Tracce d' Esame per il
Laboratorio di Algoritmi e Strutture Dati (a.a. 2015-16)

Traccia 1: Risolvere entrambi i quesiti:

1. Costruire un vocabolario V utilizzando un **Hash Table** con il metodo della concatenazione, che abbia le seguenti funzioni:
   - Inserimento del termine
   - Cancellazione
   - Ricerca del termine. In caso di fallimento deve restituire una lista delle parole piú prossime, utilizzando un approccio basato sulla **Distanza di editing**.
     La distanza di editing (o di **Levenshtein**) misura il numero di operazioni (inserimento, cancellazione e correzione) che devono essere eseguite sulla tastiera per trasformare una stringa in un' altra. La distanza di Levenshtein tra due stringhe di caratteri $S = (S_1, \ldots, S_N)$ e $T = (T_1, \ldots, T_M)$ é definita nel modo seguente. Siano S_i e T_j i caratteri corrispondenti rispettivamente all' i-esimo carattere di S ed al j-esimo carattere di T allora la loro distanza di editing $d[i, j]$ é data da:

$$d[i,j] = min \begin{cases} d[i-1,j]+1 \\ d[i,j-1]+1 \\ d[i-1,j-1]+(S_i \neq T_j) \end{cases} \quad (1)$$

$$d[0,0] = 0 \quad (2)$$

2. Costruire un vocabolario V', utilizzando un albero **RED BLACK** che abbia le stesse funzioni, dell' esercizio precedente. Nell' implementazione di entrambi i quesiti si faccia uso delle Classi Astratte.

Traccia 2: Risolvere entrambi i quesiti:

1. Costruire un encoder (codificatore) ed un decoder (decodificatore) che utilizzi la codifica di Huffman.
   L' implementazione deve far uso di una coda di prioritá, realizzata mediante un MIN-HEAP. Definire il formato (od i formati) dei dati accettati in input e definire il formato del file codificato.
   Calcolare il tasso di compressione ottenuto. Il codificatore deve essere almeno in grado di codificare qualunque file di testo (.txt) e, facoltativamente, anche altri tipi di formati (bmp, wav, ...).
2. Si implementi l' algoritmo di Prim per l' albero di copertura minimo in un grafo non orientato. L' implementazione deve far uso di una coda di prioritá, realizzata mediante un MIN-HEAP. Si verifichi la correttezza del programma su un problema reale.

Traccia 3: Risolvere entrambi i quesiti:

1. Si implementi l' algoritmo di Prim per l' albero di copertura minimo in un grafo non orientato. L' implementazione deve far uso di una coda di prioritá, realizzata mediante un albero RED-Black. Si verifichi la correttezza del programma su un problema reale.
2. Il problema delle N-Regine
   (`https://it.wikipedia.org/wiki/Rompicapo_delle_otto_regine`)
   (`https://en.wikipedia.org/wiki/Eight_queens_puzzle`) consiste nel posizionare N Regine degli scacchi su una scacchiera NxN, in modo che nessuna di esse possa catturarne un'altra. Perció, una soluzione dovrá prevedere che nessuna regina abbia una colonna, traversa o diagonale in comune con un'altra regina. Si implementi un programma che:
   - legga da file di testo una disposizione parziale di k Regine su una scacchiera quadrata di lato L ($k < L$) valutando se il test proposto sia coerente con le regole del gioco;
   - mediante la tecnica di backtracking, completi la disposizione delle Regine sulla scacchiera, fornendo tutte le possibili soluzioni;
   - calcoli il numero dei cicli di backtracking effettuati per ogni soluzione trovata.

   Formato di Input del File

   – primo rigo: numero intero indicante il Lato della scacchiera;
   – k righi successivi ($k \leq Lato$): coppie di interi (separati da punto e virgola) indicanti Riga e Colonna di posizionamento delle prime k Regine.

   Esempio di Problema di 13-regine avente 3 soluzioni
   13
   1;1
   2;11
   3;5
   5;4

Traccia 4: Risolvere entrambi i quesiti:

1. Si implementi l' algoritmo di Prim per l' albero di copertura minimo in un grafo non orientato. L' implementazione deve far uso di una coda di prioritá, realizzata mediante un albero RED-Black. Si verifichi la correttezza del programma su un problema reale.
2. Il DOMINO (`https://it.wikipedia.org/wiki/Domino_gioco`);
   (`https://en.wikipedia.org/wiki/Dominoes`) é un gioco da tavolo che si svolge utilizzando una serie di tessere. Queste sono tutte suddivise in due sezioni recanti dei punteggi segnati con dei pallini, da 1 a 6 (o piú); una tessera puó avere due facce uguali, ma, data la

tessera [x—y], non esiste anche la tessera [y—x] (combinazioni con ripetizione); detto Max il valore piú alto delle facce, avremo:

$$TotaleTessere = \frac{(Max + 1)Max}{2}$$

Se il valore delle facce va da 1 a 6, abbiamo 21 tessere possibili; portando a 7 il Max, abbiamo 28 tessere, 36 con Max = 8, 45 con Max = 9, e cosí via. Disposta la prima tessera sul tavolo, si puó mettere una tessera attaccata a quella che c'é gi sul tavolo solo se ha una faccia con un punteggio uguale a una delle estremitá libere (costruzione del serpente).
Si implementi un programma che:

- legga da file di testo un elenco di tessere del Domino;
- mediante la tecnica del Backtracking, stabilisca se sia possibile calarle tutte sul tavolo in un unico serpente; in caso contrario cerchi il serpente di lunghezza massima;
- calcoli il numero dei cicli di backtracking effettuati per ogni soluzione trovata.

Formato di Input del File

- primo rigo: numero intero indicante il massimo valore delle facce.
- n righi successivi ($n \leq TotaleTessere$): coppie di interi (separati da punto e virgola) rappresentanti ciascuna una tessera del gioco.

Esempio di Domino

9
8 ; 3
4 ; 5
6 ; 8
9 ; 9
2 ; 2
4 ; 1
4 ; 4
9 ; 2
6 ; 9
7 ; 2
9 ; 4

Serpente Massimo: [7—2] [2—2] [2—9] [9—9] [9—6] [6—8] [8—3].

Traccia 5: Risolvere entrambi i quesiti:

1. Si risolva il problema delle selezione delle attivitá mediante la tecnica greedy. L' implementazione non puó far uso della libreria STL (sort), **ma deve utilizzare una coda di prioritá, realizzata mediante un albero RED-Black**. Si verifichi la correttezza del programma su un problema reale.

2. Si implementi l' algoritmo di Prim per l' albero di copertura minimo in un grafo non orientato. L' implementazione deve far uso di una coda di prioritá, realizzata mediante un MIN-HEAP. Si verifichi la correttezza del programma su un problema reale.