

6.

#include <stdio.h>

void prim(int n, // 입력: 정점의 수

const number W[][], // 입력: 그래프의 인접행렬식 표현

set\_of\_edges& F) // 출력: 그래프의 MST에 속한 이음선의 집합

{

index i, vnear;

number min;

edge e;

index nearest[2..n];

number distance[2..n];

F = Ø;

for (i = 2; i <= n; i++) { // 초기화

nearest[i] = 1; // vi에서 가장 가까운 정점을

v1으로 초기화

distance[i] = W[1][i]; // vi과 v1을 잇는 이음선의

//가중치로 초기화

repeat(n - 1 times) { // n-1개의 정점을 Y에 추가한다

min = ;

for (i = 2; i <= n; i++) // 각 정점에 대해서

if (0 <= distance[i] <= min) { // distance[i]를 검사하여

min = distance[i]; // 가장 가까이 있는 vnear을

vnear = i; // 찾는다.

}

e = vnear와 nearest[vnear]를 잇는 이음선;

e를 F에 추가;

distance[vnear] = -1; // 찾은 노드를 Y에 추가한다.

for (i = 2; i <= n; i++) {

if (W[i][vnear] < distance[i]) {// Y에 없는 각 노드에 대해서

distance[i] = W[i][vnear]; // distance[i]를 갱신한다.

nearest[i] = vnear;

}

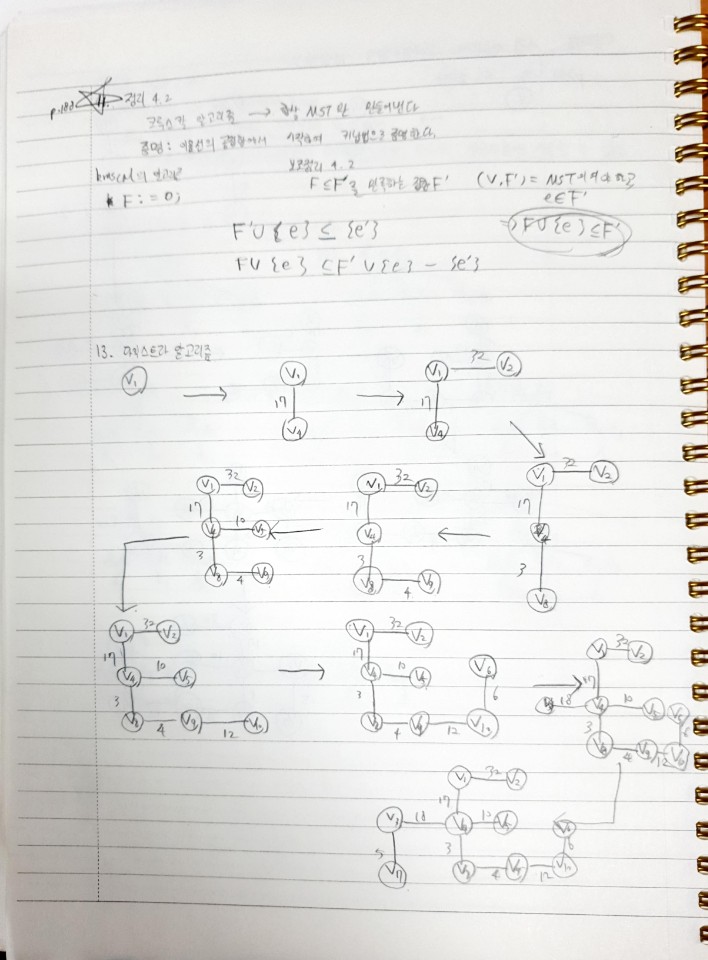
}

}

}

}

차수표기법: T(n) = 2(n-1)(n-1) 🡺 O(n의 제곱)



20.

void *schedule*(int *n*,

const int *deadline*[],

sequence\_of\_integer& *J*) {

index *i*;

sequence\_of integer *K*;

*J*= [1];

for (*i*=2; i<=*n*; *i*++) {

*K= J*에다 deadline[i]의 값이 작은 것부터 i를 추가

if (*K*가 적절하다)

*J = K*;

}

}

24.

void *schedule*(int *n*,

const int *deadline*[],

sequence\_of\_integer& *J*) {

index *i*;

sequence\_of integer *K1, K2*;

*J*= [1];

Index temp

for (*i*=2; i<=*n*; *i*++) {

*K= J*에다 deadline[i]의 값이 작은 것부터 i를 추가

if (*K*가 적절하다 & temp가 -1이다)

*J = K*;1

Temp = 1

Else if(K가 적절하다 & temp가 1이다)

J = K2

Temp = -1

}

}