

**PROGETTO PER IL LABORATORIO DI ASD**  
**A.A. 2017/18**  
**— VERSIONE 1.0 —**

ALBERTO POLICRITI  
ALBERTO.POLICRITI@UNIUD.IT

SOMMARIO. Scopo del progetto di laboratorio è verificare che lo studente sia in grado di progettare, analizzare e (soprattutto) *implementare* un programma progettato mediante una combinazione degli strumenti (algoritmi e strutture dati) illustrati durante le lezioni del corso di ASD. Il programma prodotto dovrà risolvere al calcolatore un problema di natura computazionale e, se corretto, verrà valutato sia in termini di efficienza che di eleganza della soluzione proposta.

INDICE

1. Regole generali	2
2. Il progetto	4
2.1. Specifiche per il formato di input e output	4
3. Programmi	5
3.1. Metodologie di analisi	5
Appendice A. FAQ	6

## 1. REGOLE GENERALI

- (1) Il progetto deve essere svolto singolarmente.
- (2) I linguaggi ammessi per i programmi sono: C/C++, Java, Pascal/Delphi<sup>1</sup>. L'utilizzo di altri linguaggi deve venire autorizzato preventivamente da me.
- (3) Non è ammesso l'uso di strutture dati o algoritmi già pronti (RB-tree, grafi, sorting, ecc.): dovrete implementare voi le strutture dati e gli algoritmi. Si possono usare oggetti base (stringhe e simili) e funzioni per l'input/output. È ammesso codice orientato ad oggetti (quindi classi astratte, polimorfismi, ereditarietà, ecc. sono accettati—non obbligatori).
- (4) La consegna del progetto deve essere fatta tramite posta elettronica. Si dovrà inviare un archivio compresso (.tar.gz o .zip) come allegato al mio indirizzo di posta, indicando nel corpo della mail nome, cognome e numero di matricola. Scompattando il file compresso deve esserci solo una directory radice che deve contenere il cognome e il numero di matricola dello studente usando l'underscore al posto dello spazio (es. Del\_Fabbro\_123456). Dentro questa directory dovranno esserci i sorgenti con eventuali sottodirectory e la relazione. La relazione dovrà essere in formato PDF e chiamarsi "relazione.pdf". Non includete binari: compilerò i sorgenti in ogni caso. **Allegati non conformi a queste specifiche non verranno presi in considerazione.**
- (5) La ricezione del progetto verrà confermata. Se non si riceve una risposta dopo 2 giorni *lavorativi* inviare nuovamente (facendo presente nel testo il doppio invio).
- (6) Nella relazione dovranno essere presenti, nella prima pagina, nome, cognome, numero di matricola e indirizzo di email dello studente<sup>2</sup>.
- (7) Nella relazione ci dovrà essere una opportuna sezione in cui viene spiegato come compilare il codice in oggetto.
- (8) **Il programma dovrà prendere in input da standard input (terminale) e fornire l'output su standard output (terminale).** La correttezza del risultato calcolato verrà testata su un'ampia batteria di test che verranno lanciati mediante uno script descritto più avanti nella Sezione 2.
- (9) Il progetto può essere consegnato in qualunque momento fino all'ultimo appello utile dell'anno accademico 2017/18. Correggerò e comunicherò i risultati alla prof.ssa Piazza. Metterò i risultati anche sulla mia pagina web.
- (10) Mi riservo il diritto di chiamare per un colloquio gli studenti per discutere dell'elaborato prima di decidere il voto finale.
- (11) Il progetto verrà valutato in una scala da 0 a 30. Il giudizio contribuirà al voto finale dell'esame di Algoritmi e Strutture Dati.
- (12) **È possibile fare l'esame teorico e il progetto di laboratorio in un ordine qualsiasi. Il voto del progetto vale due anni solari dalla data di consegna del progetto stesso. Se entro due anni solari l'esame non è stato registrato, il progetto è da rifare.**
- (13) Non mi occupo io della registrazione dell'esame. DOPO che avrete ottenuto il voto dello scritto (con eventuale orale) e il voto del laboratorio contattate la Prof.ssa Piazza per la registrazione con la seguente modalità: 1) iscrivetevi al prossimo appello utile 2) mandate una mail alla Prof.ssa Piazza con **i voti** che avete preso e **quando** li avete presi. I voti verranno controllati e sarà fatta la registrazione in occasione dell'appello. Non serve essere fisicamente presenti all'appello.
- (14) Il progetto è descritto qui di seguito nella Sezione 2.
- (15) Il codice da consegnare deve essere commentato in modo opportuno. (*Evitate gli estremi: né troppo, né troppo poco!*)
- (16) Anche se banale, ribadisco un concetto che dovrebbe essere chiaro a tutti: *non copiare dalle altre persone!* Ogni elaborato deve essere frutto del lavoro di una singola persona. Farò dei controlli e in caso valuti si sia copiato riterrò nulli *tutti* i progetti coinvolti.
- (17) Di seguito verrà descritto l'input e l'output. **Un programma che non acquisisce l'input in modo corretto o che non produce un output esattamente come previsto verrà valutato come non corretto.**

---

<sup>1</sup>No, C# non è ammesso.

<sup>2</sup>Si, lo so che ci sono già nella mail, ma per favore ripetete l'informazione!

- (18) Userò degli input (segreti) per controllare che il programma funzioni. Se il programma non dovesse restituire il risultato corretto per un numero sufficiente di test riterrò nullo l'elaborato (quindi fate attenzione all'output!).
- (19) Prima di consegnare l'elaborato controllate che non sia presente una nuova versione di questo file (confrontate le date) e **controllate che tutti i punti siano rispettati**. Controllate anche la sezione delle FAQ.

## 2. IL PROGETTO

**Definizione 1.** Data una stringa  $\alpha \in \Sigma^*$ , con  $\Sigma = \{a, \dots, z\}$ , indichiamo con  $T(\alpha)$  l'albero ottenuto come segue<sup>3</sup>:

- se  $\alpha = \epsilon$  ( $\epsilon$  è la stringa vuota),  $T(\epsilon) = \emptyset$ ;
- se  $\alpha \neq \epsilon$ ,
  - la radice di  $T(\alpha)$  è il nodo etichettato  $\alpha[0]$ ;
  - il sotto-albero di sinistra di  $T(\alpha)$  è  $T(\lambda)$ , con  $\lambda$  la sotto-stringa di  $\alpha[1, \dots, |\alpha| - 1]$  costituita dai caratteri lessicograficamente minori o uguali di  $\alpha[0]$  (cioè la stringa ottenuta da  $\alpha[1, \dots, |\alpha| - 1]$  cancellando tutti i caratteri lessicograficamente maggiori di  $\alpha[0]$ );
  - il sotto-albero di destra di  $T(\alpha)$  è  $T(\rho)$ , con  $\rho$  la sotto-stringa di  $\alpha[1, \dots, |\alpha| - 1]$  costituita dai caratteri lessicograficamente maggiori di  $\alpha[0]$  (cioè la stringa ottenuta da  $\alpha[1, \dots, |\alpha| - 1]$  cancellando tutti i caratteri lessicograficamente minori o uguali ad  $\alpha[0]$ ).

**Definizione 2.** Data una stringa  $\alpha$  e assunto che

$$m = \min_{s \in \Sigma} |\{i \mid \alpha[i] = s\}|,$$

$$M = \max_{s \in \Sigma} |\{i \mid \alpha[i] = s\}|,$$

costruire un grafo non orientato  $G = \langle V, E \rangle$  tale che:

- (1) la collezione dei vertici  $V$  del grafo  $G$  sono le foglie di  $T(\alpha)$ ;
- (2)  $(u, v) \in E$  se e solo se esiste un cammino in  $T(\alpha)$  da  $u$  a  $v$  e che ha peso  $p$  tale che  $p \in [m, M]$  (diciamo che il peso di un cammino è il numero dei suoi archi).

**Problema 1.**

**Input:** una stringa  $\alpha$ ;

**Output:**  $|V|, |E|$  ed il grafo  $G$  in formato dot.

**2.1. Specifiche per il formato di input e output.** L'input del programma sarà un testo inserito da linea di comando (standard input).

L'output dovrà venire scritto a terminale (standard output) e dovrà consistere:

- (1) da una riga contenente la coppia di interi  $|V|$  e  $|E|$  separati da uno spazio,
- (2) dal grafo, scritto a partire dalla seconda riga, in formato *dot* (vedi [http://en.wikipedia.org/wiki/DOT\\_\(graph\\_description\\_language\)](http://en.wikipedia.org/wiki/DOT_(graph_description_language))), i cui nodi siano etichettati con numeri interi consecutivi a partire da 0.

Il grafo in formato dot prodotto in output è necessario e sufficiente che risponda alle specifiche del formato dot.

Il programma verrà testato lanciandolo su una batteria di test mediante uno script analogo al seguente:

```
clear
path="/.../policriti/didattica/ASD/progetto2016-17/Progetti/test"
mkdir $path/$1
for i in {1..1000000000}
do
echo "run test on in-$i"
$2 $3 < $path/in-$i > $path/$1/out-$i
done
```

Dove il primo parametro ( $\$1$ ) sarà il nome dello studente mentre il secondo ed il terzo ( $\$2$  e  $\$3$ ) serviranno a lanciare il programma.

**Esempio 1.** Si consideri la stringa  $\alpha = \text{mississippi}$ .

L'albero  $T(\alpha)$  avrà radice etichettata  $m$  ed il sotto-albero di sinistra sarà  $T(\lambda)$  con  $\lambda = \text{iiii}$ , mentre il sotto-albero di destra sarà  $T(\rho)$ , dove  $\rho = \text{sssspp}$ . Proseguendo con la definizione si

<sup>3</sup>Sia  $\alpha = \alpha[0, \dots, |\alpha| - 1]$



## APPENDICE A. FAQ

- Q1** Ho un dubbio sul punto 3 delle regole: si riferisce al fatto di non usare algoritmi e strutture dati già pronti, cosa significa?
- A1** Significa che se per risolvere i problemi (ad esempio!) serve uno stack, voglio che il codice dello stack venga scritto da voi e che non si usi `java.util.Stack` o simile. Similmente sono banditi anche algoritmi già pronti come `java.util.Arrays.sort()`. Questo vale sia per oggetti presenti di default in Java, sia per oggetti che potete trovare in internet. Similmente per gli altri linguaggi di programmazione. **Dovete scrivere voi il codice!**
- Q2** In Java si possono utilizzare metodi del tipo `String.equals()` oppure `parseInt(String)` e le costanti del tipo `Integer.MAX_VALUE`?
- A2** Sono ammesse le funzioni per il parsing dell'input, per la formattazione dell'output e per la gestione dei file. E' ammesso anche l'uso delle costanti predefinite dei compilatori.
- Q3** Posso scrivere il codice usando le classi?
- A3** Siete liberi di programmare usando il paradigma orientato agli oggetti (classi, ereditarietà, interfacce, ...). Ovviamente dovete scrivere voi il codice, non usare quello pronto e scritto da altri!
- Q4** Dobbiamo eseguire la validazione dell'input e degli argomenti?
- A4** Potete dare per scontato che l'input e gli argomenti siano corretti, non serve fare nessuna validazione o controllo.
- Q5** Per misurare la durata dell'algoritmo, misuriamo il tempo utilizzato dal nostro main?
- A5** NO. Dovete predisporre due programmi. Il primo prende da input un file e produce in output la risoluzione del problema. Questo programma serve a me per verificare che gli algoritmi siano corretti. Il secondo programma serve per generare a caso gli input e misurare i tempi come spiegato a lezione e serve a voi per avere i tempi per i grafici. Ovviamente l'algoritmo di risoluzione dei due programmi deve essere lo stesso.
- Q6** Posso usare il multi-threading o qualche altro metodo di parallelizzazione?
- A6** No.
- Q7** Gli accenti o altri caratteri strani possono essere usati nei nomi delle variabili e/o nei commenti?
- A7** ARGH!!!! Mai usare accenti o caratteri strani nei nomi delle variabili. MAI! Questo perché ci sono problemi di interoperabilità fra diversi sistemi operativi! E' buona norma usare solo lettere (non accentate), numeri e l'underscore. Nei commenti potete usare qualsiasi carattere volete, perché saranno segnalati come warning e non compromettono la compilazione. **Se la presenza di accenti non mi permetterà di compilare il programma, l'elaborato risulterà nullo.**
- Q8** Posso usare i generatori di numeri casuali forniti dai compilatori?
- A8** No, per uniformità usate il generatore proposto nelle dispense (Algoritmo 8).
- Q9** Qual è *esattamente* l'alfabeto  $\Sigma$  ?
- A9**  $\Sigma = \{a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z\}$ .