In [ ]:

```
Crawler + Scrapper
HTTP - Req/Resp - Bytes - (TEXT/HTML)String => Requests, BeautifulSoup
HTML -> DOM : find~, select(CSS Selector)
HTML Tag, #id, .class, HTML Attributes, :가상선택자, [키(^$*)=밸류]
Crawler; HyperLink Web 탐색(a[href], iframe[src], img, ...)
         -> scheme://netloc(server domain)/path?params#fragment(X)
         BFS(Queue), DFS(Stack) : 검색결과 ; Depth
Focused Crawling: 전략(domain, path, 영역, Depth, ...)
Link Analysis => PageRank, 스크래핑
```

In [38]:

```
from urllib.robotparser import RobotFileParser
from requests import get
from requests.compat import urlparse, urljoin
from requests.exceptions import HTTPError
from time import sleep
import re
from bs4 import BeautifulSoup

headers = {
    'user-agent':'Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.3
}

URLs = list()
seens = list()
URLs.append({
    'url':'https://search.naver.com/search.naver?where=nexearch&query=%ED%95%9C%EC
    'depth':0
}) # 구조 변경. 기존 list에서 dict의 list로(keys:url, depth)

# 전략1. depth(0 -> 1 -> 2, ... )
#        [{url:'url', depth:0}, ...]
# 전략2. domain
#        blog.naver.com
allowedDomain = ['blog.naver.com', 'postfiles.pstatic.net']

while URLs:
    seed = URLs.pop(0) # BFS:0, DFS:-1
    seens.append(seed['url'])

    #전략1 적용
    if seed['depth'] > 3:
        continue

    # list에서 꺼낸 url은 dict 이므로, 실제 주소는 dict의 key:url
    resp = get(seed['url'], headers=headers)

    try:
        resp.raise_for_status()
    except HTTPError as e:
        print(e)
        continue

    # 전략3:텍스트/html + image/format
    if re.search('text|html|image|jpeg|png|gif|bmp',
                 resp.headers['content-type']) is None:
        continue

    if re.search('image|jpeg|png|gif|bmp', resp.headers['content-type']):
        # https://blog.naver.com/path1/path2/(image12432@!~14123)
        filename = resp.url.split('/')[-1]
        # (image12432@!~14123) => (image1243214123)
        filename = re.sub('[?#!]', '', filename)
        # image/(____)
        ext = re.search('image/(\w+);?',
                        resp.headers['content-type']).group(1)
        # filename: ./data/image1243214123.ext
        with open('./data/'+filename+'.'+ext, 'wb') as fp:
            fp.write(resp.content)
    else:
        # 위와 동일하게, filename 만들어서 w모드 저장(encoding) resp.text
        dom = BeautifulSoup(resp.text, 'html.parser')

        for link in dom.select('a[href], iframe[src], img[src]'):
            url = urljoin(seed['url'], link.attrs['href']
                          if link.has_attr('href') else link.attrs['src'])
            if len(urlparse(url).fragment) == 0 and\
               urlparse(url).scheme in ['http', 'https']:
                # {depth제한} => list의 dict 풀어서 => 주소만 있는 list
                # 전략2 적용 blog.naver.com
                if url not in [u['url'] for u in URLs] and \
                   url not in seens and \
                   urlparse(url).netloc in allowedDomain:
                    # 앞으로 방문할 URL목록에 dict로 추가
                    URLs.append({'url':url, 'depth':seed['depth']+1})
        print(len(URLs))
```

```
16
16
16
16
16
16
16
16
16
16
16
16
16
16
16
16
16
72
99
133
160
189
218
251
266
309
342
355
377
403
406
439
461
460
459
459
462
461
474
479
449
475
482
497
508
519
500 Server Error: Internal Server Error for url: https://blog.naver.com/FILEPATH
517
516
566
618
621
635
635
634
634
661
675
713
739
758
```

```
769
805
818
825
846
871
889
896
913
922
930
935
937
404 Client Error: Not Found for url: https://blog.naver.com/prologue/FILEPATH
936
935
934
938
978
988
1029
1028
1027
1026
1067
1108
1110
1131
1137
1152
1154
1180
1205
1224
1247
1265
1285
1302
1333
1358
1380
1418
1443
1477
1517
1553
1581
1612
1639
1665
1678
1697
1696
1719
1740
1756
1781
1761
1761
1761
1785
```

```
1784
1783
1931
2054
2240
2302
2458
2459
2492
2495
2495
2494
2494
2494
2493
2514
2549
2597
2611
2635
2654
2668
2686
2704
2716
2728
2749
2770
2787
2799
2810
2828
2834
2842
2844
2919
2916
2915
3012
3044
3080
3121
3162
3163
3162
3165
3164
3164
3163
3162
3188
3227
3248
3274
3292
3311
3336
3351
3379
3408
3438
```

```
3470
3497
3529
3536
3565
3572
3598
3603
3626
3644
3667
3679
3693
3703
3734
3754
3777
3782
3816
3817
3823
3840
3872
3867
3899
3906
3915
3921
3920
3921
3926
3929
3928
3913
3923
3896
3891
3890
3909
3950
3991
4023
4064
4063
4066
4066
4090
4105
4120
4140
4158
4176
4194
4224
4243
4282
4310
4334
4343
4361
4375
```

```
4381
4397
4413
4421
4433
4450
4467
4484
4499
4513
4522
4521
4503
4528
```

In [45]:

```
# 웹툰 크롤링+스크래핑
# 특정 도메인, 특정 영역, Depth X -> DHTML할때 다시 하기
URLs = ['https://comic.naver.com/webtoon']
visited = list()

while URLs:
    seed = URLs.pop(0) # Queue
    visited.append(seed)
    
    resp = get(seed, headers=headers)
    
    # 오류 처리(위 코드 참조)
    if resp.status_code != 200:
        continue
        
    if re.search('image', resp.headers['content-type']):
        filename = resp.url.split('/')[-1]
        filename = re.sub('[?#!= ]', '', filename)
        ext = re.search('image/(\w+);?',
                        resp.headers['content-type']).group(1)
        with open('./webtoon/'+filename+'.'+ext, 'wb') as fp:
            fp.write(resp.content)
    if re.search('html', resp.headers['content-type']):
        dom = BeautifulSoup(resp.text, 'html.parser')
        # 영역 제한 - 1 (웹툰 목록)
        for a in dom.select('ul[class$="R52q0"] a[href^="/webtoon/"]'):
            nurl = urljoin(seed, a.attrs['href'])
            if nurl not in URLs and\
               nurl not in visited:
                URLs.append(nurl)
        # 영역 제한 - 2 (특정 웹툰의 회차 목록)
        for a in dom.select('li[class$="M8zq4"] > a[href^="/webtoon/"]'):
            nurl = urljoin(seed, a.attrs['href'])
            if nurl not in URLs and\
               nurl not in visited:
                URLs.append(nurl)
        # 영역 제한 - 3 (특정 웹툰의 특정 회차의 이미지 목록)
        for img in dom.select('img[id^=content_image_]'):
            nurl = urljoin(seed, a.attrs['src'])
            if nurl not in URLs and\
               nurl not in visited:
                URLs.append(nurl)
        print(len(URLs))
```

```
0
```

In [56]:

```
# 뉴스 크롤링+스크래핑
# 특정 도메인, 특정 영역, Depth X
URLs = ['https://news.naver.com/']
visited = list()

while URLs:
    seed = URLs.pop(0) # Queue
    visited.append(seed)

    resp = get(seed, headers=headers)

    # 오류 처리(위 코드 참조)
    if resp.status_code != 200:
        continue

    if re.search('image', resp.headers['content-type']):
        filename = resp.url.split('/')[-1]
        filename = re.sub('[?#!= ]', '', filename)
        ext = re.search('image/(\w+);?',
                        resp.headers['content-type']).group(1)
        with open('./news/'+filename+'.'+ext, 'wb') as fp:
            fp.write(resp.content)
    if re.search('html', resp.headers['content-type']):
        dom = BeautifulSoup(resp.text, 'html.parser')
        # 영역 제한 - 1 (뉴스 카테고리)
        for a in dom.select('[role=menu] a')[1:7]:
            nurl = urljoin(seed, a.attrs['href'])
            if nurl not in URLs and\
               nurl not in visited:
                URLs.append(nurl)
        # 영역 제한 - 2 (특정 뉴스 카테고리 - 뉴스 목록)
        for a in dom.select('a.cluster_text_headline'):
            nurl = urljoin(seed, a.attrs['href'])
            if nurl not in URLs and\
               nurl not in visited:
                URLs.append(nurl)
        # 영역 제한 - 3 (특정 뉴스 한 개)
        if dom.select_one('#contents'):
            # 파일로 저장 - 뉴스
            filename = resp.url.split('/')[-1]
            filename = re.sub('[?#!= ]', '', filename)
            with open('./news/'+filename+'.txt',
                      'w', encoding='utf8') as fp:
                fp.write(dom.select_one('#contents').get_text().strip())

            for img in dom.select(
                '#contents img[src], #contents img[data-src]'):
                nurl = urljoin(seed, img.attrs['src'
                    if img.has_attr('src') else 'data-src'])
                if nurl not in URLs and\
                   nurl not in visited:
                    URLs.append(nurl)
        print(len(URLs))
```

```
6
49
82
125
152
179
210
211
212
213
214
215
216
217
218
219
220
219
220
219
220
219
220
221
220
221
220
219
220
219
218
219
220
221
222
221
222
223
222
221
220
221
220
219
218
217
216
215
214
213
212
211
210
211
212
211
212
211
210
211
212
```

```
211
212
211
212
211
210
209
210
211
210
209
208
207
206
205
206
205
204
203
204
205
204
203
202
203
202
203
202
201
200
199
198
197
196
195
194
195
194
193
192
193
192
191
190
189
190
189
188
187
186
185
186
185
184
185
186
185
184
183
182
181
```

```
180
179
178
177
176
175
174
173
172
171
172
171
170
171
170
169
168
169
168
167
166
165
164
163
162
161
160
159
158
157
156
155
154
153
152
151
150
149
150
149
148
147
146
145
144
143
142
141
140
139
138
137
136
135
134
133
132
131
130
129
128
```

```
127
126
125
124
123
122
121
122
121
120
119
118
117
116
115
114
113
112
111
110
109
108
107
106
105
104
103
102
101
100
101
100
99
98
```

In [57]:

```
실습: 다음 뉴스에서 위와 같이 크롤링(with 스크래핑) 하기
```

Out[57]:

```
['https://news.naver.com/',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=100',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=101',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=102',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=103',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=105',
 'https://news.naver.com/main/main.naver?mode=LSD&mid=shm&sid1=104',
 'https://n.news.naver.com/mnews/article/016/0002117583?sid=100',
 'https://n.news.naver.com/mnews/article/417/0000904326?sid=100',
 'https://n.news.naver.com/mnews/article/214/0001260191?sid=100']
```