Concepção e Análise de Algoritmos

Troca de Letras

Ângela Cardoso e Bruno Madeira

1 de Junho de 2015

Conteúdo

Capítulo 1

Introdução

No âmbito da disciplina de Concepção e Análise de Algoritmos do Mestrado Integrado em Engenharia Informática e Computação, foi-nos proposta a implementação de um jogo de formação de palavras.

É disponibilizada uma cadeia aleatória de letras. Alternadamente, cada um de dois jogadores pode: escolher duas letras da cadeia e trocar a sua posição e/ou retirar uma subsequência que seja uma palavra. As palavras podem ser formadas no sentido direto ou inverso. Por cada palavra que forme, o jogador ganha um número de pontos igual ao número de letras da palavra. No final, vence o jogador que tiver a maior pontuação, ou seja, aquele com o maior conjunto total de letras nas palavras que encontrou.

Para decidir quais as palavras aceitáveis, utiliza-se um dicionário. Para o efeito, escolhemos um ficheiro .dic com todas as palavras em inglês americano, organizadas por ordem crescente de tamanho e por ordem alfabética. A escolha da língua prendeu-se essencialmente com a inexistência de acentos ortográficos e com o facto de ser um dicionário mais pequeno, comparado por exemplo com o de português. No entanto, tudo funcionaria da mesma forma com um dicionário de outra língua, desde que se faça o devido tratamento de todos os caracteres especiais.

Neste documento descreve-se a forma como implementamos o jogo, desde o tipo de algoritmos utilizados até às inteligências artificiais que permitem que um único jogador defronte o computador em diversos níveis de dificuldade.

Capítulo 2

Reconhecimento de Palavras

Ainda que não fosse implementada qualquer inteligência artificial, para que o jogo funcionasse, seria sempre necessário determinar se uma determinada sequência de caracteres escolhida pelo jogador é ou não uma palavra, isto é, se faz parte do dicionário ou não. Neste aspeto o jogo funciona como um corretor ortográfico, embora não haja necessidade de sugerir alternativas quando não se trata de uma palavra.

Uma vez que apenas possuímos um ficheiro com a lista das palavras aceitáveis, a abordagem ingénua a este problema seria bastante lenta. De facto, sempre que fosse selecionada uma não palavra, um algoritmo de força bruta, percorreria toda a lista até determinar que não funciona. Neste caso, tratando-se de uma lista ordenada por tamanho, o algoritmo poderia parar assim que ultrapassasse o tamanho da não palavra. Mas ainda assim, para não palavras grandes seria um processo moroso. Em inglês, por exemplo, existem quase dez mil palavras com comprimento até cinco.

Outra alternativa seria tirar partido da ordenação do dicionário para fazer pesquisa binária, tendo em conta primeiro o tamanho da sequência e depois a sua ordem alfabética. Isso reduziria a pesquisa a um tempo logarítmico no número de palavras do dicionário. Claro que para comparar palavras do mesmo tamanho seria necessário comparar cada uma das letras, o que é linear no comprimento da sequência. Ou seja, teríamos um algoritmo de ordem $n \log(N)$, onde n é o número de letras da sequência e N é o número de palavras do dicionário. Para dicionários grandes, este processo ainda é relativamente demorado, o ideal seria obter um algoritmo que dependesse apenas do número de letras da sequência.

Por forma a acelerar este processo, decidimos construir uma árvore de prefixos para representar o dicionário. Basicamente, cada nó da árvore possui 26 filhos, correspondentes às 26 letras do alfabeto, ordenadas da forma convencional. Se não existe nenhuma palavra que comece com a sequência de letras obtidas percorrendo a árvore desde a raiz até um nó, então esse nó é nulo e não tem filhos. Por exemplo, começando com a letra 'a' existem várias palavras, então o nó 'a', filho da raiz, terá 26 filhos. Um desses filhos é também

‘a’. Mas não existem palavras começadas por ‘aa’, logo este nó ‘a’ filho do primeiro ‘a’ é nulo e não tem filhos.

Percorrendo a árvore desde a raiz até uma folha, obtém-se uma palavra. No entanto, algumas palavras são prefixos de outras. Logo, cada nó tem também um Booleano que indica se nesse nó termina alguma palavra ou não. Também podemos ver a árvore de prefixos como um autómato finito e determinístico, em que a raiz é o estado inicial e todos os nós em que podemos terminar uma palavra são estados finais.

Para determinar se uma determinava sequência de caracteres é uma palavra, basta percorrer a árvore seguindo a ordem dos caracteres da sequência. Ou seja, uma vez construída a árvore de prefixos, este processo é linear no número de letras da sequência. Ainda assim, é necessário construir a árvore, sendo esse processo linear no número de letras de todas as palavras do dicionário. Ou seja, é um processo relativamente lento.

Por forma a evitar reconstruir a árvore de cada vez que se inicia a aplicação, decidimos guardar a árvore construída num ficheiro. Para tal, usamos índices (em lugar de apontadores) para os filhos de cada nó. Dessa forma, podemos gravar os índices num ficheiro de texto, que é lido no início da aplicação. O processo de construção da árvore precisa de ser feito uma única vez e demora cerca de cinco minutos. A partir daí, a leitura do ficheiro é instantânea.

Capítulo 3

Procura de Palavras

Para criar uma inteligência artificial, isto é, um programa de computador que seja capaz de jogar o jogo, cumprindo as regras e preferencialmente sendo capaz de ganhar, é necessário criar algum algoritmo que seja capaz de reconhecer palavras na cadeia de caracteres.

Poderíamos verificar se cada subsequência da cadeia de caracteres é uma palavra. Mas da forma como temos o dicionário carregado na árvore, há um algoritmo mais eficiente. De facto, dado um caracter inicial, podemos percorrer a árvore ao longo do caminho sugerido pela cadeia e assim obter todas as palavras da cadeia que comecem com esse caracter. Depois basta repetir o processo para o caracter inicial seguinte.

Com este algoritmo simples, já poderíamos construir uma inteligência artificial muito básica. No entanto, não estaríamos a tirar qualquer proveito da possibilidade de trocar duas letras da cadeia antes de escolher uma palavra. Para isso, basta fazer uma alteração ao algoritmo de busca de palavras. Primeiro trocam-se duas letras distintas da cadeia, depois procuram-se palavras na cadeia, mas apenas aquelas que incluam uma das letras trocadas.

Com estes dois algoritmos, já é possível criar um programa capaz de jogar o jogo com todo o potencial. Apenas falta a capacidade de encontrar palavras que necessitem de mais do que uma troca. Isto é, seria necessária uma generalização daquilo que é feito quando se procuram palavras com uma troca. Naturalmente, este tipo de busca é factorial no número de letras distintas na sequência, pois é necessário efetuar todas as trocas possíveis para garantir que não há mais palavras. Embora tal busca se torne lenta em cadeias com mais de 10 letras, é útil como condição de término do jogo. Isto é, se a dada momento (com 10 letras ou menos), o computador determina que não há mais palavras, independentemente do número de trocas, o jogo pode terminar.

Capítulo 4

Inteligências Artificiais

De forma a criar uma experiência mais rica de jogo, decidimos criar várias inteligências artificiais, como diferentes níveis de dificuldades. De seguida descreve-se a lógica por detrás de cada um destes jogadores virtuais.

4.1 Nível Fácil

O jogador mais básico que criamos determina a cada momento uma palavra à sorte, sem efetuar qualquer troca. Apenas faz trocas se não encontrar nenhuma palavra desta forma. Nesse caso, também escolhe uma palavra à sorte de entre as possíveis com uma troca. A forma mais eficiente de fazer isto seria parar a busca assim que é encontrada uma palavra. No entanto, uma vez que os algoritmos que encontram todas as palavras são instantâneos (e necessários para jogadores mais inteligentes), não criamos funções específicas para este efeito.

Na prática, esta inteligência artificial escolhe frequentemente palavras pequenas, dado que estas são mais abundantes. Além disso, dado que quase não efetua trocas, também nesse aspeto encontra geralmente palavras mais pequenas. Isto faz com que seja, de facto, um nível fácil de vencer.

4.2 Nível Médio

O segundo jogador em termos de dificuldade escolhe a cada momento a melhor palavra da cadeia, mas ainda sem efetuar trocas. Mais uma vez, as trocas são reservadas para situações de recurso. Além disso, se efetuar uma troca, este jogador escolhe uma palavra aleatoriamente de entre aquelas que se conseguem formar com uma troca de letras.

Embora escolha a melhor palavra sem trocas, geralmente trocar duas letras permite obter palavras maiores, logo o jogador humano ainda tem alguma vantagem sobre esta

inteligência. Claro que, à partida, é muito mais fácil detectar palavras já formadas, do que procurar palavras com troca, por isso, de certa forma, a lógica deste jogador virtual se aproxima da lógica de um jogador humano de nível médio alto.

Na prática, este jogador já é relativamente difícil de vencer. Especialmente se tivermos em conta que o ser humano comum não conhece o dicionário na sua totalidade, ao contrário do computador, logo vai ignorar uma série de palavras.

4.3 Nível Difícil

Este jogador escolhe sempre a melhor palavra com uma troca. Se não existir, procura a melhor palavra sem trocas. Se ainda assim não encontrar nada, repete o processo com a cadeia invertida, de forma a encontrar palavras em sentido inverso.

Dado que palavras com uma troca tendem a ser maiores, este jogador já é praticamente impossível de vencer por um humano. Mesmo que o humano possa consultar o dicionário e até fazer pesquisas eficientes. A não ser que esteja disposto a fazer "manualmente" aquilo que o computador faz automaticamente, à partida será muito difícil vencer o computador. De qualquer forma, se tiver a sorte de ser o primeiro a jogar, poderá retirar daí alguma vantagem.

4.4 Nível Máximo

O último nível de dificuldade é de facto uma inteligência artificial quase perfeita e para todos os efeitos invencível. A cada momento, este jogador escolhe a melhor palavra de entre todas as possibilidades: com ou sem troca, em sentido direto ou inverso. A única coisa que não faz é planear jogadas com antecedência, por exemplo fazendo uma troca que o ajude a obter uma ótima palavra mais adiante e retirando uma boa palavra agora que não use a troca. De qualquer forma, dada a interação com o outro jogador, não há qualquer garantia que esse tipo de planeamento funcione, podia simplesmente ajudar o adversário a forma a tal palavra ótima.

Além da escolha da melhor palavra, este jogador usa dois pequenos truques no final do jogo. Se tiver mais pontos que o adversário e o adversário tiver passado a vez, então passa também a sua vez, garantindo assim que o jogo termina com a sua vitória, dado que duas passagens é uma das condições de término. Se tiver menos pontos, ainda que

não encontre nenhuma palavra, faz uma troca aleatória, para que o jogo não termine por passagens. Sendo assim, o jogo só termina quando o computador determinar que não há mais palavras.

Apesar de ser invencível, a experiência de jogar este nível de dificuldade, pode ser bastante interessante e utilizada para enriquecer o vocabulário e melhorar as técnicas de detecção de palavras.

Capítulo 5

Principais Dificuldades

Capítulo 6

Distribuição do Trabalho

Capítulo 7

Conclusão

Em contraste com o primeiro projecto, a "simplicidade" do problema permitiu uma abordagem mais intuitiva e consequentemente pudemos pensar, discutir e conceber algoritmos necessários à implementação sem dificuldades. A fácil segmentação do projecto em subproblemas concisos, permitiu um desenvolvimento linear e progressivo. Adicionalmente ainda foram polidos alguns elementos do jogo. Dada a sua completude, estamos satisfeitos com o resultado final.