

[컴퓨터공학설계 및 실험 I]
2023학년도 1학기 - 최종 프로젝트

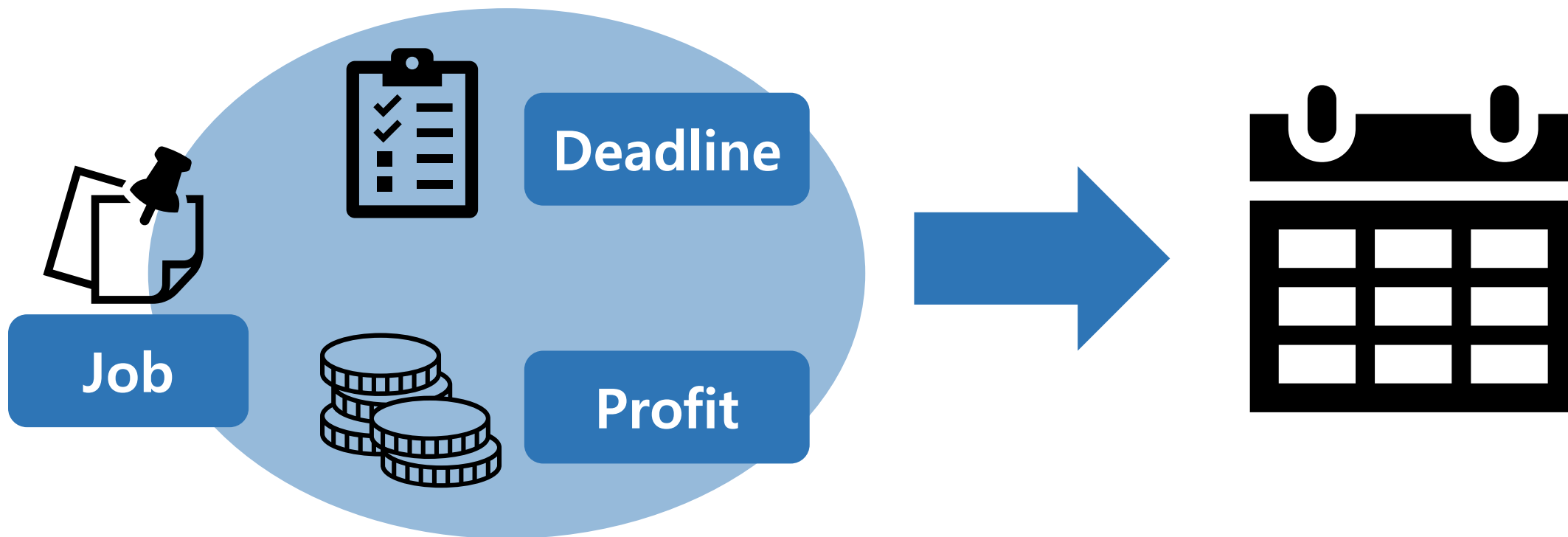
20190785 박수빈
01분반

목차

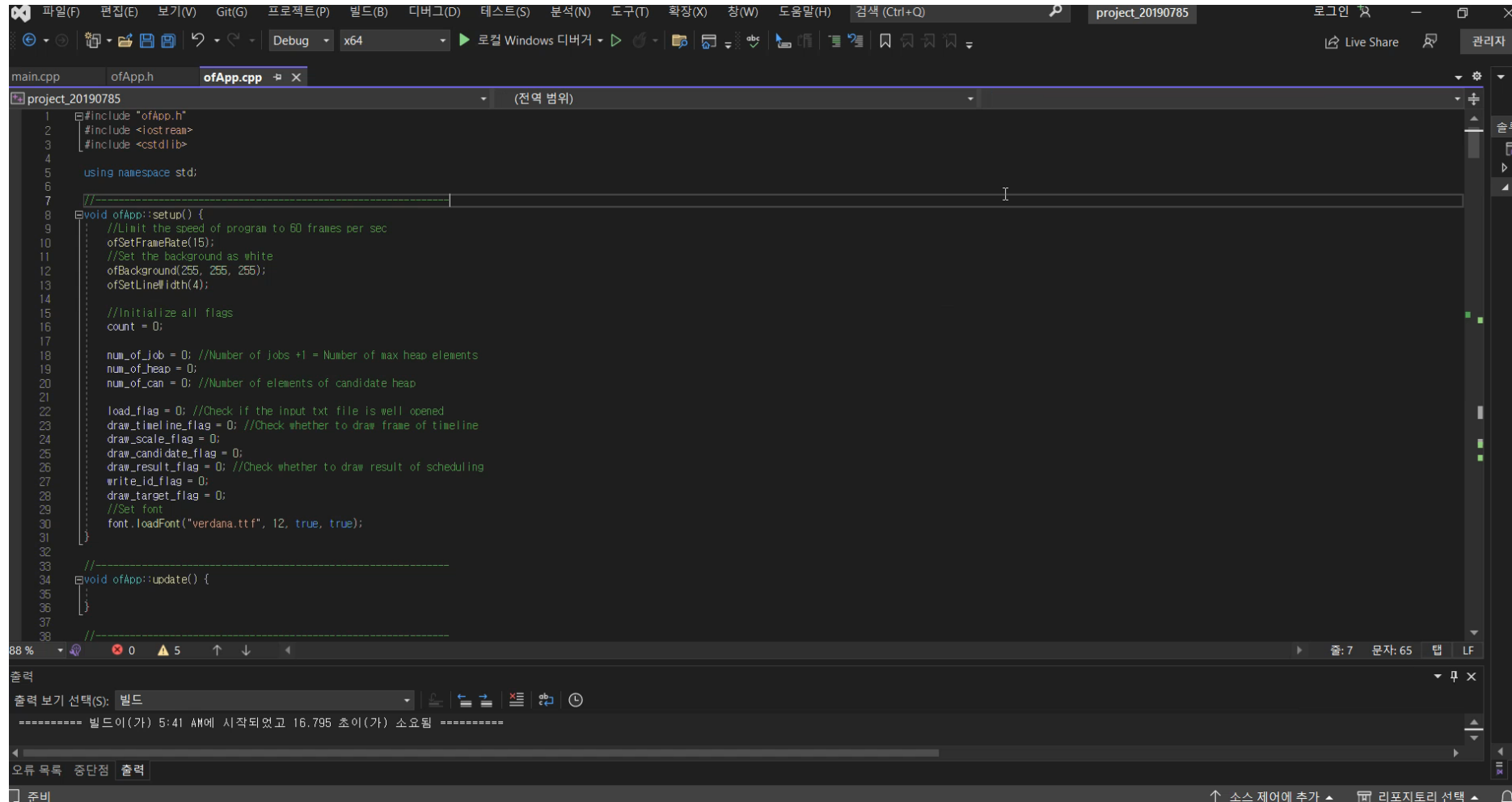
1. 프로젝트 목표
2. 프로그램 시연
3. 핵심 변수 및 자료구조
4. 알고리즘

1. 프로젝트 목표

- disjoint set 알고리즘을 이용하여 여러 개의 '작업(job)'들을 각 작업의 '마감일(deadline)'을 고려하여 최대의 '이윤(profit)'을 창출하도록 '스케줄링(scheduling)'
- 스케줄링 결과를 확인하기 쉽도록 화면에 시각화



2. 프로그램 시연



The screenshot shows a C++ IDE with the file `ofApp.cpp` open. The code is for a program named `project_20190785`. It includes `ofApp.h`, `<iostream>`, and `<cstdlib>`. The code uses the `std` namespace and defines two functions: `void ofApp::setup()` and `void ofApp::update()`. The `setup` function initializes various flags and variables, including `num_of_job`, `num_of_head`, `num_of_can`, `load_flag`, `draw_timeline_flag`, `draw_scale_flag`, `draw_candidate_flag`, `draw_result_flag`, `write_id_flag`, `draw_target_flag`, and `font`. The `update` function is currently empty. The IDE interface includes a menu bar, a toolbar, a file explorer, and a console window at the bottom.

```
1 #include "ofApp.h"
2 #include <iostream>
3 #include <cstdlib>
4
5 using namespace std;
6
7 //-----
8 void ofApp::setup() {
9     //Limit the speed of program to 60 frames per sec
10    ofSetFrameRate(15);
11    //Set the background as white
12    ofBackground(255, 255, 255);
13    ofSetLineWidth(4);
14
15    //Initialize all flags
16    count = 0;
17
18    num_of_job = 0; //Number of jobs +1 = Number of max heap elements
19    num_of_head = 0;
20    num_of_can = 0; //Number of elements of candidate heap
21
22    load_flag = 0; //Check if the input txt file is well opened
23    draw_timeline_flag = 0; //Check whether to draw frame of timeline
24    draw_scale_flag = 0;
25    draw_candidate_flag = 0;
26    draw_result_flag = 0; //Check whether to draw result of scheduling
27    write_id_flag = 0;
28    draw_target_flag = 0;
29    //Set font
30    font.loadFont("verdana.ttf", 12, true, true);
31 }
32
33 //-----
34 void ofApp::update() {
35 }
36
37
38 //-----
```

88 % 0 5 7 65 탭 LF

출력

출력 보기 선택(S): 빌드

===== 빌드가(가) 5:41 AM에 시작되었고 16.795 초이(가) 소요됨 =====

오류 목록 중단점 출력

준비

↑ 소스 제어에 추가 리포지토리 선택

3. 핵심 변수 및 자료구조

- 1) Job
- 2) Job* job_heap
- 3) char schedule[21]
- 4) Node
- 5) Node* disjoint_set
- 6) Job* candidate

ofApp.h 파일

```

/*Class Job is needed to store information of each job
which is the element of max heap*/
class Job {
public:
    char id; //ID of the job
    int dead; //Deadline of job
    int profit; //Profit of job
};
    
```

Input : ".txt" file

<Job Information>		
ID	deadline	profit
a	6	82
b	9	20
c	2	95
⋮	⋮	⋮

read file
line by line

1. Make a node of each job

ID
deadline
profit

↳ 'Job' class

a	b	...
6	9	
82	20	

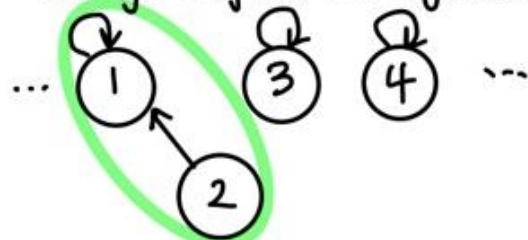
2. Insert it into max heap

get the first node
of max heap

[0]	[1]	[2]	[3]	
⋮	c	a	b	...
	2	6	9	
	95	82	20	

Max heap : sort jobs in non-ascending
order by profit

3. Check if there is a place to be
inserted in the 'schedule'
through disjoint set algorithm



[0]	[1]	[2]	[3]	[4]	
⋮					...

4. Put the node in slot of 'schedule'
as late as possible

5. Repeat 2~4 process
until number of elements
of max heap becomes 0.

[0]	[1]	[2]	[3]	[4]	
⋮		c			...

ofApp.h 파일

```
Job* job_heap;
```

ofApp.cpp

```
job_heap = (Job*)malloc(sizeof(Job) * (num_of_job + 1));
```

Input : ".txt" file

<Job Information>		
ID	deadline	profit
a	6	82
b	9	20
c	2	95
⋮	⋮	⋮

read file
line by line

1. Make a node of each job

ID
deadline
profit

↳ 'Job' class

a
6
82

b
9
20

...

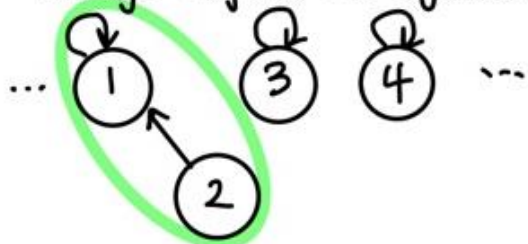
2. Insert it into max heap

get the first node
of max heap

[0]	[1]	[2]	[3]	
⋮	c	a	b	...
⋮	2	6	9	...
⋮	95	82	20	...

Max heap : sort jobs in non-ascending
order by profit

3. Check if there is a place to be
inserted in the 'schedule'
through disjoint set algorithm



[0]	[1]	[2]	[3]	[4]	
⋮					...

'schedule'

4. Put the node in slot of 'schedule'
as late as possible

5. Repeat 2~4 process
until number of elements
of max heap becomes 0.

[0]	[1]	[2]	[3]	[4]	
⋮		c			...

'schedule'

ofApp.h 파일

```
class Node {
public:
    int parent; //Slot number of parent node
    int rank; //Height of the node
};
```

Node* disjoint_set;

ofApp.cpp

```
Node* ofApp::initDisjoint(Node* disjoint_set, int n) {
    /**Purpose of the function: Initialize all elements of disjoint set***/

    disjoint_set = (Node*)malloc(sizeof(Node) * n);
    for (int i = 0; i < n; i++) {
        disjoint_set[i].parent = i;
        disjoint_set[i].rank = 0;
    }
    return disjoint_set;
}
```

Input : ".txt" file

<Job Information>		
ID	deadline	profit
a	6	82
b	9	20
c	2	95
⋮	⋮	⋮

read file
line by line

1. Make a node of each job

ID
deadline
profit

↳ 'Job' class

a
6
82

b
9
20

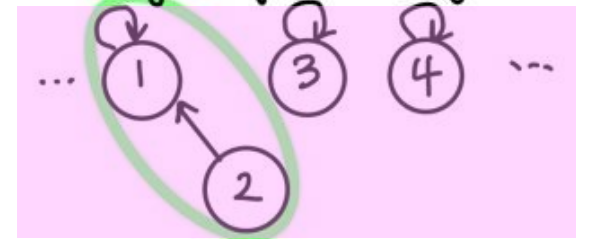
...

2. Insert it into max heap

get the first node
of max heap

[0]	[1]	[2]	[3]	
⧘	c	a	b	...
	2	6	9	
	95	82	20	

3. Check if there is a place to be
inserted in the 'schedule'
through disjoint set algorithm



Max heap : sort jobs in non-ascending
order by profit

[0]

[1]

[2]

dummy
node

parent,
rank

parent,
rank

the
late

'schedule'

ofApp.h 파일

```
char schedule[21];
```

Input : ".txt" file

<Job Information>		
ID	deadline	profit
a	6	82
b	9	20
c	2	95
⋮	⋮	⋮

read file
line by line

1. Make a node of each job

ID
deadline
profit

↳ 'Job' class

a
6
82

b
9
20

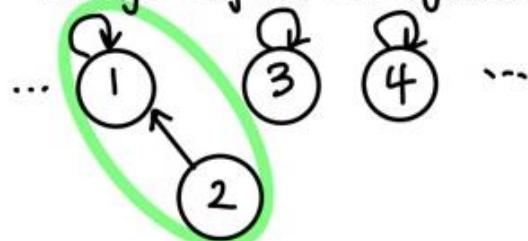
...

2. Insert it into max heap

[0]	[1]	[2]	[3]	
⋮	c	a	b	...
	2	6	9	
	95	82	20	

get the first node
of max heap

3. Check if there is a place to be
inserted in the 'schedule'
through disjoint set algorithm



Max heap : sort jobs in non-ascending
order by profit

[0]	[1]	[2]	[3]	[4]	
⋮					...

'schedule'

4. Put the node in slot of 'schedule'
as late as possible

[0]	[1]	[2]	[3]	[4]	
⋮		c			...

'schedule'

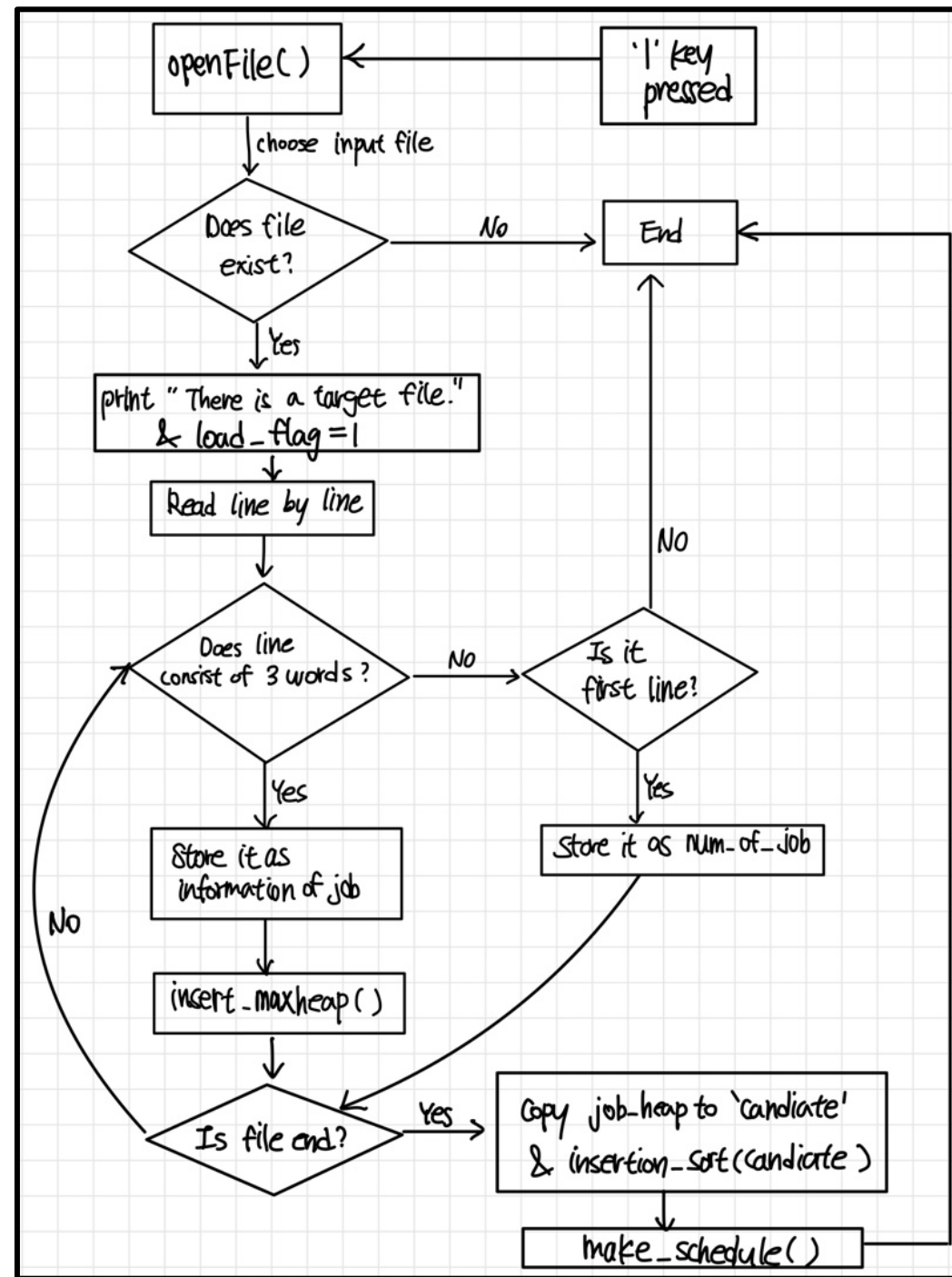
5. Repeat 2~4 process
until number of elements
of max heap becomes 0.

Key operation	Flag이름	1	0
'l' key	load_flag	입력 파일이 정상적으로 로드됨	정상적으로 로드되지 않음
't' key	draw_timeline_flag	화면에 timeline 주눈금을 그림	그리지 않음
's' key	draw_scale_flag	화면에 timeline의 수직선을 보조선으로 그림	그리지 않음
'c' key	draw_candidate_flag	화면에 입력 파일로부터 입력 받은 모든 job을 스케줄링의 후보로 그림	그리지 않음
'r' key	draw_result_flag	화면에 스케줄링 결과를 timeline 바로 위에 사각형으로 그림	그리지 않음
	write_id_flag	스케줄링 결과에서 각 job의 ID를 나타냄	나타내지 않음
'<-' or '->' key	draw_target_flag	스케줄링 결과에서 좌우 방향으로 targeting한 job을 연한 하늘색 사각형으로 그림	그리지 않음

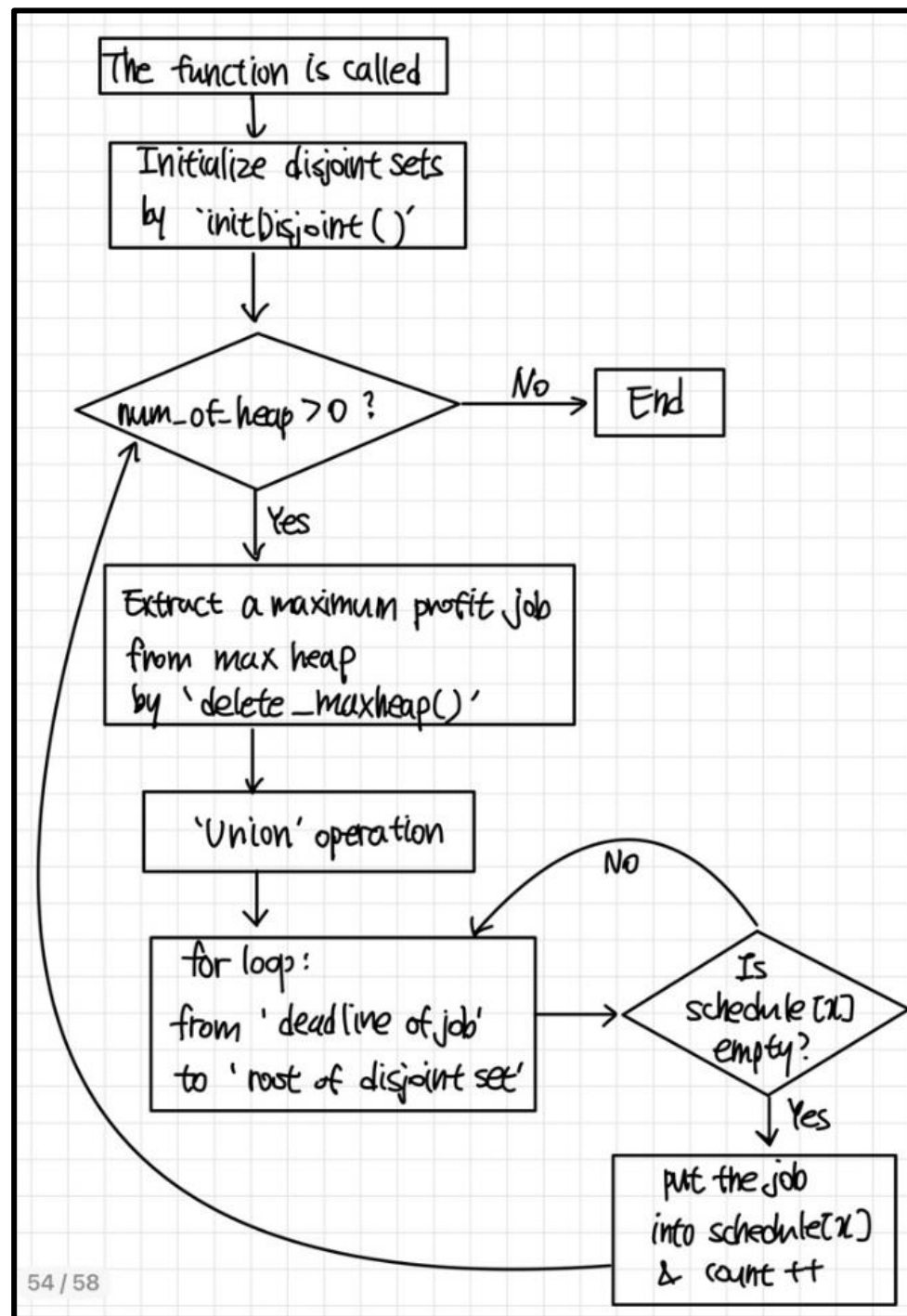
4. 알고리즘

- 1) `openFile()`
- 2) `make_schedule()`
- 3) 프로그램 전체

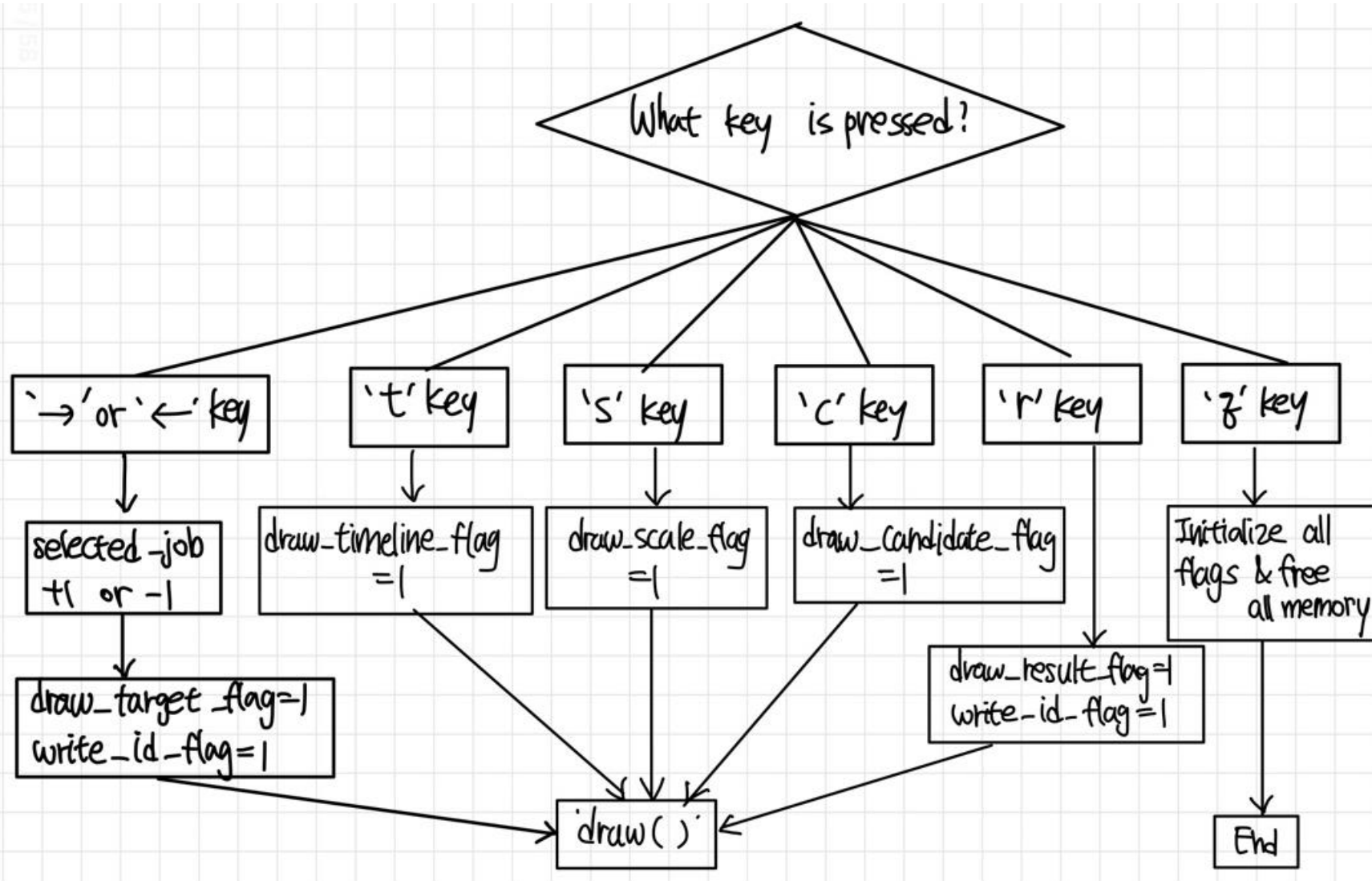
openFile()



make_schedule()



전체 프로그램



- The End-

감사합니다.