Prova Finale di Algoritmi e Strutture Dati

A.A. 2020/2021

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 (strette e vincolanti):
    - 4 luglio (ore 23.59) per i laureandi di luglio
    - 7 settembre (ore 23.59) per tutti gli altri
    - piattaforma riaperta tra il 31 gennaio e 11 febbraio 2022 (fino alle 23.59) per i SOLI laureandi di febbraio
      - Occorre essere iscritti all'appello di laurea perché il voto sia registrato
  - Tenete conto dei vincoli temporali per gestirvi!
    - Es. "inizio a pensarci a settembre" = fallimento quasi certo

Esecuzione del progetto

- Esecuzione del progetto
  - implementazione nel linguaggio C (C11, VLA ammessi)
  - esclusivamente con libreria standard (libc), niente librerie esterne
  - no multithreading o tecniche di parallelizzazione
  - dati in ingresso ricevuti via stdin, risultati da fornire via stdout

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
  - verrà fornito anche uno strumento di generazione automatica di casi ti test, per facilitarvi il testing in locale
- 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
  - il voto è assegnato in modo **automatico** in base a come vanno i test
- non c'è recupero, ma il numero di "appelli" (= sottomissioni) è praticamente illimitato

Sito per la sottomissione del progetto

- https://dum-e.deib.polimi.it/
- Ogni studente ha le proprie credenziali per accedere al sito

Operazioni che si possono fare sul sito

- scaricare le specifiche del progetto
- caricare (*submit / sottoponi*) il file da compilare e lanciare
  - si può usare il nome del file che si vuole, viene rinominato in automatico
  - dopo il caricamento, in automatico il file viene compilato e testato su un insieme di test “pubblici”
- lanciare (*play / usa*) i test “privati”, quelli su cui viene valutato il progetto

Gestione del codice

- Vi è **richiesto** di condividere il codice con il docente tramite GitHub (www.github.com)
- A questo fine occorre:
  - Registrarsi su github, usando l'email del Politecnico
    - l'email del Politecnico dà la possibilità di creare repository privati gratis
    - guardate questo link: https://education.github.com/pack
  - Creare un repository **privato**, chiamandolo "PFAPI21_<cognome>_<idnumber>"
  - "Invitare" il docente a condividere il repository
    - l'id del docente è matteo-g-rossi
- Se già non avete dimestichezza con git, online ci sono diversi tutorial
  - ma le operazioni che dovete fare voi per questo progetto sono molto semplici, di fatto solo delle "push" di codice sul repository

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**
  2. Non passarlo per "ispirazione" a colleghi
  3. Non utilizzate frammenti di codice reperito online
- controllo plagi automatizzato
- in caso di copiatura **tutti** i progetti coinvolti vengono **annullati**

Scadenze

- Martedì 7 settembre, ore 23.59
  - poi il sito per la sottomissione verrà chiuso

- Per i laureandi di luglio: domenica 4 luglio, ore 23.59
  - avvisate (mandando un email al docente) che volete avere la valutazione a luglio

- Per i laureandi di febbraio 2022: riapertura del sito da lunedì 31 gennaio, fino a venerdì 11 febbraio, ore 23.59
  - avvisate (mandando un email al docente) che volete avere la valutazione a febbraio
  - Perché il voto (se positivo) sia registrato, occorre essere iscritti all'appello di laurea di febbraio

Tutoraggio

- Possibili interazioni:
  - BeeP (**specifico per la Prova Finale**!)
    - Forum su BeeP: https://bit.ly/3hPqZ0T
  - Chat di Teams
    - Specifica per la Prova Finale: https://bit.ly/3wzUwj1
  - Email (per esempio per prendere appuntamento)
  - Tutoraggio in presenza (vedi slide successiva)

Tutoraggio in presenza

- Luogo: ufficio docente (ufficio 20, secondo piano)
- Quando: tipicamente in corrispondenza ad esami:
  - venerdì 4/6, durante orario di lezione
  - venerdì 11/6, dopo esame (fin che serve)
  - martedì 29/6, dopo esame (fin che serve)
  - se necessario, una data a luglio (chiedere)
  - mercoledì 25/8, dopo esame (fin che serve)
- Siete invitati a mandare un messaggio per avvisare che avete bisogno di un incontro

GraphRanker

- L'obiettivo del progetto di quest'anno è la gestione di una classifica tra grafi diretti pesati
  - La classifica tiene traccia dei k "migliori" grafi
- Il programma da realizzare riceve in ingresso
  - due parametri, una sola volta (sulla prima riga del file, separati da spazio)
    - d: il numero di nodi dei grafi
    - k: la lunghezza della classifica
  - Una sequenza di comandi tra
    - `AggiungiGrafo [matrice-di-adiacenza]`
    - `TopK`

d, k e il numero di grafi sono rappresentabili con interi a 32 bit.

AggiungiGrafo

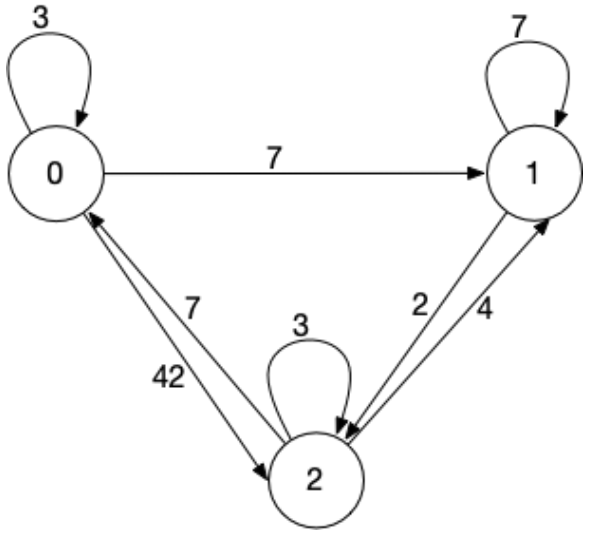
Richiede di aggiungere un grafo a quelli considerati per stilare la classifica. È seguito dalla matrice di adiacenza del grafo stesso, stampata una riga per ogni rigo, con gli elementi separati da virgole.

I nodi del grafo sono da considerarsi etichettati logicamente con un indice intero tra 0 e d-1; il nodo in posizione 0 è quello la cui stella uscente è descritta dalla prima riga della matrice.

I pesi degli archi del grafo elementi sono interi nell'intervallo $[0, 2^{32} – 1]$.

- Esempio per d=3

```
AggiungiGrafo
3,7,42
0,7,2
7,4,3
```



TopK

- Si consideri ogni grafo dall'inizio del programma fino al comando TopK etichettato con un indice intero corrispondente al numero di grafi letti prima di esso (partendo da 0)
- TopK richiede al programma di stampare gli indici interi dei k grafi aventi i k valori più piccoli della seguente metrica:
  - Somma dei cammini più brevi tra il nodo 0 e tutti gli altri nodi del grafo raggiungibili da 0
- Se ci sono più grafi con lo stesso valore della metrica, si dà la precedenza ai primi arrivati
- Le distanze dei nodi non raggiungibili da 0 sono considerate nulle
- I k indici interi sono stampati, su un unico rigo, separati da uno spazio, in un qualunque ordine

Un'esecuzione d'esempio

Input ricevuto

```
3,2
AggiungiGrafo
0,4,3
0,2,0
2,0,0
AggiungiGrafo
0,0,2
7,0,4
0,1,0
AggiungiGrafo
3,1,8
0,0,5
0,9,0
TopK
```

Commenti e Output Atteso

Si richiede di manipolare grafi da 3 nodi e riportare i k=2 migliori
Aggiunta del primo grafo (indice 0, somma cammini = 7)

Aggiunta del secondo grafo (indice 1, somma cammini = 5)

Aggiunta del terzo grafo (indice 2, somma cammini = 7)

0 1 Oppure 1 0

Breve tutorial del sito

Overview

- “Tutorial” è un semplicissimo problema, creato al solo scopo di sperimentare il funzionamento del sito
- "Open" è il task con i test pubblici, e con il codice per la generazione automatica dei test
- Gli altri sono i problemi da risolvere (i test che **devono** essere eseguiti con successo)

Task overview

Task	Name	Time limit	Memory limit	Type	Files	Tokens
Tutorial	Tutorial	N/A	N/A	Batch	Tutorial[.c]	Yes
Open	Open	N/A	N/A	Batch	Open[.c]	Yes
UpTo18	UpTo18	2.400 seconds	90.0 MiB	Batch	UpTo18[.c]	Yes
UpTo30	UpTo30	2.080 seconds	82.0 MiB	Batch	UpTo30[.c]	Yes
CumLaude	CumLaude	1.700 seconds	1.00 MiB	Batch	CumLaude[.c]	Yes

“Tutorial”

- Problema: leggere da standard input 2 numeri interi separati da uno spazio; produrre su standard output la loro somma

Some details

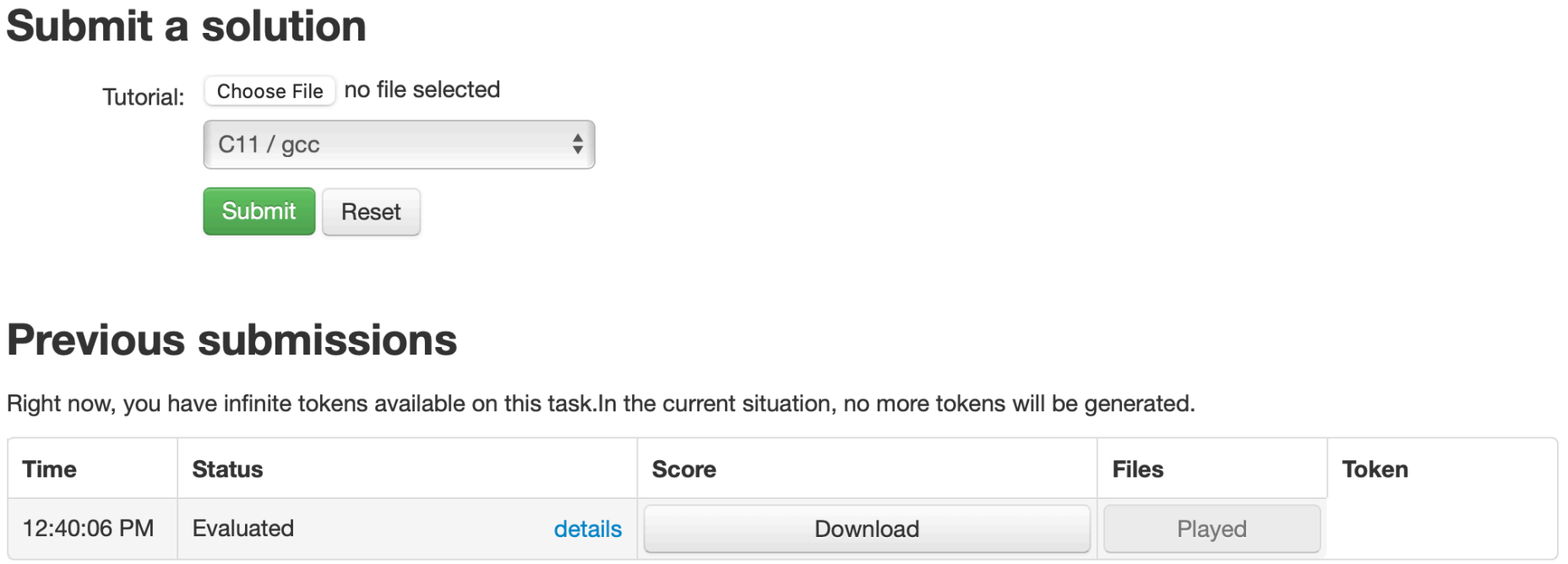
Type	Batch	
Compilation commands	C11 / gcc	`/usr/bin/gcc -DEVAL -std=gnu11 -O2 -pipe -static -s -o Tutorial Tutorial.c -lm`
Tokens	You have an infinite number of tokens.	

Passi da fare

- Step 1: creare il file `Tutorial.c`
- Step 2: compilare il file con la riga di comando (presa dal sito)
  `/usr/bin/gcc -DEVAL -std=gnu11 -O2 -pipe -static -s -o Tutorial Tutorial.c -lm`
- Step 3: preparate un file di test, per esempio `Tutorial_test.in`, con una riga di testo, in cui ci sono 2 interi separati da uno spazio; preparate anche un file `Tutorial_test.out` con il risultato atteso della computazione (una riga con il risultato della somma)
- Step 4: lanciate il comando
  `cat Tutorial_test.in | ./Tutorial > Tutorial_test.res`
- crea un file, `Tutorial_test.res`, con il risultato della computazione

Passi da fare (cont.)

- Step 5: confrontate il risultato atteso con quello ottenuto mediante il comando `diff Tutorial_test.out Tutorial_test.res`
  - se non ci sono differenze il risultato è la stringa vuota
  - il server usa il comando *wdiff* invece del comando *diff*
- Step 6: caricate ("sottoponi") il file `Tutorial.c` sul sito, e lanciate ("usa") i test



Possibili strumenti utili

- valgrind (memory debugging, profiling)
  - http://valgrind.org
- gdb (debugging)
  - https://www.gnu.org/software/gdb/
- AddessSanitizer (ASan, memory error detector)
  - https://github.com/google/sanitizers/wiki/AddressSanitizer

Calcolo del voto

- Il task di tutorial non contribuisce in alcun modo all'esito della prova
- La valutazione è immediatamente calcolata (e subito visibile), mediante 3 batterie di test (task):
  - il primo task, *UpTo18*, vale 18 punti (**pass** or **fail**)
  - il secondo task, *UpTo30*, vale fino a 12 (6 test da 2 punti ognuno)
  - l'ultimo task, *CumLaude*, vale 1 punto per la lode
- Il punteggio massimo ottenibile è 30 punti + 1 punto extra per la lode ottenibile superando il task *CumLaude*
  - il punteggio è dato dalla somma dei punti dei test superati dei primi 2 task
  - se tutti i test sono superati, inclusi i test per la lode, 30 e lode
    - Il punto della lode viene assegnato solo se tutti gli altri test sono superarti

Avvertenze

- Nessun limite al numero di sottoposizioni, né penalità per sottoposizioni multiple
- È possibile migliorare la valutazione quante volte si desidera
- Viene valutata **l'ultima** sottoposizione fatta ad ogni batteria di test
- Tutte le sottoposizioni valutate devono utilizzare **lo stesso sorgente**
  - se siete in dubbio, ri-sottoponete lo stesso sorgente a tutte le batterie di test per buona misura