International Olympiad in Informatics 2011
2011年7月22-29日, タイ, パタヤ
Day 2 競技課題
日本語版

CROCODILE

バージョン: 1.3

ワニの地下都市 (Crocodile's Underground City)

考古学者の Benjamas は, 謎のワニの地下都市を探検していたが, 命の危険を感じ逃げることにした. 都市には **N** 個の部屋があり, **M** 本の双方向に通行可能な廊下で結ばれている. 各々の廊下は 2 つの異なる部屋を結んでおり, 結んでいる部屋の対はどの廊下についても異なる. 廊下ごとに, そこを通るための時間は異なるかもしれない. **N** 個の部屋のうち **K** 個の部屋は「出口の部屋」であり, 彼女はそこから脱出することができる. Benjamas は最初は部屋 0 におり, できる限り早く出口の部屋にたどり着きたい.

門番のワニは, Benjamas が脱出するのを防ぎたい. 彼は秘密のドアをコントロールし, 廊下を **1 本だけ**封鎖することができる. つまり, 彼が新たに廊下を封鎖すると, その前に封鎖されていた廊下は再び通れるようになる.

Benjamas のおかれている状況を正確に記述すると次のようになる: 彼女が部屋から移動しようとすると, 門番はその部屋に接続する廊下のうち 1 本を選んで封鎖することができる. その後, Benjamas は封鎖されていない廊下のうち 1 本を選んでその先の部屋へ移動する. Benjamas が一度廊下に入ったら, 彼女が廊下を抜けるまで門番はその廊下を封鎖することはできない. Benjamas が次の部屋にたどり着けば, 門番はまた封鎖する廊下を選ぶことができる (Benjamas が直前に通った廊下でもよい).

彼女は単純な形の脱出計画をあらかじめ立てておきたいと思っている. より正確には, 部屋ごとに, そこにたどり着いたときに何をするかの指示を決めておこうと思っている. ある部屋 **A** について考える. **A** が出口の部屋ならそこから都市を脱出できるので, もちろん指示は必要ない. そうでない部屋については, 指示は次のいずれかの形でなければならない:

- 部屋 **A** にたどり着いたら, 部屋 **B** に通じる廊下を選べ. ただし, その廊下が封鎖されている場合は, 部屋 **C** に通じる廊下を選べ.
- 部屋 **A** のことはどうでもよい. 指示にしたがっている限り部屋 **A** にたどり着くことはない.

場合によっては, 門番は Benjamas が出口の部屋にたどり着くのを防ぐことができる (たとえば, 計画に従うと彼女が堂々巡りに入る場合などがそうである). Benjamas が門番の行動によらず必ず出口の部屋にたどり着くような計画を「良い」計画と呼ぶ. 良い計画に対し, **T** を「**T** 以下の時間で Benjamas が**必ず**出口の部屋にたどり着く」ような最小の値とする. この場合, その良い計画には「**T** の時間がかかる」と言う.

課題 (Your task)

次のパラメータをもつプロシージャー **travel_plan(N, M, R, L, K, P)** を実装せよ:

- **N** — 部屋の個数. 部屋には **0** から **N−1** までの番号が付けられている.
- **M** — 廊下の本数. 廊下には **0** から **M−1** までの番号が付けられている.
- **R** — 廊下の情報を表す整数の 2 次元配列. **0 ≦ i < M** に対し, 廊下 **i** は異なる2つの部屋 **R[i][0]** と **R[i][1]** を結んでいる. 結んでいる部屋の対はどの廊下についても異なる.
- **L** — 廊下を移動するのにかかる時間を表す整数の 1 次元配列. **0 ≦ i < M** に対し, **1 ≦ L[i] ≦ 1 000 000 000** は Benjamas が廊下 **i** を通って移動するのにかかる時間である.

- **K** — 出口の部屋の個数. **1 ≦ K < N** であるとしてよい.
- **P** — **K** 個の異なる値からなる, 出口の部屋を表す整数の 1 次元配列. **0 ≦ i < K** に対し, **P[i]** は **i** 番目の出口の部屋の番号である. 部屋 0 が出口の部屋であることはない.

プロシージャーは, **T** の時間がかかる良い計画が存在するような最小の **T** を返さなければならない.

出口の部屋でない部屋には必ず 2 本以上の廊下が接続しているとしてよい. また, **T ≦ 1 000 000 000** なる良い計画が必ず存在するとしてよい.

例 (Examples)

例 1

例として, 図 1 に示される **N = 5, M = 4, K = 3,**

$$R = \begin{matrix} 0 & 1 \\ 0 & 2 \\ 3 & 2 \\ 2 & 4 \end{matrix} \quad L = \begin{matrix} 2 \\ 3 \\ 1 \\ 4 \end{matrix} \quad P = \begin{matrix} 1 \\ 3 \\ 4 \end{matrix}$$

の場合を考える.

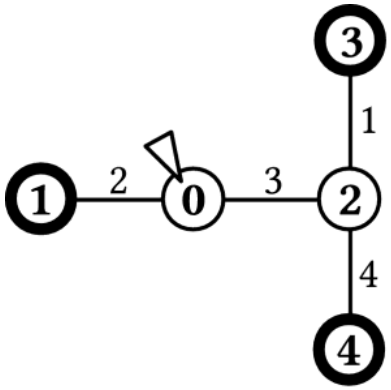

図 1

部屋は丸で表され, それらを結ぶ廊下は線で表されている. 出口の部屋は太線
の丸で表されている. Benjamas は最初は三角のついた部屋 0 にいる. 以下の計画は最適な計画のひとつである:

- 部屋 0 にたどり着いたら, 部屋 1 に通じる廊下を選べ. ただし, その廊下が封鎖されている場合は, 部屋 2 に通じる廊下を選べ.
- 部屋 2 にたどり着いたら, 部屋 3 に通じる廊下を選べ. ただし, その廊下が封鎖されている場合は, 部屋 4 に通じる廊下を選べ.

最悪の場合では, Benjamas は 7 単位時間で出口の部屋にたどり着く. よって, **travel_plan**は **7** を返さなければならない.

例 2

例として, 図 2 に示される **N = 5, M = 7, K = 2,**

$$R = \begin{matrix} 0 & 2 \\ 0 & 3 \\ 3 & 2 \\ 2 & 1 \\ 0 & 1 \\ 0 & 4 \\ 3 & 4 \end{matrix} \quad L = \begin{matrix} 4 \\ 3 \\ 2 \\ 10 \\ 100 \\ 7 \\ 9 \end{matrix} \quad P = \begin{matrix} 1 \\ 3 \end{matrix}$$

の場合を考える.

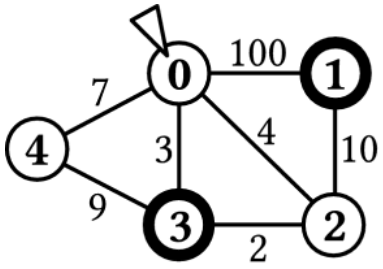

図 2

以下の計画は最適な計画のひとつである:

- 部屋 0 にたどり着いたら, 部屋 3 に通じる廊下を選べ. ただし, その廊下が封鎖されている場合は, 部屋 2 に通じる廊下を選べ.
- 部屋 2 にたどり着いたら, 部屋 3 に通じる廊下を選べ. ただし, その廊下が封鎖されている場合は, 部屋 1 に通じる廊下を選べ.
- 部屋 4 のことはどうでもよい. 指示にしたがっている限り部屋 4 にたどり着くことはない.

Benjamas は 14 単位時間以内に出口の部屋にたどり着く. よって, **travel_plan** は **14** を返さなければならない.

小課題 (Subtasks)

小課題 1 (46 点)

- **3 ≦ N ≦ 1 000.**
- 地下都市は木である. つまり, **M = N−1** であり, 各部屋の組 **i** と **j** に対して **i** と **j** をつなぐ廊下の列が存在する.
- 各出口の部屋はちょうど 1 つの他の部屋と結ばれている.
- 他の部屋は 3 つかそれよりも多くの部屋と結ばれている.

小課題 2 (43 点)

- **3 ≦ N ≦ 1 000**
- **2 ≦ M ≦ 100 000**

小課題 3 (11 点)

- **3 ≦ N ≦ 100 000**
- **2 ≦ M ≦ 1 000 000**

実装の詳細 (Implementation details)

制限 (Limits)

- CPU 時間制限 (CPU time limit): 2 秒
- メモリ制限 (Memory limit): 256 MB
  注意: スタックのサイズには決められた制限はない. スタックとして使用されたメモリは, メモリ総使用量に含まれる.

インターフェース (API) (Interface (API))

- 実装フォルダ (Implementation folder): `crocodile/`
- 選手が実装するファイル: `crocodile.c` または `crocodile.cpp` または `crocodile.pas`
- 提出ファイルのインターフェース (Contestant interface): `crocodile.h` または `crocodile.pas`
- 採点プログラムのサンプル (Sample grader): `grader.c` または `grader.cpp` または `grader.pas`
- 採点プログラムの入力のサンプル (Sample grader input): `grader.in.1`, `grader.in.2`, ...
  注意: 採点プログラムのサンプルは次の書式の入力を読み込む.
  - 1 行目: **N** と **M** と **K**.

- 2 行目から **M＋1** 行目: **0 ≦ i < M** に対し, **i＋2** 行目には **R[i][0]** と **R[i][1]** と **L[i]** が空白区切りで書かれている.
- **M＋2** 行目: **K**個の整数 **P[0]**, **P[1]**, ..., **P[K－1]** が空白区切りで書かれている.
- **M＋3** 行目: 期待される解.

- 採点プログラムの入力のサンプルに対して, 期待される出力: `grader.expect.1`, `grader.expect.2`, ...
  この課題において, これらのファイルはいずれも文字列 **“Correct.”** のみを含むファイルである.