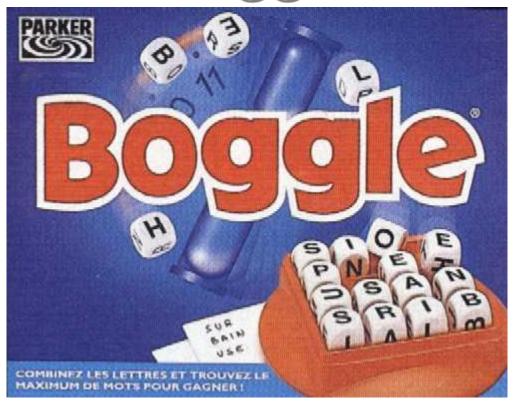


## Universidade de Évora Estrutura de Dados e Algoritmos I

# Boggle



Trabalho realizado por:

Nuno Lopes

### Introdução

No âmbito da disciplina de EDA 1 têm-se como objetivo usar a linguagem java e tipos abstratos de dados para produzir soluções para o jogo Boggle. Jogo que aparece em alguns jornais americanos e rivaliza com as habituais palavras cruzadas ou as mais antigas sopas de letras.

### **Objetivos**

Com este trabalho pretendo resolver o problema proposto escrevendo um condigo simples, fácil de entender, comentado e que execute a tarefa no menor tempo possível, aumentando a performance e aprender melhor a usar as hashtables.

#### **Classes**

Para este trabalho irei usar 7 classes sendo elas:

Esta classe vai ser umas das principais, tendo o Boggle em si, numa matriz 4x4 de posições e a função solve que faz o print de todas as soluções.

Esta função solve vai percorrendo as todas as posições do boggle até deixarem de ser encontradas na hashtable, por exemplo, se a palavra encontrada for "monkew" e se não a encontrar a palavra na hashtable(sendo prefixo ou não) não vale a pena estar a procurar mais hipóteses porque não há nenhuma palavra com prefixo "monkew" permitindo assim aumentar o desempenho do programa.

Usada para os elementos da hashtable, que vai conter um elemento que serão as palavra do dicionário(Strings), um booleano ativo, e outro booleano prefixo que vai permitir verificar se os elementos encontrados são prefixos ou são palavras, estes prefixos uma vez postos a falso, já não poderão ser alterados para evitar sobreposição com os prefixos de outras palavras que sejam iguais a palavra

- △ TabelaElementos : Elemento <T>[]
- △ ocupados: int
- C HashTable()
- C HashTable(int)
- ocupados(): int
- factorCarga(): float
- alocarTabela(int) : void
- tornarVazia(): void
- procurar(T) : Elemento <T>
- contains(T) : boolean
- remove(T) : void
- insere(T, boolean) : void
- NextPrime(int): int
- SDBMHash(String) : long
- rehash(): void
- print(): void

#### 

- △ count:int
  - C LinHashTable()
  - C LinHashTable(int)
  - <sup>₽F</sup> FNV\_64\_INIT : long
  - SF FNV\_64\_PRIME : long
  - SDBMHash(T): int
  - procPos(T) : int

```
Posicao:
             Posicao
                  letra : char
                  coluna: int
                  linha : int
                     repetido : boolean
                  Posicao(char, int, int, boolean)
                  toString(): String
                      getLetra() : char
                     setLetra(char) : void
                      getColuna(): int
                     setColuna(int) : void
                      getLinha(): int
                     setLinha(int) : void
                  isRepetido(): boolean
                      setRepetido(boolean) : void
▲ array : E[]
                  tamanho: int
                  C ArrayList(int)
                  size(): int
                  clear(): void
                  ExpandSize(int) : void
                  insert(E) : void
                  insert(E, int) : void
                     contains(E): boolean
                     procurar(E): int
                  set(E, int) : void
                  get(int) : E
                  a toString(): String
```

Usada para a matriz do Boggle 4x4, contem uma char que vai ser a letra, inteiro coluna, inteiro linha e um booleano repetida para enquanto se estiver a percorrer o boggle

#### Teste:

Esta classe é a que contem o main e que nos permite resolver o problema proposto, sendo que no inicio insere todos as palavras do dicionário numa hashtable com os seus prefixos, e no final resolve o boggle.