

ASD Laboratorio 10

Cristian Consonni/Marta Fornasier

UniTN

2019-05-06

04/03	Programmazione dinamica
06/03	Programmazione dinamica
10/04	Ricevimento (facoltativo)
06/05	Algoritmi approssimati
22/05	Progetto alg approssimati
29/05	Progetto alg approssimati

PROGETTO ALGORITMI APPROSSIMATI

- Algoritmi approssimati (ultima parte del corso);
- Assumiamo gli stessi gruppi del primo semestre, in caso di cambiamenti, avvisare **entro il 19/05**;

- 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)

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.

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 `#`.

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:

Testing

somma	sofseq	sofomat
flatland	sort	intervall
sortpesato	vista	diametro
numcammini	space	pokemon
componente	toporder	camminolungo
batman	zaino	sotlocres
pileole	sherlock	fiera
ics	mincover	tip
mincoverpesato		

Submit a test

tip No file selected.

input No file selected.

Previous tests

Time	Status	Execution time	Memory used	Input	Output	Files
11:05:22	Executed	details 0.000 s	128 KiB	<input type="button" value="Download"/>	<input type="button" value="Download"/>	<input type="button" value="Download"/>

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

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

ALPINOCLYPSE NOW (ALPINI)

3° progetto a. a. 2017/2018

Nota: Contrariamente alle istruzioni originali, non è necessario chiamare `helpers::setup()` all'inizio del `main`. È sufficiente includere `alpini.h`.

```
... 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.

ALPINO CALYPSE NOW (ALPINI)

3° progetto 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 #.