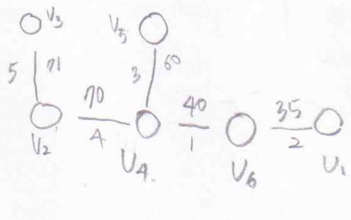
컴퓨터 알고리즘 (과제 #4)

학번 : 2016246[illegible]
이름 : 이재원
학과 : 정보컴퓨터공학부

4-3 Spanning tree (n개의 값은 n-1 개의 간선으로 연결)
MST · 최소 가중. 트리구성

	1	2	3	4	5	6
1	0	∞	72	50	90	35
2	∞	0	71	70	73	75
3	72	71	0	∞	77	90
4	50	70	∞	0	60	40
5	90	73	77	60	0	80
6	35	75	90	40	80	0

b)



c) cost.

40+35+60+70+71 = 276

∴ 276.

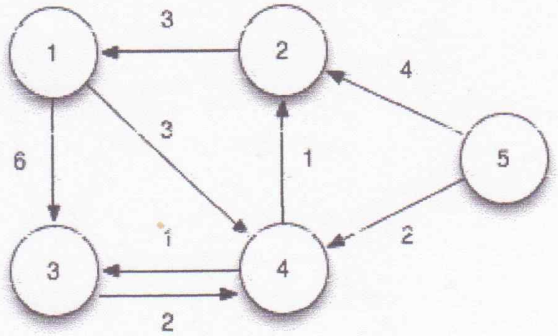
a)

	1	2	3	4	5	6
1	0	∞	∞	∞	∞	35
2	∞	0	71	70	∞	∞
3	∞	71	0	∞	∞	∞
4	∞	70	∞	0	60	40
5	∞	∞	∞	60	0	∞
6	35	∞	∞	40	∞	0

4-15 4.3 알고리즘을 수정. 최단 경로의 길이를 계산. 순서표기법을 사용하여. 결과를 표시.

알고리즘은 주석을 첨가하여 출력하였다.

다익스트라 알고리즘 (4-15)



이 그림을 기준으로 알고리즘을 작성하였다.

V_5 에서의 최단거리들을 구했다.

Python 3. Source 이다.

주석은 사용자의 알고리즘을 설명하였다.

```
import sys

#dijkstra algorithm 4.3을 약간 수정하였다.
def dijkstra(v_start, arr) :
    # 무한대를 지칭하는 INF 정의
    INF = sys.maxsize
    # 총 vertex의 개수 Algorithm 4.3에선 n과 대치
    length = len(arr)
    # 현재 vertex이다
    nvertex = v_start-1
    # 방문한 vertex 저장
    visit = [False] * length
    # 최단거리를 저장하는 배열. 출발점을 0으로 둔다
    Result = [INF] * length
    Result[nvertex] = 0
    visit[nvertex] = True


    print("초기 결과표 :", Result)
    print()
    for _ in range(length-1) :
        mini = INF
        # 현재 좌표에서 갈 수 있는 vertex에 대해 최단거리를 Result에 대입한다.
        for i in range(length) :
            Result[i] = min( Result[i], Result[nvertex] + arr[nvertex][i] )

        # 방문하지 않은 노드 중 다음 방문할 vertex와 minimum을 정한다.
        for i in range(length) :
            if visit[i] == False and Result[i] < mini :
                mini = Result[i];
                vnear = i
        # 방문 체크
        visit[vnear] = True

        # 최단거리 최신화
        for i in range(length) :
            Result[vnear] = min( Result[vnear], Result[nvertex] + mini )
        # 현재 vertex 업데이트
        nvertex = vnear
        print(_+1,"번째 선택된 vertex :", nvertex)
        print(_+1,"번째 진행한 결과표 :", Result)
        print()

    return Result

def main() :
    INF = sys.maxsize
    arr = [[INF,INF,6,3,INF], #
           [3,INF,INF,INF,INF], #
           [INF,INF,INF,2,INF], #
           [INF,1,1,INF,INF], #
           [INF,4,INF,2,INF], #
           ]

    print(dijkstra(5, arr))

    return

main()
```

출력은 이상 없이 나온 것 같다.

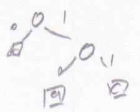
```
초기 결과표 : [2147483647, 2147483647, 2147483647, 2147483647, 0]

1 번째 선택된 vertex : 3 +1 =4
1 번째 진행한 결과표 : [2147483647, 4, 2147483647, 2, 0]

2 번째 선택된 vertex : 1 +1 =2
2 번째 진행한 결과표 : [2147483647, 3, 3, 2, 0]

3 번째 선택된 vertex : 2 +1 =3
3 번째 진행한 결과표 : [6, 3, 3, 2, 0]

4 번째 선택된 vertex : 0 +1 =1
4 번째 진행한 결과표 : [6, 3, 3, 2, 0]

[6, 3, 3, 2, 0]
```

huffman code

27. 허프만 알고리즘 사용. 다음의 문자에 대한 최적의 이진접두사 코드를 찾아

optimal binary code: 주어진 문자들을 이진코드로 표현했을때 비트의 개수가 최소.
prefix code: 한 문자의 코드가 다른 문자의 코드워드의 앞부분이 될 수 없다.

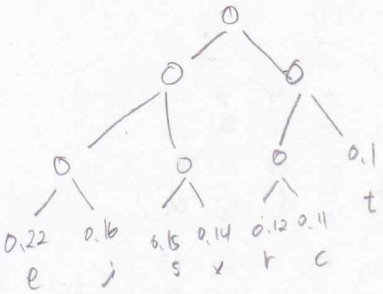
ex)



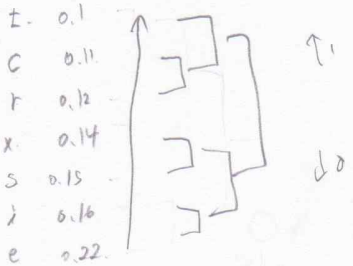
c e i r s t x

0.11 0.22 0.16 0.12 0.15 0.10 0.14 → 1

□: code.



t 0.1
c 0.11
r 0.12
x 0.14
s 0.15
i 0.16
e 0.22



28 다음 주어진 코드들은 터무니 하다.

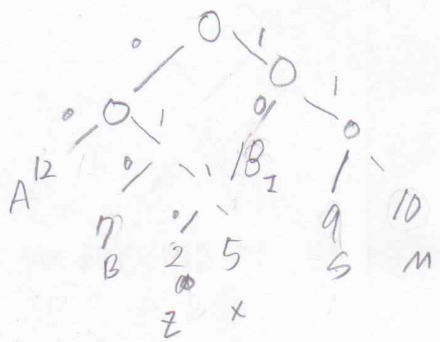
binary code Exercise 24. ? 코드가 안보임

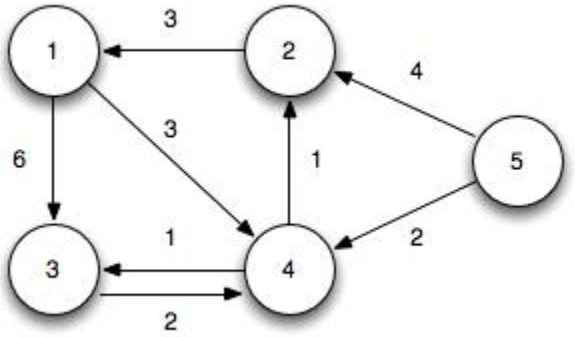
a) 0110 00 010 10 10 10
Z A B I I I

b) 10 00 10 00 010 10
I A I A B I

c) 111 00 10 0111 101
e A x ?

d) 10 00 010 0111 00
I A B Y A






```
import sys

#dijkstra algorithm 4.3을 약간 수정하였다.
def dijkstra(v_start, arr) :
    # 무한대를 지칭하는 INF 정의
    INF = sys.maxsize
    # 총 vertex의 개수 Algorithm 4.3에선 n과 대치
    length = len(arr)
    # 현재 vertex이다
    nvertex = v_start-1
    # 방문한 vertex 저장
    visit = [False] * length
    # 최단거리를 저장하는 배열, 출발점을 0으로 둔다
    Result = [INF] * length
    Result[nvertex] = 0
    visit[nvertex] = True


    print("초기 결과표 :", Result)
    print()
    for _ in range(length-1) :
        mini = INF
        # 현재 좌표에서 갈 수 있는 vertex에 대해 최단거리를 Result에 대입한다.
        for i in range(length) :
            Result[i] = min( Result[i], Result[nvertex] + arr[nvertex][i] )

        # 방문하지 않은 노드 중 다음 방문할 vertex와 minimum을 정한다.
        for i in range(length) :
            if visit[i] == False and Result[i] < mini :
                mini = Result[i];
                vnear = i
        # 방문 체크
        visit[vnear] = True

        # 최단거리 최신화
        for i in range(length) :
            Result[vnear] = min( Result[vnear], Result[nvertex] + mini )
        # 현재 vertex 업데이트
        nvertex = vnear
        print(_+1,"번째 선택된 vertex :", nvertex)
        print(_+1,"번째 진행한 결과표 :", Result)
        print()

    return Result

def main() :
    INF = sys.maxsize
    arr = [[INF,INF,6,3,INF], #
           [3,INF,INF,INF,INF], #
           [INF,INF,INF,2,INF], #
           [INF,1,1,INF,INF], #
           [INF,4,INF,2,INF], #
          ]

    print(dijkstra(5, arr))

    return

main()
```

```
초기 결과표 : [2147483647, 2147483647, 2147483647, 2147483647, 0]

1 번째 선택된 vertex : 3
1 번째 진행한 결과표 : [2147483647, 4, 2147483647, 2, 0]

2 번째 선택된 vertex : 1
2 번째 진행한 결과표 : [2147483647, 3, 3, 2, 0]

3 번째 선택된 vertex : 2
3 번째 진행한 결과표 : [6, 3, 3, 2, 0]

4 번째 선택된 vertex : 0
4 번째 진행한 결과표 : [6, 3, 3, 2, 0]

[6, 3, 3, 2, 0]
```

ㄴㄴ