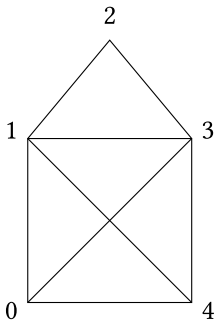
Disegna senza sollevare la matita (Oii 2012, selezioni nazionali) (`matita` [66 punti])

Sai risolvere il classico puzzle di disegnare la casetta senza mai sollevare la matita dal foglio?

Più in generale, dati N vertici numerati da 0 a $N-1$ e M lati che li collegano, sapresti indicare in quale sequenza attraversare i lati, e per ciascuno di essi in quale direzione, in modo che il vertice cui ci conduce ciascun lato coincida col vertice da cui ci preleva il lato successivo (senza alzare quindi la matita)? La sfida è di riuscire ad attraversare ciascun lato una ed una volta sola.



Input

Si legga l'input da `stdin`. La prima riga contiene T, il numero di testcase (istanze) da risolvere. Seguono T istanze del problema. Per ogni istanza, la prima riga contiene quattro numeri interi N, M, A e B separati da uno spazio: il numero di vertici, il numero di lati che si richiede di attraversare (ogni lato una ed una volta sola, dovrai decidere tu in quale direzione), il vertice di partenza, e quello di arrivo. Le successive M righe offrono, uno per riga, gli M lati del puzzle, ciascuno rappresentato da una coppia non-ordinata di interi nell'intervallo $[1, N]$, separati da uno spazio (questi interi sono i nomi dei vertici collegati dal lato in questione). Per ciascun lato "X Y" dovrai scegliere se attraversarlo da X ad Y oppure da Y ad X, oltre che stabilire in quale ordine vadano attraversati i lati.

Output

Per ciascuna istanza, prima di leggere l'istanza successiva, scrivi su `stdout` il tuo output così strutturato: per ogni riga dell'input intesa a rappresentare un lato del puzzle (ossia nel formato "X Y"), l'output dovrà contenere una riga "X Y" se il lato viene percorso da X a Y, oppure una riga "Y X" se il lato viene percorso da Y a X. Le righe in output saranno pertanto M, ma riordinate rispetto alle M rispettive righe dell'input in modo da specificare inoltre in quale ordine gli M lati dell'input vengano attraversati: per la prima di queste righe deve valere la condizione che $X = A$. Il vertice X di ciascun altra riga deve coincidere col vertice Y della riga precedente (non devi staccare mai la matita dal foglio), e per l'ultima riga deve valere anche che $Y = B$.

Prevediamo due goal:
goal 1 [decide]: basta saper distinguere se esista o meno una qualche soluzione
goal 2 [any_sol]: serve anche produrre una qualsiasi soluzione ogniqualvolta ne esista una

Oltre alle dimensioni delle istanze, ogni subtask precisa quanti punti competono ai vari goal. Il punteggio ottenuto per la generica istanza di quel subtask sarà la somma dei punti dei goal a risposta corretta (purchè si rispetti almeno il formato in tutte le righe di output, incluse quelle che competono agli altri goal – altrimenti salta il protocollo di comunicazione tra il tuo programma risolutore e il server).

Esempio di Input/Output

Input da `stdin`

```
-start-
2
5 8 1 5
1 4
2 3
5 4
2 1
2 4
3 4
1 5
5 2
-more-
```

```
6 6 1 2
2 1
2 3
2 5
2 6
4 3
-more-
```

```
4 5
6 6 1 6
2 1
2 3
2 5
2 6
4 3
4 5

-end-
```

Output su `stdout`

```
1 2
2 3
3 4
4 5
5 2
2 4
4 1
1 5

1 2
2 3
3 4
4 5
5 2
2 6
```

Note

1. Se non vi è alcuna soluzione valida, l'output da inviare su `stdout` dovrà consistere di una singola riga vuota. Viene garantita la connettività: dal vertice di partenza è sempre possibile portarsi in un qualsiasi altro vertice senza alzare la matita (ma non è detto che il percorso che condurrebbe al nodo di arrivo possa percorrere precisamente una volta ciascuno dei lati.
2. Nessun testcase presenta lati multipli che collegano la stessa coppia di vertici (se un lato di un'istanza è descritto dalla stringa "$X\ Y$", nella stessa istanza non potrai ritrovare la stessa stringa e nemmeno la stringa "$Y\ X$"). Inoltre non forniremo mai lati della forma "$X\ X$". Tutti i vertici e tutti i lati devono essere attraversati dalla matita.
3. Per chi non lo avesse riconosciuto, questo è il noto problema affrontato dal matematico Eulero in un lavoro (1735) considerato l'atto di nascita della teoria dei grafi e della topologia.

Subtask

Il tempo limite per istanza (ossia per ciascun testcase) è sempre di 2 secondi.

I testcase sono raggruppati nei seguenti subtask.

1. **[6 pts← 3 istanze da 1 + 1 punti] `esempi_testo`:** i tre esempi del testo
2. **[16 pts← 8 istanze da 1 + 1 punti] `smallest`:** $n \leq 4, m \leq 6$
3. **[8 pts← 4 istanze da 1 + 1 punti] `small`:** $n \leq 6, m \leq 12$
4. **[12 pts← 6 istanze da 1 + 1 punti] `medium`:** $n \leq 100, m \leq 500$
5. **[12 pts← 6 istanze da 1 + 1 punti] `big`:** $n \leq 1,000, m \leq 5,000$
6. **[12 pts← 6 istanze da 1 + 1 punti] `large`:** $n \leq 10,000, m \leq 50,000$

In generale, quando si richiede la valutazione di un subtask vengono valutati anche i subtask che li precedono, ma si evita di avventurarsi in subtask successivi fuori dalla portata del tuo programma che potrebbe andare in crash o comportare tempi lunghi per ottenere la valutazione completa della sottomissione. Ad esempio, chiamando[1, 2] :

```
rtal -s <URL>  connect -x <token> -a size=small
       matita  -- python my_solution.py
```

[1] <URL> server esame: `wss://ta.di.univr.it/esame`

[2] <URL> server esercitazioni e simula-prove: `wss://ta.di.univr.it/algo`

vengono valutati, nell'ordine, i subtask:

`esempi_testo`, `smallest`, `small`.

Il valore di default per l'argomento `size` è `large` che include tutti i testcase.