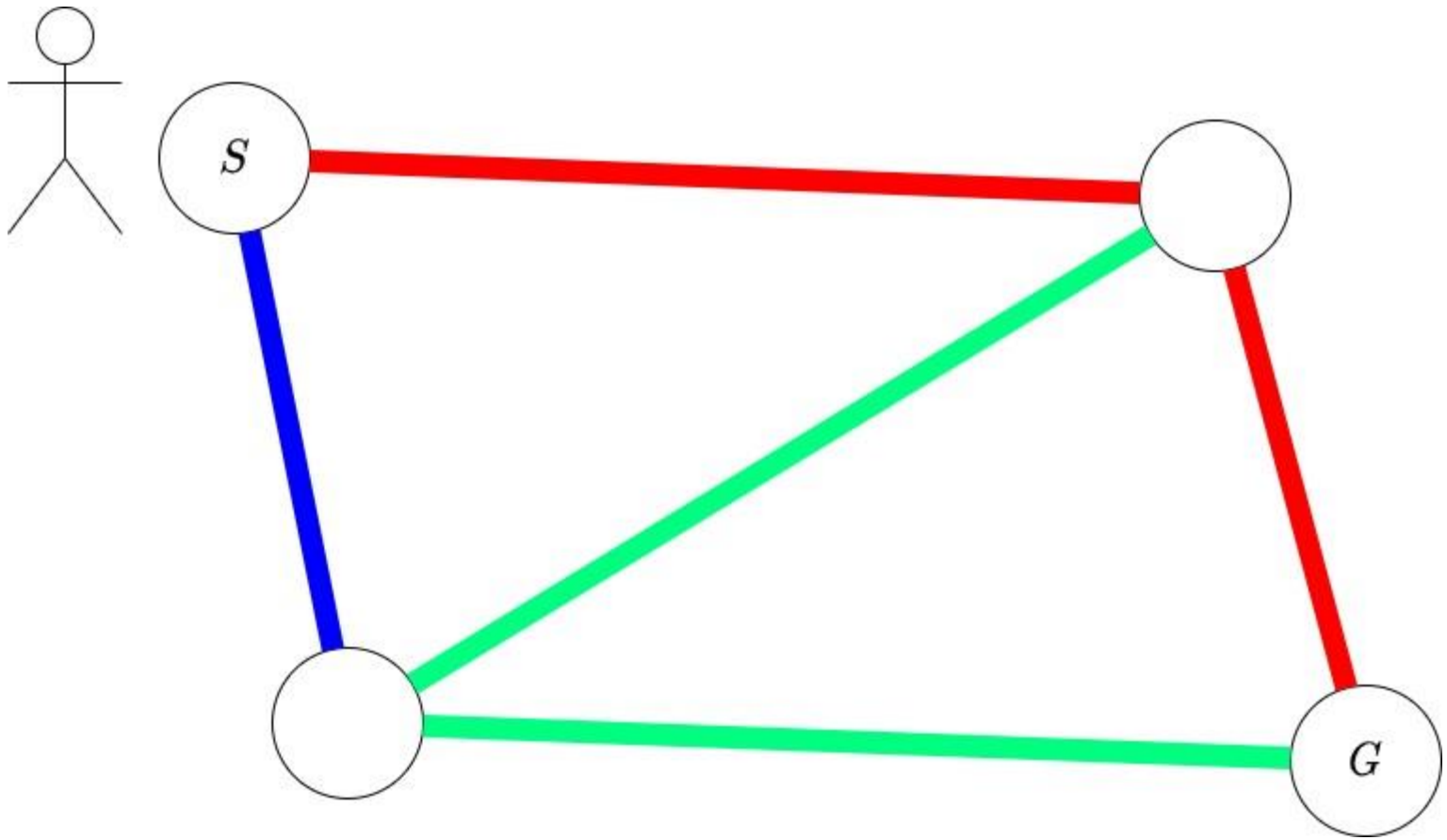
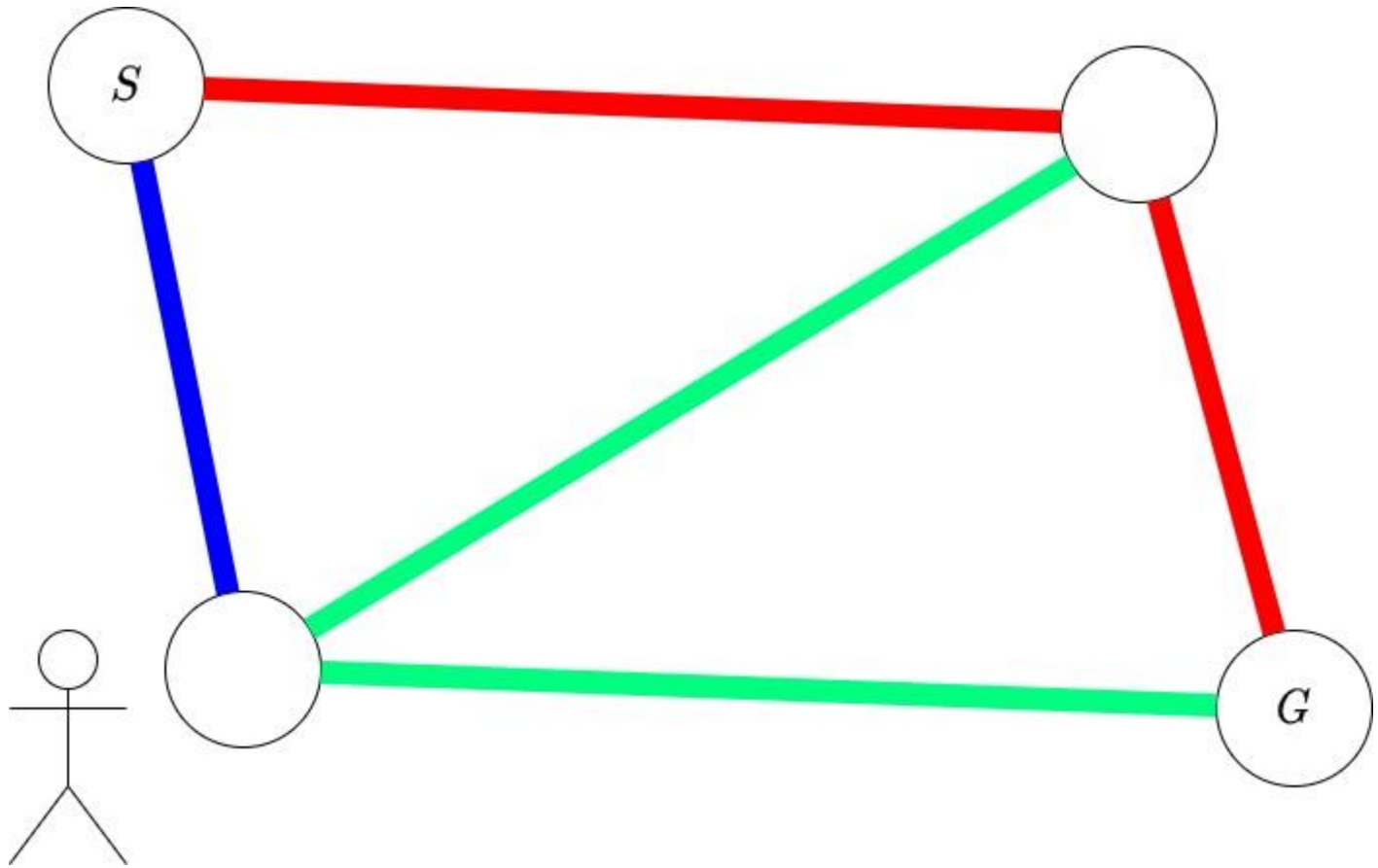
JOI 2020/2021 本選 4
ロボット (Robot)

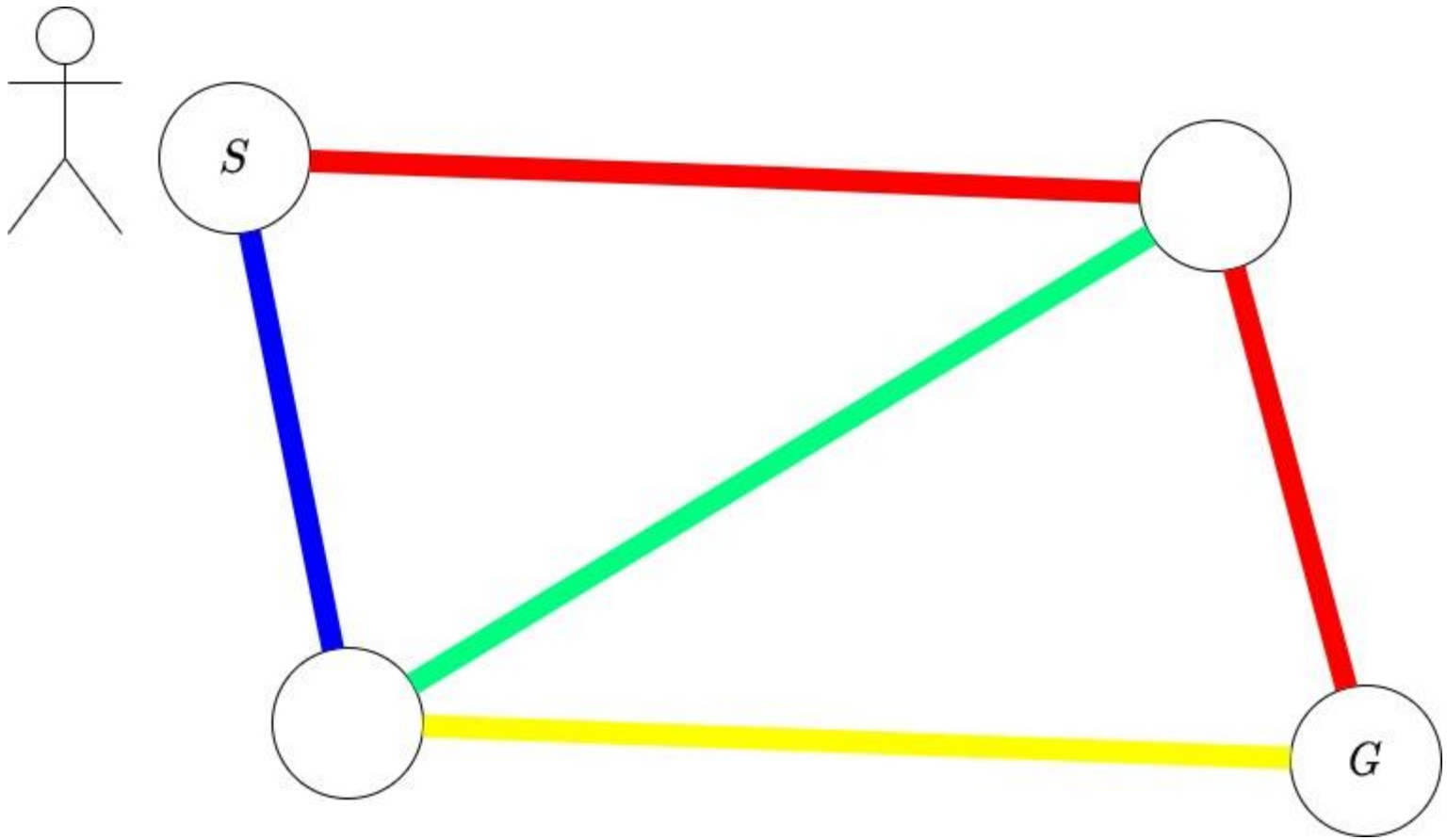
解説: 木ノ下 恭範

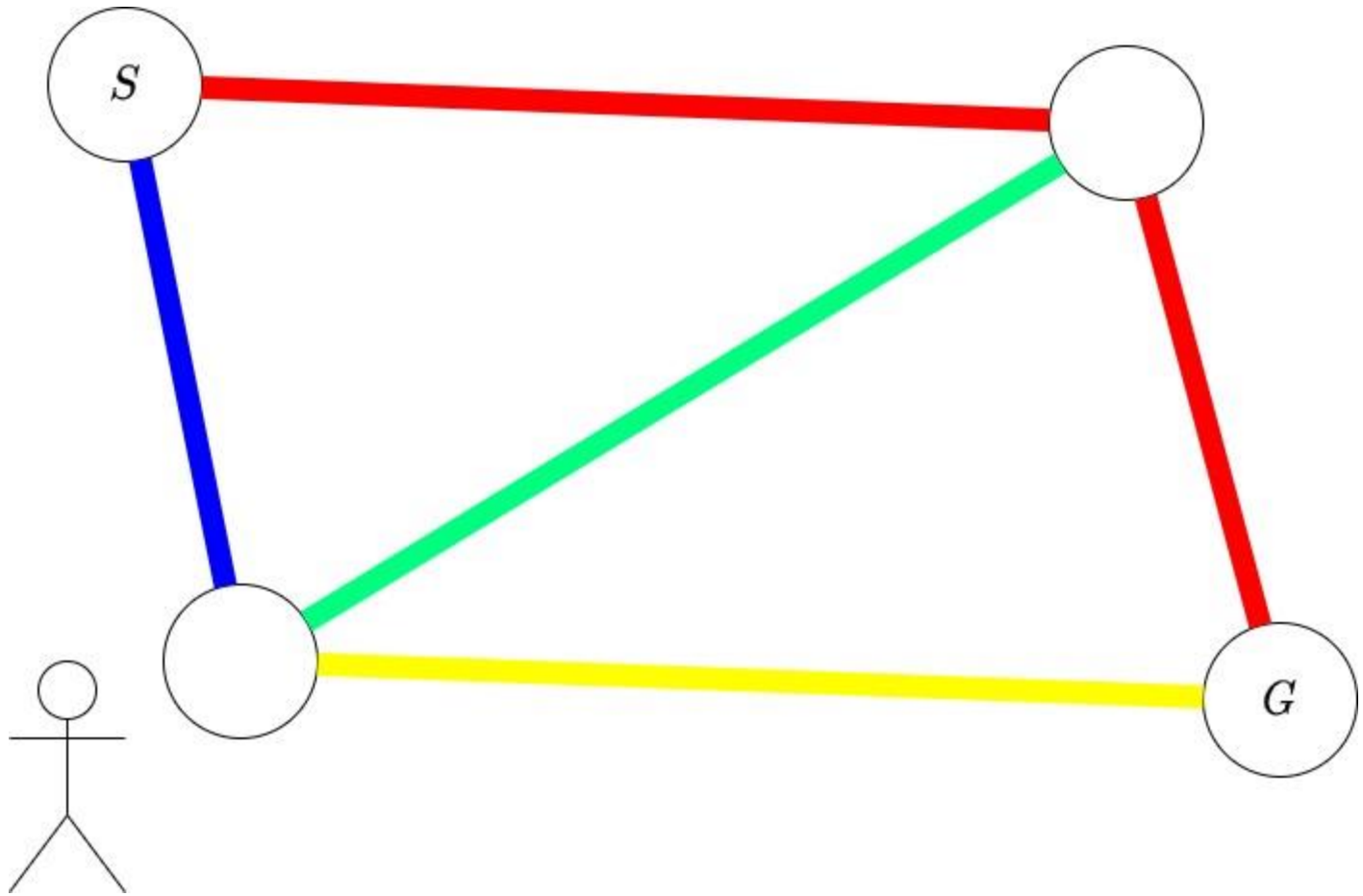
問題概要

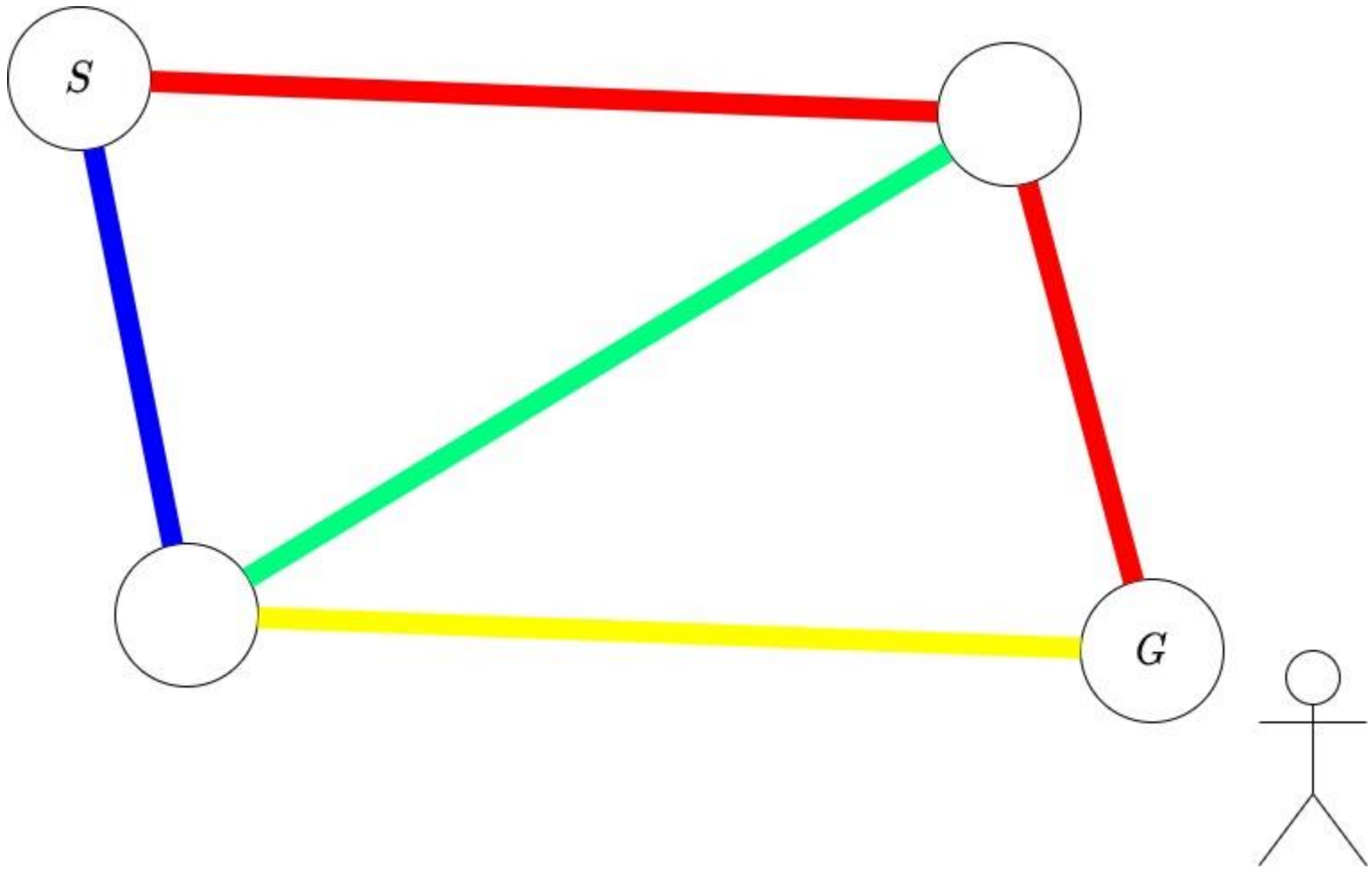
- 辺に色のついた単純無向グラフが与えられる。
- ロボットに色を指示すると、「ロボットがいる頂点から、その色の辺が丁度 1 本伸びている」ときに限り、その辺を選んで進む。
- ロボットを 1 から N に届けられるように、事前にいくつかの辺を塗り替えたい。
- それぞれの辺について、塗り替えに掛かる費用が決まっているので、費用の合計を最小化してください。


S
G


S
G


S
G


S
G


S
G

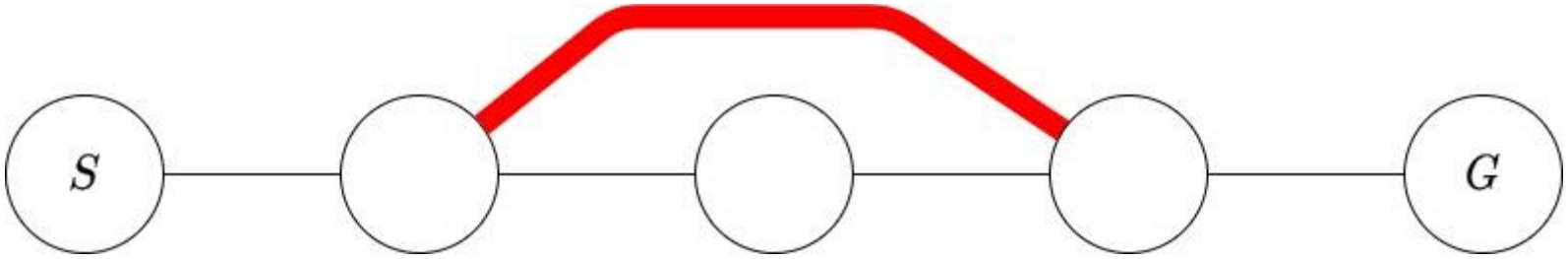
観察

- 色は M 色あるので、必ず、他のどの辺とも異なる色に塗り替えるとしてよい。
- 全ての辺を塗り替えれば、1 と N が非連結な場合以外、常に移動可能。

観察

- 色は M 色あるので、必ず、他のどの辺とも異なる色に塗り替えるとしてよい。
- 最適な塗り替え方の上で 1 から N への最短経路を考える。
- これは同じ点を二度通らないパスになる。
- パス上の頂点に接続していない辺は塗り替えていない (塗り替えても得るものが無いため)。
- パス上で距離が 2 以上離れた点同士をつなぐような辺は、塗り替えられてない。

- 他のどの辺とも異なる色に塗り替えるとしてよい。
- 最適な塗り替え方の上で 1 から N への最短経路を考える。
- パス上で距離が 2 以上離れた点同士をつなぐような辺は、塗り替えられてない。: もしそうだとすると、他と異なる色なのでこの辺は自由に通ることが出来るため、より短い経路が見つかってしまう。



観察

- 最適な塗り替え方の上で 1 から N への最短経路を考える。
- これは同じ点を二度通らないパスになる。
- パスの頂点に接続していない辺は塗り替えていない (塗り替えても得るものが無いため)。
- パス上で距離が 2 以上離れた点同士をつなぐような辺は、塗り替えられてない。

→ パスを通りながら、必要になってから色を塗り替えるとしてよい。さらに、今までに塗り替えた辺は、直前の頂点と今いる頂点を結ぶ辺以外覚えなくてよい。

小課題 1

- N <= 1000, M <= 2000
- ある頂点から辺を選んで進むとき、大きく分けて 2 つの方法がある。
  - X: 進みたい辺を塗り替えて進む。
  - Y: 進みたい辺以外で同じ色の辺を全て塗り替えて進む。
- 今までに塗り替えた辺は、パス上で直前の 1 本しか気にしなくてよいことに注意すると、この 2 つ以外は考えなくてよい。
- (問題文のグラフとは異なる) グラフ上での最短路問題に帰着することが出来る。

小課題 1

- (頂点 v に居る, 辺 e が塗られている) を頂点とするグラフで、最短距離を求めればよい。
- 塗られている辺によって、移動 Y のコストが安くなる可能性があることに注意。
- 頂点の数は N+2M, 辺の数は (N+2M)N 程度になる
- Dijkstra 法を用いると O(NM log(NM))

小課題 2

- P ＝ 1
- 0 円で進めないのなら、移動 X を使ってしまえばよい。次に移動するときに手前の辺が塗られているのを活用できる可能性がある分、移動 X は移動 Y より優秀であるため。
- 移動 X を行うなら、直前に何が塗られていたかは忘れても良い。移動 Y を行うときも、塗られている辺と進みたい辺の色が同じときだけが問題である。

小課題 2

- 移動 X を行うなら、直前に何が塗られていたかは忘れても良い。移動 Y を行うときも、塗られている辺と進みたい辺の色が同じときだけが問題である。
- (v に居る, e が塗られている) から (v に居る, どこも塗られていない) にコスト 0 の辺を張ると、(v に居る, e が塗られている) からは、e と同じ色の辺への移動 Y だけを考えればよい。
- さらに、e と同じ色の辺が v に 3 本以上接続しているなら、移動 Y はしない。よって、移動 Y は高々 1 つ。

小課題 2

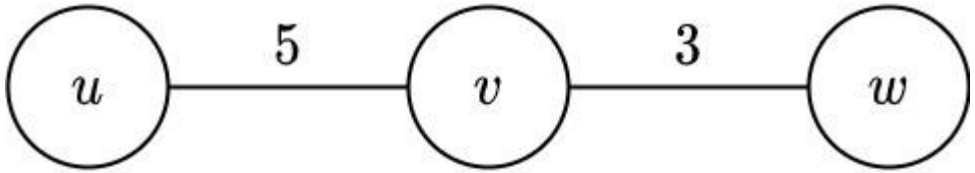
- 頂点の数 N+2M, 辺の数 4M+2M+2M 程度のグラフ上で最短距離を求めればよい。
- O(M log(M))

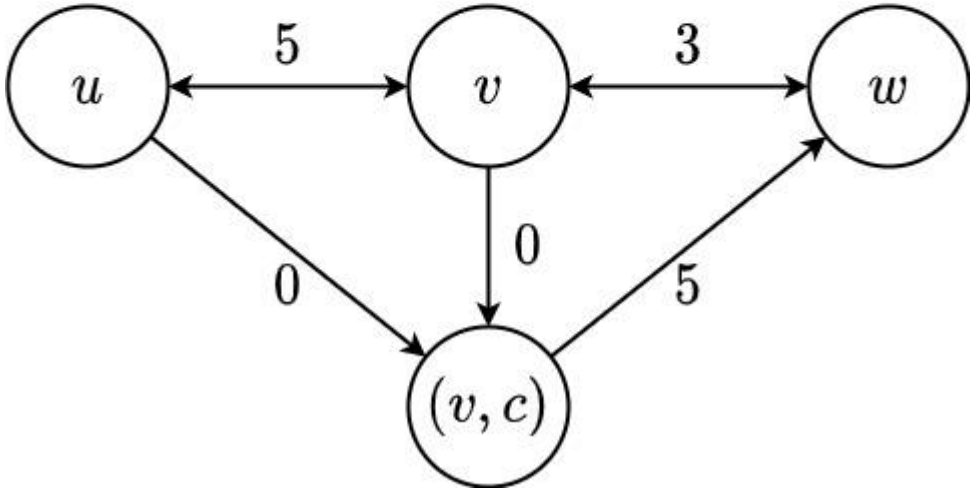
小課題 3

- (再掲) 直前に塗られた辺が必要になるのは、同じ色の移動 Y を行う場合のみである。
- u → v (移動 X), v → w (移動 Y) という遷移を、全ての u に対して上手くまとめ上げる。
- u → v の移動 X をコスト 0 ということにしてしまい、v → w の移動 Y において uv を塗ったことを忘れて普通にコストを計算すれば、つじつまがあう。
- (v に居る, 次は色 c についての移動 Y を行う) という頂点を追加し、u からコスト 0 で遷移させればよい。

小課題 3

- (v に居る, 次は色 c についての移動 Y を行う) という頂点を追加し、u からコスト 0 で遷移させればよい。





小課題 3

- (v に居る, 次は色 c についての移動 Y を行う) という頂点を追加し、u からコスト 0 で遷移させればよい。
- 頂点の数は N+2M 以下、辺の数は 2M+4M 程度になる。
- O(M log(M))

- 全体を通して、番号の振りにくい頂点を上手く管理する必要があります。
- C++ なら std::map<std::pair<int, int>, int> などを活用すると良いかもしれません。

小課題 3 (別解)

- 頂点 v に接続する色 c の辺のうち、重みが上位 2 つに入ってない辺へは、移動 Y を行う意味がない。一番重い辺が直前に塗られていたとしても、なお移動 X の方が安いため。
- 小課題 1 で作成した拡張グラフの辺を削減する。
- (v に居る, e を塗った) から (v に居る, 塗られた辺なし) にコスト 0 で辺を張る。
- (v に居る, e を塗った) からは、v に接続する e と同じ色の辺のうち、重み上位 2 つへの移動 Y を行う。
- 辺の本数が線形になり、十分高速。

得点分布