ASD Laboratorio 10

Antonio Bucchiarone/Cristian Consonni/
Francesco Lotito/Giovanni Zotta

UniTN

2021-05-13

Calendario

18/03	Programmazione dinamica
22/04	Programmazione dinamica
29/04	Ricevimento (facoltativo)
13/05	Algoritmi approssimati
27/05	Progetto alg approssimati
01/06	Progetto alg approssimati

Progetto algoritmi approssimati

- Assegnazione progetto l'**26/05/2021** e consegna il **04/06/2021**;
- Algoritmi approssimati (ultima parte del corso);
- Assumiamo gli stessi gruppi del primo semestre, in caso di cambiamenti, avvisare **entro il 24/05/2021**;

Approssimazione in Judge

- Non abbiamo una soluzione ottima!
- Pertanto è impossibile raggiungere 100 punti
- La vostra soluzione confrontata con lower bound

Soluzioni possibili:

- Soluzione "greedy"
- Soluzione esponenziale (es: branch and bound)

Problema del commesso viaggiatore

Dato un grafo non-orientato, pesato e completo, trovare percorso minimo che parte dal nodo X, visita tutti i nodi e torna in X.

- **Soluzione greedy**: muoviti sempre verso il nodo più vicino non visitato.
- **Soluzione branch and bound**: soluzione ricorsiva con taglio grazie a lower bound.

COME FUNZIONA

Soluzioni approssimate:

- Importate `tsp.h` (scaricabile da judge);
- Man mano che migliorate la soluzione, scrivetela in output terminando la riga con `#`;
- La libreria arresterà il programma prima del timeout.

```
... include delle librerie di sistema ...
#include "tsp.h"

int main() {
  ...
}
```

Note:

- il `main` va sempre dichiarato come `int main()` o `int main(void)`
- Il correttore considererà l'ultima riga di output che finisce con `#` quindi, anche se non appendete soluzioni multiple, terminate l'output con `#`.

Compilare in locale

Per testare le vostre soluzioni in locale (supponiamo che il vostro file si chiami `tsp.cpp`):

- Scaricate `grader.cpp`
- Il comando di compilazione è il seguente

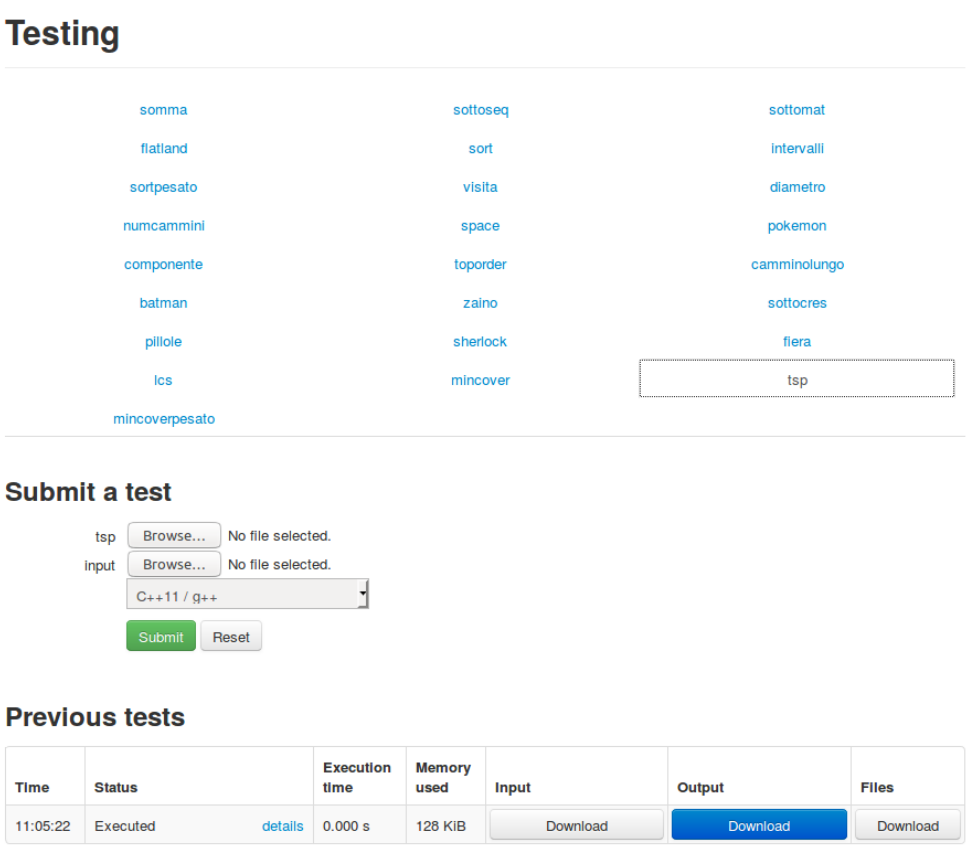
```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o
    tsp grader.cpp tsp.cpp
```

I file `tsp.cpp`, `grader.cpp` e `tsp.h` devono essere nella stessa cartella.

Per sistemi Mac OS X vedere la nota nel testo.
Nota: Per questo esercizio è necessario usare il `C++`, non è possibile usare il `C`.

Usare l'interfaccia di test di CMS

Potete testare le vostre soluzioni su CMS usando l'interfaccia di test:



- `TSP`: sorgente (`tsp.cpp`)
- `INPUT`: input come da specifica

Nota: Dopo aver caricato i file la pagina viene ricaricata nell'interfaccia generale di test.

Progetto dell'anno 2017/2018 (I)

Alpinocalypse Now (alpini)

Progetto algoritmi approssimati a.a. 2017/2018

```
... include delle librerie di sistema ...

#include "alpini.h"
int main() {
  ...
}
```

Note:

- il `main` va sempre dichiarato come `int main()` o `int main(void)`
- Anche per questo esercizio è necessario usare il `C++`, non è possibile usare il `C`.

Progetto dell'anno 2017/2018 (II)

Alpinocalypse Now (alpini)

Progetto algoritmi approssimati a.a. 2017/2018

Il comando di compilazione è il seguente:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o
    alpini grader.cpp alpini.cpp
```

I file `alpini.cpp`, `grader.cpp` e `alpini.h` devono essere nella stessa cartella.
Per sistemi Mac OS X vedere la nota nel testo.

Nota: Il correttore considererà l'ultima riga di output che finisce con `#` quindi, anche se non appendete soluzioni multiple, terminate l'output con `#`.

Progetto dell'anno 2018/2019 (I)

Game of (Approximated) Thrones (got)

Progetto algoritmi approssimati a.a. 2018/2019

```
... include delle librerie di sistema ...
#include "got.h"
int main() {
  ...
}
```

Note:

- Il `main` va sempre dichiarato come `int main()` o `int main(void)`.
- Anche per questo esercizio è necessario usare il `C++`, non è possibile usare il `C`.

Progetto dell'anno 2018/2019 (II)

Game of (Approximated) Thrones (got)

Progetto algoritmi approssimati a.a. 2018/2019

Per testare le vostre soluzioni in il comando di compilazione è il seguente:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o
    got grader.cpp got.cpp
```

I file `got.cpp`, `grader.cpp` e `got.h` devono essere nella stessa cartella.
Per sistemi Mac OS X e Windows vedere la nota nel testo.

Nota: Il correttore considererà l'ultima riga di output che finisce con $***$ quindi, anche se non stampate soluzioni multiple, terminate l'output con $***$.

Progetto dell'anno 2019/2020 (I)

Star Wars: Il Risveglio dell'Algoritmo (swrace)

Progetto algoritmi approssimati a.a. 2019/2020

```
... include delle librerie di sistema ...
#include "swrace.h"
int main() {
  ...
}
```

Note:

- Il `main` va sempre dichiarato come `int main()` o `int main(void)`.
- Anche per questo esercizio è necessario usare il `C++`, non è possibile usare il `C`.

Progetto dell'anno 2019/2020 (II)

Star Wars: Il Risveglio dell'Algoritmo (swrace)

Progetto algoritmi approssimati a.a. 2019/2020

Per testare le vostre soluzioni in il comando di compilazione è il seguente:

```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o
    swrace grader.cpp swrace.cpp
```

I file `swrace.cpp`, `grader.cpp` e `swrace.h` devono essere nella stessa cartella.
Per sistemi Mac OS X e Windows vedere la nota nel testo.

Nota: Il correttore considererà l'ultima riga di output che finisce con `#` quindi, anche se non appendete soluzioni multiple, terminate l'output con `#`.