컴퓨터 알고리즘 HW#6

학번 : 201624548
이름 : 이재원
학과 : 정보컴퓨터 공학부

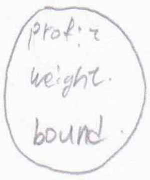
1. W=13

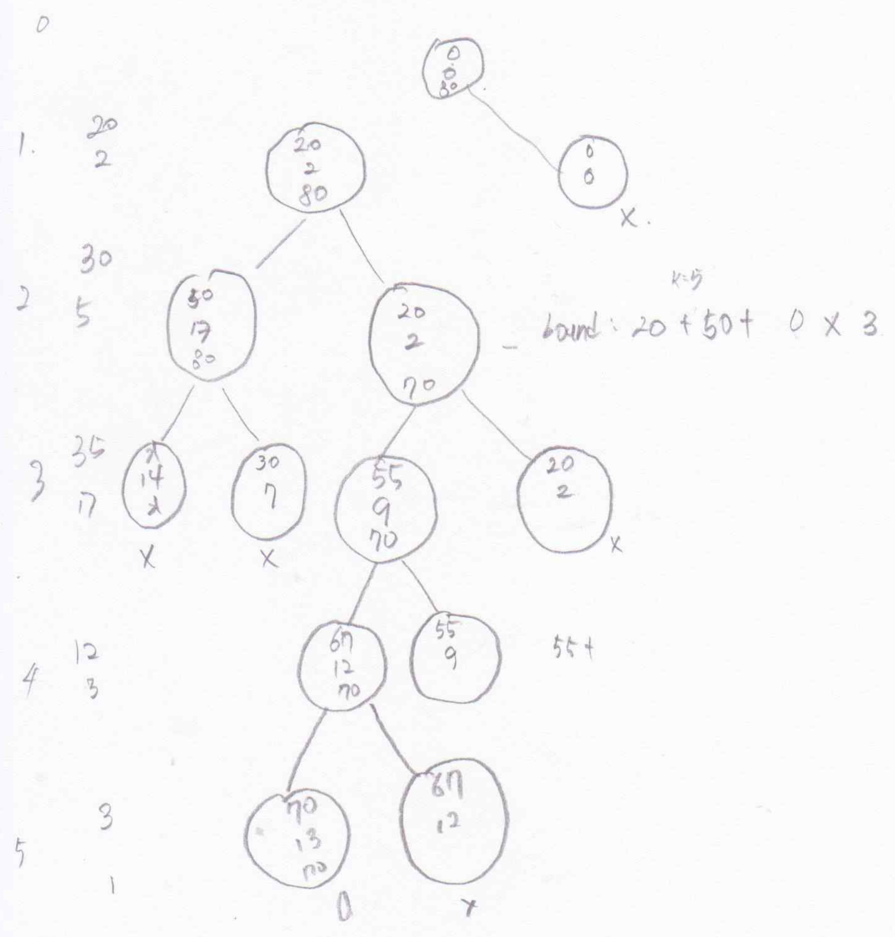
tot weight = 7., K=3.

bound: profit + $\sum_{1}^{2} p_i$ + $6 \times \frac{p_3}{w_3}$

0. 50 $\frac{}{5}$. 30

node





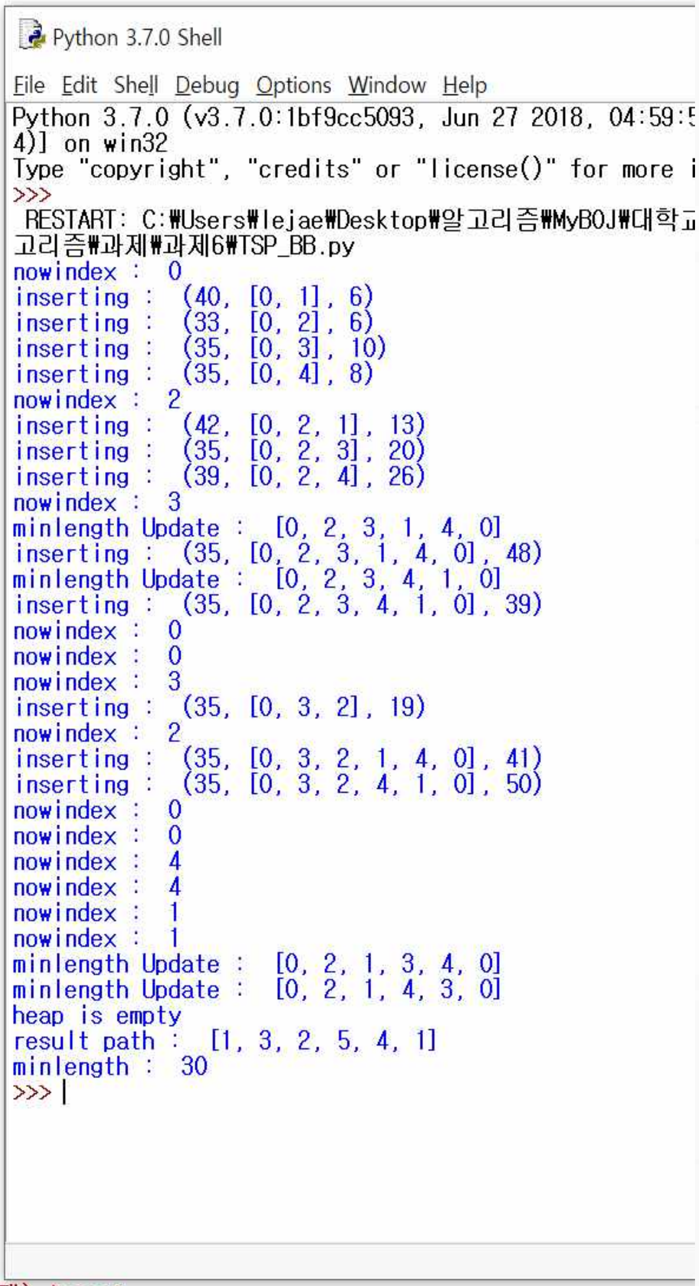
8. 11번을 위해 구현한 알고리즘에서 스텝별로 로그를 띄웠다.. (손으로 하기엔 양이 많다..)

```python
import heapq
import sys
import copy
INF = sys.maxsize
# 책에서 설명한 대로 Bound를 반환한다.
def bound(route, path) :
    _bound = list()
    for tmplist in route :
        tmparray = list()
        for i in range(len(tmplist)) :
            if i not in path and tmplist[i] != 0 :
                tmparray.append(tmplist[i])
        _bound.append(min(tmparray))
    return sum(_bound)

# Travel algorithm
# heap (bound == boundsum, path, length)
def travel(route,v_start) :
    heap = list()
    len_route = len(route)
    path = [v_start]
    boundsum = bound(route, path)
    heapq.heappush(heap,(boundsum, path, 0))
    minlength = INF
    while heap :
        nxt = heapq.heappop(heap)
        print("nowindex : " ,nxt[-1-1][-1])
        for i in range(1, len_route) :
            # copy된 값 에러를 없애기 위해 deepcopy 사용.
            boundsum, path, length = tuple(copy.deepcopy(nxt))
            if i in path : continue
            # path, length 갱신
            path.append(i)
            length = length + route[path[-1-1]][path[-1]]
            # 마지막 vertex만 남았을 때
            if len(path) == len_route-1 :
                v_last = 0
                for i in range(len_route) :
                    if i not in path :
                        v_last = i
                        length += route[path[-1]][v_last]
                        path.append(i)
                path.append(0)
                length += route[v_last][0]
                if length < minlength :
                    minlength = length
                    result_tour = path
                    print("minlength Update : ", path)
            else :
                boundsum = bound(route, path)
            # bound가 minlength보다 작을때 (방문해볼 가치가 있을 때) insert
            if boundsum < minlength :
                print("inserting : ",(boundsum, path,length))
                heapq.heappush(heap, (boundsum, path, length))
    print("heap is empty")
    return path, minlength


def main() :
    route = [ [0,6,6,10,8],
              [3,0,12,7,6],
              [8,7,0,14,20],
              [5,13,9,0,8],
              [9,8,10,6,0]
            ]
    length = 0
    result = travel(route,0)
    for i in range(len(result[0])) :
        result[0][i] += 1
    print("result path : ", result[0])
    print("minlength : ", result[1])

main()
```



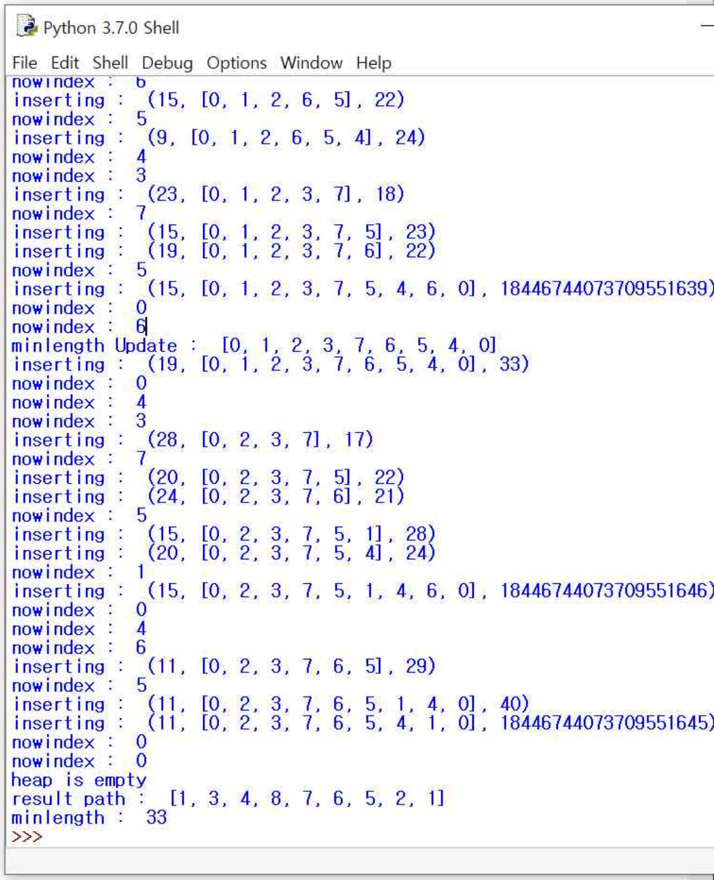
```
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:59:5
4)] on win32
Type "copyright", "credits" or "license()" for more i
>>>
 RESTART: C:\Users\lejae\Desktop\알고리즘\MyBOJ\대학교
고리즘\과제\과제6\TSP_BB.py
nowindex :  0
inserting :  (40, [0, 1], 6)
inserting :  (33, [0, 2], 6)
inserting :  (35, [0, 3], 10)
inserting :  (35, [0, 4], 8)
nowindex :  2
inserting :  (42, [0, 2, 1], 13)
inserting :  (35, [0, 2, 3], 20)
inserting :  (39, [0, 2, 4], 26)
nowindex :  3
minlength Update :  [0, 2, 3, 1, 4, 0]
inserting :  (35, [0, 2, 3, 1, 4, 0], 48)
minlength Update :  [0, 2, 3, 4, 1, 0]
inserting :  (35, [0, 2, 3, 4, 1, 0], 39)
nowindex :  0
nowindex :  0
nowindex :  3
inserting :  (35, [0, 3, 2], 19)
nowindex :  2
inserting :  (35, [0, 3, 2, 1, 4, 0], 41)
inserting :  (35, [0, 3, 2, 4, 1, 0], 50)
nowindex :  0
nowindex :  0
nowindex :  4
nowindex :  4
nowindex :  1
nowindex :  1
minlength Update :  [0, 2, 1, 3, 4, 0]
minlength Update :  [0, 2, 1, 4, 3, 0]
heap is empty
result path :  [1, 3, 2, 5, 4, 1]
minlength :  30
>>> |
```

11. 8번의 문제에서 bound 알고리즘과 main 함수만 수정되었다.
함수

```python
import heapq
import sys
import copy
INF = sys.maxsize
'''
# 문제 8번
# 책에서 설명한 대로 Bound를 반환한다.
def bound(route, path) :
    _bound = list()
    for tmplist in route :
        tmparray = list()
        for i in range(len(tmplist)) :
            if i not in path and tmplist[i] != 0 :
                tmparray.append(tmplist[i])
        _bound.append(min(tmparray))
    return sum(_bound)
'''
# 문제 11번
# 연결되지 않은 부분을 INF로 정의
# 리스트가 빈 경우 0을 append
# INF는 고려 X
def bound(route, path) :
    _bound = list()
    for tmplist in route :
        tmparray = list()
        for i in range(len(tmplist)) :
            if i not in path and tmplist[i] != 0 and tmplist[i] != INF :
                tmparray.append(tmplist[i])
        if tmparray : _bound.append(min(tmparray))
        else : _bound.append(0)
    return sum(_bound)

# Travel algorithm
# heap (bound == boundsum, path, length)
def travel(route,v_start) :
    heap = list()
    len_route = len(route)
    path = [v_start]
    boundsum = bound(route, path)
    heapq.heappush(heap,(boundsum, path, 0))
    minlength = INF
    while heap :
        nxt = heapq.heappop(heap)
        print("nowindex : " ,nxt[-1-1][-1])
        for i in range(1, len_route) :
            # copy된 값 에러를 없애기 위해 deepcopy 사용.
            boundsum, path, length = tuple(copy.deepcopy(nxt))
            if i in path : continue
            # path, length 갱신
            path.append(i)
            length = length + route[path[-1-1]][path[-1]]
            #INF일 경우 continue
            if route[path[-1-1]][path[-1]] == INF : continue
            # 마지막 vertex만 남았을 때
            if len(path) == len_route-1 :
                v_last = 0
                for i in range(len_route) :
                    if i not in path :
                        v_last = i
                        length += route[path[-1]][v_last]
                        path.append(i)
                path.append(0)
                length += route[v_last][0]
                if length < minlength :
                    minlength = length
                    result_tour = path
                    print("minlength Update : ", path)
            else :
                boundsum = bound(route, path)
            # bound가 minlength보다 작을때 (방문해볼 가치가 있을 때) insert
            if boundsum < minlength :
                print("inserting : ",(boundsum, path,length))
                heapq.heappush(heap, (boundsum, path, length))
    print("heap is empty")
    return path, minlength


def main() :
    route = [ [INF,5,8,INF, INF, INF, INF, INF],
              [INF, INF, 4, INF, 4, INF, INF, INF],
              [INF, INF, INF, 2, INF, INF, 5, INF],
              [INF, INF, INF, INF, INF, INF, INF, 7],
              [1, INF, INF, INF, INF, INF, INF, INF],
              [INF, 6, INF, INF, 2, INF, INF, INF],
              [INF, INF, INF, INF, INF, 8, INF, INF],
              [INF, INF, INF, INF, INF, 5, 4, INF]
              ]
    for i in range(len(route)) :
        route[i][i] = 0
    length = 0
    result = travel(route,0)
    for i in range(len(result[0])) :
        result[0][i] += 1
    print("result path : ", result[0])
    print("minlength : ", result[1])
main()
```



```
nowindex :  6
inserting :  (15, [0, 1, 2, 6, 5], 22)
nowindex :  5
inserting :  (9, [0, 1, 2, 6, 5, 4], 24)
nowindex :  4
nowindex :  3
inserting :  (23, [0, 1, 2, 3, 7], 18)
nowindex :  7
inserting :  (15, [0, 1, 2, 3, 7, 5], 23)
inserting :  (19, [0, 1, 2, 3, 7, 6], 22)
nowindex :  5
inserting :  (15, [0, 1, 2, 3, 7, 5, 4, 6, 0], 18446744073709551639)
nowindex :  0
nowindex :  6
minlength Update :  [0, 1, 2, 3, 7, 6, 5, 4, 0]
inserting :  (19, [0, 1, 2, 3, 7, 6, 5, 4, 0], 33)
nowindex :  0
nowindex :  4
nowindex :  3
inserting :  (28, [0, 2, 3, 7], 17)
nowindex :  7
inserting :  (20, [0, 2, 3, 7, 5], 22)
inserting :  (24, [0, 2, 3, 7, 6], 21)
nowindex :  5
inserting :  (15, [0, 2, 3, 7, 5, 1], 28)
inserting :  (20, [0, 2, 3, 7, 5, 4], 24)
nowindex :  1
inserting :  (15, [0, 2, 3, 7, 5, 1, 4, 6, 0], 18446744073709551646)
nowindex :  0
nowindex :  4
nowindex :  6
inserting :  (11, [0, 2, 3, 7, 6, 5], 29)
nowindex :  5
inserting :  (11, [0, 2, 3, 7, 6, 5, 1, 4, 0], 40)
inserting :  (11, [0, 2, 3, 7, 6, 5, 4, 1, 0], 18446744073709551645)
nowindex :  0
nowindex :  0
heap is empty
result path :  [1, 3, 4, 8, 7, 6, 5, 2, 1]
minlength :  33
>>>
```