabalho. Além disso, o gráfico foi ilustrado de forma a demonstrar a região pontuada em que o dardo caiu.

Gráfico 1. Gráfico de dispersão gerado por um script em Python utilizando 10^5 dardos.



Fonte: Autor

Verificando quais dardos caíram na Área pontuada

Foram criadas duas variáveis, o raio, uma constante de valor 0.5, e o vetor resultados, de tamanho variável. O tamanho do vetor resultados muda de acordo com a quantidade de dardos inseridos no programa.

Essas variáveis foram implementadas para a realização do cálculo que verifica se determinado dardo caiu dentro do círculo de raio 0.5, ou seja, na área pontuada. Os valores do cálculo são armazenados no vetor resultados, para que posteriormente seja determinada a pontuação obtida em cada um dos lançamentos.

Inicialmente o cálculo foi feito usando a Equação 1, porém após reunião com o professor a equipe foi orientada a fazer a troca para a Equação 2, uma vez que a Equação 1 retornava o valor errado. Para isso foi usada a função hypot, presente na biblioteca math.h.

$$resultados[i] = ((X^2 - R) + (Y^2 - R)) \quad (1)$$

Onde, resultados[i] é o vetor que armazena o resultado da operação, X é a posição do dardo no eixo x, R é o raio do círculo e Y é a posição do dardo no eixo y.

$$resultados[i] = \sqrt{(X^2 + R^2) + (Y^2 - R^2)} \quad (2)$$

Realizando a seleção e atribuindo pontos aos lançamentos

As verificações foram feitas usando a estrutura condicional *if* e *else*. Apesar de intuitiva, o grupo encontrou problemas em sua implementação, uma vez que o código retornava uma pontuação fixa independentemente da quantidade de dardos lançados.

Após realização de alguns testes, o erro foi encontrado na transferência de valores entre funções, os valores do vetor resultados estavam se perdendo e assumindo o valor 0. A equipe, então, decidiu unir os elementos das funções e, após essa alteração, o código se comportou e realizou a atribuição das pontuações corretamente.

Erro na execução do código

Apesar de realizar o cálculo da pontuação corretamente, o código estava apresentando erro de segmentação ao serem atribuídos valores maiores que 10^6 para a quantidade de dardos lançados.

Após consulta ao professor, a equipe foi direcionada a declarar um vetor estático global de tamanho 10^7+1 ou processar os dados sem a utilização de um vetor, pois se estivesse sendo usado um vetor de tamanho variável declarado na *main.c*, função principal, o tamanho do vetor não seria suficiente para o armazenar de todos os valores e ocasionando a falha de segmentação.

Para a solucionar o problema, no entanto, a equipe alterou o tamanho do vetor resultados para que fosse alocado dinamicamente, de acordo com a quantidade de dardos lançados. Nesse processo foi usada a função malloc presente na biblioteca stdlib.h [4].

O cálculo da Esperança

Para o desenvolvimento do cálculo da esperança o grupo realizou o cálculo da área dos anéis e círculo presente no tabuleiro do jogo de dardos. A área contabilizada considera somente a área de atribuição de pontuação, ou seja, região que quando atingida por um dardo atribui ao jogador uma determinada quantidade de pontos. O cálculo da área dos anéis foi realizado através da Equação 3 e a Equação 4 foi utilizada para determinação da área do círculo central, os resultados estão presentes na Tabela 1.

$$AN = (\pi R^2) - (\pi r^2) \quad (3)$$

Onde AN é a área do anel, R é o raio do círculo externo e r é o raio do círculo interno.

$$AC = \pi t^2 \quad (4)$$

Onde AC é a área do círculo central e t é o raio desse mesmo círculo.

Tabela 1. Área das partições pontuadas.

Pontuação atribuída	Raio do círculo	Área
5	0.5	$9\pi/100$
15	0.4	$7\pi/100$
30	0.3	$\pi/20$
50	0.2	$3\pi/100$
100	0.1	$\pi/100$

Fonte: Autor

Os valores das pontuações atribuídas em cada uma das áreas foram armazenados em um vetor e o cálculo da esperança foi realizado através do somatório presente na Equação 5, transcrito no código como um laço do tipo *for*.

$$Esperança = \sum_{V=1}^{5} V * p \quad (5)$$

Onde, V é a pontuação atribuída a área, variado nos valores 5, 15, 30, 50 e 100 a cada execução, e p é a área do círculo ou anel que possui a pontuação usada anteriormente.

Revisão final do código com o professor

Ao fim do desenvolvimento do código a equipe se reuniu novamente com o professor, onde foi orientada a fazer alterações pontuais no código. A primeira mudança deveria ser feita nas condições usadas nos *if* e *else* usados para atribuição das pontuações, uma vez que as condições usadas se repetiam, se tornando redundantes. A segunda mudança deveria ser realizada no cálculo da esperança, pois o valor que ela retornava estava incorreto, uma vez que dependia da quantidade de dardos jogados e a equação correta não deveria depender desse valor. A mudança foi efetuada e o cálculo foi feito como descrito em *O cálculo da Esperança.*

III. RESULTADOS

Ao fim dos cálculos e dos processos de criação do código, foi possível obter um código que roda no ambiente Linux e estima a pontuação de um jogo de dardos usando dois métodos: Monte Carlo e a pontuação estimada analiticamente.

A partir da execução do código, uma mensagem é exibida no terminal pedindo ao usuário que digite a quantidade de dardos a serem jogados, para a realização dos cálculos detalhados anteriormente. Após o processamento dos dados, a pontuação real e o valor estimado para ela são exibidos nesse mesmo terminal.

A pontuação obtida através da esperança analítica é fixa e tem como valor 17,278760. Já as pontuações calculadas por meio do método de Monte Carlo variam a cada execução e podem ser analisadas na Tabela 2.

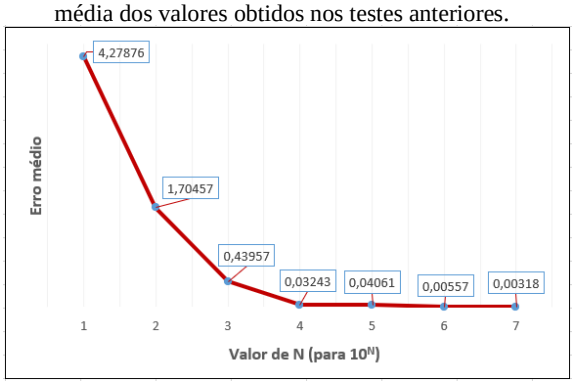
Tabela 2. Amostras de pontuação obtidas em testes com o algoritmo do Método de Monte Carlo.

Quantidade de dardos lançados	Teste 1	Teste 2	Teste 3
10^1	6.50000	22.00000	10.50000
10^2	17.20000	17.60000	22.15000
10^3	17.97500	17.94500	17.23500
10^4	17.13950	17.35000	17.24950
10^5	17.19890	17.17525	17.34030
10^6	17.28630	17.28341	17.28328
10^7	17.27527	17.27126	17.28020

Fonte: Autor

Ao realizar a comparação dos valores entregues pelo algoritmo aqui desenvolvido, pode-se constatar que o cálculo de Monte Carlo foi realizado corretamente e apresenta uma baixa taxa de erro, demonstrada no Gráfico 2.

Gráfico 2. Taxa de erro médio entre a esperança e a média dos valores obtidos nos testes anteriores.



Fonte: Autor

IV. CONSIDERAÇÕES FINAIS

Ao fim do desenvolvimento do trabalho, o grupo observou que o método de Monte Carlo pode realmente ser utilizado para criar a estimativa da pontuação final de um jogo de dardos.

Outro aspecto interessante é que calcular a esperança a partir desse método se torna algo simples. O grupo conseguiu descobri-la antes de realmente fazer qualquer cálculo, simplesmente observando os resultados das jogadas, fornecidos pelo código.

Em suma, o grupo percebeu que o método de Monte Carlo possui muitas utilidades e grande versatilidade quando se trata de calcular estimativas, possibilitando a geração de um código estruturado e funcional.

REFERÊNCIAS

[1] RAVENZWAAIJ, D. van; CASSEY, P; BROWN, S. D. **A Simple Introduction to Markov Chain Monte-Carlo sampling.** [S.I.], Psychonomic Bullentin and Review. 25, p. 143-154, 2018. Disponível em: https://link.springer.com/content/pdf/10.3758/s13423-016-1015-8.pdf. Acesso: 30 mar. 2022

[2] **C reference**: cmath. [20-?]. Disponível em: https://en.cppreference.com/w/cpp/header/cmath. Acesso em: 30. mar. 2022

[3] **C reference**: time. [20-?]. Disponível em: https://en.cppreference.com/w/c/chrono. Acesso em: 07. mar. 2022

[4] **Amostra simplificada da interface da biblioteca stdlib**. [20-?]. Disponível em: https://www.ime.usp.br/~pf/algoritmos/apend/stdlib.h.html. Acesso em: 07. mar. 2022