D :Revenge of the Broken Door

原作：壊れたドア
(国内予選2011)
改題：tokoharu
実装：tokoharu
kyuridenamida

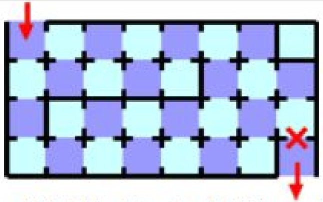


図 G-3: 通り抜けられなくなる可能性のある迷路

図G-3の場合，故障しているドアが赤いバツ印(✕) の位置のドアの場合は入口から出口に行けなくなるが，図G-1の迷路では故障しているドアがどれであっても，入口から出口に行くことはできる．図G-1 の迷路で，図G-2の最短経路を行こうとして図G-4の赤いバツ印のドアが故障していることを発見した場合，緑の矢印の経路で出口にたどり着くことになるだろう．この場合，カードは20枚必要になる．

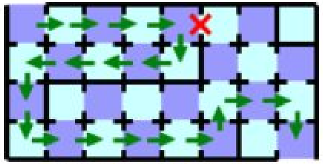


図 G-4: ドアが故障した迷路

しかし，この迷路はもっと少ない枚数のカードで抜けることができる．まず，ドアの故障を見つけるまでは図G-5の経路にしたがって進む．これは故障がなくても12枚のカードが必要なので最短経路ではない．経路の途中で故障しているドアを見つけたら，そこからそのドアを使わないときの出口への最短経路で進む．この戦略ならば故障しているドアがどこであっても16枚のカードで抜けることができる．この戦略で最悪となる16枚のカードが必要なケースを図G-6に示す．

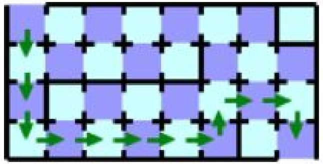
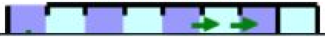


図 G-5: 故障発見前の経路



概要

- 重み付き無向グラフと始点終点が与えられる。
- sからtへ行きたい。
- どれか1本の枝は通行止め。これは事前にどれかわからない。
- s-tへ行くときの移動距離の最悪ケースを最小化するといくらになるか？
- N,M <= 10^5オーダー

- 要するに壊れたドアの設定を一般化

解法(1)

- 最大値を最小化 → 二分探索
- 新しい問題
  - 距離Dが与えられる
  - 1回の通行止めされてもs-t間を総距離D以内で到達可能か判定せよ
- これを解くためには次のf(v,u)を求める必要がある。
  - f(v,u) := 頂点vと頂点uの間に辺があるときに定義され、
    頂点vに到達した時点でこの辺が通行禁止と分かった時に
    頂点tまで到達可能な時間の最小値
- d[e.from] + f(e.from, e.cost) <= Dであるときに
  d[e.to] = min(d[e.to], d[e.from] + e.cost)の更新をするDijkstra。
  最後、d[t] <= DならOK.

解法(2)

- 前処理としてfを求める必要がある(再掲)
    - f(v,u) := 頂点vと頂点uの間に辺があるときに定義され、
      頂点vに到達した時点でこの辺が通行禁止と分かった時に
      頂点tまで到達可能な時間の最小値
- 明らかに、v-u辺がv-tの最短経路に含まれないなら最短経路を求めればよい

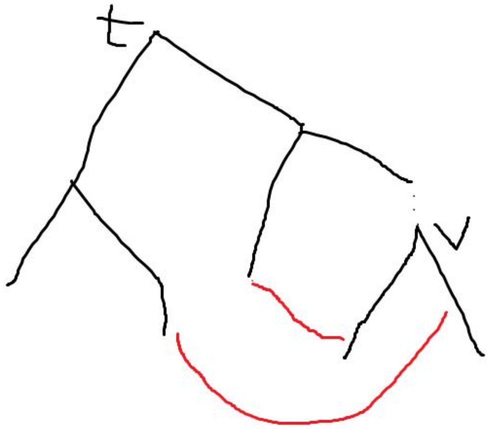
- とりあえず tからDijkstraをして、tを根とする最短路木を求めてみる
    - この最短路木に入る辺以外はv-t最短経路の値を返す。
    - では、最短路木に含まれる場合は？

解法(3)

- 問 :「t始点最短経路木に含まれるvの親辺が使えないとき
  vからtへの最短距離は？」
- 答 : 最短経路木におけるvをrootとする部分木にある頂点と、
  それに含まれない頂点を結ぶ辺eに対して、v-e-t最短路の最小値
- イメージ :

赤辺をeとして用いて、
それ以外は最短路木をそのまま用いる

二回以上最短路木以外の辺を通るのが無駄



解法(4)

- 赤線eがf(v,vの親)に影響を与えるvがどの範囲か考えてみる
- eの両端点(x,y)のLCA点zをとったとき、vはx-z-yパスのどれか（z含まず）
- ex) z-y上の各vは、(t-x).cost + (e).cost + (y-v).cost をクエリとして更新

- このままだと一定コストで更新できない
- f(v, vの親) + (最短路木上t-vパス長) を求めようと思えば
  (t-x).cost + (e).cost + (t-y).cost　でvによらない一定のコストになる

解法(5)

- パス上一定のクエリに落とせたので、木のクエリの形になった
  - この問題はAOJ2259 (Minimum Spannning Tree) と同等
- HL分解など様々な方針がつかえるが、次のやり方が楽。

- eに対してコストは一定で各点で最小化したいから、
  クエリはコストが小さい順に投げればよい。
- 一度更新したらその頂点は縮約していけばよい。
- この処理は例えばUnion-Findで実現可能。

想定解法まとめ

- 外側 : 二分探索, s始点の変則Dijkstra
- 前処理 : t始点Dijkstra, LCA, (HL分解 or Union-Find or その他)

- CxivDxiv解は二分探索を飛ばしていてより高速でした

ジャッジ解

tokoharu C++   303 lines, 5.6kB
kyuri   JAVA 369 lines, 13.0kB

FA

hankan_rta : 222:46

CxivDxiv : 269:31