かんたんにしたつもりの Python講座

BinNスタートアップゼミ#2

担当：植田瑞貴（M1）

2018/4/16

もくじ

第一部：Pythonについて

Pythonとは？なぜRだけではだめなのか？という疑問から始まり，具体的な基礎文法などについてちょこっと説明します．ついでにBinNで一年過ごして「早く調べればよかった…」と思ったことや，つまづいたことに対するヒントをご紹介します．

第二部：最短経路探索法

交通研では道路ネットワークを扱う機会が多くなります．最短経路を探索する，ダイクストラのアルゴリズムと，そのために必要なデータ処理構造をご紹介します．

課題説明：

Pythonの最短経路探索とRの目的地選択離散モデル

第一部 Pythonについて

基礎の基礎とBinN的Tips

Pythonとは

- スクリプト言語の名前
- パッケージとよばれる，便利な定義のセットがたーくさん！ あります
- IDE（開発環境：編集しやすくなるソフト）もいろいろ
  Atomとか，**PyCharm**とか

わたしが使っているIDEです 

- インストールすべきものは三つ
  ①Python本体，②パッケージ，③IDE

→ですが，「**Anaconda**」を使えば，
  ①Python本体と②データサイエンスおすすめパッケージ
  を一気にインストールできます！

（補）Anacondaの役割は？

- 大きなデータの分析に使うには，早いことが重要です
- しかしPythonはわりと遅めの言語です　※Rよりは早いです
- そこで，C/C++ といった言語で開発された「拡張モジュール」とつなげて使うと早くなります
- そのようなパッケージをインストールするには，自分のOSに対応するようにコンパイルしてからインストールする必要がありますが，Anaconda はコンパイル済みのパッケージを出してくれています

いろいろ言いましたが

プログラミングにあまり詳しくなくても

すぐ研究に使えて便利！ということです！

そもそもなんでPython？

- わたしにとってのPythonのいいところ

①**わかりやすい**：書きやすいし読みやすい

②Google三大言語のひとつ：今後しばらく安泰そう

③**ほかの言語と組み合わせやすい**

④これらの状況から，どんどん知識が蓄積されている

⑤**ArcGISと組み合わせられる**

ほかにもコンパイル不要であることとか，
PyCharmがよくできていてストレスが少ないことなど，
いろいろすてきなポイントがあります！

RとPython

特徴の比較

- Rはデータ分析関係のコマンドが最初から入っている
  ↔Pythonはパッケージに依存している
- Rは統計寄り
  ↔統計以外ならPythonの方が単純

Rだけだと，今後の研究テーマによっては以下のような問題がでることがあります.

- Rは大規模なデータを扱うには適していない
- 基本のRは並列プログラミングに対応していない
- Rは非常に簡単だがブラックボックス化しやすい

のでPythonもやっておくのがおすすめです！

(補)インストールしてみよう

Python本体と主要パッケージ

- Pythonには歴史があり，大きく分けて2.x系と3.x系のふたつのバージョンがあります
- 2.x系は「古い」ので3.x系を入れましょう
- Anacondaによるインストールの仕方はPython Japanがまとめてくださっています↓

https://www.python.jp/install/windows/anaconda/install_anaconda.html

IDE

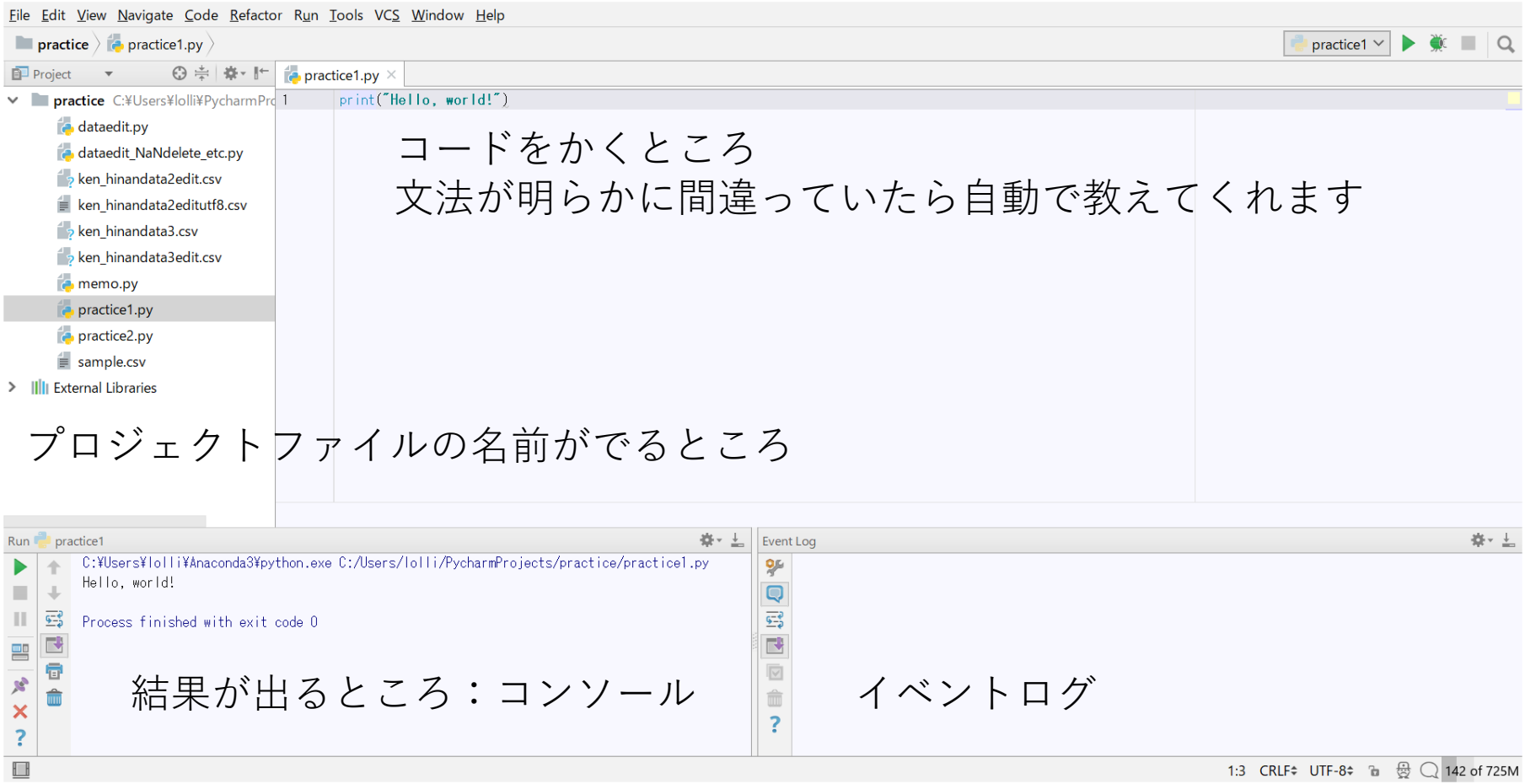
PyCharmにしたい方はこちらを参考に．※以下ではPycharmで解説します

https://pythondatascience.plavox.info/python%E3%81%AE%E9%96%8B%E7%99%BA%E7%92%B0%E5%A2%83/pycharm%E3%81%AE%E3%82%A4%E3%83%B3%E3%82%B9%E3%83%88%E3%83%BC%E3%83%AB

Community Editionでいいと思います（無料です）

ファイル自体はJetBrains…という名前になるのでお間違いなく！

PyCharm画面はこんな感じ



背景カラーはいくつかあるので好みで変えられます

コマンドなどのRとの違い

R	Python

R

```r
# パッケージのインストール
install.packages("name")

# パッケージの読み込み
library(name)

# 作業ディレクトリの確認
getwd()

# 作業ディレクトリの変更
setwd()

# ディレクトリ内に格納されているファイルの確認
dir()

# 全ての作成済みオブジェクトを確認
ls()

# オブジェクトの削除
rm('name')
```

Python

```python
# パッケージのインストール（ターミナルから実行）
pip install name

# パッケージの読み込み
import name as other_name

# 作業ディレクトリの確認
import os
os.getswd()

# 作業ディレクトリの変更
os.chdir()

# ディレクトリ内に格納されているファイルの確認
os.listdir()

# 全ての作成済みオブジェクトを確認
globals()

# オブジェクトの削除
del('name')
```

https://pythondatascience.plavox.info/pythonとrの違い　より

こんな感じで文法はちょこっと違います．その都度確認しましょう．

注意

エンコーディング

- 3.x系のデフォルトエンコーディングはUTF-8

インデント

- Pythonでは，インデントに意味があります．
- インデントには4文字分のスペースを使うのが一般的です．

PyCharmを使っている場合は，だいたい自動で処理してくれます．

コメント

- 一行におさまるものは，#を先頭につけます．
- 複数行にわたるものは，”””か’’’でコメントを囲みます．

書き方

- カンマ後ろには半角スペースを入れねばなりません．

①プロジェクトをつくります

- 上のタブから

[file]->[New Project]でプロジェクト名を入力,

[完了]すると左側にプロジェクトフォルダができます

②ファイルをつくります

- 左のプロジェクト名を右クリック
- [New]->[Python File]でファイル名を入力，[完了]

③実行します

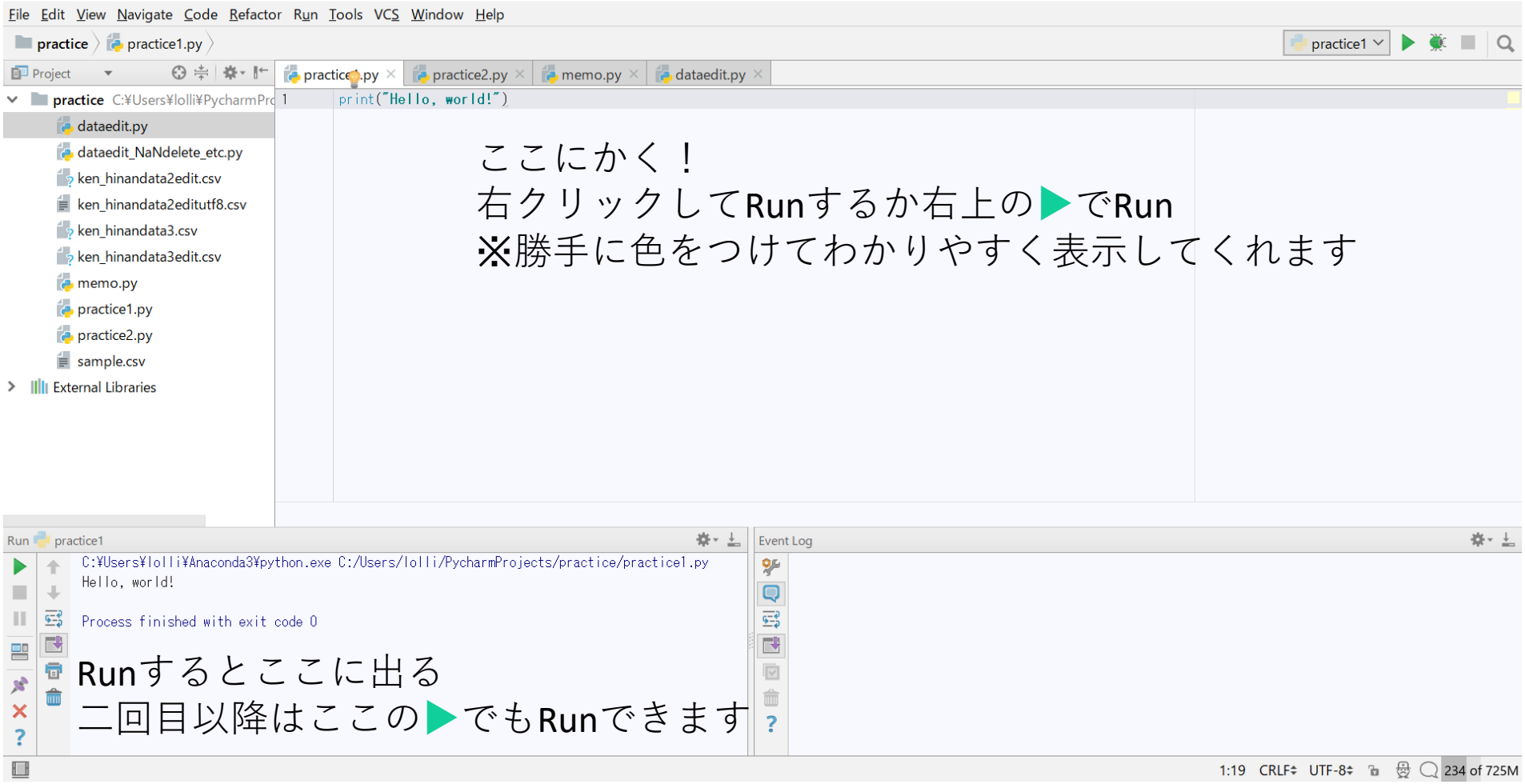
- コードをかいて，右クリックで出てくるメニューか右上のボタンから

  [▶Run]すれば実行されます

（コンソール左の▶では，直前に実行したファイルを再実行できます）

動かしてみましょう

まずは　print(“Hello, world!”)



まず知りたい①識別子

識別子：クラス、関数、変数などの名前

名前の付け方ルール

- アルファベット，アンダースコア_，半角数字が使えます（3.x系では日本語もOK）
- 大文字と小文字は区別されます
- 数字は識別子の先頭には使えません
- 予約語（python内で既定の言葉）は使えません

  Ex. False, True, import, if, …
- アンダースコアを先頭におくと特別な意味がつきます

たとえば，thetaとThetaは
別の変数として扱われます

※import()やクラスを利用するとき

まず知りたい②データ型

Pythonは変数の型を自動で判定します.
型が異なるもの同士の演算はできないこともあります.

整数 int
長整数 long
小数 float
複素数 complex ※虚数をjで表すので注意
真理値 bool：True,　False

数値

文字列 str：”abcde”　“10”

数字も””または”’の間に書くと
文字列として認識されます

リスト list：[“today”, 200,“next", 250]
タプル tuple：(“today”, 200 , “next”, 250)

リストは変更可能
タプルは変更不可だが早い
という違いがあります

シーケンス

辞書 dict：{"dog":4,"cat": 4,"bird": 2}

値とキーを紐づけできます
Ex “dog”キーで4が呼び出せる

まず知りたい③比較演算子

- 比較演算子 http://www.isl.ne.jp/pcsp/python/python09.html より

演算子	説明	例	
		プログラム	実行結果
==	左辺と右辺が等しい	1 == 1	True
!=	左辺と右辺が等しくない	1 != 1	False
<	左辺が右辺より小さい	1 < 2	True
>	左辺が右辺より大きい	1 > 2	False
<=	左辺が右辺以下	2 <= 2	True
>=	左辺が右辺以下	2 >= 2	True
is	厳密一致	1 is 1	False
is not	厳密不一致	1 is not 1	True

practice1.py practice2.py me

```
1  a = "Hello"
2  b = "hello"
3  print(a==b)
4  if a==b:
5      print("That's right")
6
7  c = 1000
8  d = 1000
9  print(c==d)
10 if c==d:
11     print("That's right")
```

Run practice1

```
C:¥Users¥lolli¥Anaconda3¥python.exe C:/Users/lolli/PycharmProjects/practice/practice1.py
False
True
That's right

Process finished with exit code 0
```

TrueまたはFalseを返します．if文やwhile文に多用します．

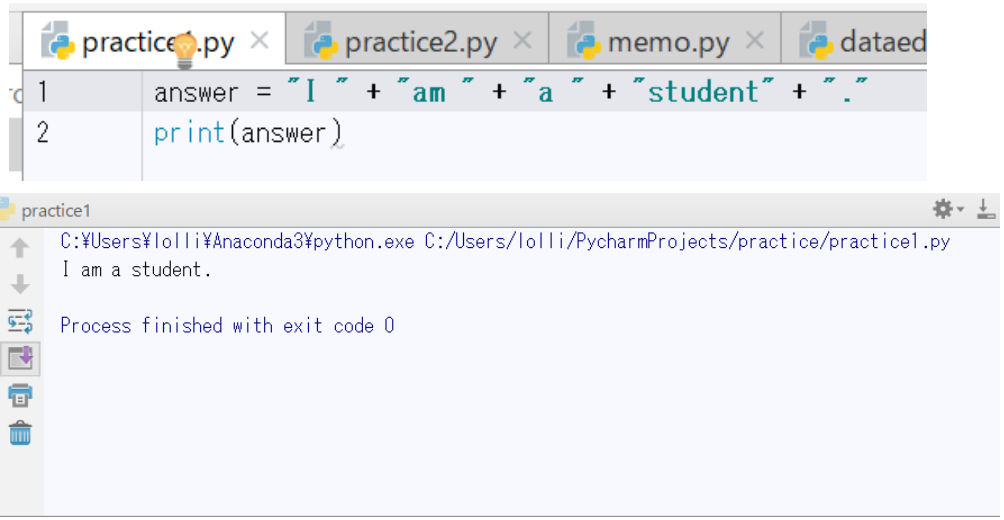
まず知りたい④算術演算

- 数値型の算術演算

http://www.isl.ne.jp/pcsp/python/python09.html より

演算子	説明	例	
		プログラム	実行結果
+	足し算	1 + 2	3
-	引き算	3 - 1	2
*	掛け算	2 * 3	6
/	割り算	4 / 2	2
%	剰余	10 % 3	1
**	べき乗	2 ** 3	8

（補）文字列同士の場合も演算（=連結）ができます.



（補）自主学習

- range()とprint()
- for文，if文，while文，try文
- データ型の判定
- 関数の定義の仕方

（補）わたしの勉強の仕方

①一つの言語（RでもPythonでもJavaでも）についてはまず一冊本を読むと
なんとなく何をすればいいのかわかってくると思います.

実際に書いて実行してみましょう
このとき，全部を暗記するのではなく，「こういうことができるらしい」と把握すればよいと思います

②わからなかったら検索エンジン

「やりたいこと＋python」で検索すればたいていヒットします
エラーがでたらその文をコピーして検索

③遠慮せずに人にきく！

使っているコード，データをもって聞きにいきましょう

次に知りたい①モジュールの使い方

- **モジュール**：関数やクラスなどを書いたファイル．

クラスとは，現実世界のものをプログラミング世界に落とし込むときに行う，状態と性質の定義のことです．これをあらかじめ書いておいて，呼び出して使うことで，Pythonでも高速計算が可能です！

- **パッケージ**：モジュールをまとめたもの．
- **ライブラリ**：パッケージをまとめたもの．プログラム群のことをいいます．

モジュールの使い方

```
import モジュール名 as 別名
```

自分で決められて，そのファイル内では
その名前で使うことができます．
pandasをpd，numpyをnpとつけるのが定石です

データ解析系ライブラリ　　配列処理系ライブラリ

(補) 未インストールエラーの時は？

1　まず綴り間違いじゃないか確認する

- File > Settings > project: ○○ > Project Interpreter
  からPackageを確認．ここにあったらたぶん綴り間違い．
- ここになかったら，右上のLatestセルの右横にある＋を押してみる．

2　Available Packagesの確認

- 開いたウィンドウで検索してみる．ここにあったら話は早くて，選択して左下のInstall Package.
- このAvailable Packageにもなかったら，コマンドプロンプトからインストールする必要がある．

3　コマンドプロンプトからのインストール

- cortanaの検索フォームみたいな，画面左下の「ここに入力して検索」というところに「コマンドプロンプト」と入力．出てきたものをクリックしてコマンドプロンプトを立ち上げる．PATHを通していればここからいじれる．
- pip install パッケージ名
  を入れて，エンターキーを押すだけ．こうしたあとで1の手順のようにPycharmから確認するときちんと入っていて，そのパッケージが使えるはず．おしまい！

（※windows10，PyCharm，pipが入っている，
かつPythonをダウンロードしたときにPATHを通している前提）

次に知りたい②csvファイルの読み書き

今回はnumpyを使うものを紹介します！

※pandasにもあります．こちらを使うことも多いです[**自主学習**]

まずは

```
import numpy as np
```

※ファイル中で一回書けばOKです

- 今回使うモジュールはどれもcsv限定ではなく，テキストに対して区切り文字を指定して読みこむものです．

※csvファイルはセルの区切り文字がカンマ，になっているテキストです．メモ帳でひらいてみると確認できます

次に知りたい②csvファイルの読み書き

1. 読み取り

空白セルがあるとき

```
import numpy as np
np.genfromtxt("sample.csv", delimiter="","", filling_values=0)
```

np.genfromtxt(“ファイル名.csv”, delimiter=“区切り文字”, filling_values=代わりの文字)

- filling_values：なんにもはいってないセルがあったときに代わりに詰めてもらう文字の指定．何も入れたくないときは””でいけます

列名ヘッダを飛ばしたりしたいとき

```
import numpy as np
np.loadtxt("sample.csv", delimiter="","", skiprows=1)
```

np.loadtxt(“ファイル名.csv”, delimiter=“区切り文字", skiprows=飛ばしたい行数)

2. 書き込み

```
np.savetxt("sample.csv", dataname,
delimiter=",")
```

np.savetxt(“ファイル名.csv", data名, delimiter=",")

- すでに存在しているのと同じファイル名を指定する場合，そのファイルを開いた状態で実行すると，エラーになるので注意．

次に知りたい③データフレーム

pandasのデータフレームはとても便利です．[自主学習]

- **データを行で取り出したい**
  データ名[始まり行番号:終わり行番号]
- **データの列を取り出したい**
  データ名[”列名”]
  複数列を取り出したいときは
  データ名[['列名1', '列名2']]
  ※かっこ[]が二重なのは，列名のリストを指定する形になっているから．
- **ある列が条件を満たす行だけ取り出す**
  データ名[“列名”] ==0 や >= 80 などとすると，
  その列の各行に対してTrue or Falseが返る．
  これを利用して，Trueなデータのみ抽出．
  データ名[データ名["列名"] ==0 とか >= 80とか]

```
data3 = data2[data2['A'] != "不明"]
```

次に知りたい④NaNの判定法

- **欠損データNaNを含むデータを取り除く処理**

data4 = data3[data3['A'] == data3['A’]]

とすると，data3の各行のうち，列名AのセルがNaNでないもののみdata4に代入されます．

※どういうこと？

普通は同じものを比べたらTrueが返ります（0==0, “Hello” == “Hello”などの結果はTrue）．

しかし， PythonではNaNとNaNを比較する（NaN==NaN）とFalseが返る仕様になっているらしいです．

つまりある同一セルを比較した場合，NaNが入っていた場合のみFalseになります．

これを利用して，NaNでないものだけを取っています．

調査データ原本では欠損データが含まれることがよくあり，そのままでは推定が回りません．この処理をしておくとストレスなく推定に入れます．

データ編集コードの例

```
# dataedit
# パッケージpandasを使うのが便利
import pandas
# 書き込み先ファイル閉じないと回らないので注意
data2 = pandas.read_csv("data2.csv", encoding="utf_8")
# 文字コードのutf-8への変更は，メモ帳で開いて新規保存のときに文字コード指定が簡単
# print(csv)
data3 = data2[data2['A'] != "不明"]
data3 = data3[data3['A'] == data3['A'] ]
data3 = data3[data3['B'] == data3['B'] ]
data3 = data3[data3['C'] == data3['C'] ]
data3 = data3[data3['D'] == data3['D'] ]
data3 = data3[data3['D'] != "?" ]
print(data3)

# csvファイルとして出力
data3.to_csv("data3.csv")
```

（補）自主学習

- range()とprint()
- for文，if文，while文，try文
- データ型の判定
- 関数の定義の仕方
- pandasのデータフレームの使い方
- pandasのcsvファイルの扱い方
- 日付データの扱い方

（おまけ）

第三週のGIS講義後，PythonとArcGISの組み合わせについて調べてみましょう！

練習問題

1. Welcome to BinN!とコンソール画面に表示
2. 1~100の数字をコンソール画面に表示
3. 1~100の数字をカンマで区切った一行でコンソール画面に表示(1, 2, …, 100)
4. 1~100の素数を表示

（発展）

緯度経度を変数に入力して，東京大学工学部一号館までの距離を算出してみてください.

これができたら，緯度経度が入力されたcsvファイルを読み込み，東京大学工学部一号館までの距離を算出し，csvファイルに書き込むコードを書いてみてください.

練習問題ヒント

1. print()
2. range()とfor文：PythonではRと違い，指定番号が0から始まるので注意.
3. print()の引数とif文・while文：print()ではそのままだと毎回改行されます．引数endを指定しましょう．また，100のあとにカンマを入れないためには，if文かwhile文を使うとよいでしょう.
4. if文とfor文：素数の求め方にはいろいろあります．まずもっとも実直なのは2-99の数で割り切れないものを素数と判断するもの．ネットで検索して効率のいいやり方もさがしてみてください.

（発展）

http://ikasumiwt.hatenadiary.jp/entry/2012/06/25/031357

緯度経度による距離算出はこのあたりを参考に．途中に書いてあるコードはPython対応じゃないので，Pythonでうごくように直しましょう.

第二章 最短経路探索法

ダイクストラのアルゴリズム実装

交通ネットワーク研究での大きな問題

計算量が，多すぎる！

- 交通ネットワークはノード（点）とリンク（線）で記述することが多いです
- 実際のネットワークでは無数のノードとリンクの処理が必要です
- しかも経路選択となると，リンクの組み合わせ問題になるので，膨大な量の計算を行わねばなりません
- PCは人間の処理能力を大幅に超えていますが，それでも複雑な計算を大量にすることはスペック上難しいことがあります

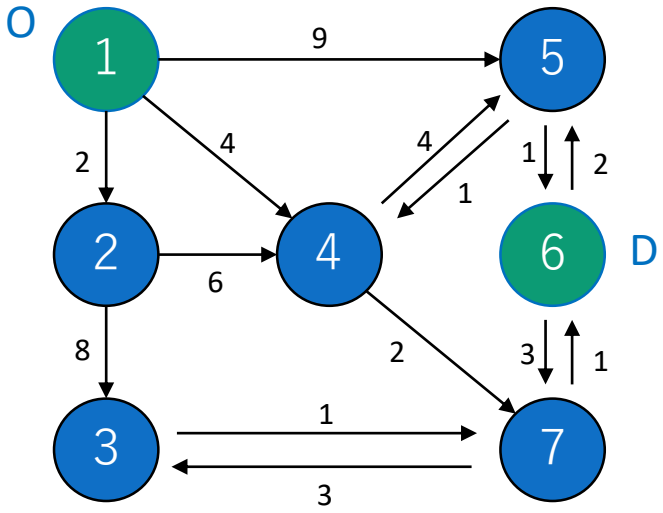
→そこで，膨大な組み合わせのなかから，
必要なルートを効率よく列挙する必要があります

たとえば

- Origin：ノード1
- Destination：ノード6

このとき，最短経路は？

※最短経路を通ろうとする人は多いので，最短経路がわかっているとうれしいことが多いですね！

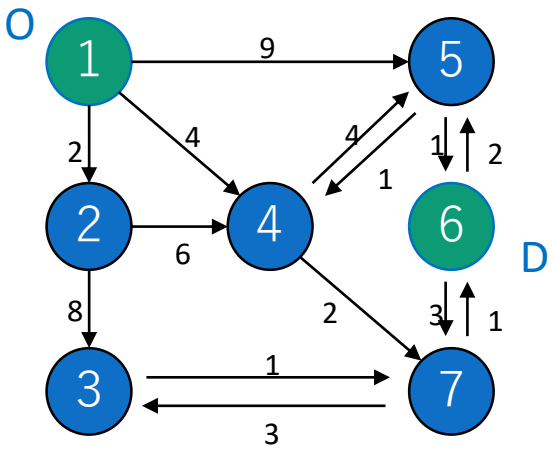


黒数字：リンクコスト

・距離
・運賃
・それらを重みづけしたもの
などさまざまなものを入れられます

たとえば答えは

- 答えは1 → 4 → 7 → 6



黒数字：リンクコスト

この答えを求めるべく全ての経路を書き出した場合

- 1 → 2 → 4 → 7 → 6 (11)
- 1 → 2 → 3 → 7 → 6 (14)
- 1 → 4 → 7 → 6 (7)
- 1 → 5 → 4 → 7 → 6 (16)
- 1 → 5 → 6 (10)

今回は列挙できましたが,
・全ノード数が4000個くらいあったら？
・Dが1以外すべての場合について
　計算する必要があったら？
…計算量が，多すぎる！

→最短経路探索法

Dijkstra法：最短経路探索法のひとつ

「最短とわかっているところから確定していく」

すべてのノードは、既に最短経路が求まっているノードの集合Kと
まだ最短経路が求まっていないノードの集合$\overline{K}$のどちらかに分類できる

各ノードiについて以下の変数を定めます．

・ c_i：始点からノードiまでの最小交通費用

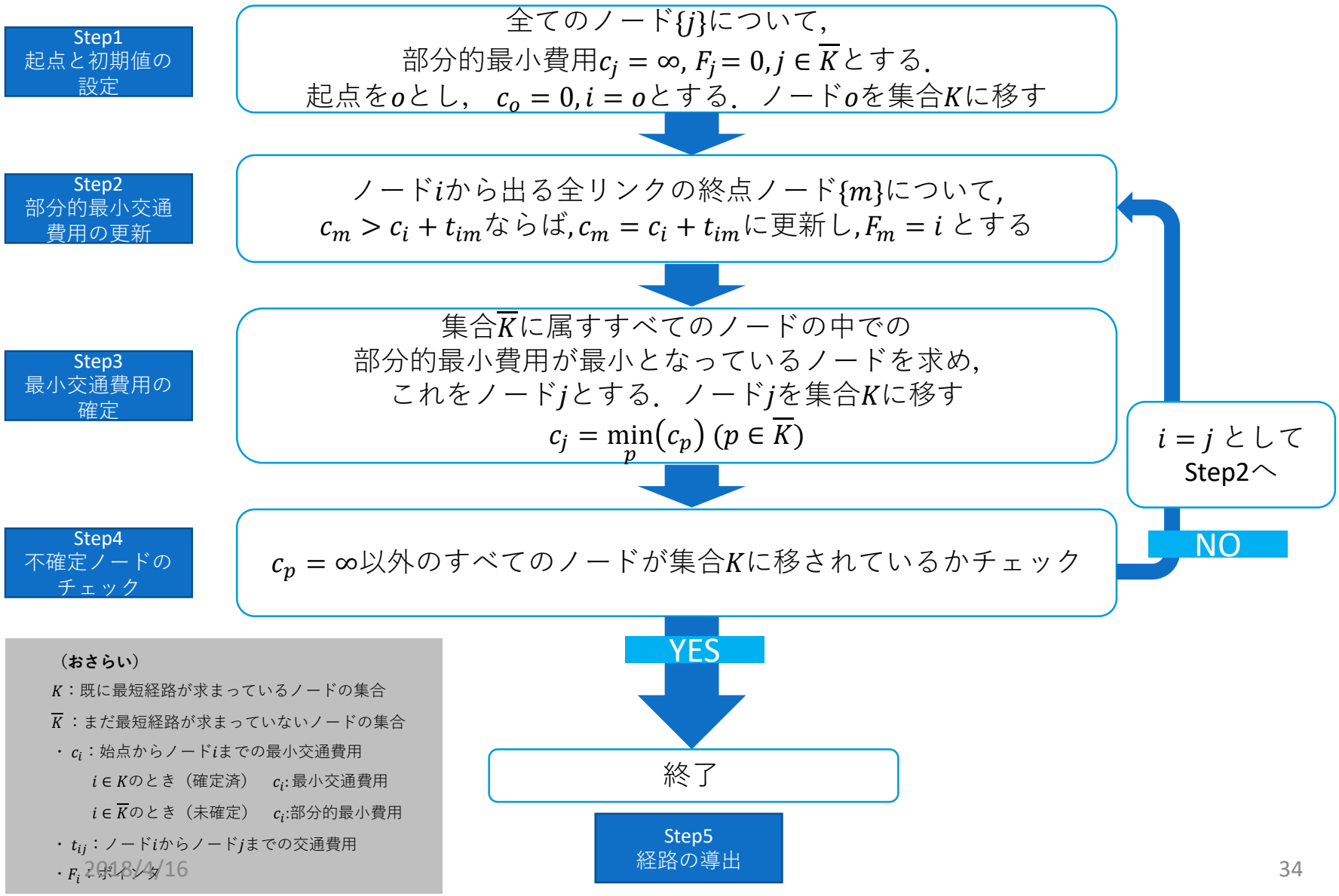
$i \in K$のとき（確定済）　c_i:最小交通費用

$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用…ここまでの最小費用

・ t_{ij}：ノードiからノードjまでの交通費用

・F_i：ポインタ（最短経路を列挙する際に使う．後述）

アルゴリズム



（おさらい）

K：既に最短経路が求まっているノードの集合

$\overline{K}$：まだ最短経路が求まっていないノードの集合

・c_i：始点からノードiまでの最小交通費用

$i \in K$のとき（確定済）　c_i: 最小交通費用

$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用

・t_{ij}：ノードiからノードjまでの交通費用

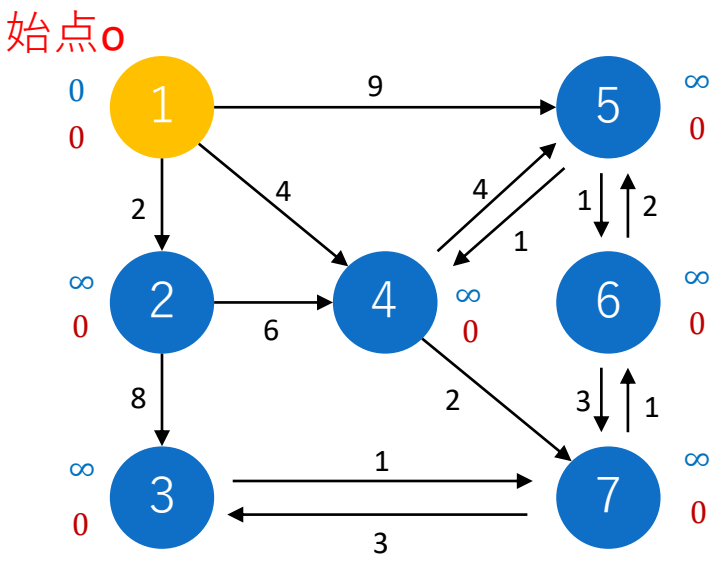
・F_i：ポインタ

Step1 起点と初期値の設定

全てのノード$\{j\}$について，
部分的最小費用$c_j = \infty, F_j = 0, j \in \overline{K}$とする．
起点をoとし，$c_o = 0, i = o$とする．ノードoを集合Kに移す

<Step1>

$i = 1$



（おさらい）
K：既に最短経路が求まっているノードの集合
$\overline{K}$：まだ最短経路が求まっていないノードの集合
・c_i：始点からノードiまでの最小交通費用
　$i \in K$のとき（確定済）　c_i:最小交通費用
　$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用
・t_{ij}：ノードiからノードjまでの交通費用
・F_i：ポインタ

黒数字：リンクコスト
赤数字：F_m（ポインタ）
青数字：c_m（最小費用）

c_m							F_m						
1	2	3	4	5	6	7	1	2	3	4	5	6	7
0	∞	∞	∞	∞	∞	∞	0	0	0	0	0	0	0

K	1
$\overline{K}$	2,3,4,5,6,7

Step2 部分的最小交通費用の更新

ノードiから出る全リンクの終点ノード$\{m\}$について，
$c_m > c_i + t_{im}$ならば，$c_m = c_i + t_{im}$に更新し，$F_m = i$ とする

ノード 1 から行けるのはノード2, 4, 5→ $m = 2,4,5$

$$c_2' = c_1 + t_{12} = 0 + 2 = 2 < \infty (= c_2)$$
$$c_4' = c_1 + t_{14} = 0 + 4 = 4 < \infty (= c_4)$$
$$c_5' = c_1 + t_{15} = 0 + 9 = 9 < \infty (= c_5)$$

$m = 2,4,5$すべてで$c_m' < c_m$なので，すべて更新

また，$F_m = i = 1$とする

（おさらい）

K：既に最短経路が求まっているノードの集合

$\overline{K}$：まだ最短経路が求まっていないノードの集合

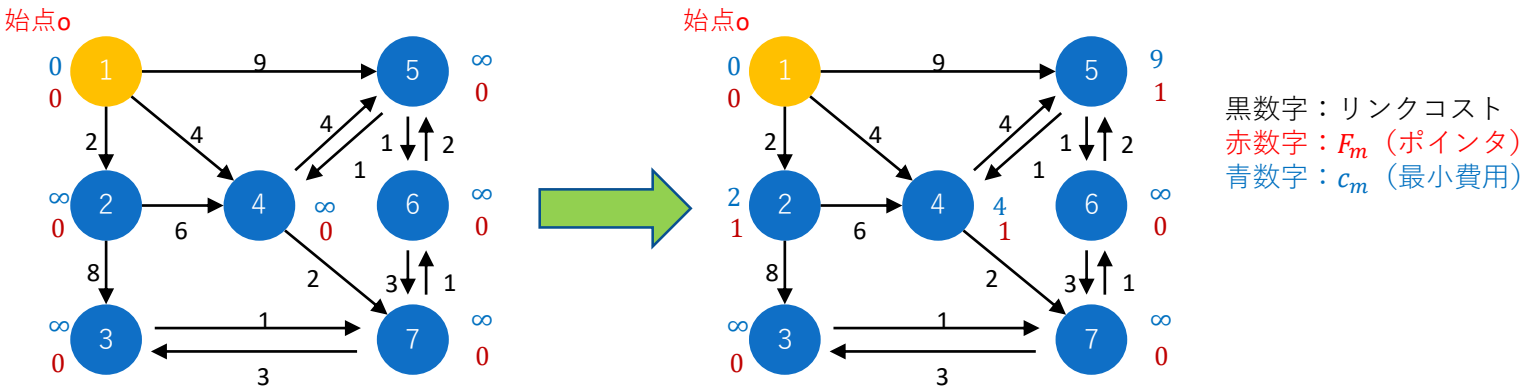
・c_i：始点からノードiまでの最小交通費用

　$i \in K$のとき（確定済）　c_i:最小交通費用

　$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用

・t_{ij}：ノードiからノードjまでの交通費用

・F_i：ポインタ



Step3 最小交通費用の確定

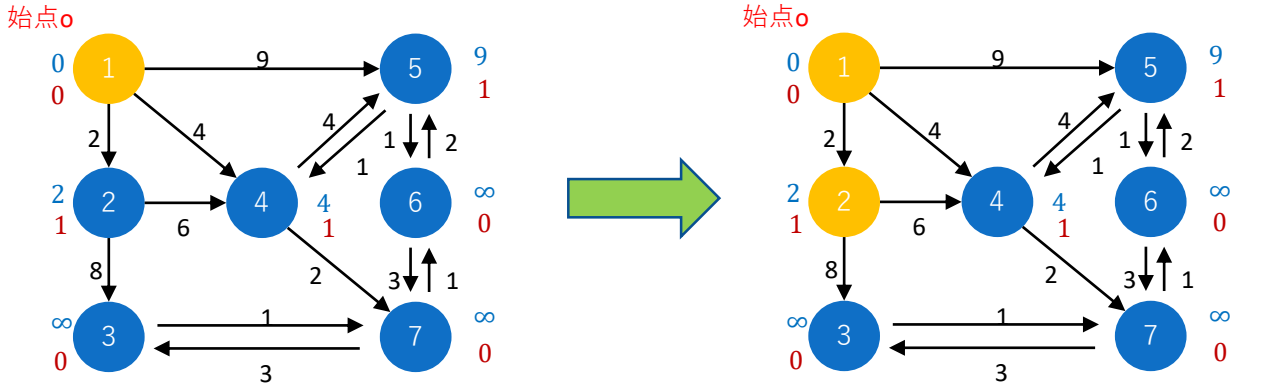
集合$\overline{K}$に属すすべてのノードの中での
部分的最小費用が最小となっているノードを求め，
これをノードjとする．ノードjを集合Kに移す

$$c_j = \min_p (c_p)\ (p \in \overline{K})$$

$\overline{K} = \{2,3,4,5,6,7\}$に属すノードの中で費用が最小となるのは、ノード２

$\rightarrow j = 2$ として、ノード2を集合Kへ移す

ノード2を含む最短経路：$1 \rightarrow 2$　最小交通費用：$c_2 = c_1 + t_{12} = 0 + 2 = 2$



（おさらい）

K：既に最短経路が求まっているノードの集合

$\overline{K}$：まだ最短経路が求まっていないノードの集合

・c_i：始点からノードiまでの最小交通費用

　$i \in K$のとき（確定済）　c_i: 最小交通費用

　$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用

・t_{ij}：ノードiからノードjまでの交通費用

・F_i：ポインタ

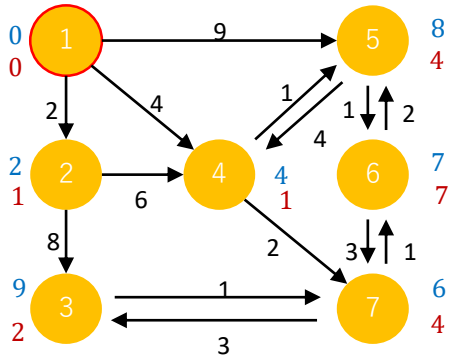
Step4 不確定ノードのチェック

$c_p = \infty$以外のすべてのノードが
集合Kに移されているかチェック

$c_p \neq \infty$かつ$p \in \overline{K}$なるpが残っている

→次は$i = 2$でstep2～step4を行う

$\rightarrow i = 4 \rightarrow i = 7 \rightarrow i = 6 \rightarrow i = 5$→終了



Step5 最短経路の導出

ポインタF_mを目印に遡っていく

・ノード 6 までの最短経路

$$6 \rightarrow (F_6 =)7 \rightarrow (F_7 =)4 \rightarrow (F_4 =)1$$

（おさらい）

K：既に最短経路が求まっているノードの集合

$\overline{K}$：まだ最短経路が求まっていないノードの集合

・c_i：始点からノードiまでの最小交通費用

　$i \in K$のとき（確定済）　c_i:最小交通費用

　$i \in \overline{K}$のとき（未確定）　c_i:部分的最小費用

・t_{ij}：ノードiからノードjまでの交通費用

・F_i：ポインタ

$c_j = \min_p(c_p)\ (p \in \overline{K})$の求め方：ヒープ構造

ダイクストラ法の中のStep3の「最小」の求め方は？

Step3
…部分的最小費用が最小となっているノードを求め…
$c_j = \min_p(c_p)\ (p \in \overline{K})$

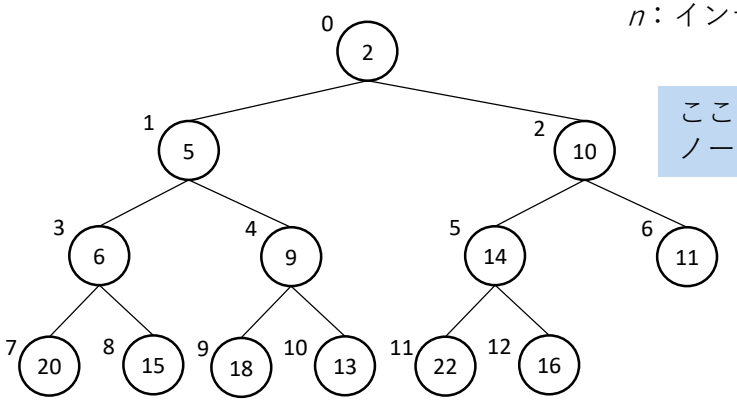
ヒープ構造

- 最小値を効率的に求めるために用いるデータ構造

「**ヒープ条件**」を満たすように配列（○つき数字）を並べ，インデックスをつける

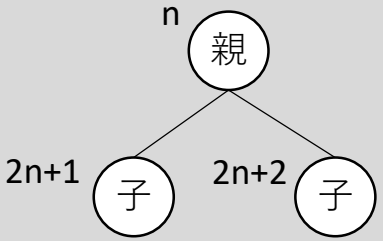
$$B(n) \geq B(2n+1), B(2n+2)$$

n：インデックス，*B(n)*：配列



ここで○の中に入っている数字はノード番号ではなく，**配列（値）**

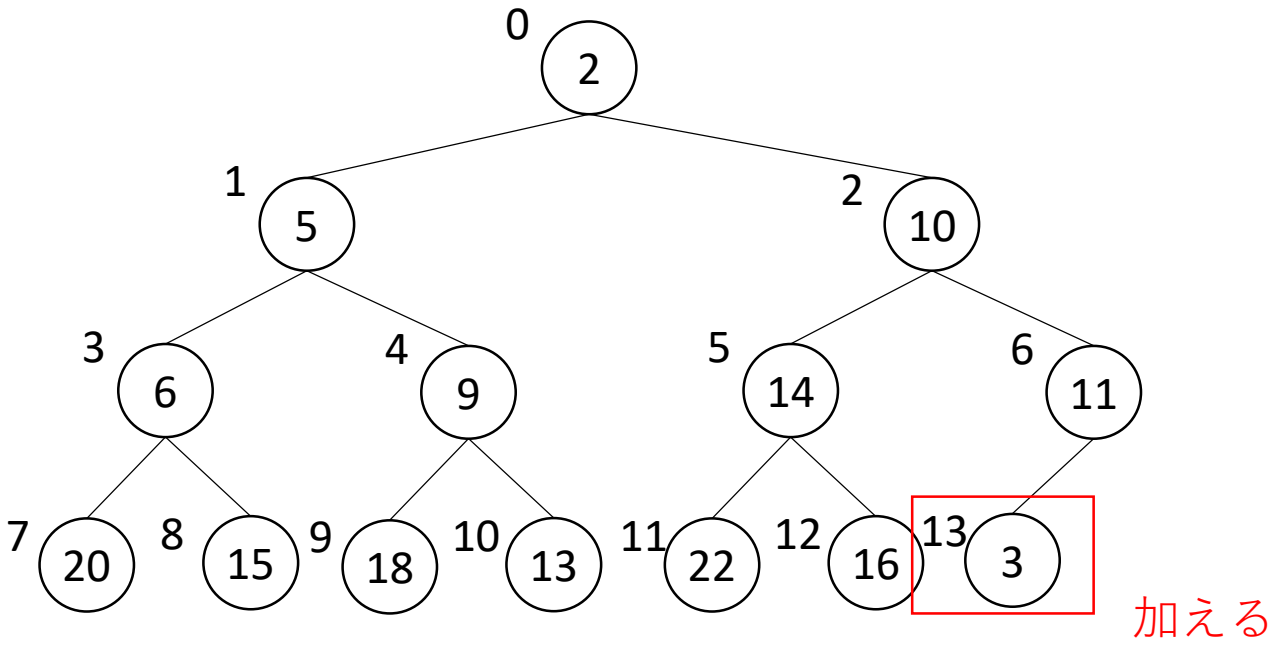
必ず子供よりも親が大きい構造を作る

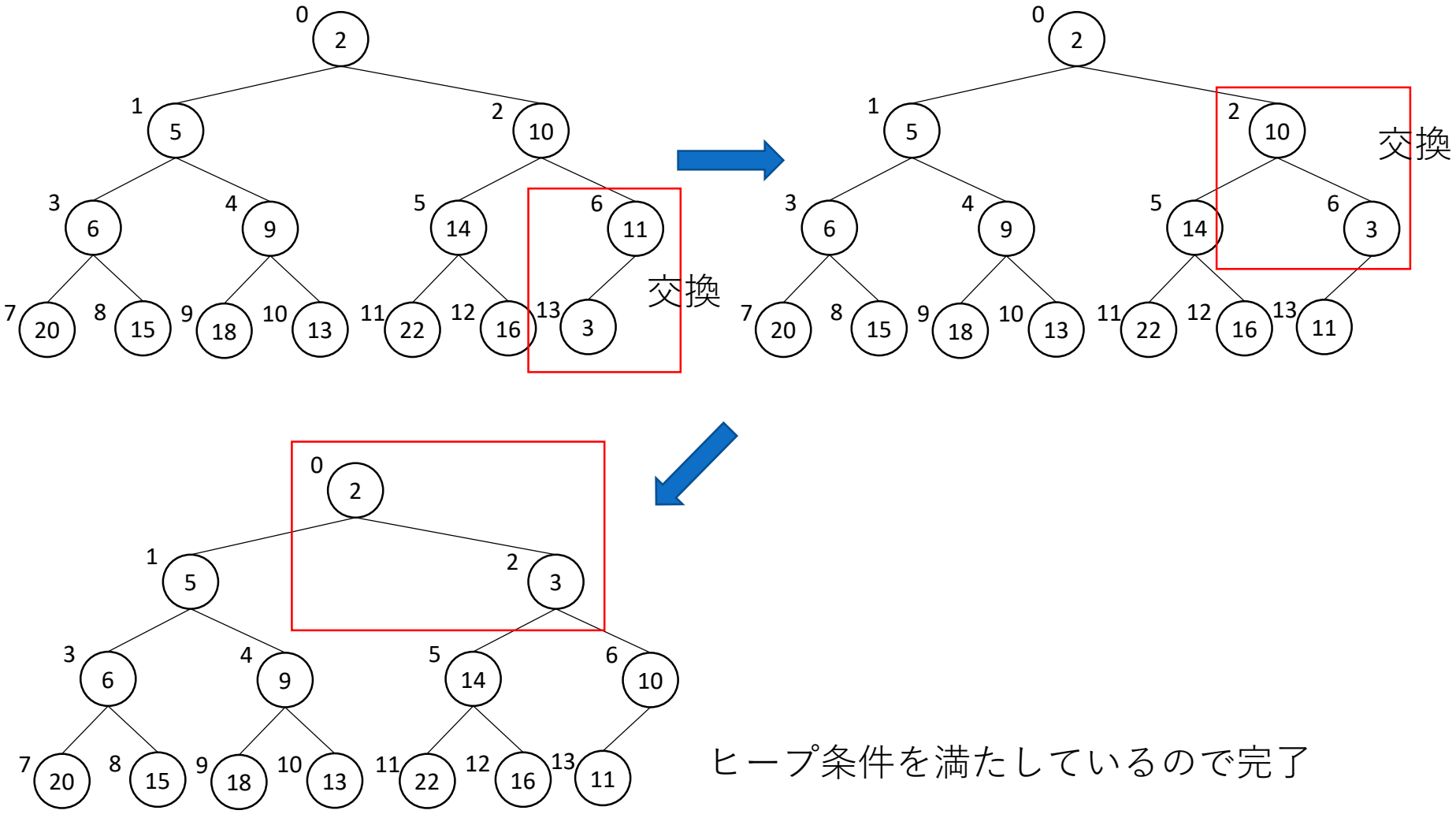


n	0	1	2	3	4	5	6	7	8	9	10	11	12
B(n)	2	5	10	6	9	14	11	20	15	18	13	22	16

数字追加のとき

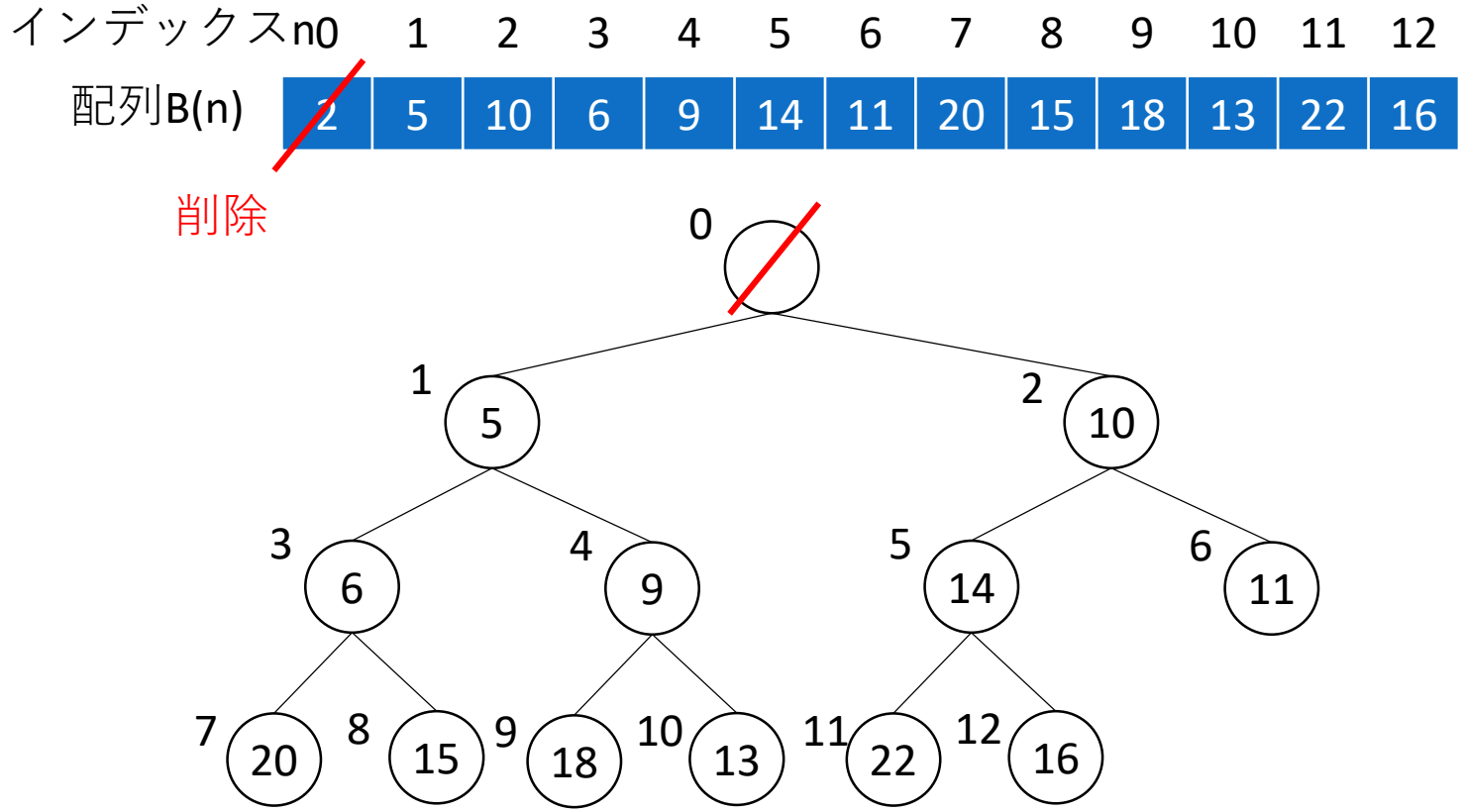
インデックスn	0	1	2	3	4	5	6	7	8	9	10	11	12	13
配列B(n)	2	5	10	6	9	14	11	20	15	18	13	22	16	3

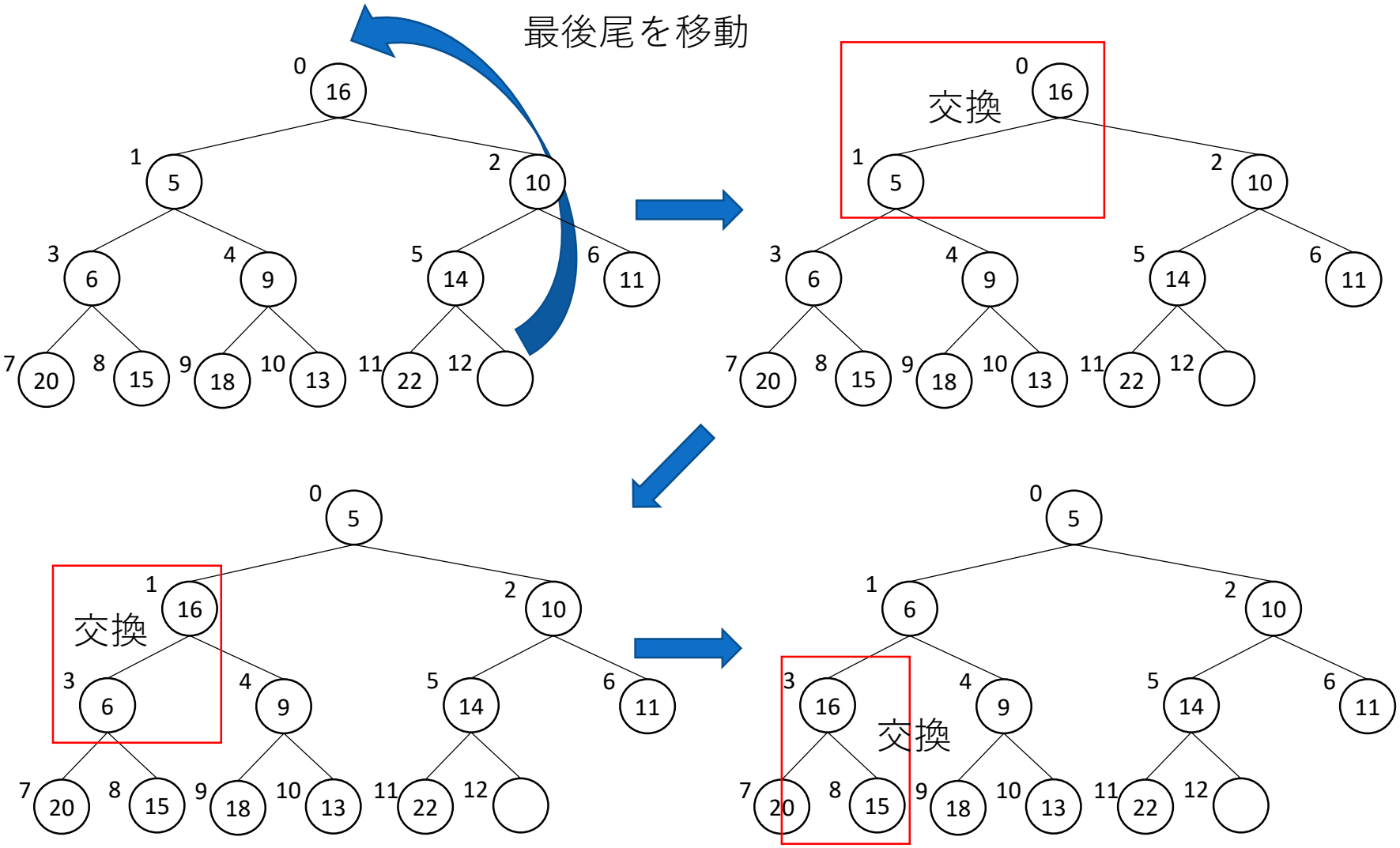


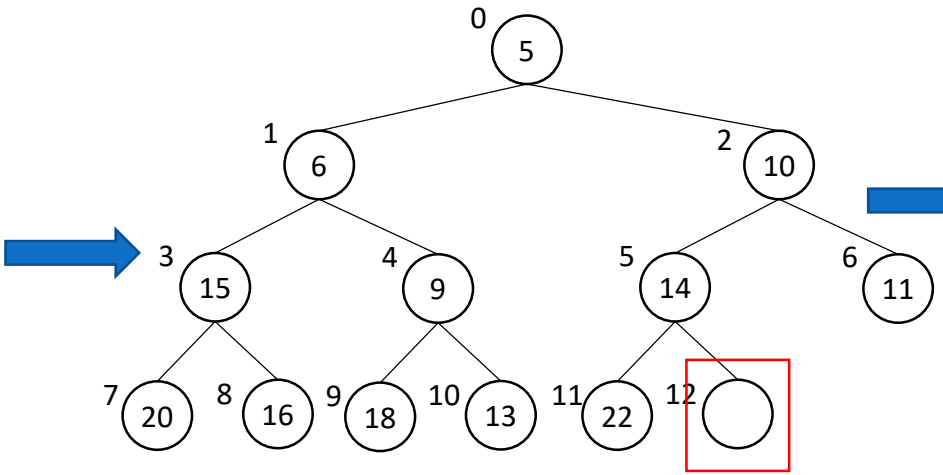


ヒープ条件を満たしているので完了

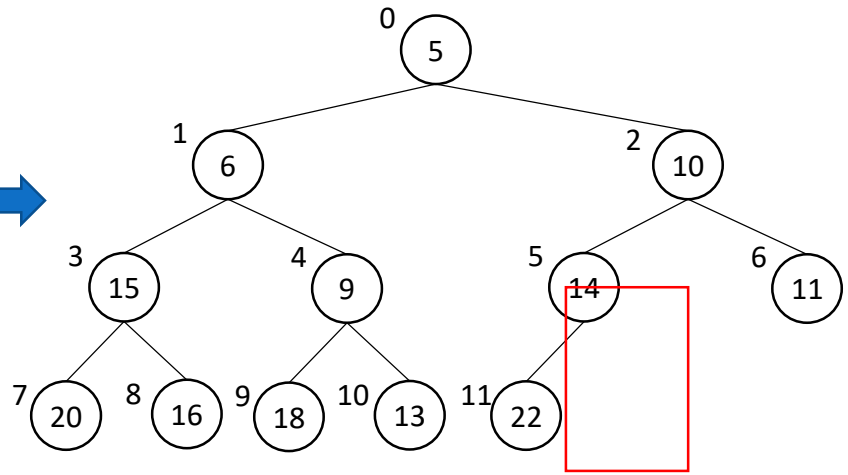
数字削除のとき


インデックスn0 1 2 3 4 5 6 7 8 9 10 11 12
配列B(n) 2 5 10 6 9 14 11 20 15 18 13 22 16
削除
0
1
5
2
10
3
6
4
9
5
14
6
11
7
20
8
15
9
18
10
13
11
22
12
16


最後尾を移動
交換
交換
交換
0 16
1 5
2 10
3 6
4 9
5 14
6 11
7 20
8 15
9 18
10 13
11 22
12



削除



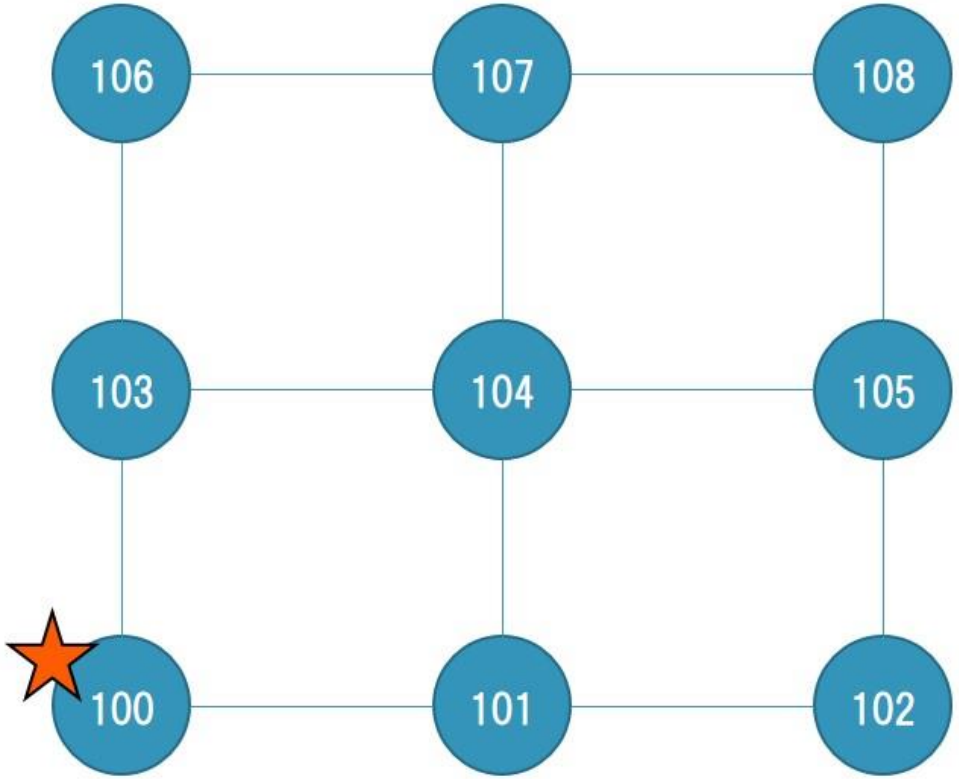
ヒープ条件を満たしているので完了

課題

仮想ネットワークに対する離散選択モデルの推定

（復習）前回配布データ

発表課題　データ概要



・円内の番号はNodeIDを表す
・現在★の位置にいて他のエリアに行く場合の目的地選択を考えます
・その他の詳細はデータのzip内にあるREADME.txtを参照

（復習）csvファイルの中身

- **Node.csv**

NodeID：各ノードにユニークなID，Lat:ノードの緯度，Lon:ノードの経度

- **Link.csv**

LinkID:各リンクにユニークなID，sNode:リンクの始点ノードID，eNode:リンクの終点ノードID

- **Area.csv**

AreaID:各目的地のユニークなID，distance:各目的地の出発地Node100からの距離，scale:各目的地の規模(どれほど栄えているか)の指標，NodeID:Areaに最も近いNodeのID（今回は各ノードとの一致を仮定）

- **Trip.csv**

TripID:各トリップにユニークなID，male:性別[個人属性1](男性なら1,女性なら0)，age:年齢[個人属性2]，oNode:トリップの出発地ノードID，dNode:トリップの目的地ノードID

- **Linkdetail.csv**

Link.csvに始点終点ノードの緯度経度データを結合させたファイル

- **mnl.csv**

Trip.csvに対して，各トリップが選択した目的地のノードID，距離，規模のデータを6行目から8行目に結合した．さらに，各トリップで選択されなかった目的地のノードID，距離，規模のデータを9行目以降に結合した．

（補）ネットワーク図の表現

隣接行列

各ノードが隣接しているか否かを0-1で表した行列

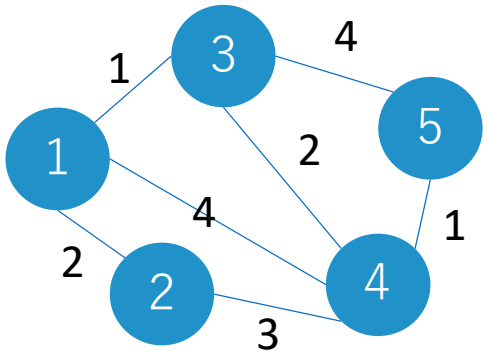
- 隣接している（経路がある）：1
- 隣接していない（経路がない）：0

※プログラミングにおいてノード・リンクからなるネットワークを処理するには，行列がよくつかわれます

※今回は考慮しませんが，一方通行を考慮することもできます

ラベル付き隣接行列

- 隣接を表す1の代わりに，各ルートに対応する費用（コスト）を用いたもの



隣接行列

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

ラベル付き隣接行列

$$\begin{bmatrix} 0 & 2 & 1 & 3 & 0 \\ 2 & 2 & 0 & 3 & 0 \\ 1 & 0 & 0 & 2 & 4 \\ 3 & 3 & 2 & 0 & 1 \\ 0 & 0 & 4 & 1 & 0 \end{bmatrix}$$

宿題

コードを２つお渡しするので，各コードを解読したうえで以下をやってみてください！

課題：目的地離散選択モデルの推定

ごめんなさい
書き直す時間がなかったのでRです

①Link.csvとNode.csvから「Linkの緯度経度」を引き出し，Link間距離を求める：コード１

②Link間距離をコストとした隣接行列を作成し，ダイクストラ法により，Trip.csvの各Tripに対する最短経路長を計算：コード２

③目的地選択MNLモデルを推定．説明変数を各自工夫し，結果からどのようなことが言えるのかまとめる．：前回コード

第三回（4/30）で全員に簡単な発表をしていただきます．

不明点があれば山野さんかわたしまでお気軽に～