前処理

アルゴリズム論

中野

Note 8　前処理

2020.5.11 作成

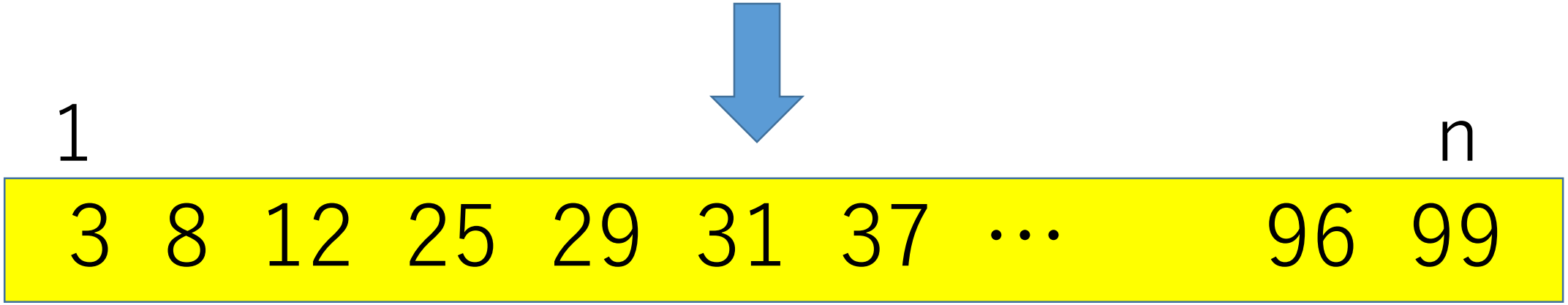
2020.6.14 update 6.15　2021.5.31

今回の講義の概要

前処理(preprocessing)を使う手法.
問題をk回解くとき,
前もって**一回だけ**準備をするのに苦労すれば,
あとのk回それぞれについては, ラクに解けるよ！
という手法です.

（学生時代は、前処理の時代です。。。）

2分探索



ソート　　O(n log n)
探索　　O(log n)

vs

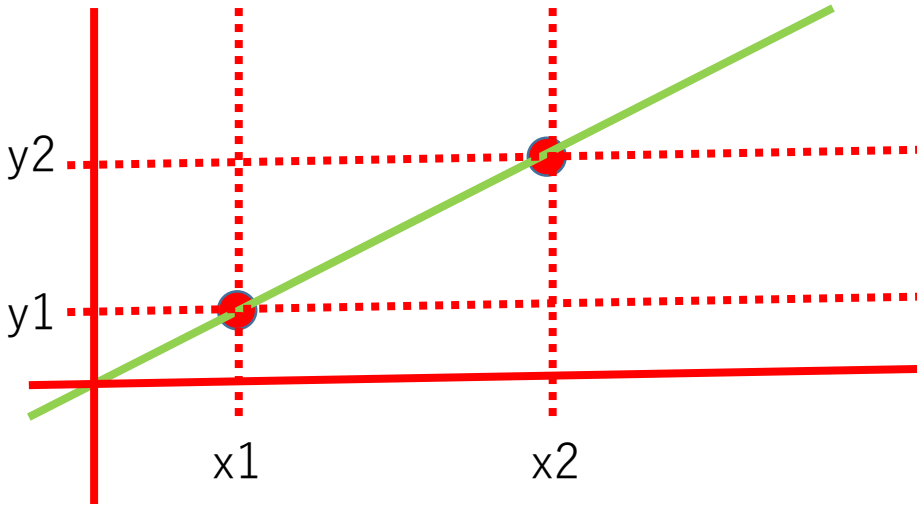
線形探索 O(n)

n	10^3	10^6	10^9	10^{12}
log n	10	20	30	40

CONVEX POLYGON INCLUSION問題(準備)

平面上の点の座標を(x,y)のように表す。xはx座標、yはy座標である。
点(0,0)から点(x1,y1)への有向線分 p1と、
点(0,0)から点(x2,y2)への有向線分 p2が　あるとしよう。
x1,x2,y1,y2は0でないとしよう。
もし、p1とp2が、同一線上にあるならば、
　　x1 : y1 = x2 : y2　であるから
　　x1・y2 = y1・x2　である。
また、上記等号が成立しないときには、
x1・y2 - y1・x2　の符号を調べることによって、
有向線分 p1の、左右のどちらに有向線分 p2があるかどうか
わかる。(ホントはコレにも証明が必要です。)

つまり、同一点を始点とする2本の線分が与えられたとき、定数個の計算で、
一方が他方の左右どちらにあるか判定できる。

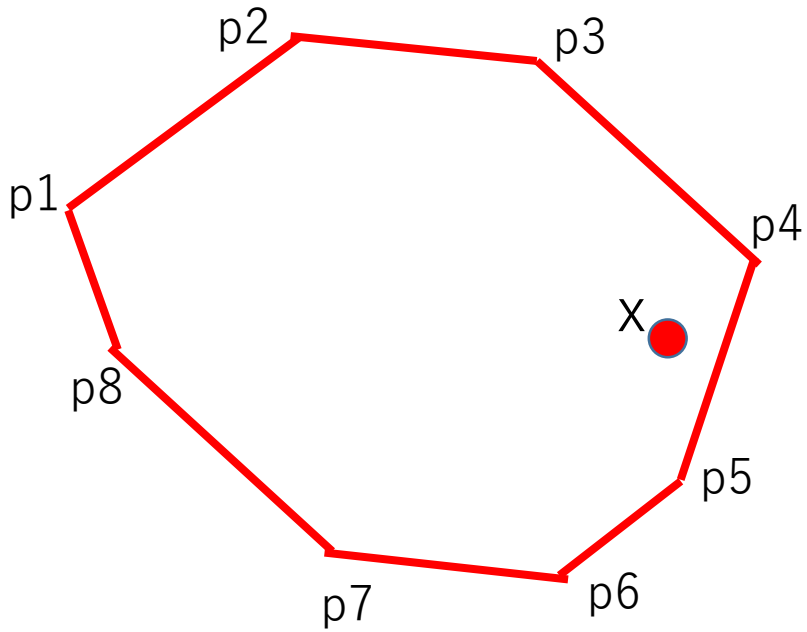


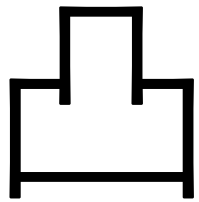
CONVEX POLYGON INCLUSION問題

CONVEX POLYGON INCLUSION問題

入力: 凸n角形Pの外周を構成する
　　n本の線分 と 1点x

出力: x は P の内部にあるかどうか
　　判定する



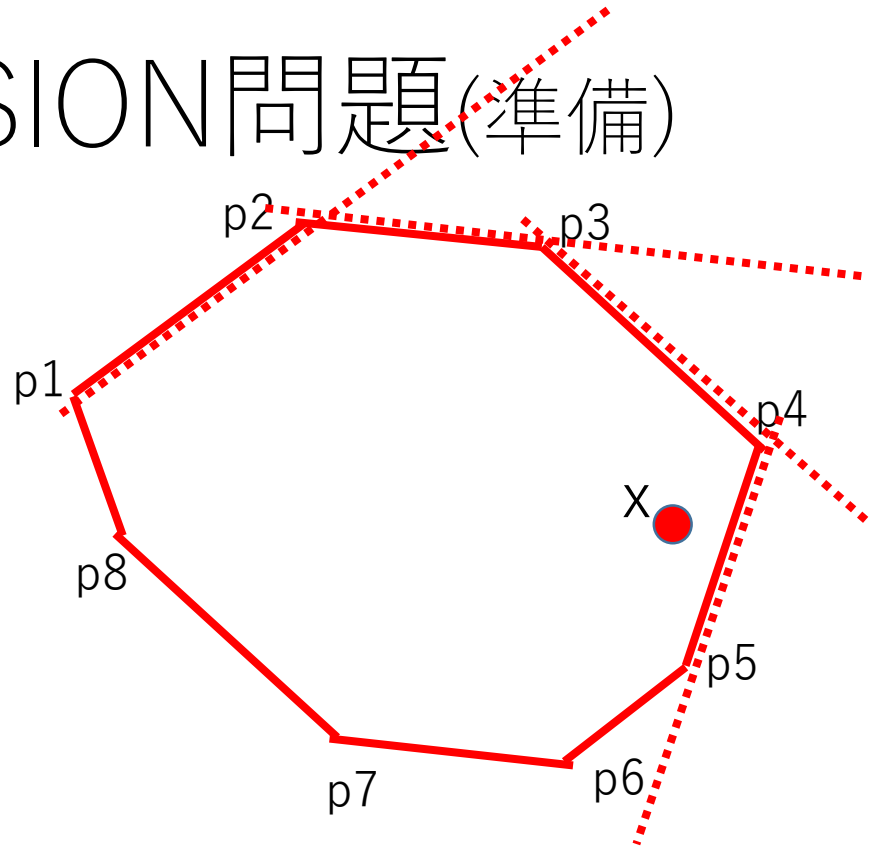


CONVEX POLYGON INCLUSION問題(準備)

素朴なアルゴリズム

凸n角形の外周上の各線分に対して、
点xが右にあるかどうかをチェック。
n回のチェックがすべてYESならば、
xはPの内部にある。
そうでないなら、xはPの内部にない。

計算時間は O(n)

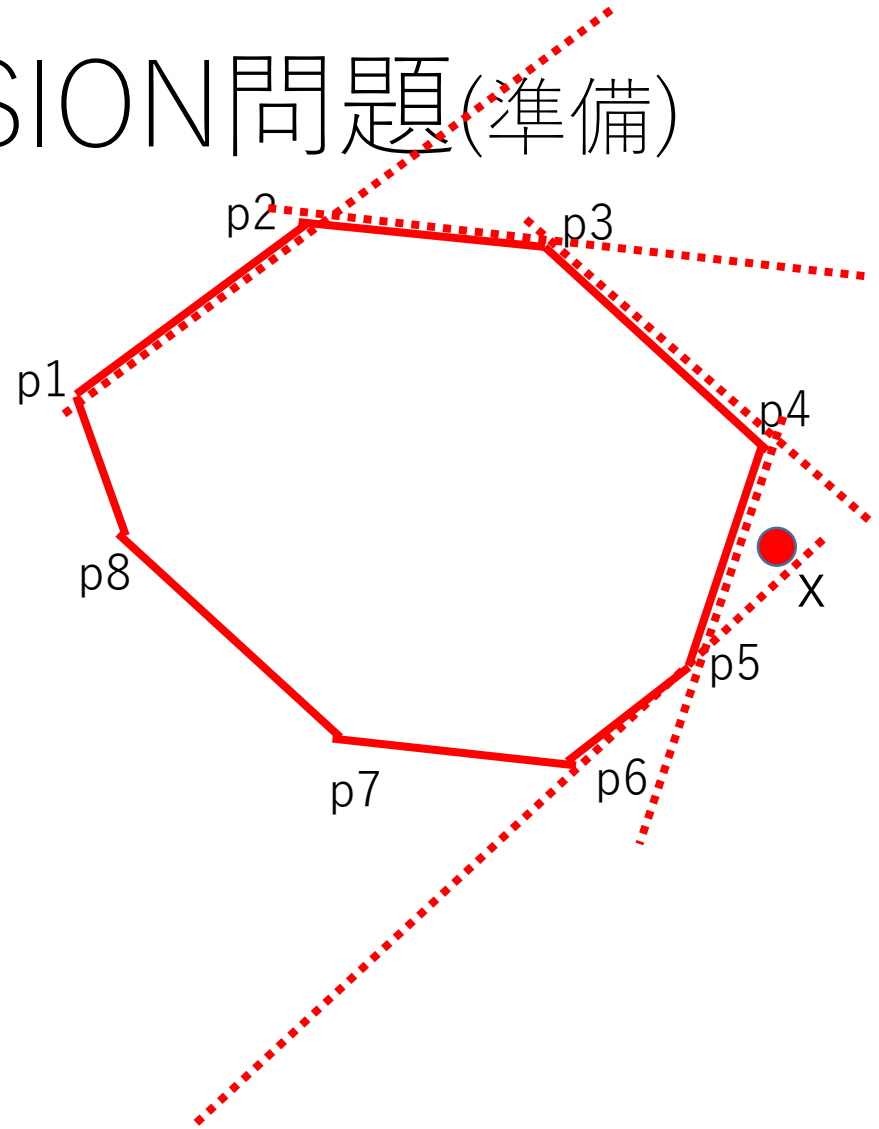


CONVEX POLYGON INCLUSION問題(準備)

素朴なアルゴリズム

凸n角形の外周上の各線分に対して、
点xが右にあるかどうかをチェック。
n回のチェックがすべてYESならば、
xはPの内部にある。
そうでないなら、xはPの内部にない。

計算時間は O(n)



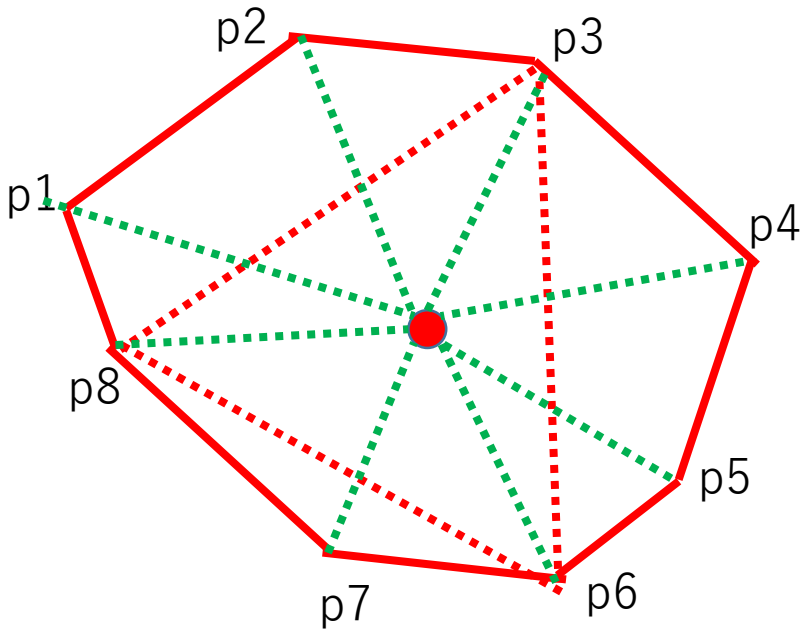
CONVEX POLYGON INCLUSION問題
(前処理を使ったアルゴリズム)

(前処理)
凸n角形Pの外周を構成するn本の線分上の、任意の3点を選ぶ。
この3点からなる3角形の重心を求める。
この重心を中心として、重心とP上のn点を結ぶn本の辺を書き、角度でソートし、
外周上に時計回りに現れる順に並べる。
(前処理終 以上でn log n時間)

2分探索にて、xがどのクサビ型(もしくはピザカット)にはいっているか調べる。
そのクサビ型内の外周上の線分の左右いずれにxがあるか判定し、
xがPの内部にあるかどうか判定する。
(この部分は log n 時間)

もし、同じ、Pに対して、点 x1,x2,...,xkが
Pの内部にあるかどうか調べるときは、前処理は一回だけでよいので、
kが大きければ、素朴なアルゴリズムよりも効率的になる。

このアルゴリズムは簡易に星型n角形にも適用できる。



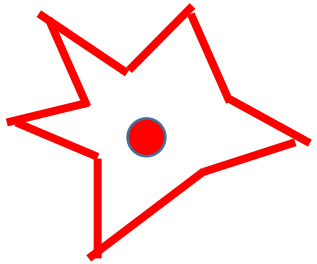
CONVEX POLYGON INCLUSION問題
(前処理を使ったアルゴリズム)

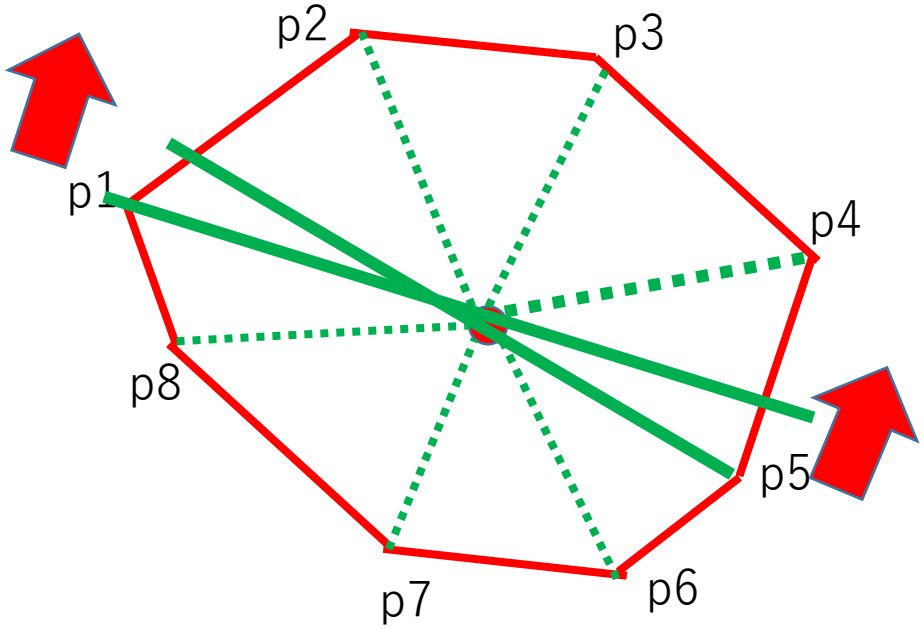
(前処理)
凸n角形Pの外周を構成するn本の線分上の、任意の3点を選ぶ。
この3点からなる3角形の重心を求める。
この重心を中心として、重心とP上のn点を結ぶn本の辺を書き、角度でソートし、
外周上に時計回りに現れる順に並べる。
(前処理　おしまい　以上でn log n時間)

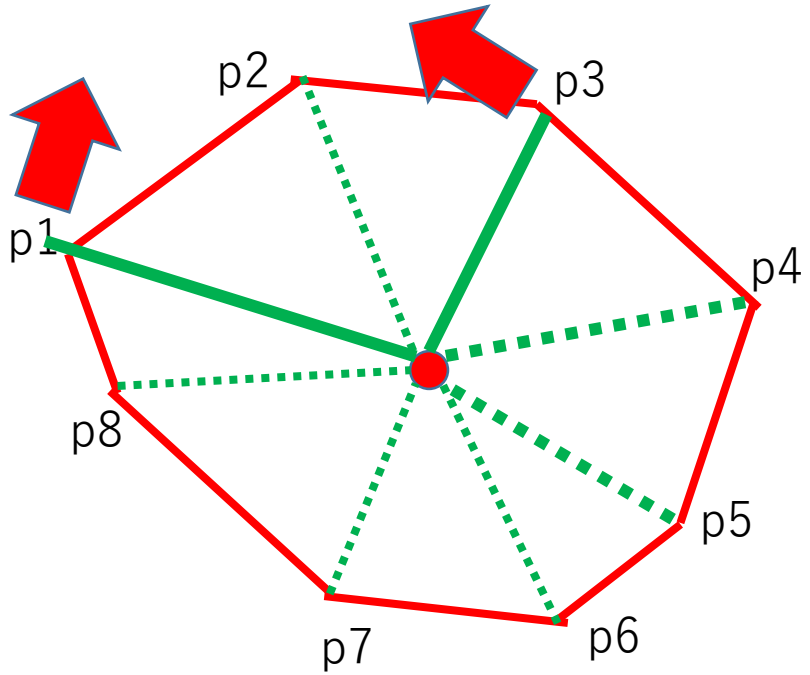
2分探索にて、xがどのクサビ型(もしくはピザカット)にはいっているか調べる。
そのクサビ型内の外周上の線分の左右いずれにxがあるか判定し、
xがPの内部にあるかどうか判定する。
(この部分は　1+ log n　回の判定が必要)

もし、同じPに対して、点 x1,x2,...,xkが
Pの内部にあるかどうか調べるときは、前処理は一回だけでよいので、
kが大きければ、素朴なアルゴリズムよりも効率的になる。

このアルゴリズムは簡易に星型n角形にも適用できる。

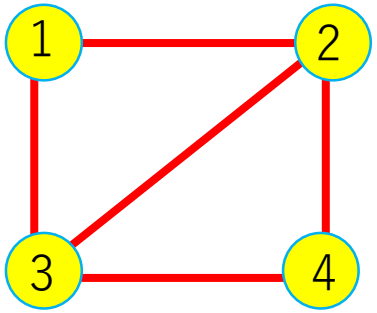






(もっと、巧妙な手法を使うと
(文献
Computational Geometry: An introduction
Author:　Franco P. Preparata, Michael Ian Shamos
Hardcover, 398 pages, ISBN: 338796131-3の
page56)
平面グラフを各辺が直線であるように平面に描画したとき、
与えられた点が、どの面にはいるかという問題も
O(n lon n)時間の前処理を使って、O($\log^2 n$)時間で解ける。

平面グラフの辺を(1)時間でさがす

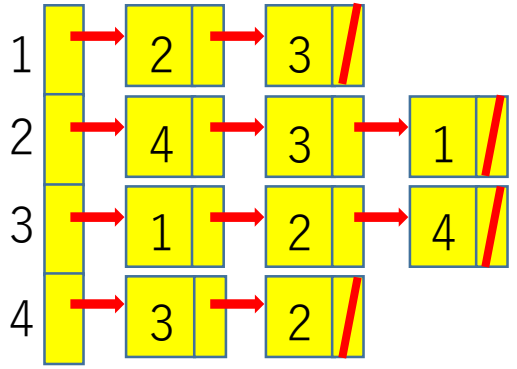


グラフを表現するデータ構造には、
隣接行列と 隣接リストがある。

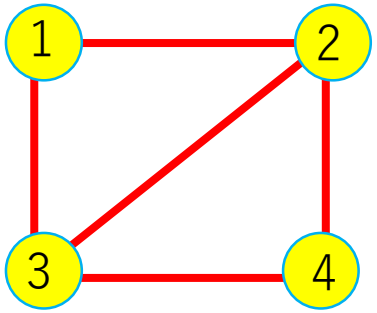
隣接行列は $O(n^2)$のメモリを必要とする。nは点の個数。
2点u,vが与えられたとき、この間に辺があるかどうかを
O(1)時間で判定できる。

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

隣接リストは O(m)のメモリを必要とする。mは辺の本数。
2点u,vが与えられたとき、この間に辺があるかどうかを
O(d)時間で判定できる。
dはグラフの最大次数。
(点vの隣接点の個数をvの次数という。)



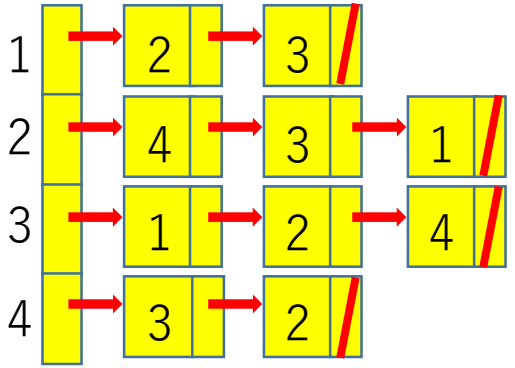
平面グラフの辺を(1)時間でさがす

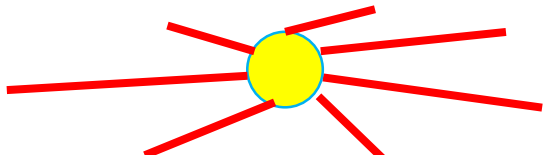


平面グラフにO(n)時間の前処理をすることにより、
O(n)のメモリを使って、
2点u,vが与えられたとき、この間に辺があるかどうかを
O(1)時間で判定できるようなデータ構造を作る。

	1	2	3	4
1	0	1	1	0
2	1	0	1	1
3	1	1	0	1
4	0	1	1	0

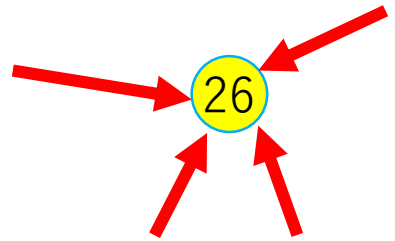
平面グラフとは、平面に辺の交差なく描画できるグラフです。
オイラーの公式よりn - m + f = 2　である。
nは点の個数、mは辺の本数、fは面の個数である。
全ての面は3本以上の辺で囲まれているから3f≦2mである。
この2式よりfを消すと　m ≦ 3n – 6 である。
である。つまり、平面グラフには 5次以下の点がある。
すべて6次以上だと　6n ≦ 2m となり上式に矛盾する。





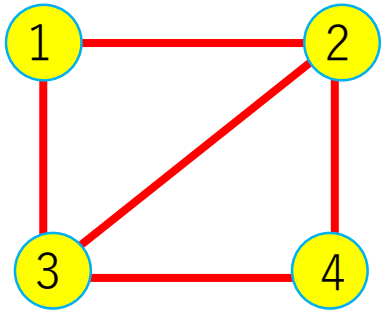
平面グラフの辺を(1)時間でさがす

平面グラフGの各辺に
向きを以下のようにつける。

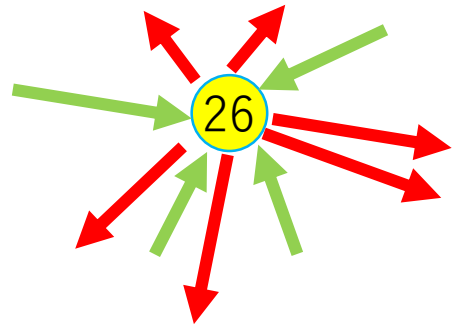


G中の5次以下の点vをひとつ取り除く。
(vに接続していた辺にはvに向かう向きをつける。)
残りのグラフも平面グラフなので、やはり、5次以下の点がある。
同様に5次以下の点vをひとつ取り除く。
(vに接続していた辺にはvに向かう向きをつける。)
。。。くりかえし。。。
これで、全部の辺に向きがついた！
かつ、各点に向かう辺は高々5本である。

平面グラフの辺を(1)時間でさがす



n x 5 の2次元配列Aを作成し、
A[v,1],A[v,2],A[v,3],A[v,4],A[v,5]に、
vに向かう辺の番号と辺の両端点を記入する。
この配列はO(n)のサイズをもつ。　n行5列



2点u,vが与えられたとき、
この間に辺があるかどうかは、
2次元配列Aの、10か所
A[u,1],A[u,2],A[u,3],A[u,4],A[u,5],
A[v,1],A[v,2],A[v,3],A[v,4],A[v,5],
をチェックすればわかる。
すなわち、O(1)時間で判定できる。