Prova Finale di Algoritmi e Strutture Dati

note generali

Introduzione

- Obiettivo: implementazione efficiente (e corretta!) di un algoritmo
- Logistica
  - codice sorgente sarà caricato su un server, compilato e fatto girare automaticamente
  - Scadenze:
    - 11 luglio (ore 24) per i laureandi di luglio
    - 12 settembre (ore 24) per tutti gli altri
- Esecuzione del progetto
  - implementazione nel linguaggio C
  - esclusivamente con libreria standard (libc)

Valutazione

- programma deve compilare e girare correttamente
  - verranno resi disponibili dei casi di test come prova per controllare il corretto funzionamento del programma
  - dopo il caricamento sul server, il programma viene fatto girare su test, divisi in 2 parti: pubblici e privati
- si misura la correttezza (risultati in uscita) e l'efficienza (tempi di risposta e memoria occupata) del programma su vari casi di test
- dipendentemente dai risultati sui casi di test, potrete calcolare il voto
- non c'è recupero

Sito per la sottomissione del progetto

- https://dum-e.deib.polimi.it/
- Ogni studente avrà le proprie credenziali per accedere al sito
- Il sistema di sottomissione verrà presentato esaustivamente nel primo incontro coi tutor

Plagi

- progetto da svolgere singolarmente ed in totale autonomia (no a gruppi)
- siete responsabili del vostro codice, quindi vi consigliamo fortemente di:
  1. Non caricarlo in repository pubblici (es. GitHub)
  2. Non passarlo per "ispirazione" a colleghi
- controllo plagi automatizzato
- in caso di copiatura tutti i progetti coinvolti vengono annullati

Un simulatore di Macchine di Turing non-deterministiche

Prova finale di Algoritmi e Strutture Dati AA 2017-18

Il progetto

- Implementazione in linguaggio C standard (con la sola *libc*) di un interprete di Macchine di Turing non-deterministiche, nella variante a nastro singolo e solo accettori.
- Struttura del file di ingresso: prima viene fornita la funzione di transizione, quindi gli stati di accettazione e un limite massimo sul numero di passi da effettuare per una singola computazione (per evitare in maniera banale il problema delle macchine che non terminano), infine una serie di stringhe da far leggere alla macchina.
- In uscita ci si attende un file contenente 0 per le stringhe non accettate e 1 per quelle accettate; essendoci anche un limite sul numero di passi, il risultato può anche essere *U* se non si è arrivati ad accettazione.

Convenzioni utili

- Per semplicità i simboli di nastro sono dei *char*, mentre gli stati sono *int*. Il carattere "_" indica il simbolo "blank".
- La macchina parte sempre dallo stato 0 e dal primo carattere della stringa in ingresso.
- Si assume, come al solito, che il nastro sia illimitato sia a sinistra che a destra e che contenga in ogni posizione il carattere "_".
- I caratteri "L", "R", "S" vengono usati, come al solito, per il movimento della testina.
- Il file di ingresso viene fornito tramite lo standard input, mentre quello in uscita è sullo standard output.

Struttura del file di ingresso

Il file di ingresso è diviso in 4 parti:

- La prima parte, che inizia con "tr", contiene le *transizioni*, una per linea - ogni carattere può essere separato dagli altri da spazi.
  Per es. *0 a c R 1* significa che si va dallo stato 0 allo stato 1, leggendo *a* e scrivendo *c*; la testina viene spostata a destra (*R*).
- La parte successiva, che inizia con "acc", elenca gli stati di *accettazione*, uno per linea.
- Per evitare problemi di computazioni infinite, nella sezione successiva, che inizia con "max", viene indicato il numero di mosse massimo che si possono fare per accettare una stringa.
- La parte finale, che inizia con "run", è un elenco di stringhe da fornire alla macchina, una per linea.

Esempio: MT non-det. per ww^R

```
tr
0 a a R 0
0 b b R 0
0 a c R 1
0 b c R 2
1 a c L 3
2 b c L 3
3 c c L 3
3 a c R 4
3 b c R 5
4 c c R 4
4 a c L 3
5 c c R 5
5 b c L 3
3 _ _ R 6
6 c c R 6
6 _ _ S 7
```

```
acc
7
max
800
run
aababbabaa
aababbabaaaababbabaa
aababbabaaaababbabaab
aababbabaaaababbabaabbaababbabaaaababbabaa
aababbabbaaababbabaabbaababbabaaaababbabaa
```

Standard output:

```
1
1
0
U
0
```

Esempio: MT per ww

tr

```
0 a c R 1
0 b c R 2
1 a a R 1
1 a d L 3
1 b b R 1
2 a a R 2
2 b b R 2
2 b d L 3
3 a a L 3
3 b b L 3
3 c c R 4
4 d d R 10
4 a c R 5
4 b c R 6
5 a a R 5
5 b b R 5
5 d d R 7
6 a a R 6
6 b b R 6
6 d d R 8
7 d d R 7
7 a d L 9
8 d d R 8
8 b d L 9
9 d d L 9
9 a a L 3
9 b b L 3
9 c c R 10
10 d d R 10
10 _ _ S 11
acc
11
max
2000
run
aabaab
bbabbb
ababa
babaaababaaa
ababaababababbabbbbbabbbbabaaababaaaababaabababbabbbbbba-
bbbbabaaababaaa
```

Standard output:

```
1
0
0
1
U
```