Relatório do trabalho T1 (Os Macaquinhos)

Felipe Freitas e Luiza Heller

PUCRS

1. Introdução

Neste trabalho, o objetivo era ajudar meu primo antropólogo explorador da selva que havia observado diversos macacos por um longo periodo de tempo, e percebido um dos jogos que eles realizavam. O jogo em questão era uma troca de cocos entre muitos macacos contendo uma quantidade par ou impar de pedrinhas e, ao final do jogo, o macaco que tivesse mais cocos seria eleito o campeão do bando por uma semana.

2. O problema

O problema resolvido foi o de descobrir, ao final de diversas rodadas em que múltiplos macacos cada um com muitos cocos e ainda mais pedrinhas, qual macaco terminava o jogo com mais cocos. Isto é algo praticamente impossível de um ser humano contar, então um programa que possa fazer essa análise em no menor tempo possível é algo muito útil para os pesquisadores.

3. Processo de solucao

Para resolver o problema, o primeiro passo lógico é ler o arquivo com os macacos em questão e interpretá-lo. Então o programa começa lendo o arquivo definido (ou todos na pasta de testes) e precisa entender duas coisas: a quantidade de rodadas do jogo, e as informações de cada macacão. Para o primeiro: basta ler a segunda palavra (considerando uma palavra por espaço) da primeira linha como um número, e para as informações de cada macaco, dividimos a linha de cada macaco, que tem o formato:

```
Macaco {id} par -> {evenTarget} impar -> {oddTarget} :
{coconutAmount} : {stoneAmount} {stoneAmount} {stoneAmount}
```

A partir desse formato, como sabemos que todos os macacos têm esse formato, separamos inicialmente a linha pelo " : ", e temos um vetor com 3 posições resultantes:

```
0:    Macaco {id} par -> {evenTarget} impar -> {oddTarget}
1:    { coconutAmount }
2:    {stoneAmount} {stoneAmount} {stoneAmount}
```

A quantidade total de cocos de cada macaco é simplesmente a segunda posição, e depois para distribuir em questão da paridade basta iterar sobre a terceira posição e verificar: caso seja um número par, soma um no total de cocos pares desse macaco. Ao final, subtrai-se do total da posição 1 o total de cocos pares.

Agora, sobre a linha dos alvos e id de cada macaco, dividimos ela em " -> ", tendo como resultado também um vetor de 3 posições:

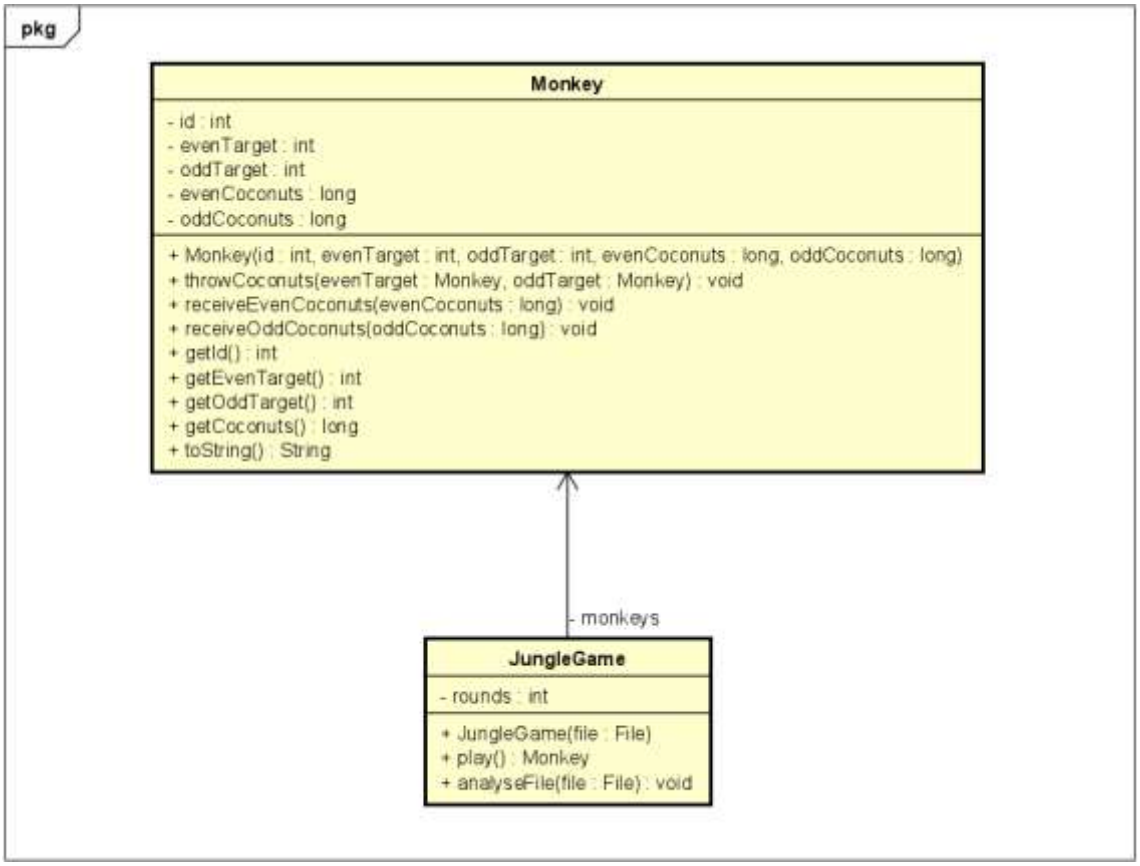
```
0:        Macaco {id} par
1:        {evenTarget} impar
2:        {oddTarget}
```

Novamente, temos um caso que já é o resultado que queremos (oddTarget), e só precisamos converter, e depois pegamos as posições 2 e 1 respectivamente das posições 0 e 1 do vetor, separando assim o id e o alvo par.

Com tudo isso separado, podemos montar nosso macaco e repetir para todas as próximas linhas(macacos) do arquivo.

Finalizada a leitura do arquivo, criamos um laço que se repete uma quantidade de vezes igual ao número de rodadas lidas na primeira linha do arquivo, e passamos por todos os macacos separados anteriormente. Para cada macaco, separamos seus alvos par e impar (que são um número equivalente a uma posição na lista de macacos) e convertemos em macaco, para passar adiante para a função "throwCoconuts" do macaco em questão. Em seguida, ele passa todos seus cocos pares para o alvo par e todos os cocos impares para o alvo impar, e tem todos seus cocos zerados, dando inicio a vez do próximo macaco.

Inicialmente, o problema havia sido "resolvido" de uma maneira mais complexa, com classes adicionais para os cocos de cada macaco, por exemplo, mas foi descoberto que isso não era performático ou mesmo necessário, visto que os cocos são equivalentes; importa a quantidade desses e não sua identidade.



4. Evidências de que o problema foi resolvido

Resultados do arquivo gerado pelo programa em modo de teste:

Tempo decorrido para analisar o arquivo "0004macaquinhos.txt": 00:00.041
Vencedor: Macaco 1 - 5 cocos.
Tempo decorrido para analisar o arquivo "0006macaquinhos.txt": 00:00.012
Vencedor: Macaco 0 - 43 cocos.
Tempo decorrido para analisar o arquivo "0050macaquinhos.txt": 00:00.030
Vencedor: Macaco 14 - 1847 cocos.
Tempo decorrido para analisar o arquivo "0100macaquinhos.txt": 00:00.091
Vencedor: Macaco 88 - 16327 cocos.
Tempo decorrido para analisar o arquivo "0200macaquinhos.txt": 00:00.098
Vencedor: Macaco 9 - 17043 cocos.
Tempo decorrido para analisar o arquivo "0400macaquinhos.txt": 00:00.272
Vencedor: Macaco 109 - 402270 cocos.
Tempo decorrido para analisar o arquivo "0600macaquinhos.txt": 00:00.690
Vencedor: Macaco 49 - 307830 cocos.
Tempo decorrido para analisar o arquivo "0800macaquinhos.txt": 00:00.986
Vencedor: Macaco 33 - 626323 cocos.
Tempo decorrido para analisar o arquivo "1000macaquinhos.txt": 00:01.1528
Vencedor: Macaco 77 - 333394 cocos.
Tempo decorrido para analisar o arquivo "2000macaquinhos.txt": 00:06.6456
Vencedor: Macaco 539 - 4844003 cocos.
---------- RESULTADOS ----------
Total de arquivos analisados: 10
Tempo total de execução: 00:10.10210
Tempo médio de execução: 00:01.1021

5. Conclusões

Podemos concluir do trabalho diversas coisas, mas principalmente a importância do uso das estruturas de dados corretas de acordo com o problema proposto. Aprendemos que estruturas como Listas (sejam elas encadeadas ou de array) são exponencialmente mais complexas quando aplicadas sem o endendimento adequado do problema, e, ainda,

que sempre há o que pode ser melhorado, por menor que seja, visto que será importante conforme o projeto amplia.