情報実験Ⅰ：最適化手法

電子情報システム学科　相場　亮

「最適化手法」とは（1）

- **最適化手法**
  - ある制約条件のもとで特定の状態に到達することを目的とし、それを達成するための方法
- **最適化手法の重要性**
  - 最適化手法はほとんどあらゆる工学・産業に深く関連する非常に重要な技術である

「最適化手法」とは（2）

- **最適化の方法**
  - 「最適化問題」を解くには、問題自体が成立している分野の性質などによって非常に多くの技法がある
- **情報実験Ｉの目的**
  - この実験では、代表的な手法を実践的に学び、それによって最適化技術についての理解を深めることを目的とする

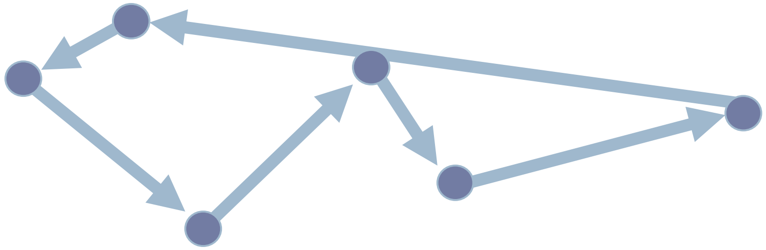
巡回セールスマン問題（1）

- **15年後、ある企業の主任プログラマになっているあなたはある日上司から海外出張を命じられた**
  - 出張の目的はある巨大プロジェクトの打ち合わせのため、参加企業の全てを回ることにある
- そこであなたは目的地を全て1回ずつ訪れ、かつその道筋が最短になるような出張経路を見つけなければならなくなったとしよう

巡回セールスマン問題（2）

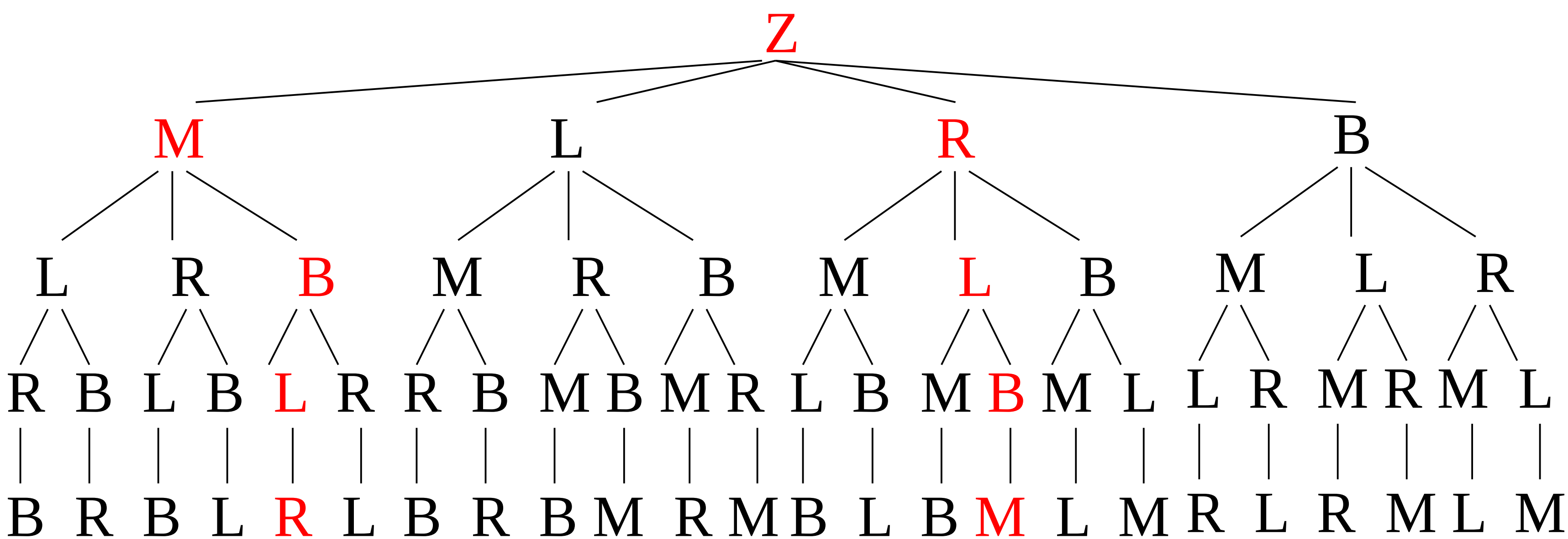
- こういった問題のことを「巡回セールスマン問題 (Traveling Salesman Problem : TSP) という
  - TSPは最適化問題のひとつの典型である

目的地を全て1回ずつ訪れ、かつその道筋が最短になるような出張経路を見出す



巡回セールスマン問題（3）

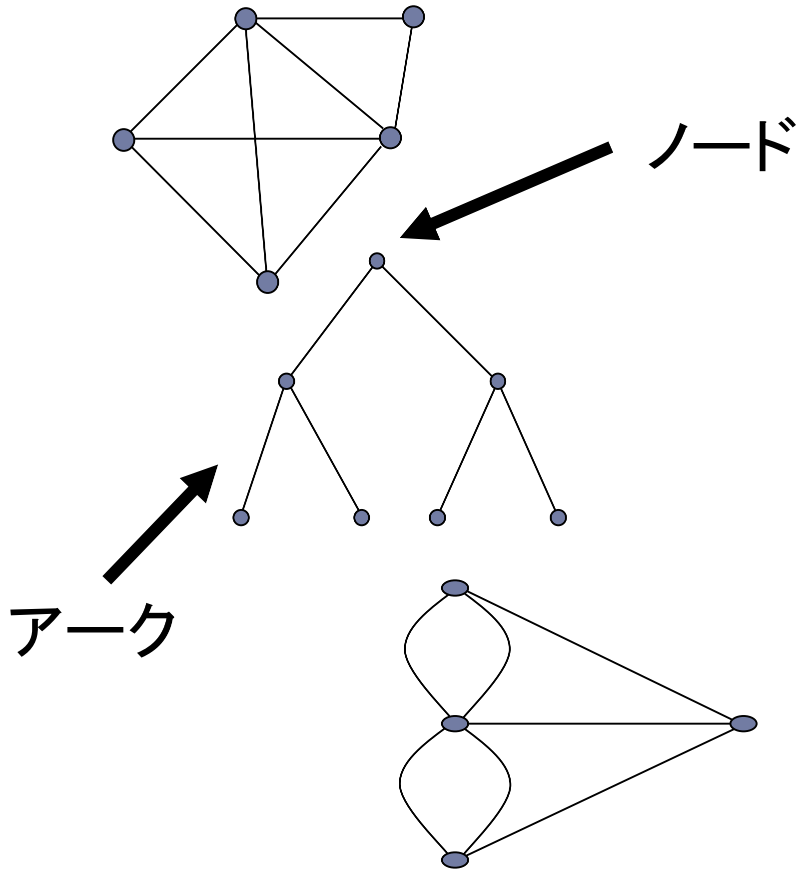
- たとえばチューリッヒを出発点とし、 マドリッド，ロンドン，ローマ，ベルリンを回るとすると全ての巡回路は次のようになる



最短経路はZ→R→L→B→M→Zまたはその逆順で3047マイルとなる

TSPの定義（１）

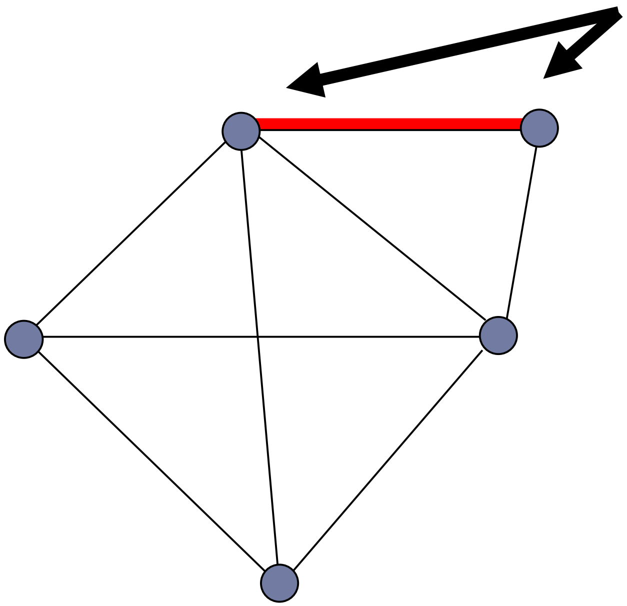
- **TSPの定義**
  - ではここでTSPを数学的にきちんと定義してみる
- **グラフ**
  - グラフとは、**点**および点の間に引かれた**線**とから構成されるものである
  - グラフの点のことを***ノード*（node）**、線のことを***アーク*（arc）**と呼ぶ



TSPの定義（2）

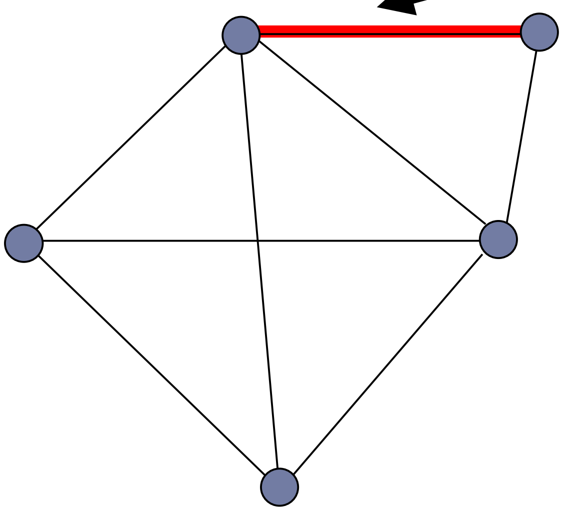
- アークの両方の端にあるノードは**互いに隣接している（adjacent）**という

これらのノードは互いに
隣接しているという



TSPの定義（３）

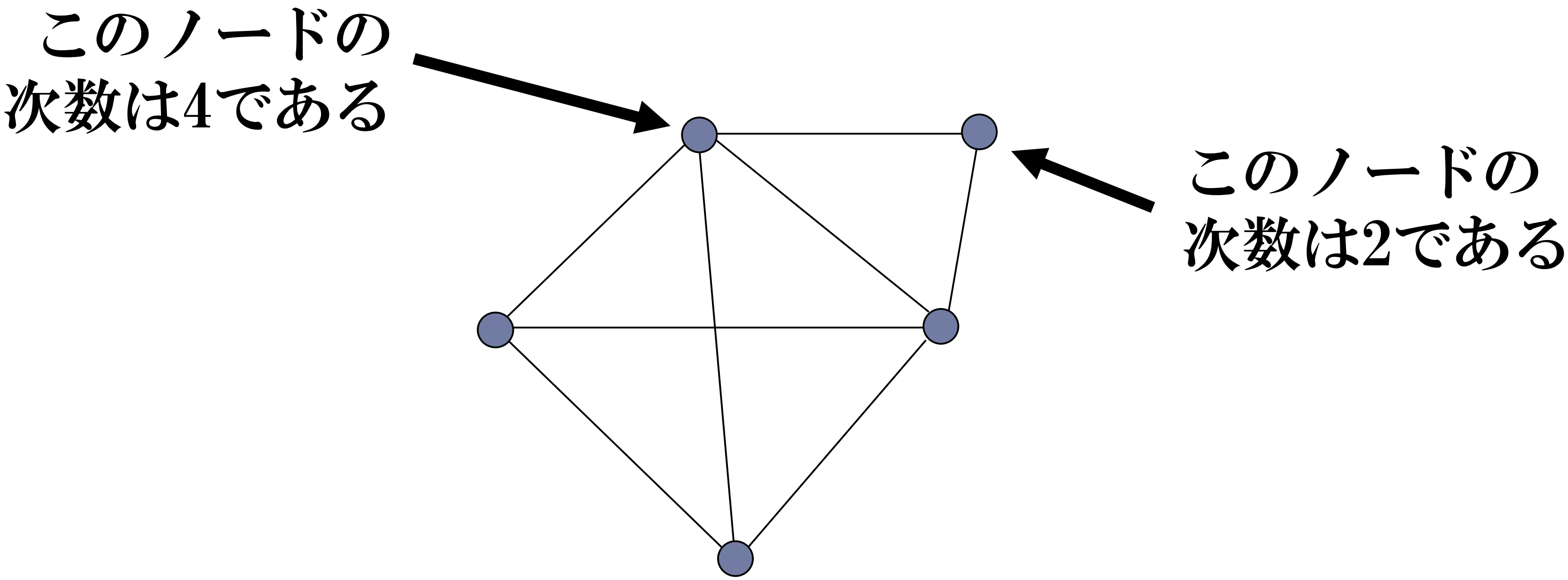
- また、アークはその両端のノードに**接続している (incident)** という



このアークは両端のノードに接続しているという

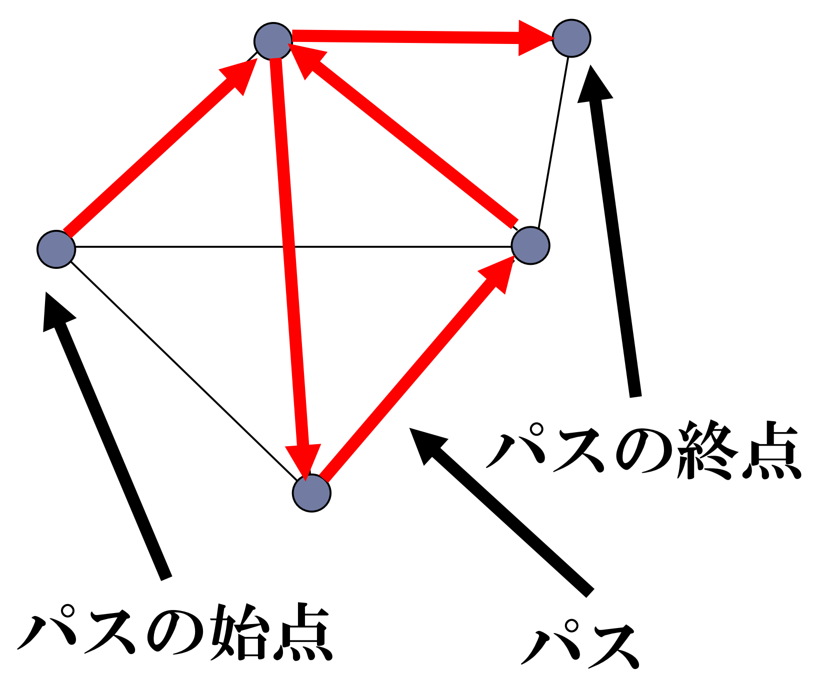
TSPの定義（4）

- あるノードに接続しているアークの本数のことを**そのノードの次数（degree）**という



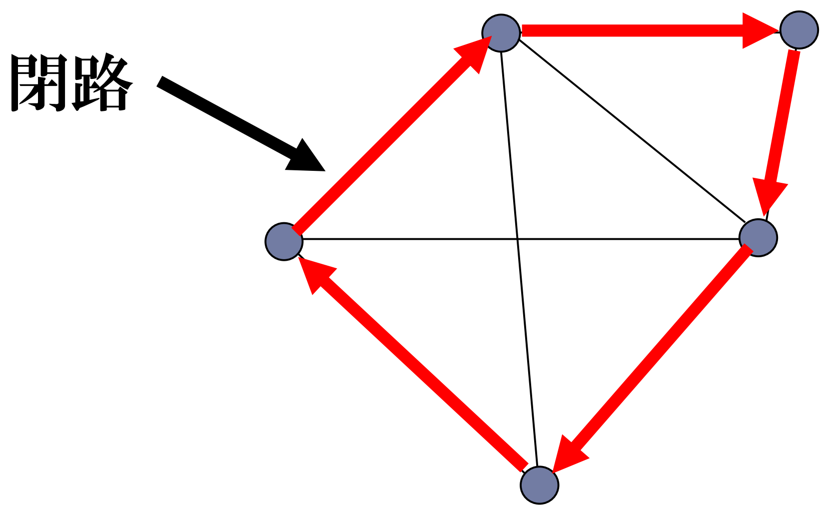
TSPの定義（５）

- グラフの上をノードからアークを伝ってノードを巡る、ノードとそれに接続するアークが交互に並んだものを**パス（path）**という
- パスの最初のノードをパスの**始点（source）**という
- パスの最後のノードをパスの**終点（sink）**という



TSPの定義（６）

- 始点と終点が一致しているようなパスのことを**閉路 (circuit)** という



TSPの定義（７）

- これらの定義のもとで巡回セールスマン問題（TSP）とは次のような問題である

> グラフとそのグラフの全てのアークの長さが与えられたとき、そのグラフの全てのノードをちょうど1回ずつ経由するような閉路（これをHamilton閉路と呼ぶ）のうち、アークの長さの合計（閉路の長さ）が最小になるものを求めよ

TSPを解く（１）

- **問題を解く前に考えること（１）：計算の手間**
  - まず問題を解く前に、この問題にはどういう性質があるか考えてみる
  - 巡回セールスマン問題を解く最も単純な方法は次によるものである

アルゴリズム1

１．適当に出発点を選択し、その出発点からの閉路を全て求める
２．求めた全ての閉路の全長を計算する
３．閉路の全長が最も短いものを選択する

TSPを解く（２）

- この最も単純な解法について考察してみよう
  - 「全ての閉路を求める」と書いたが、閉路は一体いくつあるのだろうか？
  - 都市数を N とすると、出発点から次の都市を選択する場合の数は $N-1$ だけあることになる
- その次の都市の選択に関する場合の数は$N-2$である
- その次は $N-3$ となる…
- このように考えていくと、出発点を固定した場合、全ての閉路の個数は $(N-1)!$ あることになる

$$(N-1)! = (N-1)\times(N-2)\times\ldots\times 2\times 1$$

TSPを解く（3）

- この表を見てもわかるように、$N!$ という数は N の値が大きくなるにつれ、急激に大きくなっていく
  - たとえば15都市の場合、1都市の場合の約1兆3千億倍の時間がかかることがわかる

N	*N!*
1	1
2	2
3	6
4	24
5	120

N	*N!*
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800

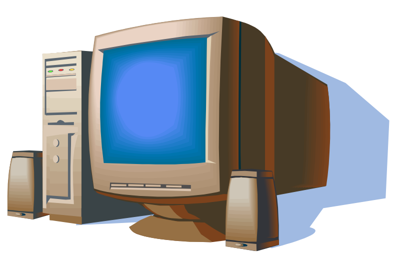
N	*N!*
11	39,916,800
12	479,001,600
13	6.227.020.800
14	87,178,291,200
15	1,307,674,368,000

TSPを解く（４）

- このように巡回セールスマン問題とは単純な解法を採用すると、都市の数が増えると急激に計算時間が増加するような性質を持つ

実験１：TSPの複雑さ（1）

- TSPの複雑さを実感してみよう
  - 現在のコンピュータの速度はおよそ100MIPS (Mega Instructions Per Second) だといわれ、1秒間に基本的な命令を1億回実行することができる
- コンピュータの性能は現状技術のままでどこまで向上するだろうか
  - 真空中の光速はおよそ3.0×10^{10}cm/秒である
  - 一辺が0.5cmのメモリ・チップを考えるとその一辺を光が横切るのにおよそ1.67×10^{-11}秒かかる

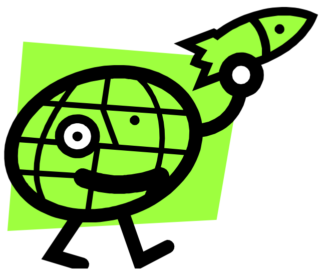


実験１：TSPの複雑さ（２）

- するとメモリから情報を取り出すのにも速くても1.67×10^{-11}秒かかる
- メモリから情報を取り出し、直ちに計算が行われると考えても、1秒間に実行できる命令はその逆数6.0×10^{10}回程度ということになる
- この速度は 60GIPS (Giga Instructions Per Seconds) である
- 60GIPSは現在のコンピュータの速度である100MIPSのわずか600倍に過ぎない

実験１：TSPの複雑さ（３）

- この60GIPSの速さのコンピュータが地球を埋め尽くしているとしよう
  - 地球の赤道半径は6378Km なので、地球を球体と考えると0.5cm × 0.5cmのチップをおよそ2×10^{19}個地球上に置くことができる
- それぞれのチップが60MIPSで、理想的な並列処理が可能だとするとこの「超地球コンピュータ」は1秒間におよそ1.2×10^{30}回の命令を実行することができる



実験１：TSPの複雑さ（４）

- しかもこの「超地球コンピュータ」は巡回セールスマン問題のために作られていて、ひとつの順回路の長さを1つの命令で計算することができるとする

実験１：この「超地球コンピュータ」を使ったとして、10都市、100都市、1000都市、そして世界記録の7397都市についてＴＳＰを解くのに必要な時間を求めなさい。ただし、nが大きな数の場合、n!の計算には次のスターリングの公式を使いなさい

$$n! \approx \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n$$

TSPを解く（５）

- 問題を解く前に考えること（２）：データ構造
  - データ構造の選択も計算時間に影響を与える場合があり、またデータを格納するための記憶領域の大きさにも関わってくる

●都市の配置をどのようなデータ構造で表わすか？
●閉路をどのようなデータ構造で表わすか？

TSPを解く（６）

- 都市の配置を表わすデータ構造について
  - 都市の配置を示す最も単純な方法は各都市の座標を用いて表わす方法である
  - しかしこの方法では各都市の位置を表わすのに少なくとも2つの浮動小数点数が必要になる
- そもそもこの問題を解くために各都市の座標が必要だろうか？
- 求めたいのは閉路の全長であり、これを求めるには各都市間の距離さえわかればいい
- では各都市間の距離をどのように表わしたらいいのだろうか？

TSPを解く（７）

- 都市の間の距離を$N \times N$の行列で表わす
  - データ量は各都市の座標を表わす場合が$2N$であるのに対しN^2だけ必要だが都市間の距離を計算する必要がなくなる
  - TSPでは計算時間が問題になるためデータ量を犠牲にしても計算時間を少なくしたい

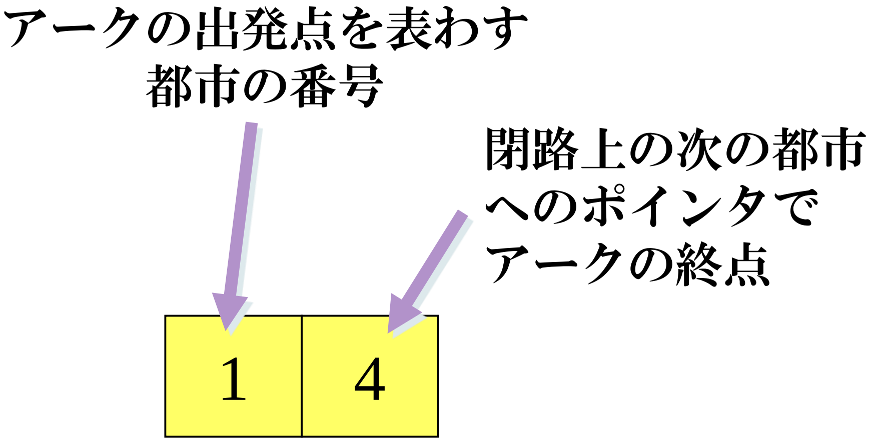
ここは都市3と都市3の間の距離なので0となる

$$[d_{ij}] = \begin{pmatrix} 0 & 3 & 8 & 4 & 6 & 1 \\ 3 & 0 & 6 & 3 & 3 & 3 \\ 8 & 6 & 0 & 4 & 3 & 9 \\ 4 & 3 & 4 & 0 & 1 & 4 \\ 6 & 3 & 3 & 1 & 0 & 5 \\ 1 & 3 & 9 & 4 & 5 & 0 \end{pmatrix}$$

ここは都市5と都市2の間の距離を示している

TSPを解く（８）

- 閉路を表わすデータ構造について
  - 閉路はノードとアークを交互に並べた列で出発点と終点が同じものである
  - 閉路を表わす単純な方法としてアークを次のようなデータ構造で表わす



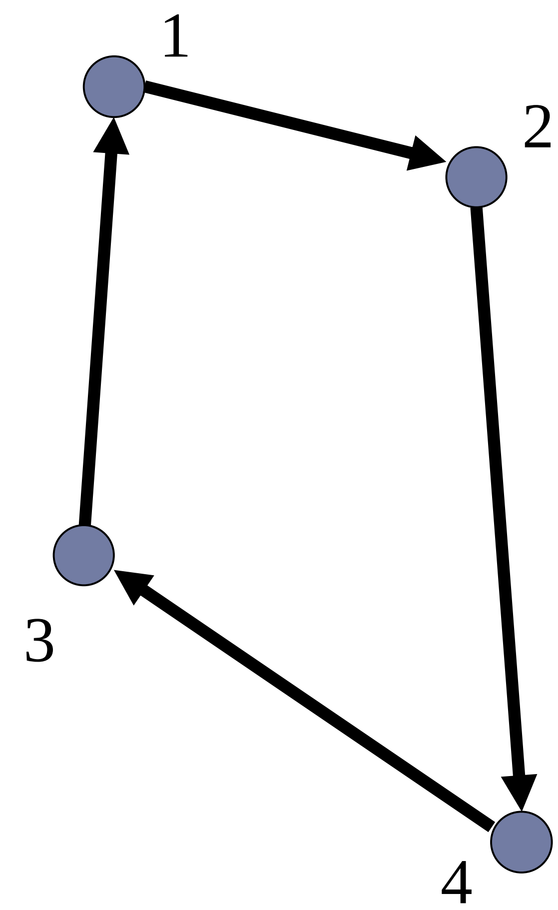
- この例では閉路上の都市1の次が都市4であることを表わしている

TSPを解く（9）

- 例えば次のようなデータは右図のような閉路を表わしていることになる

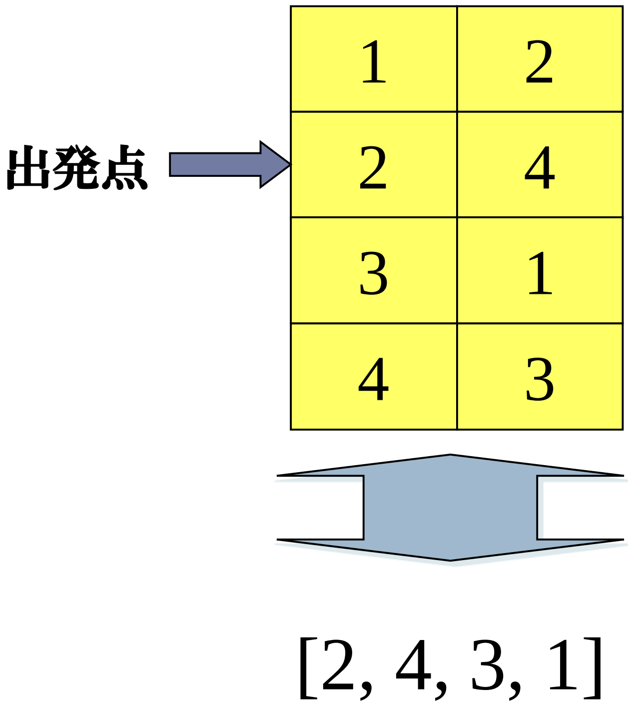
出発点

1	2
2	4
3	1
4	3



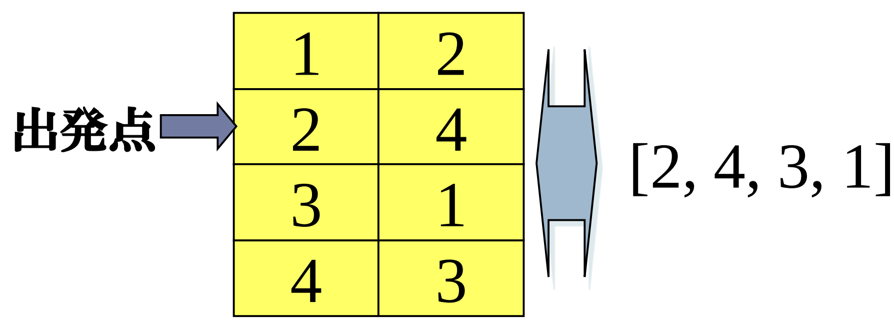
TSPを解く（１０）

- だがよく考えてみると
  - 閉路を表わすためには巡回する都市の番号が順番に決められていれば充分であるし、その方がデータ量も少ない
  - 前の例でいえば［2，4，3，1］という都市の順番さえわかれば閉路を決定することができる



TSPを解く（１１）

- この2つの表現方法を比較すると、データ量が半分で抑えられていることがわかる
  - またこのような方法で表わされた全ての閉路は都市の番号からなる列の「順列(permutation)」になる
- たとえば4都市からなるTSPであれば数列［1，2，3，4］の順列を求めれば全ての閉路を求めることができる



TSPを解く（１２）

- この「順列を用いた閉路の定義」を使うと、TSPは次のように定義することができる

TSPのもうひとつの定義：
n×nの行列［dij］が与えられたとき、V={1，2，…，n}からVへの1対1写像（これを順列と呼ぶ）ρのうち、以下を最小にするものを求めよ

$$\sum_{i=1}^{n-1} d_{\rho(i)\rho(i+1)} + d_{\rho(n)\rho(1)}$$

TSPの新しい定義について（1）

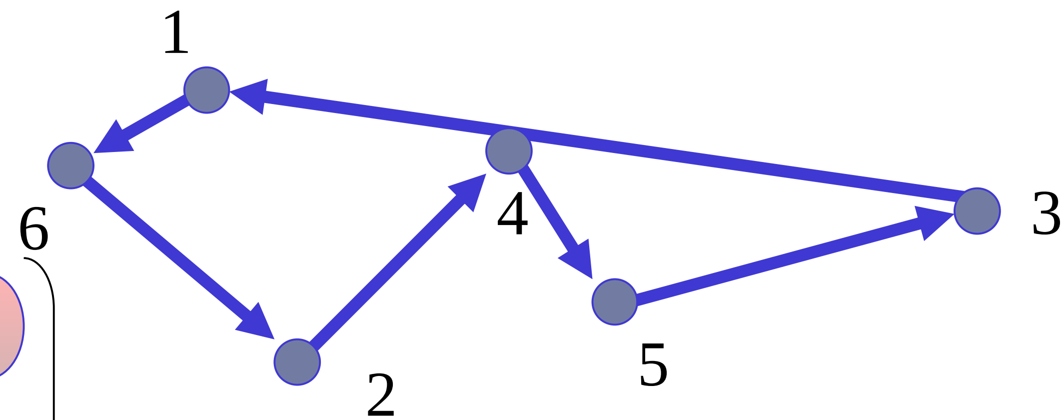
TSPのもうひとつの定義：
n×nの行列［d_{ij}］が与えられたとき、V=｛1，2，…，n｝からVへの1対1写像（これを順列と呼ぶ）ρ のうち、以下を最小にするものを求めよ

$$\sum_{i=1}^{n-1} d_{\rho(i)\rho(i+1)} + d_{\rho(n)\rho(1)}$$

- 配列［d_{ij}］は都市間の距離を表わす
- $\rho(i)$ はある閉路において i番目に訪れる都市を示す、
- したがって $d_{\rho(i)\rho(i+1)}$は i 番目に訪れる都市と $i+1$ 番目に訪れる都市の間の距離を表わす
- 最後の$d_{\rho(n)\rho(1)}$は閉路を閉じるためのもの

TSPの新しい定義について（2）

$$[d_{ij}] = \begin{pmatrix} 0 & 3 & 8 & 4 & 6 & 1 \\ 3 & 0 & 6 & 3 & 3 & 3 \\ 8 & 6 & 0 & 4 & 3 & 9 \\ 4 & 3 & 4 & 0 & 1 & 4 \\ 6 & 3 & 3 & 1 & 0 & 5 \\ 1 & 3 & 9 & 4 & 5 & 0 \end{pmatrix}$$



V=[1, 6, 2, 4, 5, 3]

丸で囲まれた数値の総和 19 が上の閉路の長さになる

TSPを解く①：完全列挙法

- 完全列挙法
  - 解の候補となる全てを作り出し、その中から解を見つけ出すような解法のことを「完全列挙法 (Complete Enumeration Method)」、あるいは「力ずく法 (Brute Force Method)」という
- TSPのように解の候補の数が爆発的に増加するようなことを「組み合わせ的爆発 (Comninatorial Explosion)」と呼んでいる

実験２：完全列挙法（１）

- 完全列挙法を用いてTSPを解くようなプログラムを作成しなさい
  - プログラミング言語は何を用いてもよい
  - 適当に出発点を設定し、そこから出発する全ての閉路を求め、その中で最短のものを選択して出力する
- 詳細は各自で定める
  - 都市数については定数としても定義してもよいし、引数として与えてもよい
  - また外部から入力してもよい

注意：このプログラムを動かすときには決して大きな都市数を与えてはならない

実験２：完全列挙法（２）

- 作成したプログラムを適切に小さな都市数Nに対して動かし、その実行時間を求めなさい
  - 実行毎のばらつきを抑えるため、適当な回数ずつ実行したらその平均値を求め、縦軸を処理時間、横軸をNとしてグラフを作りなさい
- そしてその結果から考えられる事柄は何か考察しなさい

たとえば
●計算時間はNの増加に対してどのように増加していくか？
●1都市あたりの計算時間はNの大きさに依存しているだろうか？

TSPを解く②：順次生成・比較法（1）

- 問題の規模
  - 完全列挙法を用いたプログラムによる実験からもわかるように、この方法では大規模なTSPを扱うことはできない
  - ではもっと大規模なTSPを扱うにはどうしたらいいのだろうか？
- 考えられるひとつの方法は完全列挙法で必要となる「全ての閉路を記憶しておくスペースを効率化する」というものである

TSPを解く②：順次生成・比較法（２）

- 記憶スペースの効率化
  - 完全列挙法では全ての閉路を生成し、その中で閉路長が最短になるものを選択していた
  - これを閉路を生成するたびにそれまでの最短のものと比較することを繰り返すという方法が考えられる
- そうすると閉路を記憶しておくスペースは高々閉路ふたつ分で済むことになる

このようなTPSの解法を「順次生成・比較法」と呼ぶことにする

実験３：順次生成・比較法（１）

- 「順次生成・比較法」により、閉路を順次生成し、これまでの最短経路長をもった閉路と閉路長を比較することによってTPSを解くようなプログラムを作りなさい

- 適当に出発点を定め、そこを出発点とする閉路をひとつ生成し、それをとりあえず最短閉路とする
- 新たに閉路を生成し、その閉路長をその時点での最短閉路長と比較し、短い方を最短閉路とする
- これを全ての閉路について繰り返し、最短閉路を求める

実験３：順次生成・比較法（２）

- 作成したプログラムを適切に小さな都市数に対して実行し、その実行時間を求めなさい
  - 実験2と同様、実行時間のばらつきを抑えるため、適当な回数ずつ実行したらその平均値を求め、縦軸を処理時間、横軸をNとしてグラフを作りなさい
- さらにその結果を実験2の結果と比較し、そこから考えられる事柄は何か考察しなさい

さらに大規模なTSPを解くには（1）

- 実験2で使った「順次生成・比較法」でも大規模なTSPを解くことはできない
  - 実行時間に影響を与えるのは主として記憶のためのスペースではなく、生成される閉路の数だからである
- そういった意味で生成される閉路の数は「全ての閉路を生成する」方法においては一括にしろ順次にしろ、いずれも同じである

もっと大規模なTSPを解くためには全ての閉路を生成せずにTSPを解く方法を考えださなければならない

さらに大規模なTSPを解くには（2）

- 厳密解
  - TSPの厳密な意味での解(これを厳密解という)を求めるのは一般的にいってとても難しい
  - そこで｢厳密に最短であること｣をあきらめ、その代わりより短時間に比較的短い解を求める方法を考える
- この方法は最短な閉路長の定数倍以下の閉路長を持った閉路だということを保証された、しかもその定数が出来るだけ1に近いような解を求めるものである

これを「精度保証付き近似計算法」という

さらに大規模なTSPを解くには（３）

- **精度保証付き近似計算法**
  - この方法を使っても一般的なTSPを解くのはとても難しい
  - そこで問題自体を限定することにする
  - 扱う問題は距離が非負で、かつ対称性があり、しかも三角不等式を満たすものに限定する
- 距離が非負とは、距離が0以上だということ
- 距離に対称性があるとは、たとえばAからBまでの距離とBからAまでの距離が同じということ
- 三角不等式を満たすとは、AからBまでの距離とBからCまでの距離の和がAからCまでの距離よりも短くないということ

さらに大規模なTSPを解くには（4）

- **普通距離は非負で対称性があり、三角不等式を満たすのだが**
  - TSPにおける節間の距離は図形的な距離に限られていない
  - たとえば運賃、輸送コスト、所要時間などを考えてみると、必ずしも対称性や三角不等式を満たすわけではない
- たとえば山の上り下りを考えればその所要時間には対称性はないことが多い
- また、このような制限を加えたとしても扱うことのできる問題の範囲は充分に広いと考えられる

そこで今後はこのような制限のある距離について考えることにする

さらに大規模なTSPを解くには（5）

- 精度保証付き近似計算
  - このような距離の制限を加えれば、精度保証付き計算法が存在する
  - しかも最短閉路に対して一定の精度以内の解を求める方法がある
- この実験では3つの精度保証付き近似計算法について実験を行う

●Nearest Addition法
●Greedy法
●Nearest Neighbor法

Nearest Addition法（１）

- 基本的な考え方
  - 先のことは考えず、次に追加する都市について距離の増加を最小限にとどめる
- アルゴリズム

① ひとつの都市からなる長さ0の部分閉路Tをひとつ作る

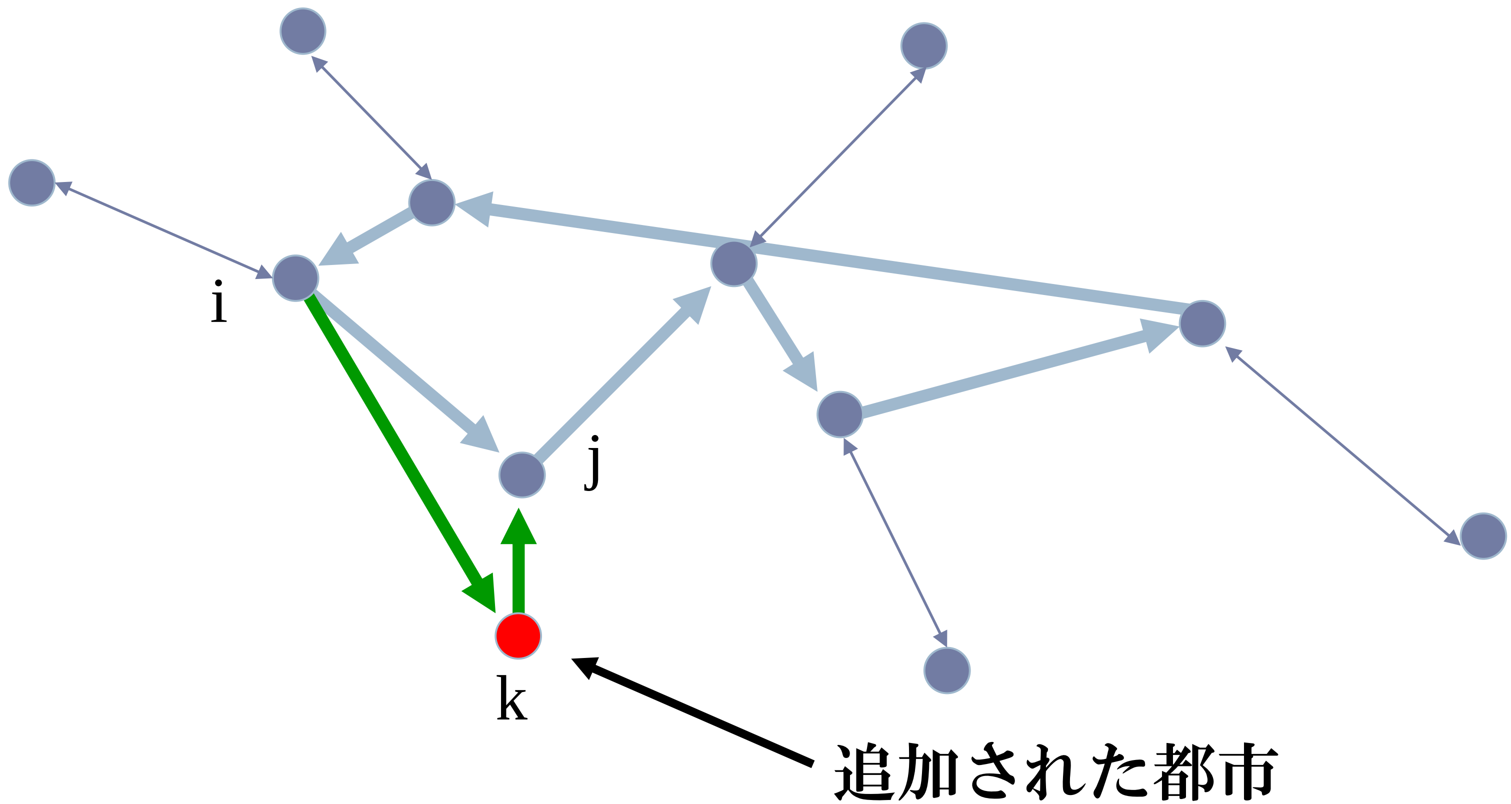
② Tが全ての都市を含めばそれが解

③ そうでないならば、Tに含まれる都市jと含まれない都市kの組み合わせのうち、jk間の距離D_{jk}が最小なものを求める

④ (i，j)をTに含まれるアークとするとき、これを2つのアーク（i，k）と（k，j）で置き換える

⑤ 以上の②～④を繰り返す

Nearest Addition法（2）



実験４：Nearest Addition法

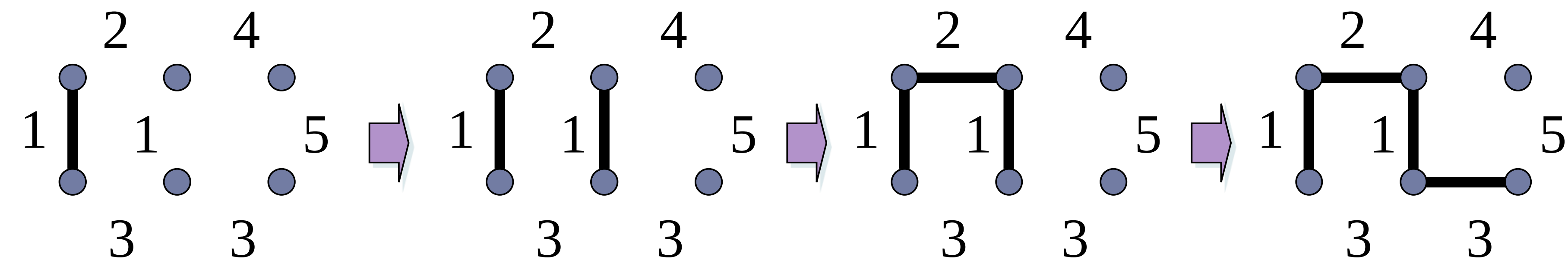
- Nearest Addition法を実装し、次の項目についての実験を実施しなさい
  ① この方法を用いると、何都市まで適切な実行時間で求めることができるか、実行データに基づき、完全列挙法と同じようにグラフを作成して推測しなさい
- 完全列挙法およびNearest Addition法の両方で計算が可能であった都市数について、その最短閉路長と計算時間を比較しなさい

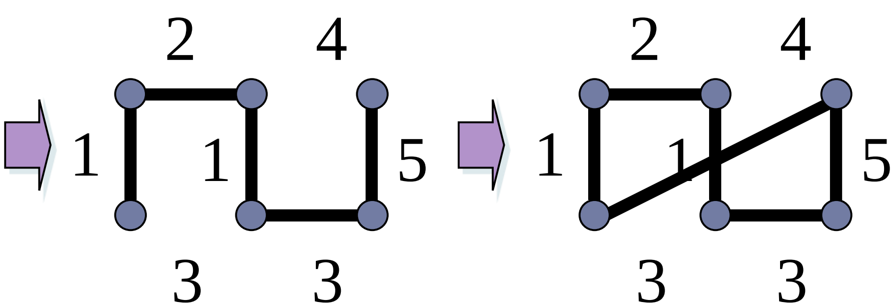
Greedy法（１）

- Greedy法は貪欲法ともいい、次のように最短閉路長を求める
  1. 全てのアークを長さ順に並べる
  2. 空の閉路から始め、アークを短い順に調べ、そのアークが次の条件を満たすならば閉路に付け加える

<u>条件</u>
a. 都市の次数が3を越えない
b. すべての都市を回らないような閉路を作らない

Greedy法（2）





Greedy法を用いると、最短閉路長の $\log n$ 倍の解を求めることができる（ただし、ここで n は都市数である）

実験５：Greedy法

- Greedy法を実装し、次の項目について実験を実施しなさい
  - ① この方法を用いると、何としまで適切な実行時間で求めることができるか、実行データに基づき、完全列挙法と同じようにグラフを作成して推測しなさい

② 完全列挙法、Nearest Addition法、およびGreedy法のすべてにおいて計算が可能であったような都市数について、その最短閉路長と計算時間を比較しなさい

Nearest Neighbor法

- Nearest Neighbor法は次のようにして最短閉路を求める
  - ① 適当な都市を選択し、その都市から出発する
  - ② まだ訪問していない都市のうち、現在いる都市から最も近いものを選択し、その都市に移動する
  - ③ これをすべての都市を訪れるまで繰り返す
  - ④ 最初に出発した都市に戻る

この方法は比較的単純であり、求められる解は最短閉路長の $m \cdot \log n$ 倍以内である（ただし m は定数、n は都市数）

実験６：Nearest Neighbor法

- Nearest Neighbor法を実装し、次の項目について実験を実施しなさい
  - ① この方法を用いると、何としまで適切な実行時間で求めることができるか、実行データに基づき、完全列挙法と同じようにグラフを作成して推測しなさい
  - ② 完全列挙法、Nearest Addition法、Greedy法のいずれでも計算可能であった都市数、および二つ以上の方法で研鑽可能であった都市数について、その最短閉路長と計算時間を比較しなさい

レポートについて（1）

- 完全列挙法（又は順次生成・比較法）以外に少なくとも一つの方法で巡回セールスマン問題を解き、比較検討する
  - 課題
  - 課題の分析
  - プログラムと実行結果（必要なものについて）
  - 考察

<u>レポート提出期限</u>：7月7日の情報実験Ｉの授業開始時点までにShare Folderにpdfまたはwordファイルなどで、学籍番号をファイル名として提出

- レポート作成上の注意
  - 分かりやすく誤解のないような日本語を心がける
  - レポートには必ず表紙をつける
  - 考察は感想ではない
  - 事実と意見を区別して記述す
  - 参考となる文献やサイトがあればそれを列挙すること

レポートは学生と教員の間のコミュニケーションである

レポートについて（2）

- たとえば考察として
  - それぞれの方法における都市数と処理時間の関係
    - それぞれの方法において、都市数が増えると処理時間はどのように増加するか？
  - それぞれの方法・都市数における一都市あたりの処理時間
- 完全列挙法（又は順次生成・比較法）で最短の巡回路が求められる場合に「精度保証付き近似計算」で求めた経路長は最短周回路長と比べてどのようになっているか
- …

レポートについて（3）

- 参考とした文献・サイトの列挙方法は、たとえば以下のとおり

[1]「TSPの効率に関する知見」
http://www.x.y.z/
[2] 芝浦太郎、大宮次郎「TSP入門」、
豊洲出版、pp.34-41、2010

- 基本的に参考とした文献、サイトについて第三者が同じものを必ず見ることができるようにすること

- 他の文献、サイトなどを参考にしながらそれらを「参考文献」として明記しない場合、剽窃と解釈される場合もあるので注意すること