# *ASD Laboratorio 09*

The A(SD)-Team

UniTN

2024-05-06

# Calendario

| | |
|---|---|
| 27/02 | Programmazione dinamica |
| 03/04 | Programmazione dinamica |
| 06/05 | Algoritmi approssimati 1 |
| 15/05 | Algoritmi approssimati 2 |
| 22/05 | Presentazione Progetto 2 (no lab) |
| 27/05 | Progetto alg approssimati |
| 29/05 | Progetto alg approssimati |

# Secondo progetto

## Progetto algoritmi approssimati

- Il secondo progetto verrà assegnato il **22/05/2025** e avrete circa una settimana di tempo. Il progetto riguarderà gli algoritmi approssimati (ultima parte del corso);
- Per iscriversi (se non vi siete iscritti al primo progetto): `https://bit.ly/ASDprog_2024-2025` (dovete essere loggati con l'account UniTN)
- Assumiamo gli stessi gruppi del primo semestre, in caso di cambiamenti, avvisare **entro il 19/05/2025**;

# Sottosequenza comune massimale

Algoritmo sulle slide del prof. Montresor

## Sottoproblema

$S[i, j] =$ soluzione per stringhe $A[1..i]$ e $B[1..j]$

$$S[i,j] = \begin{cases} 0, & i = 0 \text{ or } j = 0 \\ S[i-1, j-1], & A[i] = B[j] \\ max(S[i-1, j], S[i, j-1]), & A[i] \neq B[j] \end{cases}$$

# Node Cover su Albero non pesato

## Sottoproblema

$S[i]$ = soluzione del sottoalbero radicato in $i$ con la scelta di $i$ obbligata.
$L[i]$ = soluzione del sottoalbero radicato in $i$ con la scelta di $i$ libera.

$$S[i] = 1 + \sum_{f \in V(i)} L[f]$$

$$L[i] = min(S[i], \sum_{f \in V(i)} S[f])$$

# Node Cover su Albero pesato

## Sottoproblema

$S[i]$ = soluzione del sottoalbero radicato in $i$ con la scelta di $i$ obbligata.
$L[i]$ = soluzione del sottoalbero radicato in $i$ con la scelta di $i$ libera.

$$S[i] = Peso[i] + \sum_{f \in V(i)} L[f]$$

$$L[i] = min(S[i], \sum_{f \in V(i)} S[f])$$

# 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

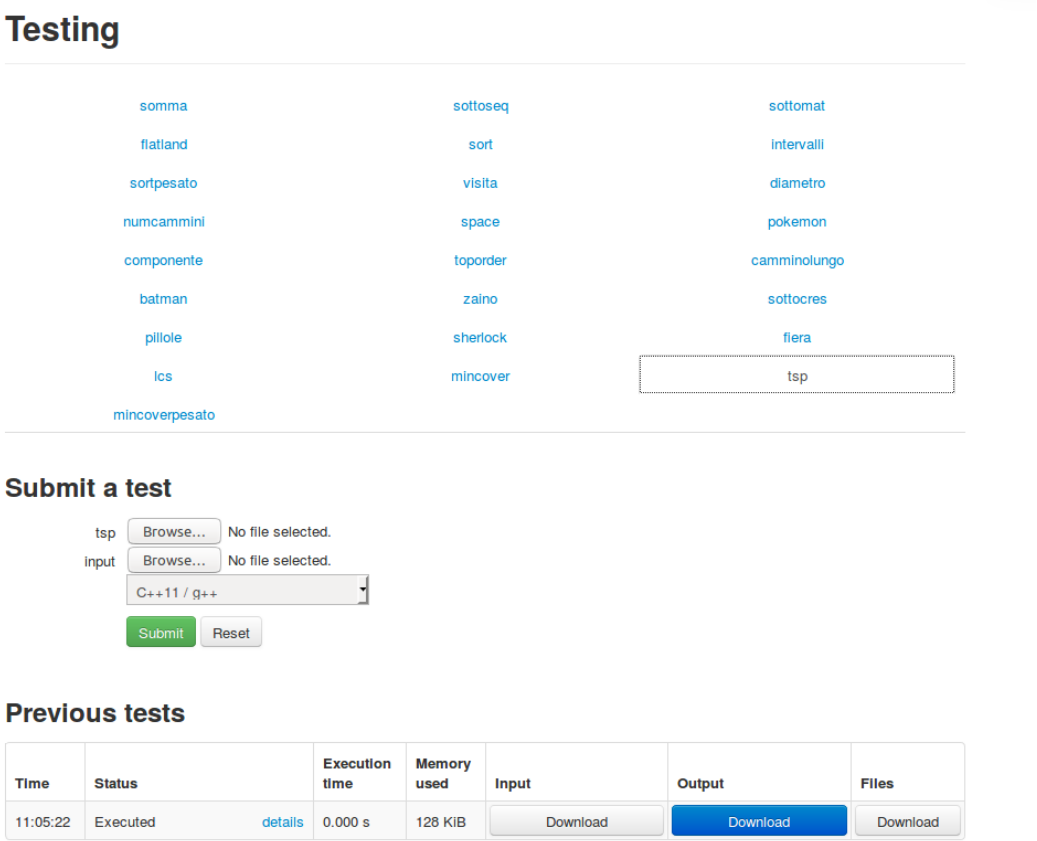
```
/usr/bin/g++ -DEVAL -std=c++11 -O2 -pipe -static -s -o
    tsp grader.cpp tsp.cpp
```

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 `#`.

# Progetto dell'anno 2023/2024 (II)

## L'uncinetto di nonna Alberta (swrace)

Progetto algoritmi approssimati a.a. 2023/2024

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
    nonna grader.cpp nonna.cpp
```

I file `nonna.cpp`, `grader.cpp` e `nonna.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 `***`.