DI PALMA MATTIA 0124002448.

TRACCIA 1.

QUESITO 2.

DESCRIZIONE DEL PROBLEMA:

Il problema posto nel quesito 2 è quello di calcolare il percorso che massimizza la soddisfazione dei turisti che visitano l'arcipelago partendo da una generica isola verso tutte le altre.

L'arcipelago in questo caso viene visto come un grafo dove le "isole" sono dei generici **NODI** e i "ponti" ovvero i collegamenti tra le isole rappresentano gli **ARCHI** appartenenti al grafo.

Il grafo come scritto nella traccia è orientato e aciclico con la "qualità" del collegamento che rappresenta il **PESO** dell'arco che può essere anche una quantità negativa.

L'output del problema consiste nella ricerca di un cammino massimo all'interno del grafo per massimizzare la soddisfazione del turista e completare la visita di tutte le isole partendo da una qualsiasi.

Il grafo essendo aciclico ha una sottostruttura ottima e quindi non è un problema **NP-HARD.**

Dato un grafo orientato, pesato e aciclico **G (V, E)** ed una funzione peso **w: E -> R**, sia $p = (v_1, \ldots, v_h)$ un cammino massimo dal vertice v_1 al vertice v_h $\forall$ i,j tale che $1 \leq i \leq j \leq h$, considerato $p_{i,j}$ un sottocammino di p tra vi e vj, allora pi,j è un cammino massimo tra v_i e v_j .

DIMOSTRAZIONE:

IPOTESI:

Supponiamo che $\mathbf{p_{i,j}}$ **sia un cammino massimo** di un grafo tra il vertice v_i e v_j e dividiamo $p_{i,j}$ in due sottocammini $p_{i,k}$ e $p_{k+1,j}$ compresi rispettivamente tra i vertici v_i e v_k e v_{k+1} e v_j. . Per la sottostruttura ottima quindi, se $p_{i,j}$ è un cammino massimo allora $p_{i,k}$ e $p_{k+1,j}$ devono essere cammini massimi.

PER ASSURDO:

Neghiamo l'affermazione dell'ipotesi asserendo che $p_{i,k}$ non sia il cammino massimo tra i vertici v_i e v_k per cui esisterà un cammino $*p_{i,k}$ con un peso maggiore di $p_{i,k}$.
Aggiungiamo adesso $*p_{i,k}$ a $p_{k+1,j}$ creando un nuovo cammino massimo $*p_{i,j}$ con un peso maggiore rispetto a $p_{i,j}$, ma questo contraddice l'ipotesi fatta in precedenza e quindi porta ad un ASSURDO.

DESCRIZIONE STRUTTURA DATI:

La struttura dati utilizzata per lo svolgimento del problema è un grafo **G**, ovvero una coppia ordinata di **V** (insieme dei nodi) ed **E** (insieme degli archi).
Gli elementi dell'insieme E sono coppie di elementi (u, v) di nodi che appartengono a V, possiamo affermare quindi che E è un sottoinsieme proprio di V x V.
Il grafo trattato nella traccia è orientato e pesato dove gli archi sono coppie ordinate (u, v) di nodi appartenenti a V.
La traccia afferma inoltre che in input vengono specificati la direzione ed i valori della qualità dei collegamenti per ogni isola che possono essere anche negativi.

DATI INPUT/OUTPUT:

L'input del programma come descritto nella traccia è rappresentato da un file di testo contenente nella prima riga due interi separati da uno spazio in cui il primo intero rappresenta **N** ovvero il numero di isole (NODI) mentre il secondo intero rappresenta il numero di ponti **P** (ARCHI).
I successivi P righi contengono ciascuna tre numeri Isola1, Isola2 e Q dove Q è la qualità del collegamento.
Sappiamo inoltre che $2 \leq N \leq 1000$ e $1 \leq P \leq 10000$.
L'input dell'isola da cui partire per la visita è deciso dall'utente tramite una scelta fatta digitando dalla tastiera.
L'output è rappresentato da un grafo diretto e aciclico con il percorso massimo che viene visualizzato a video partendo da una generica isola S (nodo Sorgente) scelta dall'utente.

DESCRIZIONE ALGORITMO:

Il problema gode della sottostruttura ottima, questo ci garantisce che, calcolato il cammino massimo tra la sorgente s ed un nodo u allora i cammini calcolati s ed i predecessori di u saranno sicuramente massimi.
Il grafo G come descritto nella traccia è un **DAG (**Direct Acyclic Graph) quindi possiamo realizzare un ordinamento topologico ovvero un ordinamento lineare di tutti i suoi vertici tale che se G contiene un arco (u, v) allora u appare prima di v nell'ordinamento così da calcolare tutti i percorsi che dal nodo sorgente s portano a v.
L'ordinamento topologico viene effettuato tramite l'algoritmo **DFS**, ovvero una visita in profondità con l'uso di una struttura dati stack per memorizzare l'ordine dei vertici.
Prima di richiamare la DFS per l'ordinamento topologico utilizziamo la funzione **Inizialized_single_source** per inizializzare la distanza della sorgente da sé stessa a 0, le distanze degli altri nodi a meno infinito e i padri a NIL per modificarli in seguito con il calcolo del percorso massimo.
Dopo aver fatto tutto questo ed aver memorizzato nello stack i nodi ordinati li estraiamo ed a ogni nodo passiamo la lista dei suoi adiacenti richiamando la funzione **RELAX** per rilassarli migliorando il cammino

massimo del nodo trovato fino a quel punto modificando l'attributo d ovvero la distanza dal nodo sorgente e il padre π.
Facendo questo verrà assegnato ad ogni nodo il valore massimo del cammino partendo dal nodo sorgente s e si determinerà il padre per il percorso massimo.

PRESENTAZIONE PSEUDO_CODICE:

```
1.  DFS_VISIT (u, K)
2.  Color(u) ← GRAY
3.  For each v ∈ adj(u)
4.      If color (v) = WHITE
5.          DFS_VISIT (v, K)
6.  Color(u) ← BLACK
7.  Push (K, u)

8.  DFS (G)
9.  K ← Ø
10. For each u ∈ V(G)
11.     Color(u) ← WHITE
12. For each u ∈ V(G)
13.     If color (u) = WHITE
14.         DFS_VISIT (u, K)
15. Return K

16. INIZIALIZED_SINGLE_SOURCE (s, G)
17. For each u ∈ V(G)
18.     d(u) ← - ∞
19.     π(u) ← NIL
20. d(s) ← 0
21. π(s) ← NIL

22. RELAX (u, v)
23. If d(v) < d(u) + w (u, v)
24.     d(v) ← d(u) + w (u, v)
25.     π(v) ← u
```

26. PERCORSO_MASSIMO (s, G)
27. INIZIALIZED_SINGLE_SOURCE (s, G)
28. K ← DFS(G)
29. while (K != Ø)
30. u = top(K)
31. Pop(K)
32. For each v ∈ adj(u)
33. RELAX (u, v)

Dalla riga 1 alla 7 viene illustrato il meccanismo della **DFS_VISIT**, la quale riceve in input il nodo da visitare u e lo stack (K) per la memorizzazione dei nodi.
Alla riga 1 viene assegnato al nodo u il colore GRAY che segna l'inizio della visita.
Tra la riga 3 e la riga 5 scorriamo la lista dei nodi adiacenti al nodo u e controlliamo quali hanno come attributo colore = WHITE per capire se la visita è iniziata, in caso affermativo richiamiamo la DFS_VISIT per visitare in profondità il grafo.
Tra le righe 6 e 7 viene assegnato ad u il colore BLACK che sta a significare che la visita di quel nodo è terminata, infine il nodo viene inserito all'interno di S.

Dalla riga 8 alla 15 viene illustrato il meccanismo della **DFS** a cui viene passato il grafo G.
Nella riga 9 viene allocato lo stack che conterrà i nodi ordinati e viene inizializzato a vuoto.
Tra le righe 10 e 11 viene inizializzato a WHITE il colore di tutti i nodi che non sono stati ancora visitati.
Tra le righe 12 e 14 si scorrono tutti nodi del grafo e su quelli che hanno colore WHITE viene effettuata la DFS_VISIT.
Nella riga 15 ritorniamo lo stack(K) contenente i nodi ordinati.

Dalla riga 16 alla 21 viene utilizzata la funzione **INIZIALIZED_SINGLE_SOURCE** a cui passiamo il nodo sorgente s e il grafo G.
Dalla riga 17 alla 19 vengono inizializzati i padri dei nodi e le distanze dalla sorgente rispettivamente a NIL e a 0 tutto all'interno di un ciclo for.
Dalla 20 alla 21 inizializziamo la distanza dalla sorgente di s a 0 e il padre a NIL.

Dalla riga 22 alla 25 viene illustrato il meccanismo della funzione **RELAX** che prende in input due nodi *u* e *v.* La funzione ha il compito di rilassare i nodi e aggiornare la stima del cammino massimo modificando la distanza dalla sorgente di *v* ovvero d(v).
Tutto questo viene realizzato all'interno di una strutta di controllo "IF " che gestisce la modifica della di d(v) solo nel caso in cui d(v) < d(u) (distanza dalla sorgente del padre) + w (u, v) (ovvero il peso dell'arco da *u* a *v*) , se questo risulta vero allora d(v) ← d(u) + w (u, v).

Dalla riga 25 alla 33 viene illustrata la procedura **PERCORSO_MASSIMO** a cui passiamo il nodo sorgente s e il grafo G.

Dalla riga 27 alla 28 richiamiamo le due funzioni precedentemente descritte per l'inizializzazione dei nodi ovvero INIZIALIZED_SINGLE_SORCE e la DFS per l'ordinamento topologico dei nodi inseriti nello stack(K).
Dalla riga 29 alla riga 31 utilizziamo un ciclo while che uscirà quando lo stack(K) diventerà vuoto, dopodichè estraiamo dalla cima il nodo che viene subito decrementato.
Nelle ultime righe ovvero 31 ,32 e 33 utilizziamo un ciclo for per scorrere tutti gli adiacenti del nodo estratto e richiamiamo la funzione RELAX precedentemente descritta che migliorerà la stima del cammino massimo dei nodi passati *v* ed *u.*
Alla fine della funzione avremo calcolato il cammino massimo da s ad ogni altro nodo del grafo G risolvendo quindi il quesito.

STUDIO DELL COMPLESSITA':

La complessità di tempo di questo algoritmo viene determinata dalle due funzioni **DFS** e **PERCORSO_MASSIMO.**
La funzione DFS usata per l'ordinamento topologico ha un costo θ (V + E), ovvero la somma dell'insieme dei vertici V ed il numero di archi E.
La funzione PERCORSO_MASSIMO utilizza un "while" per analizzare tutti i vertici estratti dallo stack (K) e inoltre esegue un ciclo "for" per scorrere la lista di adiacenza, quindi abbiamo una complessità di tempo anche in questo caso di θ (V + E).

Il totale abbiamo 2[θ (V + E)] che è asintoticamente equivalente a **θ (V + E).**

TEST E RISULTATI:

Il programma effettua un controllo tramite la funzione **LETTURA_FILE** per verificare l'apertura senza errori del file "GRAPHA-NUI.txt", in tal caso la funzione legge N nodi, P ponti e i successivi righi contenenti i vertici e gli archi.
Successivamente l'utente potrà tramite la tastiera scegliere l'isola di partenza per il calcolo del percorso massimo digitando un numero compreso tra 0 ed N-1.
Dopo la scelta il programma mostra a video il nodo sorgente (scelto dell'utente) con il valore del peso del cammino massimo per ogni isola raggiungibile.
Il programma termina quando l'utente digita il carattere "0" oppure può continuare digitando qualsiasi altro carattere per visualizzare il valore del cammino massimo partendo da un'altra isola sorgente.

ESEMPI E TEST:

Partenza isola 0.

```
"C:\Users\matti\Desktop\PROGETTO MATTIA DI PALMA\TRACCIA 1\QUESITO 2\cmake-build-debug\QUESITO_2.exe"
                      DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA.

DIGITARE SCELTA :0
PARTENDO DALL'ISOLA 0 CALCOLIAMO IL PERCORSO.

0 -> 0  0
0 -> 1  5
0 -> 2  7
0 -> 3  14
0 -> 4  13
0 -> 5  15

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

Partenza isola 1.

```
                  DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA.

DIGITARE SCELTA :1
PARTENDO DALL'ISOLA 1 CALCOLIAMO IL PERCORSO.

1 -> 0  -INF
1 -> 1  0
1 -> 2  2
1 -> 3  9
1 -> 4  8
1 -> 5  10

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

Partenza isola 2.

```
                  DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA.

DIGITARE SCELTA :2
PARTENDO DALL'ISOLA 2 CALCOLIAMO IL PERCORSO.

2 -> 0  -INF
2 -> 1  -INF
2 -> 2  0
2 -> 3  7
2 -> 4  6
2 -> 5  8

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

Partenza isola 3.

```
                  DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA.

DIGITARE SCELTA :3
PARTENDO DALL'ISOLA 3 CALCOLIAMO IL PERCORSO.

3 -> 0  -INF
3 -> 1  -INF
3 -> 2  -INF
3 -> 3  0
3 -> 4  -1
3 -> 5  1

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

Partenza isola 4.

```
                  DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA.

DIGITARE SCELTA :4
PARTENDO DALL'ISOLA 4 CALCOLIAMO IL PERCORSO.

4 -> 0  -INF
4 -> 1  -INF
4 -> 2  -INF
4 -> 3  -INF
4 -> 4  0
4 -> 5  -2

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

Partenza isola 5.

```
                  DI PALMA MATTIA  0124002448  TRACCIA 1.

QUESITO 2 :

GRAPHA-NUI CON 6 ISOLE E 10 PONTI CREATO DAL FILE 'GRAPHA-NUI.TXT'
SCEGLIERE UN ISOLA DA 0 A 5 DA CUI PARTIRE PER CALCOLARE IL PERCORSO CON LA QUALITA' MASSIMA

DIGITARE SCELTA :5
PARTENDO DALL'ISOLA 5 CALCOLIAMO IL PERCORSO.

5 -> 0  -INF
5 -> 1  -INF
5 -> 2  -INF
5 -> 3  -INF
5 -> 4  -INF
5 -> 5  0

PREMERE '0' PER USCIRE QUALSIASI ALTRO TASTO PER CONTINUARE :
```

CLASS DIAGRAM: