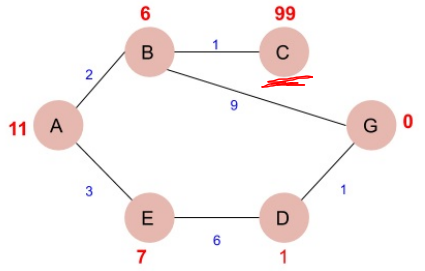
```python
def a_star(node_start ,node_goal):
    open_list = [node_start]
    closed_list = []
    g = {}
    parent = {}
    g[node_start] = 0
    parent[node_start] = node_start


    while len(open_list) > 0:
        n = None

        for i in open_list:
            if n == None or g[i] + h(i) < g[n] + h(n):
                n = i

        if n == node_goal or Graph[n] == None:
            pass
        else:
            for (m, weight) in neighbours(n):
                if m not in open_list and m not in closed_list:
                    open_list.append(m)
                    parent[m] = n
                    g[m] = g[n] + weight

                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parent[m] = n
                        if m in closed_list:
                            closed_list.remove(m)
                            open_list.append(m)
```

```python
        if n == None:
            print("Path does not exist")
            return None
        if n == node_goal:
            path = []

            while parent[n] != n:
                path.append(n)
                n = parent[n]
            path.append(node_start)

            path.reverse()
            print(path)
            return path
        open_list.remove(n)
        closed_list.append(n)
    print("Path does not exist")
    return None

def neighbours(i):
    if i in Graph:
        return Graph[i]
    else:
        return None

def h(n):
    h_dist = {
        'A' : 11,
        'B' : 6,
        'C' : 99,
        'D' : 1,
        'E' : 7,
        'G' : 0,

    }
    return h_dist[n]

Graph = {
    'A' : [('B', 2 ), ('E', 3)],
    'B' : [('C', 1 ), ('G', 9)],
    'C' : None,
    'E' : [('D', 6)],
    'D' : [('G', 1 )],

}
```



$G = \{A:0, B:2, E:3\}$

$C = [A, B \quad]$

$O = [B, E, C, G \quad]$

$P = \{A:A, B:A, E:A, G:B\}$

① $n = A$

$m = B$

$m = E$

remove A

② $len = 2$

$g[B] + h[B] = 8$ $\quad g[E] + h[E] = 10$

$n = B$

$m = C$

$m = G$

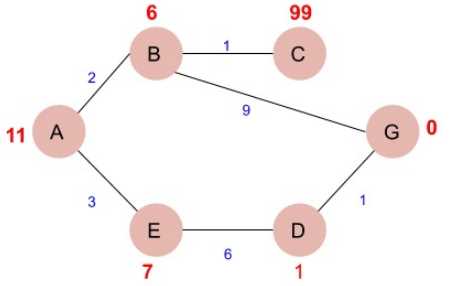
remove B

```python
def a_star(node_start ,node_goal):
    open_list = [node_start]
    closed_list = []
    g = {}
    parent = {}
    g[node_start] = 0
    parent[node_start] = node_start


    while len(open_list) > 0:
        n = None

        for i in open_list:
            if n == None or g[i] + h(i) < g[n] + h(n):
                n = i

        if n == node_goal or Graph[n] == None:
            pass
        else:
            for (m, weight) in neighbours(n):
                if m not in open_list and m not in closed_list:
                    open_list.append(m)
                    parent[m] = n
                    g[m] = g[n] + weight

                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parent[m] = n
                        if m in closed_list:
                            closed_list.remove(m)
                            open_list.append(m)

        if n == None:
            print("Path does not exist")
            return None
        if n == node_goal:
            path = []

            while parent[n] != n:
                path.append(n)
                n = parent[n]
            path.append(node_start)

            path.reverse()
            print(path)
            return path
        open_list.remove(n)
        closed_list.append(n)
    print("Path does not exist")
    return None

def neighbours(i):
    if i in Graph:
        return Graph[i]
    else:
        return None

def h(n):
    h_dist = {
        'A' : 11,
        'B' : 6,
        'C' : 99,
        'D' : 1,
        'E' : 7,
        'G' : 0,

    }
    return h_dist[n]

Graph = {
    'A' : [('B', 2 ), ('E', 3)],
    'B' : [('C', 1 ), ('G', 9)],
    'C' : None,
    'E' : [('D', 6)],
    'D' : [('G', 1 )],

}
```

O = [B, E, C, G, D]

G = {A:0, B:2, E:3, G:100, D:9}

C = [A, B, E, D]



P = {A:A , B:A , E:A , G:D , C:B , D:E}

n = none

(1) i → E

n → E

g[c] + h[c] > g[e] + h[e]

g[g] + h[g]

11 + 0 < 3 + 7

n = E

(2) i → D

n → D

n = G

g[g] > 9 + 1

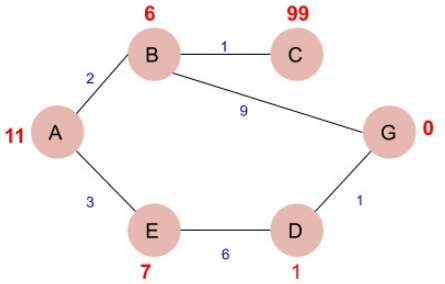
||

g[g] = 10

```python
def a_star(node_start ,node_goal):
    open_list = [node_start]
    closed_list = []
    g = {}
    parent = {}
    g[node_start] = 0
    parent[node_start] = node_start


    while len(open_list) > 0:
        n = None

        for i in open_list:
            if n == None or g[i] + h(i) < g[n] + h(n):
                n = i

        if n == node_goal or Graph[n] == None:
            pass
        else:
            for (m, weight) in neighbours(n):
                if m not in open_list and m not in closed_list:
                    open_list.append(m)
                    parent[m] = n
                    g[m] = g[n] + weight

                else:
                    if g[m] > g[n] + weight:
                        g[m] = g[n] + weight
                        parent[m] = n
                        if m in closed_list:
                            closed_list.remove(m)
                            open_list.append(m)
```

```python
        if n == None:
            print("Path does not exist")
            return None
        if n == node_goal:
            path = []

            while parent[n] != n:
                path.append(n)
                n = parent[n]
            path.append(node_start)

            path.reverse()
            print(path)
            return path
        open_list.remove(n)
        closed_list.append(n)
    print("Path does not exist")
    return None

def neighbours(i):
    if i in Graph:
        return Graph[i]
    else:
        return None

def h(n):
    h_dist = {
        'A' : 11,
        'B' : 6,
        'C' : 99,
        'D' : 1,
        'E' : 7,
        'G' : 0,

    }
    return h_dist[n]

Graph = {
    'A' : [('B', 2 ), ('E', 3)],
    'B' : [('C', 1 ), ('G', 9)],
    'C' : None,
    'E' : [('D', 6)],
    'D' : [('G', 1 )],

}
```

O = [C, G,]



G = {A:0, B:2, E:3, G:10, D:9}

C = [A, B, E, D, G]

P = {A:A, B:A, E:A, G:D, C:B, D:E}

n = none

(1) i → C

n = C

i → g

n = g

path = [G → D → E → A]

⇒ A E D G