PROGETTO PER IL LABORATORIO DI ASD
A.A. 2015/16
— VERSIONE 1.2 —

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 verrà valutato sia in termini di efficienza che di eleganza della soluzione proposta.

INDICE

1. Regole generali

(1) Il progetto deve essere svolto singolarmente.
(2) I linguaggi ammessi per i programmi sono: C/C++, Java, Pascal/Delphi[1]. 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[2].
(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 l'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 che eseguirà un loop della seguente linea di comando:
```
prompt$ eseguibile-progetto < mio-test-numero-n > output-numero-n
```
Una piccola batteria di test con il relativo script per il controllo verrà resa disponibile.
(9) Il progetto può essere consegnato in qualunque momento ma correggerò con le seguennti scadenze:
- 30/06/2016 - per la sessione di giugno/luglio.
- 01/09/2016 - per la sessione di settembre.
- 01/12/2016 - per la sessione straordinaria-

I risultati verranno resi noti tramite la 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 laboratoro 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!*)

[1] No, C# non è ammesso.
[2] Si, lo so che ci sono già nella mail, ma per favore ripetete l'informazione!

(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 funzionante**.
(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. Progetto di ASD A.A. 2015/16

Definizione 1. *Dato un grafo orientato $G = (V, E)$ diciamo che* ammette una radice *se e solo se esiste un nodo $r \in V$ tale che ogni $v \in V$ sia raggiungibile da r in G.*

Problema 1. *Dato in input un grafo orientato G,*

1. *si determini il numero minimo di archi da aggiungere a G in modo tale da ottenere $G' = (V, E')$ che ammetta una radice r;*
2. *si produca un albero $T = (V, E_T)$ dei cammini minimi in G' di radice r.*

2.1. Specifiche per il formato di input e output.

L'input del programma sarà dato in formato dot (vedi `http://en.wikipedia.org/wiki/DOT_(graph_description_language)`) e l'output dovrà essere fornito nello stesso linguaggio.

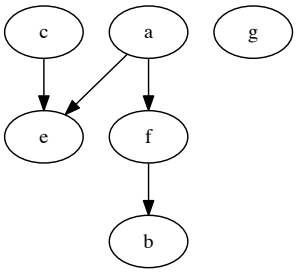
L'output dovrà essere un grafo $G' = (V, E')$ che rappresenterà contemporaneamente sia G' che T secondo le seguenti direttive:

1. G' deve ammettere una radice r etichettata con la parola *root* e con il numero naturale $|E'| - |E|$ (che deve essere il sia minimo possibile);
2. gli archi in $E' \setminus E$ devono essere colorati di rosso;
3. T deve essere un albero di radice r;
4. ogni nodo di T deve essere etichettato con la sua distanza dalla radice;
5. ogni arco di T deve essere tratteggiato.

Esempio 1.

```
digraph G1 {
c -> e;
a -> e;
a -> f;
f -> b;
g;
}
```

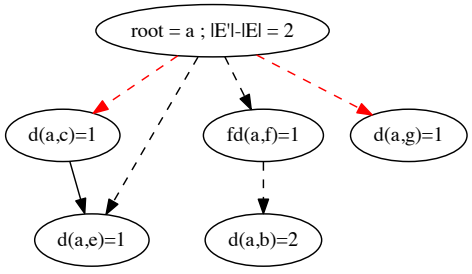
(A) Formato .dot per il grafo in input



(B) Visualizzazione

```
digraph out_G1 {
a -> c [style=dashed,color=red];
a -> e [style=dashed];
a [label="root = a ; |E'|-|E| = 2"];
e [label="d(a,e)=1"];
c [label="d(a,c)=1"];
a ->f [style=dashed];
f->b [style=dashed];
f [label="fd(a,f)=1"];
b [label="d(a,b)=2"];
a -> g [color=red; style=dashed];
c -> e;
g [label="d(a,g)=1"];
}
```

(A) Formato .dot per il grafo in output



(B) Visualizzazione

Il formato che utilizzerò per l'input sarà analogo a quello dell'esempio: parola chiave `digraph` seguita dal nome (arbitrario) del grafo, seguito da parentesi graffa aperta, seguita da un arco o un nodo per ogni riga (numero arbitrario di caratteri *blank* ovunque) e da una partentesi graffa chiusa. I nomi (etichette) dei nodi saranno solo stringhe di uno o più caratteri alfanumerici.

Il formato per l'output è sufficiente che risponda alle specifiche del formato dot.

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

```
clear
path="/.../policriti/didattica/ASD/progetto2015-16/Progetti/test"
mkdir $path/$1
for i in {1..1000000000}
do
echo "run test on in-$i.dot"
$2 $3 < $path/in-$i.dot > $path/$1/out-$i.dot
done
```

Dove il primo parametro sarà il nome dello studente mentre il secondo ed il terzo serviranno a lanciare il programma.

3. Programmi

Bisogna presentare due programmi basati sullo stesso codice.

Il primo programma è quello che risolve il problema: deve prendere i dati dallo standard input e scrivere il risultato sullo standard output, risolvendo il problema a seconda del numero di parametri della prima riga. Questo programma serve a **me** (ma anche a **voi**) per controllare che gli algoritmi scritti siano corretti. Userò il programma su dei dati (segreti) di cui conosco il risultato per valutare la correttezza. Non è ammesso l'uso di programmi interattivi o di interfacce grafiche.

Il secondo programma serve a voi per la misurazione dei tempi per i grafici: dovrà produrre dei dati casuali e misurarli secondo le specifiche degli appunti delle lezioni, come specificato di seguito.

3.1. **Metodologie di analisi.** Presentare una relazione in cui viene enunciato il problema[3] e spiegata la soluzione proposta. Fare uno studio di complessità degli algoritmi coinvolti. Misurare il tempo al variare della dimensione dei dati in input e confrontare (commentando) le eventuali differenze rispetto all'analisi teorica. Includere nella relazione i tempi e i grafici ottenuti. **Per il calcolo del numero di simulazioni bisogna rifarsi alla teoria spiegata negli appunti delle lezioni** e che potete trovare nella home page del corso. In particolare dovete utilizzare **l'algoritmo 9 degli appunti**. Misurazioni ottenute con metodi diversi da quelli descritti non verranno valutate.

Includere nella relazione tutte le osservazioni che vengono ritenute opportune ed eventuali problemi riscontrati.

[3]NON copiate pedissequamente la MIA spiegazione: fatemi capire che avete compreso il problema!

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 Dobbiamo verificare che non ci siano archi ripetuti?

A5 Come per la risposta precedente, potete dare per scontato che l'input e gli argomenti siano corretti e quindi non serve nemmeno verificare che non ci siano ripetizioni di archi nell'input.

Q6 Per misurare la durata dell'algoritmo, misuriamo il tempo utilizzato dal nostro main?

A6 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.

Q7 Posso usare il multi-threading o qualche altro metodo di parallelizzazione?

A7 No.

Q8 Gli accenti o altri caratteri strani possono essere usati nei nomi delle variabili e/o nei commenti?

A8 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.**

Q9 Dobbiamo testare noi se il grafo in input è ciclico?

A9 Si.

Q10 Il grafo in input è necessariamente connesso? Nel caso non lo sia e una delle componenti sia ciclica, dobbiamo considerarlo ciclico?

A10 No. Si.

Q11 Posso usare i generatori di numeri casuali forniti dai compilatori?

A11 No, usate il generatore proposto nelle dispense (Algoritmo 8).

Q12 Posso assumere che entrambi i grafi in input abbiano almeno un arco?
A12 Si

Q13 Posso assumere che le versioni non orientate dei grafi in input abbiano una sola componente connessa?
A13 No

Q14 L'input sarà rediretto da un file?
A14 Questa è la modalità più naturale. Eseguirò i test redigendo l'input da un (alto) numero di file.