# Theseus vo Tirol

> *Wenn olle Toala vo da Aufgab Theseus mit da Zeit austauscht werdn, ab wann — wenn überhaupt — is es nimma die gleiche Aufgab?*

Wenn a nit grad über irgendwas Gscheids nachdenkt, haut da Theseus in seina Freizeit Minotauren z'samm. Desmal muas er aba zerst durch a finsteres und ganz schian verschlunganes Labyrinth durche. Des is natürlich nit gord oanfach, drum bittet er de Ariadne um a bissal a Hilfe. Ma kann sich es Laabyrinth wia an verbundenen, ungerichteten **Graphn** vorstelln, mit $n$ Knotn (nummeriert vo $1$ bis $n$) und $m$ Kantn, und oam ganz besondern Knotn $t$, der wo da Minotaurus hockt.

Da Theseus kann in Graphn selba nit segn, aba die Ariadne kann scho. Deswegn miasn sie zerst a Strategie überlegn, damit er sicha zum Knotn mim Minotaurus kimmt: Sie werd jeder vo de $m$ Kantn a Beschriftung gebn - entweder $0$ oder $1$. Da Theseus startet nacha ins Labyrinth eini am Knotn $s$. Den Knotn kenn die Ariadne vorher aba leida nitta.

Weils so dunkel isch, kann da Theseus zu jedm Zeitpunkt lei (nur) folgendes sechn:

- in Index vom Knotn auf dem er grad isch
- die Indizes vo die direkt verbundenen Knotn
- die Beschrifungen vo die Kantn, de zu de Knotn führn (welche die Ariadne dort ani gschriebn hat)

Zusätzlich kannt er sich wegn im ganzn Wirrwarr da drine **niea merkn**, was bisher alls passiert is. Hoast, an jedm Knotn is als wärs da erste.

Damit a auf alle Fälle zum Minotaurus kimmt, derf er maximal $min + C$ Schritte machn, wobei $min$ die minimal benötigte Kantnanzahl is, um vom Start $s$ bis zum Ziel $t$ zum kemma — $C$ is dabei a fixe Konstante.

## Implementierungsdetails

Du sollsch de zwoa Funktionen implementiern:

```
std::vector<int> label(int n, std::vector<std::pair<int,int>> edges, int t);
```

- $n$: wie vü Knotn im Graphn hen
- $edges$: a `vector` mit da Länge $m$, der die Kantn vom Graphn beschreibt
- $t$: da Zielknotn

- De Funktion muas an `vector` da Länge $m$ zruggebn, wobei es $i$-te Element entweder $0$ oda $1$ sein kann und de Beschriftung vo da $i$-tn Kantn für alle $0 \leq i < m$ darstellt.
- Jede Kantn muas entweder mit $0$ oder mit $1$ beschriftet sein. Die Beschriftung mi am andern Wert führt zu am **undefiniertn Verhaltn**
- De Funktion werd für jedn Testfall **genau oamal** aufgrufn

```
int travel(int n, int u, std::vector<std::pair<int,int>> neighbours);
```

- $n$: wie vü Knotn im Graphn hen
- $u$: wo da Theseus grad is (welcha Knotn)
- $neighbours$: a Listn vo Paare $(v, e)$, de angibt, dass a Kantn zwischn $u$ und $v$ verlafft, welche die Ariadne mit $e$ beschriftet hat#
- De Funktion muas genau oan Nachbarknotn zruggebn, zu dem ma sich a bewegn kann. Falls dera Knotn $t$ is, werd es Programm automatisch beendet
- Es is garantiert, dass bei jedm Aufruf dera Funktion $u$ sich vom Knotn $t$ unterscheidet
- Oa Aufruf vo dera Funktion stellt oa Bewegung im Labyrinth doa. Deswegn kann se für jedn Testfall **so oft wia nötig** aufgrufn werdn. Halt so lang bis da Zielknotn erreicht is

**Obacht!** Es Programm derf koane globaln oda statischn Variablen nutzn um zwischn die verschiedenen Insntanzn vo `label` oda `travel` zum kommuniziern. Jeda Versuch um sel zu umgehn führt zu am **undefiniertn Verhaltn**.

## Beschränkungen

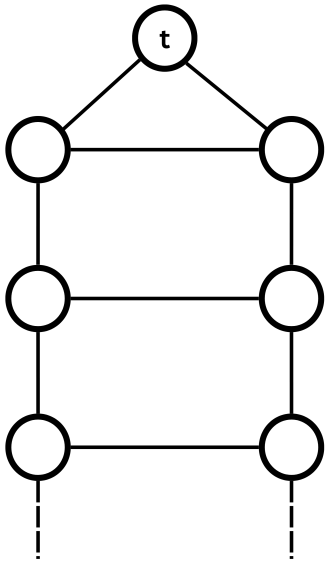
- $1 \leq n \leq 10000$
- $1 \leq m \leq 50000$
- $C = 14$
- Da Startknoten $s$ wer für jedn Testfall festglegt bevor `label` aufgrufn werd.

## Teilaufgaben

1. (4 Punkte) Da Graph is a Clique (d.h. es gibt oa Kantn zwischn alle Paaren vo Knotn).
2. (10 Punkte) Die Entfernung zwischn im Ziel und am beliebign Knotn im Graphn betragt max. 2 Kantn.
3. (11 Punkte) Da Graph is a Bam (Baum).
4. (13 Punkte) Da Graph is bipartit (d.h. es gibt oa Möglichkeit, die Knoten vom Graphen in zwoa Teilmengen zu unterteiln, so dass es koa Kantn zwischn zwoa Knotn aus da selbn Teilmenge gibt).
5. (12 Punkte) Da Graph is oa Loata (Leiter $-$ siehe Definition untn).
6. (50 Punkte) Koane zusätzlichn Beschränkungen.

**Anmerkung:** Ein Loata-Graph is an Graph, dea aus zwoa parallele Pfad (oda Kettn) gleicha Länge bsteht, wobei jeds Paar entsprechenda Knotn durch oa Kantn verbundn is und de Sprossn da Loata bildet. Zusätzlich gibt es an oam End da Loata oan spezielln Knotn $-$ in Zielknotn $t$ $-$, der mit boade

Endpunkte da Loata verbundn is und deswegn als gemeinsamer Vota (Parent) fungiert. Es is garantiert, dass $n$ für jedn solchn Graphn ungrad is. Schaugs da an dem Bild o:



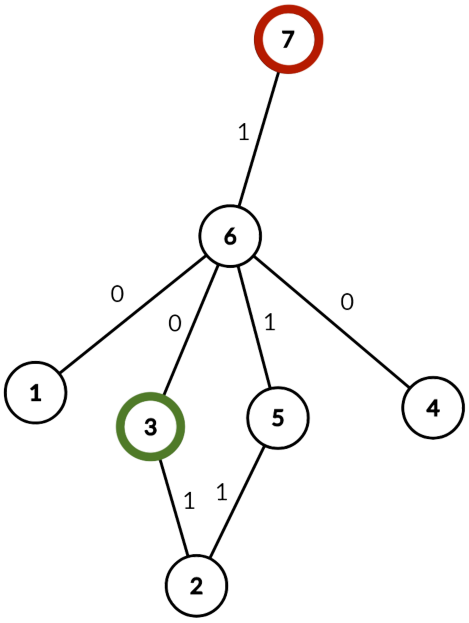

## Beispiele

### Beispiel 1

Nemm ma an, dass ma an Graphn mit $7$ Knotn und $7$ Kantn hen (sigsch untn). Da Startknotn is die $3$ (sich am **gianen** Kreis) und es Ziel is die $7$ (**rot**). Da Grader werd zerst in folgenden Funktionsaufruf mocha:

```
label(7, {{1, 6}, {7, 6}, {2, 5}, {3, 2}, {3, 6}, {6, 5}, {6, 4}}, 7)
```

Angenommen obiges liefert `{0, 1, 1, 1, 0, 1, 0}`. Dann werd da Gaph fürn Theseus folgendermasn auschaugn:




Es folgt a mögliche Abfolge an Aufrufn vo `travel`, de ihn zum Minotaurus bringt:

| Aufruf | Rückgabewert |
|---|---|
| `travel(7, 3, {{2, 1}, {6, 0}}` | 2 |
| `travel(7, 2, {{5, 1}, {3, 1}}` | 5 |
| `travel(7, 5, {{6, 1}, {2, 1}}` | 6 |
| `travel(7, 6, {{3, 0}, {5, 1}, {1, 0}, {4, 0}, {7, 1}}` | 4 |
| `travel(7, 4, {{6, 0}}` | 6 |
| `travel(7, 6, {{3, 0}, {5, 1}, {1, 0}, {4, 0}, {7, 1}}` | 7 |

Wenn da zruggebene Wert die $7$ (also es Ziel) is, dass is Ganze aus.

## Beispiel-Grader

Da Beispiel-Grader liest die Eingab im folgenden Format:

- Zeile $1$: $n$ $m$
- Zeile $2$: $s$ $t$
- Zeile $3 + i$: $a$ $b$, was hoast, dass die Kantn zwischn im Knotn $a$ und $b$ gib.

Zerst ruft da Grader `label`mit die entsprechenden Parametern auf und beschriftet die Kantn vom Graphn entsprechend im zruggebenen Vektoren.

Dann ruft er `travel`mit die Parameter $n$, $s$ und im Nachbarn vo $s$ auf. Nach im erstn Aufruf werd `travel` imma und imma wieda aufgrufn, wobei da aktuelle Knotn dera is, der vom vorherigen Aufruf zruggebn wordn is. Halt so lang bisd es Ziel $t$ erreicht hast.

Am Ende gibt ea die Anzahl an Bewegungen aus, die deine Lösung gmacht hat, sowie die Knotn, in da Reihenfolge, in dea sie besucht wordn hen.