안녕하세요, 하계 대학생 S/W 알고리즘 특강의 열두 번째 시간인 오늘은 트라이에 대해 다루어보도록 하겠습니다.

**1. 기초 강의**  
동영상 강의 컨텐츠 확인 > 12. 다익스트라  
Link : <https://swexpertacademy.com/main/learn/course/subjectDetail.do?courseId=CONTENTS_REVIEW&subjectId=AYVXyY7aRHYDFARs>  
**※ 출석은 강의 수강 내역으로 확인합니다**.

**2. 실전 강의**  
**2.1. 다익스트라 알고리즘**

다익스트라 알고리즘은 대표적인 최단 경로 탐색 알고리즘입니다.   
특정한 하나의 정점에서 다른 모든 정점으로 가는 최단 경로를 알려줍니다. 간선마다 0이상의 cost를 가질 수 있으며 연결되지 않는 간선의 가중치는 Inf로 표현합니다.

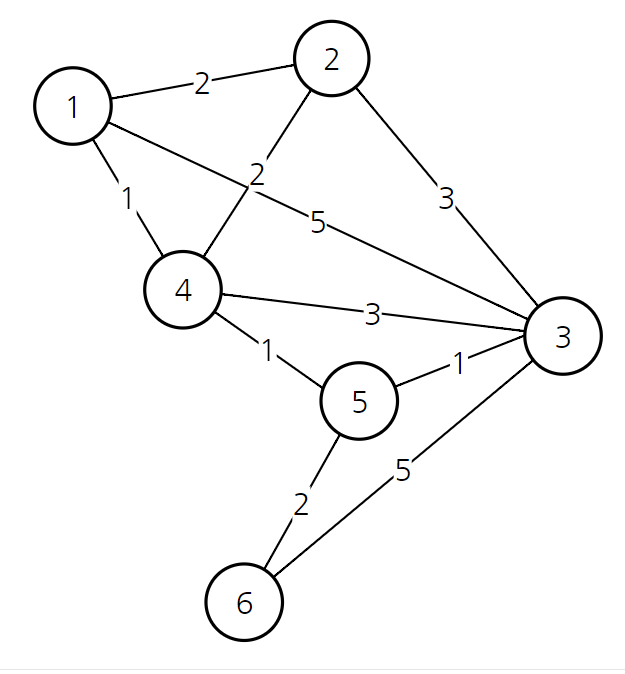
최단 거리는 여러 개의 최단 거리로 이루어져 있기 때문에 하나의 최단 거리를 구할 때 그 이전까지 구했던 최단 거리 정보를 그대로 사용하여 구한다.

**2.2. 동작 단계**

① 출발 노드를 설정한다.  
② '최단 거리 테이블'(dist)을 초기화한다.  
③ 현재 위치한 노드의 인접 노드 중 방문하지 않은 노드를 구별하고, 방문하지 않은 노드 중 거리가 가장 짧은 노드를 선택한다. 그 노드를 방문 처리한다.  
④ 해당 노드를 거쳐 다른 노드로 넘어가는 간선 비용(가중치)을 계산해 '최단 거리 테이블'을 업데이트한다.  
⑤ ③~④의 과정을 반복한다.

'최단 거리 테이블'은 1차원 배열로, start에서 N개 노드까지 가는 데 필요한 최단 거리를 기록한다. N개 크기의 배열을 선언하고 큰 값(INF)을 넣어 초기화시킨다.

'노드 방문 여부 체크 배열'(visted)은 방문한 노드인지 아닌지 기록하기 위한 배열로, 크기는 '최단 거리 테이블'과 같다. 기본적으로는 0(False)으로 초기화하여 방문하지 않았음을 명시한다.

****

**2.4. 구현**

**1) Sequential Search**

#include <stdio.h>

#define MAX\_N 6

#define INF (987654321)

int graph[MAX\_N][MAX\_N] = {

    {0,2,5,1,INF,INF},

    {2,0,3,2,INF,INF},

    {5,3,0,3,1,5},

    {1,2,3,0,1,INF},

    {INF,INF,1,1,0,2},

    {INF,INF,5,INF,2,0}

};

int getMinIdx(int nodes[MAX\_N],int visited[MAX\_N]){

    int min = -1;

    for(int i=0;i< MAX\_N;i++){

        if(visited[i]) continue;

        if(min<0 || nodes[min] > nodes[i]) min = i;

    }

    return min;

}

void dijkstra2(int arr[MAX\_N][MAX\_N], int start, int dist[MAX\_N]){

    int visited[MAX\_N] = {0,};

    for(int i=0;i<MAX\_N;i++){

        dist[i] = arr[start][i];

    }

    visited[start] = 1;

    for(int i=0;i<MAX\_N-1;i++){

        int n\_new = getMinIdx(dist,visited);

        visited[n\_new] = 1;

        for(int j=0;j<MAX\_N;j++){

            if(visited[j]) continue;

            if(dist[j] > dist[n\_new] + arr[n\_new][j])

                dist[j] = dist[n\_new] + arr[n\_new][j];

        }

    }

}

int main()

{

    int dist[MAX\_N];

    int start =0;

    dijkstra2(graph,start,dist);

    for(int i=0;i<MAX\_N;i++){

        printf("%d->%d : %d\n", start, i, dist[i]);

    }

    printf("\n");

    return 0;

}

**2) Priority Queue Search**

void dijkstra(int arr[MAX\_N][MAX\_N], int start, int dist[MAX\_N]){

    priority\_queue<pair<int,int>> pq;

    for(int i=0;i<MAX\_N;i++){

        dist[i] = INF;

    }

    pq.push({0,start}); // {dist, destination}

    while(!pq.empty()){

        int cur\_dist = -pq.top().first;

        int cur\_node = pq.top().second;

        pq.pop();

        for(int i=0;i<MAX\_N;i++){

            int nxt\_dist = cur\_dist +  arr[cur\_node][i];

            if(nxt\_dist < dist[i])

            {

                dist[i] = nxt\_dist;

                pq.push({-nxt\_dist,i});

            }

        }

    }

}

**3. 기본 문제**  
    ·  [H2255] 물류허브