[Data]

Giorgio Gandola

UpWords

Documentazione

Sommario

[Gestione Revisioni 2](#_Toc515561657)

[Problema da risolvere 3](#_Toc515561658)

[Introduzione 3](#_Toc515561659)

[Metodo di approccio 3](#_Toc515561660)

[Scelte di base 4](#_Toc515561661)

[Caricamento ed espansione dizionario 4](#_Toc515561662)

[Struttura dati dizionario 4](#_Toc515561663)

[Algoritmi di ordinamento 4](#_Toc515561664)

[Algoritmi di ricerca 4](#_Toc515561665)

[Operazioni di gioco 4](#_Toc515561666)

[Estrazione Lettera 4](#_Toc515561667)

[Calcolo punteggio 4](#_Toc515561668)

[Interfaccia grafica 4](#_Toc515561669)

[Suggerimento 4](#_Toc515561670)

# Gestione Revisioni

|  |  |  |
| --- | --- | --- |
| **VERSIONE** | **MODIFICHE** | **DATA** |
| 0.1 | Creazione scheletro documentazione | 31/05/18 |

# Problema da risolvere

## Introduzione

## Metodo di approccio

# Scelte di base

## Caricamento ed espansione dizionario

Il dizionario necessita di essere espanso con tutte le declinazioni di ogni parola. Inizialmente viene letta una parola da espandere, viene controllata tramite espressioni regolari e, nel caso necessiti di espansioni viene estesa con le sue declinazioni nel dizionario principale. In questo modo l’operazione di espansione, che ha delle tempistiche relativamente lunghe, viene effettuata una sola volta. Nel caso specifico del gioco UpWords abbiamo un campo di gioco di massimo (N) lettere sia in orizzontale che in verticale e di conseguenza le parole di lunghezza superiore vengono automaticamente eliminate durante l’espansione del dizionario stesso in modo da risparmiare spazio altrimenti inutilmente occupato sia su disco che in Memoria Centrale durante l’esecuzione dell’applicativo.

Per quanto riguarda la fase di caricamento in memoria il dizionario espanso viene letto da file per poi essere caricato e organizzato nella struttura dati sotto riportata.

## Struttura dati dizionario

Il dizionario viene organizzato con una struttura dati formata da un array di std::vector contenente le parole del dizionario ordinate in ordine alfabetico crescente.

[immagine esempio]

## Algoritmi di ordinamento

Per l’ordinamento ho scelto di utilizzare il Merge Sort, un algoritmo basato sulla tecnica del “dividi e conquista” molto efficiente. L’alternativa era l’utilizzo del Quick Sort che, nelle simulazioni fatte, era quasi alla pari in prestazioni. Per l’implementazione mi sono affidato alle librerie standard del c++ (std).

## Algoritmi di ricerca

Per la validazione di una parola mi sono affidato alla ricerca dicotomica, una ricerca che necessita di una struttura dati ordinata (come la mia) e che è particolarmente performante (caso peggiore :O(log2N)).

Questa scelta ci avvantaggia nella validazione di una parola inserita.

# Operazioni di gioco

## Estrazione Lettera

## Calcolo punteggio

## Interfaccia grafica

# Suggerimento