Risoluzione automatica del cruciverba

Progetto di Esperienze di programmazione

Davide Coffaro Matricola 556603

2020

Il problema di “Risoluzione automatica del cruciverba” è quello di riempire lo schema dato (con caselle bianche e caselle nere) con le parole disponibili, partendo da un’unica parola inserita nello schema.  
Questo problema può essere visto anche come problema che data una parola con l’iniziale o solo alcune lettere, proponga in automatico le parole disponibili (ricercandolo all’interno di una lista di parole), dando come input anche la lunghezza della parola voluta oppure con una procedura diversa tutte le parole disponibili non tenendo conto della lunghezza.

**ALGORITMI PER RISOLVERE IL PROBLEMA**

* Algoritmo1 - parto dallo schema, cerco la parola da inserire fino a completare il cruciverba:

INPUT matrice rappresentante lo schema iniziale + lista parole da inserire + parola iniziale nello schema  
OUTPUT cruciverba completato o no (true o false)

PROCEDURA ALGORITMO1:

1. Inizializzo trovato(i)=false per i=2…n, questo indicherà per le caselle di lunghezza i se sono state trovate tutte le parole. Inizializzo il timer tempoDiEsecuzione;
2. Finche trovato(i)=true per ogni i=2…n, tempoDiEsecuzione<tempoMax, cicliEseguiti<cicliMax;
3. Incremento i cicliEseguiti di 1. Prendo le parole dello schema di c numero di caselle e le inserisco in una lista (controllando solo le parole ancora da completare), con c=2…n, se c=n+1 torno al punto 2);
4. Ciclo fino a che la lista di parole trovate di c caselle non è vuota, se è vuota faccio un controllo se ho inserito nello schema tutte le parole di lunghezza c aggiorno trovato(c)=true e torno al punto 3) ricercando parole con numero di caselle c+1;
5. A parità di c caselle prendo la parola x con più lettere già inserite;
6. Controllo quali parole di c lettere del dizionario possono essere inserite nello spazio della parola x con c caselle, facendo il confronto dei caratteri lettere-caselle;
7. Se è stata trovata una parola la inserisco nello schema, aumento di 1 il numero di parole di lunghezza c inserite, aggiorno il dizionario (perché alcune parole potrebbero essersi completate automaticamente) e torno al punto 4), altrimenti torno al punto 4), prendendo la prossima parola di c caselle se presente.

CONDIZIONI DI TERMINAZIONE:  
1) trovato(i)=true per i=2…n;  
2) tempo di esecuzione algoritmo>tempoMax???;  
3) cicliEseguiti>cicliMax.

COMPLESSITA’ algoritmo1:  
analizzando l’algoritmo si può notare che ci sono vari cicli annidati necessari per poter completare lo schema e possono dipendere da:   
- cicliMax indicati con n, i cicli massimi superati i quali l’algoritmo termina determinando il completamento o non dello schema;   
- lunghezzaMax indicati con L, la lunghezza massima possibile per le parole dello schema, per il momento impostata a priori dal programma;  
- parole schema e parole disponibili entrambi indicati con m, che indicano il numero delle parole dello schema e quelle della lista di parole da inserire.  
Per approssimare al caso peggiore imposto sia L=O(n) che m=O(n), quindi sia L che m sono dell’ordine di n, numeri molto alto rispetto agli altri input e trovo che la complessità dell’algoritmo1 è O(n^5)

* Algoritmo3 - Parto invece dalle parole disponibili nella lista delle parole da inserire, le analizzo e le inserisco fino a completare il cruciverba:

1)

* Algoritmo2 – procedura simile all’algoritmo1 ma:

1. Utilizzo di lista(i) che contengono le parole di lunghezza i per i=1…n
2. Quando inserisco la parola di lunghezza i nello schema, la elimino anche dalla sua lista(i)
3. Quando lista(i) è vuota per i=1…n termino il ciclo

Problemi relativi all’algoritmo2

* + Riempire lista(i) all’inizio durante la creazione delle caselle ecc…
  + Memoria occupata da listaParolaSchema, listaParoleSchemaDaCompletare, listaParoleDaInserire (dizionario input)

COMPLESSITA’ algoritmo2:  
Anche per questo algoritmo ho diversi cicli annidati che al caso peggiore ha complessità O(n^5) così come l’algoritmo1

* Algoritmo4 (con AI) - utilizzo di algoritmi di intelligenza artificiale per cercare la soluzione in modo più efficiente, correggendo gli errori per cui gli altri algoritmi si bloccavano ad un certo punto non trovando altre parole da inserire nello schema del cruciverba. Con questi algoritmi (che sfruttano la strategia CSP – Constraint Satisfaction Problem) analizzo le parole che possono essere inserite nello schema e nel caso in cui arrivassi ad un punto in cui non è possibile inserire altre parole ma lo schema non è ancora completo, si possa tornare indietro e provare le parole alternative fino al completamento dello stesso.

PROCEDURA ALGORITMO4:

al momento della creazione dello schema inserisco tutte le parole in orizzontale o in verticale (non ancora completate) all’interno di una lista di variabili; dopo analizzo i possibili valori che possono essere inseriti all’interno delle variabili inserendo anch’essi in una lista collegata alle singole variabili, creando così tanti sottodomini delle parole da inserire nello schema, uno per ogni variabile.  
Passi di soluzione dell’algoritmo:  
1) ricerca della variabile a cui assegnare un valore (STRATEGIA MRV e grado maggiore a parità di MRV);

2) inserimento di un valore del sottodominio all’interno della variabile (STRATEGIA di scelta valore per la variabile attuale per minimizzare i vincoli sulle altre variabili ad essa collegata);

3) meccanismo di inferenza in cui vengono ridotti i sottodomini delle variabili collegate a quella corrente, in caso di sottodominio di una variabile collegata vuoto e schema del cruciverba non ancora completo, significa che uno degli assegnamenti di valori alle variabili precedentemente effettuato non era corretto e bisogna fare dei passi all’indietro provando un altro valore (sfrutta il backtracking su una variabile collegata a quella corrente;

4)se tutti i passi precedenti sono andati a buon fine proseguo la ricerca sul sottoproblema in cui adesso ho assegnato un valore alla variabile su cui stavo lavorando.

**SCELTE IMPLEMENTATIVE**

Nell’implementazione del cruciverba ho deciso di scomporre lo schema del cruciverba nelle parole e nelle caselle nere, all’interno delle parole ho la stringa della parola associata che uso per cercare la parola da inserire nello schema ma scompongo ulteriormente la parola nelle singole caselle in cui ho il carattere ad esse collegate che vengono poi stampate a video. Ho deciso di utilizzare come struttura dati l’ArrayList per tutte le collezioni di oggetti utilizzate dal programma (perché struttura dati di dimensione variabile?) e di utilizzare varie classi per implementare il cruciverba:

* InterfacciaCruciverba, è la classe che contiene il metodo Main e che viene utilizzata per inizializzare i componenti grafici della maschera del programma;
* Cruciverba, contiene lo schema del cruciverba e la lista delle parole da inserire, al suo interno vengono implementati gli algoritmi utilizzati nella risoluzione del cruciverba, oltre a quelli di aggiornamento della lista delle parole da inserire e di controllo di completamento o meno del cruciverba;
* Schema, contiene le parole (completate o meno) inserite nello schema del cruciverba e le caselle dello schema stesso, si occupa della creazione dello schema a partire dalla matrice in input e dell’inserimento nello schema della parola iniziale, collega le caselle dello schema alla lista delle parole dello schema, facendo si che l’aggiornamento di uno sia collegato all’altro;
* Parola, ha al suo interno i riferimenti della stringa, caselle dello schema associate, posizione all’interno dello schema, orientamento orizzontale o verticale, lunghezza parola e numero di lettere inserite; può aggiornare le caselle dello schema con la stringa attuale, oppure il contrario e aggiornare la stringa con le caselle dello schema attuali. Può verificare se due parole si riferiscono alle stesse caselle dello schema;
* Casella, si occupa della creazione del componente grafico della casella che viene visualizzata a video e della relativa posizione all’interno dello schema. Contiene il carattere associato alla casella e se è una casella nera (in cui non è possibile scrivere) o no;
* Posizione, contiene i riferimenti riga-colonna nello schema.

Risultati ottenuti:

* Credo che valga per tutti gli algoritmi, il dizionario delle parole disponibili/da inserire alla fine dell’elaborazione non è vuoto ma risulta contenere alcune parole le cui lettere sono state tutte inserite durante l’inserimento di altre parole, quindi la parola è completa ma non è stata inserita nello schema tramite la procedura degli algoritmi. Questo fa sì che ci sia una procedura di aggiornamento del dizionario delle parole disponibili che cicla tutte le parole inserite nello schema che sono complete e le elimini dal dizionario se sono contenute al suo interno. AUMENTA LA COMPLESSITA’ dell’algoritmo/algoritmi di un ordine n, nel caso del primo algoritmo si passa da O(n^4) a O(n^5).