Algoritmi e Strutture Dati

Commesso Viaggiatore (`tsp`)

Testo del problema

Dato un grafo completo pesato non orientato di N nodi, trovare il percorso minimo che partendo dal nodo i, visiti tutti i nodi del grafo.

Formato dell'input

La prima linea contiene il numero di nodi N. Le $N-1$ linee successive contengono i pesi degli archi: La prima riga contiene il peso dell'arco da 1 a 0; la seconda riga contiene i pesi degli archi da 2 a 0 e 1, la terza riga contiene i pesi degli archi da 3 a 0, 1 e 2, ecc. .

Formato dell'output

L'output può contenenere varie soluzioni, ognuna descritta da una riga terminata dal simbolo `#`. Il correttore prenderà **l'ultima riga terminata da `#`** come soluzione da valutare. La soluzione è costituita da $N+1$ interi, ovvero il percorso di nodi da visitare.

Assunzioni

- $3 < N \leq 50$

Istruzioni per l'output

Se scrivete una soluzione esponenziale (tipo branch and bound) 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.
Il main va sempre dichiarato come `int main()` o `int main(void)`. Questo esercizio deve essere svolto in C++, non è possibile usare il C.

```
# include "tsp.h"

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

Supponendo che il sorgente con il vostro codice si chiami file `tsp.cpp`, i file `tsp.cpp`, `grader.cpp` e `tsp.h` devono essere nella stessa cartella e vegono compilati con il seguente comando:

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

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

Esempi di input/output

File `input.txt`	File `output.txt`

`4`
`1`
`1 3`
`2 4 1`
	`0 1 3 2 0#`