JAG ICPC模擬国内予選2024

I: 橋の建造計画 2

原案：mtsd
問題文：riantkb
データセット：mtsd
解答：hos, hotman, mtsd, smiken
解説：mtsd

問題概要

- N 頂点 M 辺の単純連結グラフが与えられる
- 各辺に以下の条件を満たすように色を塗りたい
  - 各色について、補グラフが連結である
- 条件を満たす色数の最小値とその塗り方を答えよ
  - ただし，不可能な場合は -1 を返すこと
- 制約
  - N <= 150
  - N-1 <= M <= 1500

不可能な場合

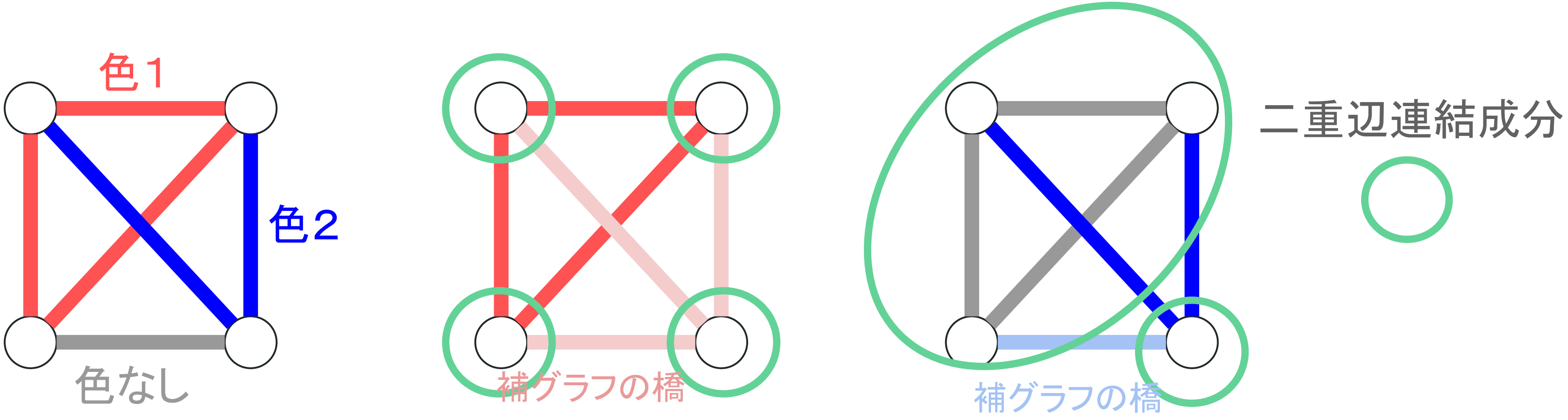
- グラフに橋がある場合は条件を満たすような色の塗り方は不可能である
- 橋の列挙は O(N+M) で可能
  - 後述する色数最小化のアルゴリズムにおいても橋の列挙，二重辺連結成分分解 の計算が必要
- 以下ではグラフに橋がない場合を考える

解法の概要

- 現在 K 色を使って M 辺のうち m 辺が塗られているとする
- このとき, K 色で m+1 辺を塗ることが出来るかをO((N+K)M) で計算可能である

<u>アルゴリズム</u>

1. 各色の補グラフに対して、二重辺連結成分分解を計算する

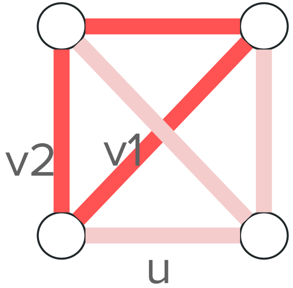


解法の概要

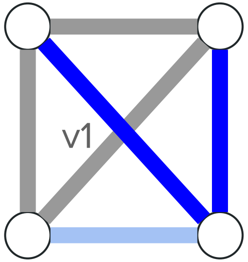
- 現在 K 色を使って M 辺のうち m 辺が塗られているとする
- このとき, K 色で m+1 辺を塗ることが出来るかをO((N+K)M) で計算可能である

<u>アルゴリズム</u>

2. 各辺 u に対して、辺 u を色 x (1 ≤ x ≤ K)で塗り替えた際に、以下を調べる
    a. 色 x について補グラフの連結性が満たされたままであるか(この情報を、(x, u -> ok) と表現する)
    b. 色 x について補グラフの連結性を満たさない場合
    色 x の辺 v のうち、「辺 u を色 x で塗り，辺 v の色を消したとき」に「色x の補グラフの連結性が満たされる」辺 v を二重辺連結成分分解を用いて列挙する（この情報を (x, u->v) と表現する）



(1, u -> v1),
(1, u -> v2)



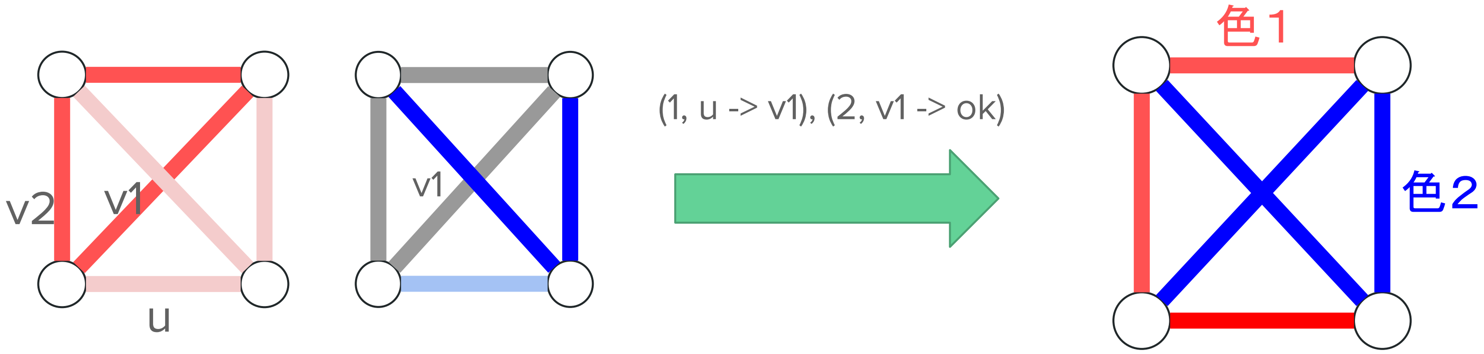
(2, v1 -> ok)

解法の概要

- 現在 K 色を使って M 辺のうち m 辺が塗られているとする
- このとき, K 色で m+1 辺を塗ることが出来るかを O((N+K)M) で計算可能である

<u>アルゴリズム</u>

3. 2 で列挙された情報から(x1, u1 -> u2), (x2, u2 -> u3), ... , (xe, ue -> ok) となる**最短の情報列**を計算する
   ただし, u1 は色がまだ塗られていない辺とする
   a. 見つかった場合、その情報列に従って色を塗り替えることが可能（色xi で辺 ui を塗り替え）
   b. 見つからなかった場合、K 色で m+1 辺を塗ることは不可能



解法の正当性

- 特定の色 x については複数回塗り替えられる可能性があるが、
  最初に作成した二重辺連結成分分解をそのまま使ってよいのか？
  -> 最短の情報列であれば、問題ない
- 「補グラフが連結である辺集合」はマトロイドを成す (cographic matroid, bond matroid)
- 本問題は cographic matroid に対するマトロイド分割に関する問題である
- 前述のアルゴリズムはマトロイド分割のアルゴリズムでの増加道探索の計算に対応する
  (※マトロイド分割はマトロイド交差の特殊ケースであるが、本問題においてはマトロイド交差として考えても解くことが出来る)
- 参考
  : https://topcoder-g-hatena-ne-jp.jag-icpc.org/spaghetti_source/20121124/1353741121.html

計算量解析

- 色数 K の場合における、最短の情報列の計算 1 回あたりの時間計算量
  - 各色での二重辺連結成分分解にかかる時間計算量は O(KM)
  - 各色の補グラフの橋の本数の総和は N で抑えられるため，(x, u->v) の情報の総数は高々NM
  - 塗られていない辺の距離を 0 とする BFS として最短の情報列を計算すれば 1回につき O((N+K)M)
- 色数 K で条件を満たす塗り方が存在するかを判定する時間計算量
  - 色を塗っていない状態から m = M になる、もしくは不可能だと分かるまで適用すると、色数 K で条件を満たすかどうかを判定できる
  - この場合の時間計算量は O((N+K)M^2) となる
    - 色数の最小値を求める場合に、例えば K について 1 〜 M で二分探索すると、時間計算量が O(M^3 log M) となり TLE するので、計算量改善が必要

時間計算量の改善

以下の工夫をすると、時間計算量O(N^2M) で色数の最小値を求めることができる

（※国内予選形式では O(NM^2) も実装によっては間に合う）

- 色数の最大値の正しい見積り
  - 全域木を一つ考えると、N 色以下で必ず条件を満たす塗り方ができる
- 最短の情報列の計算の回数の削減
  - K 色で m+1 辺を塗ることが出来るかの判定アルゴリズムでは、任意の「現在 K 色を使って m 辺が塗られているとする」状態を開始地点としても問題なく動く
そこで「ある全域木を残して、それ以外の辺を色 1 で塗った」状態からスタートすることで、n-1 回の反復まで減らすことが出来る
- 色を増やす際の二分探索の回避
  - 二分探索を行わずに、K=2 から色を 1 色ずつ増やして最大まで塗った場合の解を得ることが出来る
K - 1 色で最大まで塗った時の解を K 色の初期値として再利用すればよい

乱択・ヒューリスティック解法について

- 辺を一列に並べて、順に、その時点で条件を満たすように塗ることができる最小の色番号の色で塗る貪欲法によるヒューリスティック解法
  - 辺をシャッフルする乱択を一定時間行い、ベストな解を出力する
    - 1 回の成功確率が 10^-12 程度になるケースを複数追加したため、通らないはず
  - 山登り
    - 近傍: 枝リストの i 番目の枝と j 番目の枝を swap する
    - 遷移条件: (必要な色数, 最も少ない色の辺数) が辞書順で遷移前以下になる
    - 数秒間回した場合に WA になるケースを作成できなかった

      (※ただし, 遷移条件において辞書順で一致する場合も遷移しないと登りきらなかった)

ジャッジ解

- mtsd (C++, N^2M) : 240 lines, 7031 bytes
- smiken (C++, N^2M) : 301 lines, 6277 bytes
- hos (C++, N^2M logN) : 291 lines, 7554 bytes
- hotman (C++, NM^2) : 101 lines, 2234 bytes
- mtsd (C++, 山登り) : 174 lines, 4935 bytes

統計情報

- AC teams / Trying teams
  - 0 / 2
- First Acceptance
  - なし