
알고리즘분석 실습 자료
4주차
Photo by Piotr Guzik on Unsplash


실습 소요 시간 100분

2장 분할정복법 (divide-and-conquer)

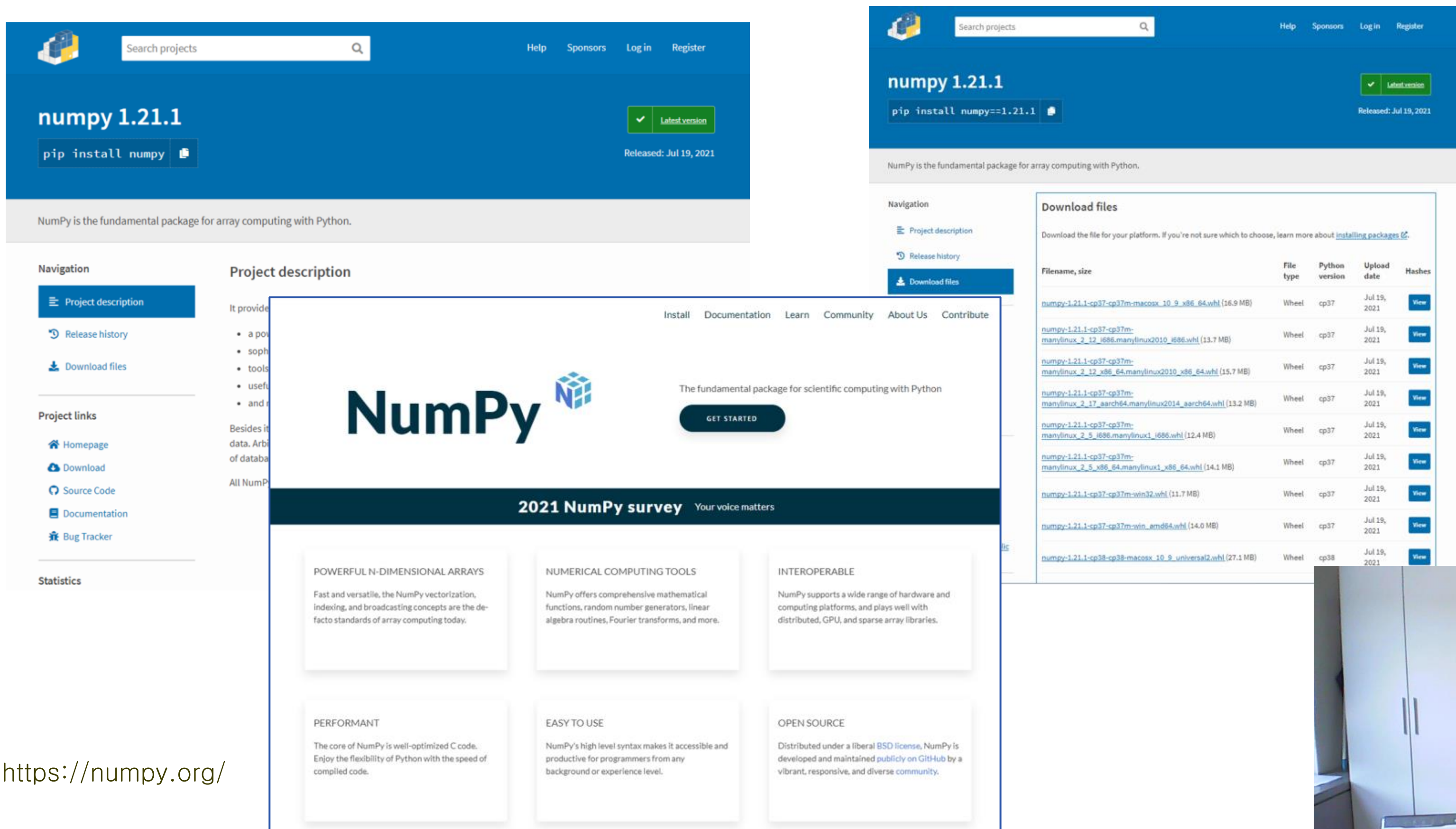
실습프로그램

- ✓ 빠른정렬
- ✓ 쉬트라센 알고리즘
- ✓ 큰 정수 곱셈

numpy

- numerical python–파이썬으로 과학,공학 계산을 하기 위해 개발된 기본적인 패키지
- 다운로드 사이트: https://pypi.org/project/numpy/
- 파이썬의 해당 버전에 연결된 numpy 설치
- 설치는 command 창에서 파일이 저장된 디렉토리로 이동 후 “pip install 다운로드한 파일명”
- 프로그램 내에 사용 시 상단에 import numpy

  또는 import numpy as np 또는 from numpy import *



https://numpy.org/

array

- 파이썬에는 array 자료구조가 없으나, numpy 에서는 array 구조 지원
- array 구조는 수학적인 표현처럼 행렬, 행렬 원소를 표시할 수 있다.
- numpy 를 줄여 쓰기 위해 import numpy as np 사용

```
import numpy
a=[1,2,3]
print(a)
b=numpy.array(a)
print(b)
```

```
[1, 2, 3]
[1 2 3]
>>>
```

```
import numpy as np
a=[1,2,3]
print(a)
b=np.array(a)
print(b)
```

```
[1, 2, 3]
[1 2 3]
>>>
```

```
import numpy as np
c=[[1,2,3],[5,6,7]]
print(c)
print(c[1][2])
d=np.array(c)
print(d)
print(d[1,2])
print(d[1][2])
```

```
[[1, 2, 3], [5, 6, 7]]
7
[[1 2 3]
 [5 6 7]]
7
7
>>>
```

- from numpy import * 사용 시에는 numpy의 함수명을 numpy 또는 np 없이 사용
- numpy 의 함수를 강조하기 위해(혼돈하지 않기 위해) import numpy as np 권장

```python
from numpy import *
a=[1,2,3]
b=array(a)
print(a)
print(b)
```

```
[1, 2, 3]
[1 2 3]
>>>
```

@ 는 numpy의 행렬의 곱셈 operator

```
import numpy as np
a=[[1,2],[3,4]]
b=[[1,0],[2,5]]
c= np.array(a)
d= np.array(b)
f = c @ d
print(f)
```

```
[[ 5 10]
 [11 20]]
>>>
```

행렬의 덧셈, 뺄셈 가능

```
import numpy as np
a=[  [1,2],[3,4]]
b=[[1,0],[2,5]]
c= np.array(a)
d= np.array(b)
f = c + d
print(f)
g = c-d
print(g)
```

```
[[2 2]
 [5 9]]
[[ 0  2]
 [ 1 -1]]
>>>
```

행렬의 초기화

```
import numpy as np
A = [[2 for j in range(3)] for i in range(2)]
print(A)
B=np.array(A)
print(B)
```

```
[[2, 2, 2], [2, 2, 2]]
[[2 2 2]
 [2 2 2]]
>>>
```

열(column)의 인덱스 행(row)의 인덱스

2	2	2
2	2	2

```
import numpy as np
A = np.array([[i+j for j in range(3)] for i in range(2)])
print(A)
```

```
[[0 1 2]
 [1 2 3]]
>>>
```

0	1	2
1	2	3

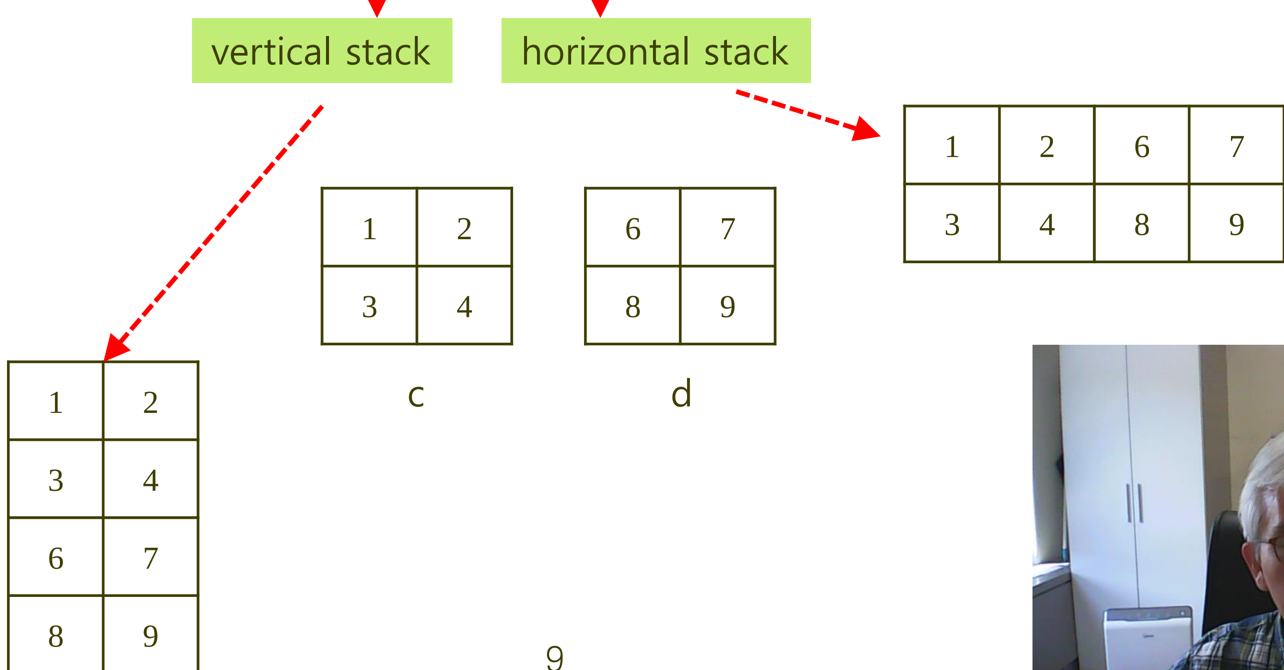
행렬의 병합

```
import numpy as np

a=[[1,2],[3,4]]
b=[[6,7],[8,9]]
c=np.array(a)
d=np.array(b)
print(np.hstack([c,d]))
print()
print(np.vstack([c,d]))
```

```
[[1 2 6 7]
 [3 4 8 9]]

[[1 2]
 [3 4]
 [6 7]
 [8 9]]
>>>
```



빠른정렬 알고리즘

- 문제: n개의 정수를 비내림차순으로 정렬
- 입력: 정수 $n > 0$, 크기가 n인 배열 S[1..n]
- 출력: 비내림차순으로 정렬된 배열 S[1..n]
- 알고리즘:

```
void quicksort (index low, index high) {
     index pivotpoint;
     if (high > low) {
          partition(low,high,pivotpoint);
          quicksort(low,pivotpoint-1);
          quicksort(pivotpoint+1,high);
     }
}
```

분할 알고리즘

- 문제: 빠른정렬을 하기 위해서 배열 S를 둘로 나눈다.
- 입력: (1) 첨자 low, high (2) S의 부분배열 (첨자는 low에서 high)
- 출력: 첨자 low에서 high까지의 S의 부분배열의 기준점(pivot point), pivotpoint

```
void partition (index low, index high, index& pivotpoint) {
      index i, j;
      keytype pivotitem;
      pivotitem = S[low];        //pivotitem으로 첫번째 항목을 고른다
      j = low;
      for(i = low + 1; i <= high; i++)
            if (S[i] < pivotitem) {
               j++;
               exchange S[i] and S[j];
            }
      pivotpoint = j;
      exchange S[low] and S[pivotpoint];// pivotitem 값을 pivotpoint에 넣는다
}
```

j: pivotitem 보다 작은 그룹의 제일 우측끝 데이터의 위치

– not stable

[실습프로그램] 빠른정렬

```
def quickSort(s,low, high):



     구현


def partition(s,low,high):


     구현

s=[3,5,2,9,10,14,4,8]
quickSort(s,0,7)
print(s)
```

[선택]

- 빠른 정렬의 평균 시간 복잡도 O(nlogn)을 확인하는 프로그램을 작성

행렬 곱셈(matrix multiplication)

- 단순한 행렬곱셈 알고리즘
  - ✓ **문제**: $n \times n$ 크기의 행렬의 곱을 구하시오.
  - ✓ **입력**: 양수 n, $n \times n$ 크기의 행렬 A와 B
  - ✓ **출력**: 행렬 A와 B의 곱인 C

```
void matrixmult (int n, const number A[][], const number B[][],
                                          number C[][]) {
    index i, j, k;
     for (i = 1; i <= n; i++)
          for (j = 1; j <= n; j++) {
              C[i][j] = 0;
              for (k = 1; k <= n; k++)
                  C[i][j] = C[i][j] + A[i][k] * B[k][j];
           }
}
```

$n \times n$ 행렬곱셈: 쉬트라쎈의 방법

- 문제: n이 2의 거듭제곱이고, 각 행렬을 4개의 부분행렬(submatrix)로 나눈다고 가정하자. 두 $n \times n$ 행렬 A와 B의 곱 C:

$$\left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right] = \left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \times \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]$$

- 쉬트라쎈(Strassen)의 해:

$$C = \begin{bmatrix} M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\ M_2 + M_4 & M_1 + M_3 - M_2 + M_6 \end{bmatrix}$$

여기서

$$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$
$$M_2 = (A_{21} + A_{22}) \times B_{11}$$
$$M_3 = A_{11} \times (B_{12} - B_{22})$$
$$M_4 = A_{22} \times (B_{21} - B_{11})$$
$$M_5 = (A_{11} + A_{12}) \times B_{22}$$
$$M_6 = (A_{21} - A_{11}) \times (B_{11} + B_{12})$$
$$M_7 = (A_{12} - A_{22}) \times (B_{21} + B_{22})$$

$$\begin{array}{c} A_{11} \\ \\ \\ A_{21} \end{array} \left(\begin{array}{cc|cc} x & x & x & x \\ x & x & x & x \\ \hline x & x & x & x \\ x & x & x & x \end{array}\right) \begin{array}{c} A_{12} \\ \\ \\ A_{22} \end{array}$$

쉬트라쎈의 알고리즘

- **문제**: n이 2의 거듭제곱일 때, $n \times n$ 크기의 두 행렬의 곱을 구하시오.
- **입력**: 정수 n, $n \times n$ 크기의 행렬 A와 B
- **출력**: 행렬 A와 B의 곱인 C

```
void strassen (int n, n*n_matrix A, n*n_matrix B, n*n_matrix& C) {
     if (n <= 임계점)
          단순한 알고리즘을 사용하여 C = A * B를 계산;
     else {
          A를 4개의 부분행렬 A11, A12, A21, A22로 분할;
          B를 4개의 부분행렬 B11, B12, B21, B22로 분할;
          쉬트라쎈의 방법을 사용하여 C = A * B를 계산;
          // 되부르는 호출의 예: strassen(n/2, A11+A12, B11+B22, M1)
     }
}
```

$$M_1 = (A_{11} + A_{22}) \times (B_{11} + B_{22})$$

- 용어: 임계점(threshold)이란? 두 알고리즘의 효율성이 교차하는 문제의 크기.

[실습프로그램] 쉬트라센 알고리즘

```
from numpy import *
def strassen (n, A, B, C):
    threshold = 2
    A11 = array([[A[rows][cols] for cols in range(int(n/2))]for rows in range(int(n/2))])

          구현

#    print(A11, A12, A21, A22, B11, B12, B21, B22)
    if (n <= threshold):
        C = array(A) @ array(B)
    else:
        M1 = M2 = M3 = M4 = M5 = M6 = M7 = array([])
        M1=strassen(int(n/2), (A11 + A22) , (B11 + B22), M1)


          구현

        C =  vstack([ hstack([M1+M4 -M5 + M7, M3 + M5]), hstack([M2 + M4, M1 + M3 - M2 + M6]) ])
    return C
n = 4
#A = [[1 for cols in range(n)]for rows in range(n)]
#B = [[2 for cols in range(n)]for rows in range(n)]
A=[ [1,2,0,2], [3,1,0,0], [0,1,1,2],[2,0,2,0]]
B=[ [0,3,0,2], [1,1,4,0], [1,1,0,2],[0,5,2,0]]
C = array(A)@array(B)
D = [[0 for cols in range(n)]for rows in range(n)]
print(C)
D=strassen(n, A, B, D)
print(D)
```

```
[[ 2 15 12  2]
 [ 1 10  4  6]
 [ 2 12  8  2]
 [ 2  8  0  8]]
[[ 2 15 12  2]
 [ 1 10  4  6]
 [ 2 12  8  2]
 [ 2  8  0  8]]
>>>
```

큰 정수 계산법

- 하드웨어의 용량을 초과하는 정수연산 – 천문학
  - ✓ 정수 배열을 이용한 큰 정수의 표현
  - ✓ (예) 543,127

5	4	3	1	2	7
S[6]	S[5]	S[4]	S[3]	S[2]	S[1]

  - ✓ n: 큰 정수의 숫자(digit) 개수
    - 단순 곱셈은 n^2시간 걸림. 덧셈/뺄셈은 1차 시간에 수행 가능
    - 1차시간 가능: $u \times 10^m$, u divide 10^m , u mod 10^m
  - ✓ $567{,}832 = 567 \times 10^3 + 832$,
  - ✓ $9{,}423{,}723 = 9423 \times 10^3 + 723$

$$\underbrace{u}_{n \text{ digits}} = \underbrace{x}_{\lceil n/2 \rceil \text{ digits}} \times 10^m + \underbrace{y}_{\lfloor n/2 \rfloor \text{ digits}}$$

$$m = \left\lfloor \frac{n}{2} \right\rfloor$$

```
    aaaa
x   bbbb
--------
    cccc
   cccc
  cccc
 cccc
```

$$u = x \times 10^m + y, \; v = w \times 10^m + z$$

$$u \times v = (x \times 10^m + y) \times (w \times 10^m + z)$$

$$= xw \times 10^{2m} + (xz + wy) \times 10^m + yz$$

큰 정수곱셈

- **문제**: 2개의 큰 정수 *u*와 *v*를 곱하라
- **입력**: 큰 정수 *u*와 *v*, 크기 *n*
- **출력**: prod(*u*와 *v*의 곱)

```
large_integer prod(large_integer u, large_integer v){
        large_integer x, y, w, z;
        int n, m;

        n = maximum(u의 자리수, v의 자리수);
        if(u == 0 || v == 0) return 0;
        else if( n <= threshold)
            return 일반적인 방법으로 구한 u × v ;
        else{
          m = ⌊n/2⌋;
          x = u divide 10^m; y = u mod 10^m;
          w = v divide 10^m; z = v mod 10^m;
          return prod(x, w) × 10^2m +
                (prod(x, z)+prod(w, y)) × 10^m + prod(y, z);
          }
}
```

- prod 최악의 경우 시간복잡도 분석:
  - **단위연산**: 덧셈, 뺄셈, divide 10^m, mod 10^m, $\times 10^m$
  - **입력크기**: 정수의 자리수, n
  - n이 2의 거듭제곱 형태라고 가정
  - 덧셈, 뺄셈, divide 10^m, mod 10^m, $\times 10^m$ 에 있는 1차시간 연산은 모두 cn으로 표시

$$W(n) = 4W(\frac{n}{2}) + cn, \quad n > s \text{ 이고}, n\text{이 2의 거듭제곱인 경우}$$

$$W(s) = 0$$

s=threshold보다 작거나 같은 문제크기. W(s)의 단위연산 횟수는 0

$$W(n) \in \Theta(n^{\lg 4}) = \Theta(n^2)$$

Appendix Theorem B.5
The Master Theorem

- 개선된 방법:
  - ✓ 이전 방법에서는 xw, $xz+yw$, yz의 계산 필요 ➔ 4회의 곱셈
  - ✓ 개선방법: r 계산 추가

$$r = (x+y) \times (w+z) = xw + (xz+yw) + yz$$

① ② ③

$$xz+yw = r - xw - yz$$

$$u \times v = (x \times 10^m + y) \times (w \times 10^m + z)$$
$$= xw \times 10^{2m} + (xz+wy) \times 10^m + yz$$

(1) ① 계산 수행

(2) ②, ③ 계산 : xw, yz 구함

(3) r의 값에서 ②, ③의 계산 결과를 빼줌 : $xz+yw$ 구함

- 결과적으로 xw, $xz+yw$, yz을 계산하는데, 덧셈/뺄셈의 회수는 증가지만, 곱셈은 3회 필요

큰 정수곱셈2

- **문제**: 2개의 큰 정수 *u*와 *v*를 곱하라
- **입력**: 큰 정수 *u*와 *v*, 크기 *n*
- **출력**: prod2(*u*와 *v*의 곱)

```
large_integer prod2(large_integer u, large_integer v){
        large_integer x, y, w, z, r, p, q;
        int n, m;
        n = maximum(u의 자리수, v의 자리수);
        if(u == 0 || v == 0) return 0;
        else if( n <= threshold)
            return 일반적인 방법으로 구한 u × v ;
        else{
          m = ⌊n/2⌋;
          x = u divide 10^m; y = u mod 10^m;
          w = v divide 10^m; z = v mod 10^m;
          r = prod2(x+y,w+z);
          p = prod2(x, w);
          q = prod2(y, z);
          return p×10^2m + (r-p-q)×10^m + q;
          }
}
```

[실습프로그램] 큰 정수 곱셈

```
def prod2(a,b):




     구현






a=12345678123456 78
b=23456789234567 89

print(prod2(a,b))
print(a*b)
```

threshold=4

강의가 곧 시작됩니다.