이동규제 R 미니 프로젝트

BAEKJOON사이트를 활용한 효율적인 공부방법 제시

멀티캠퍼스 빅데이터 분석서비스개발 이동규
멀티캠퍼스 빅데이터 분석서비스개발 이지혜

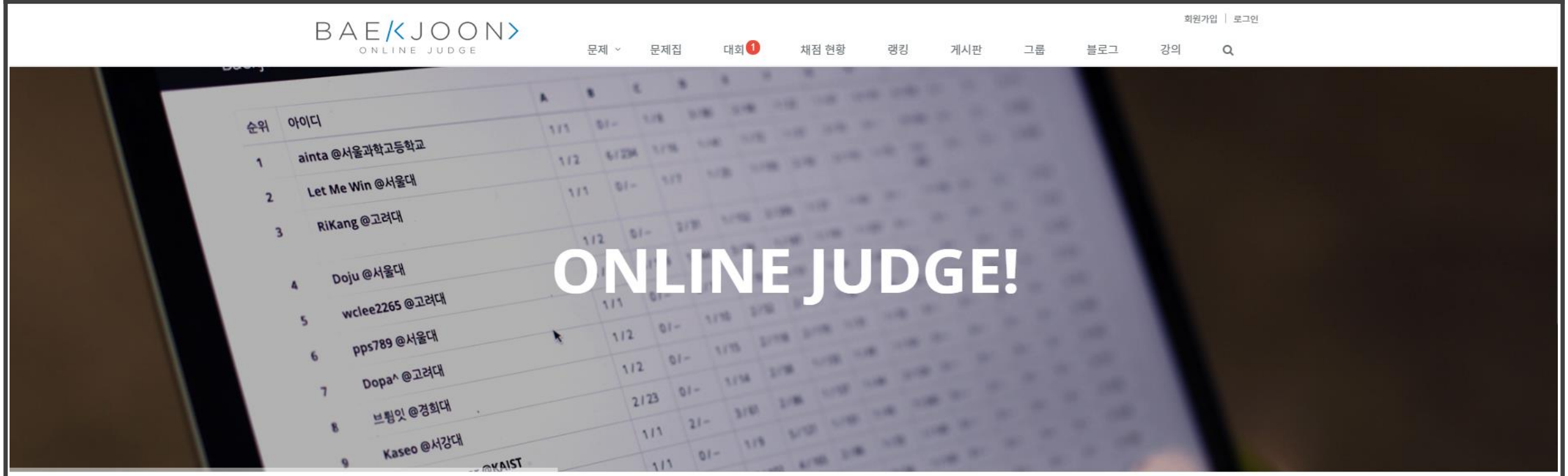
이동규
데이터 수집
데이터 전처리
Selenium을 활용한 웹 크롤링

이지혜
데이터 분석
데이터시각화
ggplot2를 활용한 그래프 작성

분석 필요성

알고리즘을 처음 공부할 때 공부 순서를 정하기 어려웠습니다.
이때 정답율이 높은 문제 순으로 공부한다면
접근이 용이할 것이라고 생각했습니다.
따라서 알고리즘을 공부하는 학생 입장에서
정답율을 분석해 각 알고리즘의 난이도를 가늠해보고 보고,
이에 따라 공부 순서와 보완해야 할 부분을 계획하고자 합니다.
우리의 분석 결과를 보고, 알고리즘을 처음 공부하는 학생들이
공부 계획을 세우는 데에 참고할 수 있을 것입니다.

BAEKJOON Homepage



온라인으로 제시된 문제를 풀어보고
공부하면서 코딩 능력을 확인하고
실력향상 할 수 있는 사이트

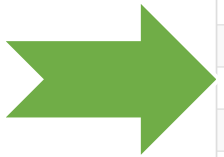
가설 1

알고리즘 별로 정답 비율을 분석했을 때
정답률이 높은 문제가 많을수록 쉬운 알고리즘이고,
정답률이 낮은 문제가 많을수록 어려운 알고리즘이다.

크롤링 과정_1

```
library(RSelenium)
remDr <- remoteDriver(remoteServerAddr = "localhost" ,
                      port = 4445, browserName = "chrome")
remDr$open()
site <- 'https://www.acmicpc.net/problem/tags'
remDr$navigate(site)
```

알고리즘별 분류 탭으로 이동



BAEKJOON ONLINE JUDGE

문제 | 출처 | 단계 | 분류 | 추가된 문제 | 추가된 영어 문제 | 문제 순위 | 더 보기

검색

태그		문제
수학	Mathematics	1513
다이나믹 프로그래밍	Dynamic programming	1278
구현	Implementation	1138
그래프 이론	Graph theory	1115
자료 구조	Data structures	923
그래프 탐색	Graph traversal	550
그리디 알고리즘	Greedy	515
문자열	String	469
세그먼트 트리	Segment tree	431
브루트포스 알고리즘	Bruteforcing	426
이분 탐색	Binary search	374
기하학	Geometry	369
트리	Tree	369
정렬	Sorting	362
정수론	Number theory	341

크롤링 과정_2

```
for (n in 1:30) {
  pageLink <- NULL
  algo_title <- NULL
  problem_num <- NULL
  answer_percent <- NULL

  #태그로 이동_문제수 100문제이상
  Sys.sleep(5)
  pageLink <- remDr$findElements(using='xpath',
                                 value= paste0('/html/body/div[3]/div[2]/div[5]/div/div/table/tbody/tr[', n, ']/td[1]/a'))

  #알고리즘 태그명
  algo_titles <- sapply(pageLink, function(x) {x$getElementText()})
  print(algo_titles)
  algo_title <- append(algo_title, unlist(algo_titles))

  #태그별 문제수
  algo_node <- remDr$findElements(using='xpath',
                                  value= paste0('/html/body/div[3]/div[2]/div[5]/div/div/table/tbody/tr[', n, ']/td[3]'))

  problem_nums <- sapply(algo_node, function(x) {x$getElementText()})
  print(problem_nums)
  problem_num <- append(problem_num, unlist(problem_nums))

  #태그 클릭
  remDr$executeScript("arguments[0].click();",pageLink)

  Sys.sleep(3)
  pageLink_next <- NULL
  curr_PageOldNum <- 0
```

알고리즘별 태그명과 총 문제 수를 각각 변수에 저장

저장 후 태그명 클릭

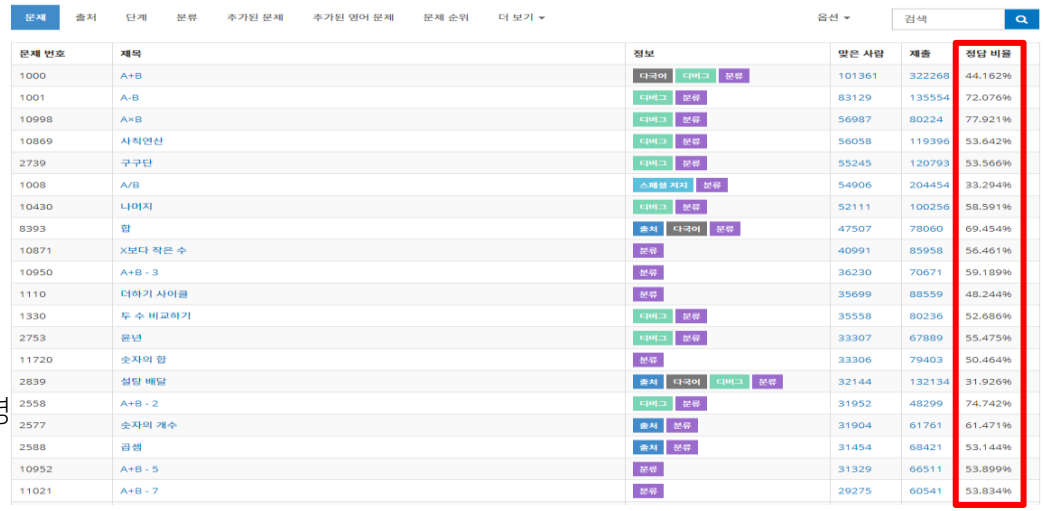
태그		문제
수학	Mathematics	1513
다이나믹 프로그래밍	Dynamic programming	1278
구현	Implementation	1138
그래프 이론	Graph theory	1115
자료 구조	Data structures	923
그래프 탐색	Graph traversal	550
그리디 알고리즘	Greedy	515
문자열	String	469
세그먼트 트리	Segment tree	431
브루트포스 알고리즘	Bruteforcing	426
이분 탐색	Binary search	374
기하학	Geometry	369
트리	Tree	369
정렬	Sorting	362
정수론	Number theory	341

크롤링 과정_3

```
repeat{
  #정답 비율
  problem_nodes <- remDr$findElements(using='xpath',
                                      value= paste0('//*[@id="problemset"]/tbody/tr/td[6]'))
  answer_percents <- sapply(problem_nodes, function(x) {x$getElementText()})
  answer_percent <- append(answer_percent, unlist(answer_percents))

  #다음페이지
  pageLink_next <- remDr$findElements(using='css',"#next_page")
  remDr$executeScript("arguments[0].click();",pageLink_next)
  Sys.sleep(1)

  curr_PageElem <- remDr$findElement(using='css',
  'div.wrapper > div.container.content > div:nth-child(6) > div:nth-child(2) > div > ul > li.active')

  curr_PageNewNum <- as.numeric(curr_PageElem$getElementText())

  if(curr_PageNewNum == curr_PageOldNum)  {
    cat("종료\n")
    #태그 하나 종료 시 다시 처음 화면으로
    site <- 'https://www.acmicpc.net/problem/tags'
    remDr$navigate(site)
    df <- data.frame(problem_num, answer_percent, check.rows = FALSE)
    # 파일명 생성
    file_name <- paste0(df[n,"algo_title"],".csv")
    # 저장 경로지정 + 순서 + 파일 이름
    save_name <- paste0("./BEAKJOON/", n, "_", file_name)
    # 파일 저장
    write.csv(df, save_name)
    break;
  }
  curr_PageOldNum <- curr_PageNewNum;
}
```

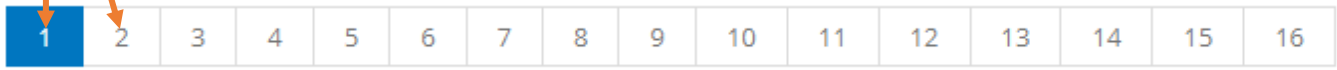
자동으로 다음 페이지의
태그명이 #next_page로 변경

마지막 페이지 까지 이동 후
각 알고리즘별 csv파일 생성



문제 번호	제목	정보	맞은 사람	제출	정답 비율
1000	A+B	다국어 디버그 분류	101361	322268	44.162%
1001	A-B	디버그 분류	83129	135554	72.076%
10998	A×B	디버그 분류	56987	80224	77.921%
10869	사칙연산	디버그 분류	56058	119396	53.642%
2739	구구단	디버그 분류	55245	120793	53.566%
1008	A/B	스페셜 저지 분류	54906	204454	33.294%
10430	나머지	디버그 분류	52111	100256	58.591%
8393	합	출처 다국어 분류	47507	78060	69.454%
10871	X보다 작은 수	분류	40991	85958	56.461%
10950	A+B - 3	분류	36230	70671	59.189%
1110	더하기 사이클	분류	35699	88559	48.244%
1330	두 수 비교하기	디버그 분류	35558	80236	52.686%
2753	윤년	디버그 분류	33307	67889	55.475%
11720	숫자의 합	분류	33306	79403	50.464%
2839	설탕 배달	출처 다국어 디버그 분류	32144	132134	31.926%
2558	A+B - 2	디버그 분류	31952	48299	74.742%
2577	숫자의 개수	출처 분류	31904	61761	61.471%
2588	곱셈	출처 분류	31454	68421	53.144%
10952	A+B - 5	분류	31329	66511	53.899%
11021	A+B - 7	분류	29275	60541	53.834%

현재 활성화된 페이지



dplyr 패키지 전처리 과정

```
# answer_percent의 %제거, 숫자로 변환
dataset$answer_percent=as.numeric(gsub("%","",dataset$answer_percent))
answer_percent <- round(dataset$answer_percent,2)
head(dataset)

#전체 문제 중 각 정답율이 차지하는 문제 비율을 구한다.
a = dataset %>%
  filter(answer_percent >= 90) %>%
  tally()*100/dataset$problem_num
b = dataset %>%
  filter(answer_percent >= 80 & answer_percent < 90) %>%
  tally()*100/dataset$problem_num
c = dataset %>%
  filter(answer_percent >= 70 & answer_percent < 80) %>%
  tally()*100/dataset$problem_num
d = dataset %>%
  filter(answer_percent >= 60 & answer_percent < 70) %>%
  tally()*100/dataset$problem_num
e = dataset %>%
  filter(answer_percent >= 50 & answer_percent < 60) %>%
  tally()*100/dataset$problem_num
f = dataset %>%
  filter(answer_percent >= 40 & answer_percent < 50) %>%
  tally()*100/dataset$problem_num
g = dataset %>%
  filter(answer_percent >= 30 & answer_percent < 40) %>%
  tally()*100/dataset$problem_num
```

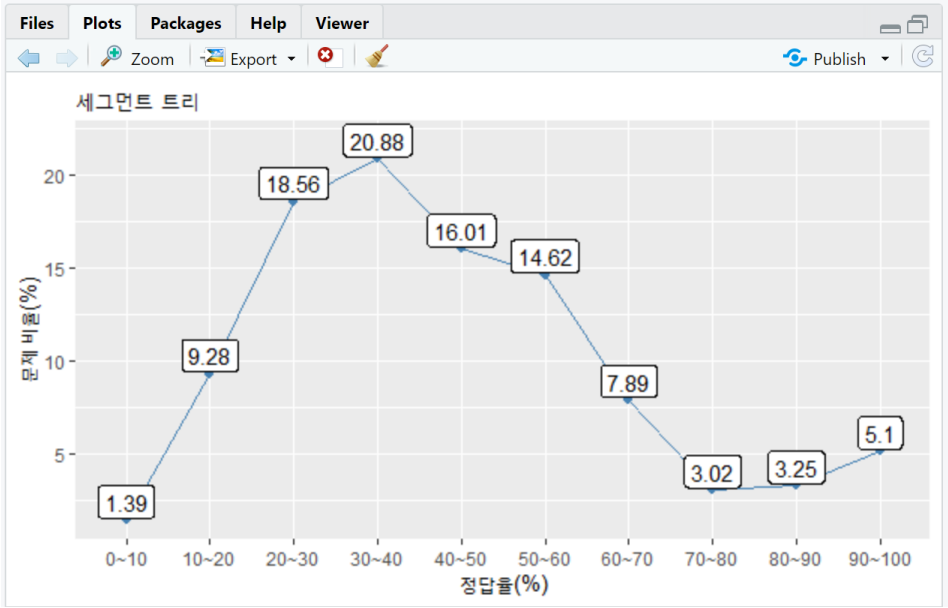
정답율에 표기된 % 문자를 삭제하고, 문자열을 숫자로 변환함

```
> a = dataset %>%
+     filter(answer_percent >= 90) %>%
+     tally()*100/dataset$problem_num
> a
         n
1 5.104408
```

전체 데이터 셋 중 정답율이 차지하는 비율을 구함

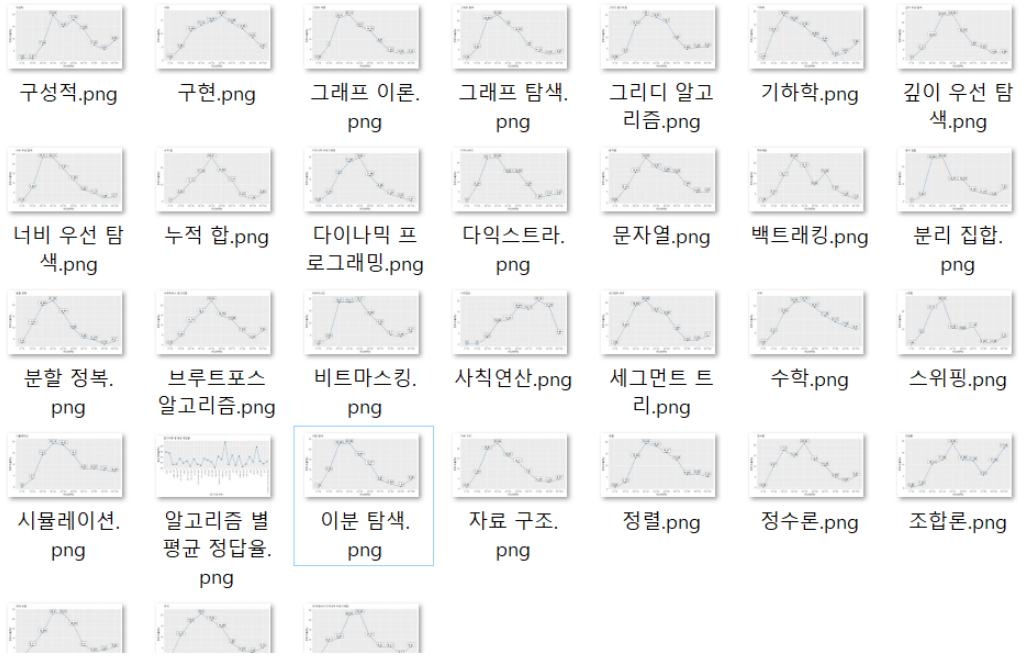
ggplot 시각화 과정

```
#이름 테이블로 만든다.
tab <- data.frame(tag = c('90~100','80~90','70~80',
                          '60~70','50~60','40~50',
                          '30~40','20~30','10~20',
                          '0~10'),
                  percent = rbind(round(a,2),round(b,2),round(c,2),
                                  round(d,2),round(e,2),round(f,2),
                                  round(g,2),round(h,2),round(j,2),round(k,2)))
names(tab) <- c("tag", "percent")
print(head(tab))


#테이블을 그래프로 그린다.
baekjoon <- ggplot(tab, aes(x=tag,y=round(percent,2),group=1))+
  geom_point(color="steelblue",stroke=1)+
  geom_line(color="steelblue")+
  geom_label(aes(label=percent), nudge_y=1)+
  labs(x="정답율(%)", y="문제 비율(%)", title=gsub("[[:digit:][:punct:][:lower:][:upper:]]
print(baekjoon)
ggsave(paste0(gsub("[[:digit:][:punct:][:lower:][:upper:]]","",src_file[i]),".png"))
```



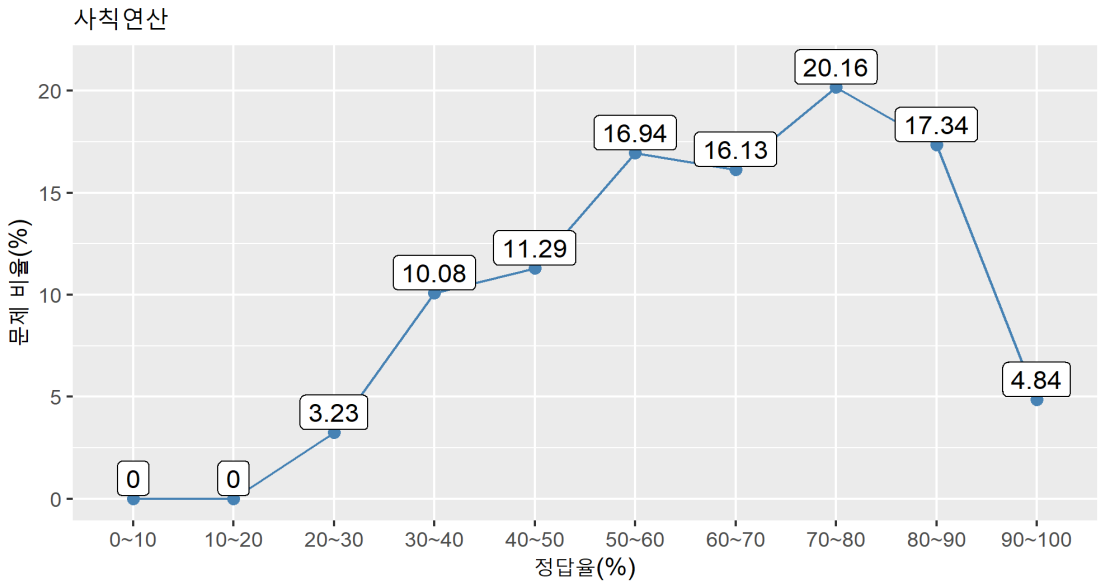
알고리즘별 그래프 생성과정

```
rm(list=ls())
src_dir <- c("C:/mini_project/data")
src_dir

src_file <- list.files(src_dir)
src_file

src_file_cnt <- length(src_file)
src_file_cnt

for(i in 1:src_file_cnt){
  dataset <- read.table(
    paste(src_dir,"/",src_file[i],sep=""),
    sep=",",header=T,stringsAsFactors = F)

  .
  .
  .

  ggsave(paste0(gsub("[[:digit:][:punct:][:lower:][:upper:]]","",src_file[i]),".png"))
}
```



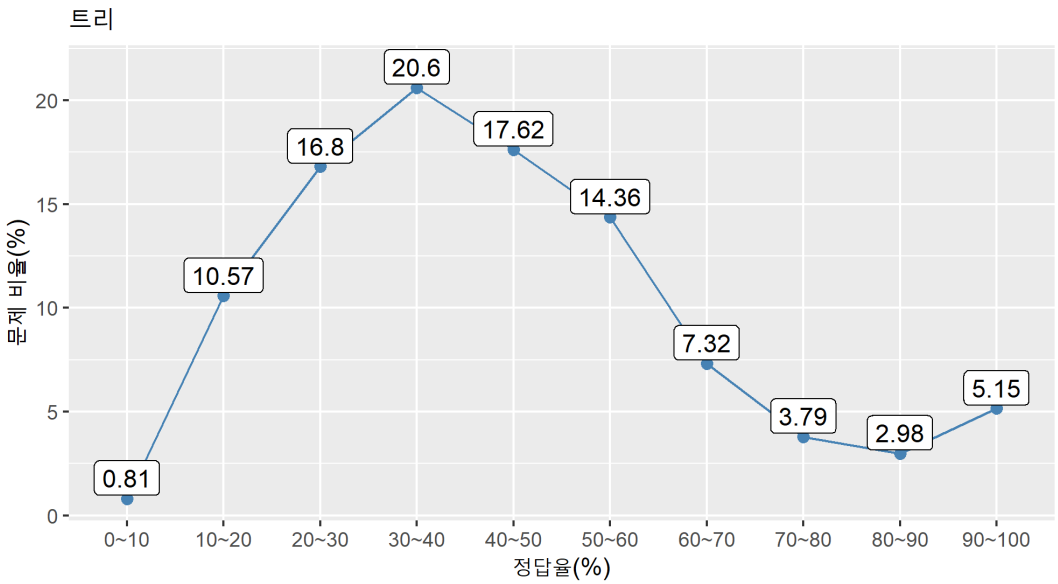
반복문을 사용하여 30개의 그래프를 생성 후, 저장

알고리즘 그래프 해석



우측이 볼록한 그래프는 난이도 쉬움을 나타냅니다.

사칙 연산의 경우, 정답 비율 70% 이상인 문제가 42.34%를 차지합니다.



좌측이 볼록한 그래프는 난이도 어려움을 나타냅니다.

트리의 경우, 정답 비율 40% 이하인 문제가 48.78%를 차지합니다.

가설 2

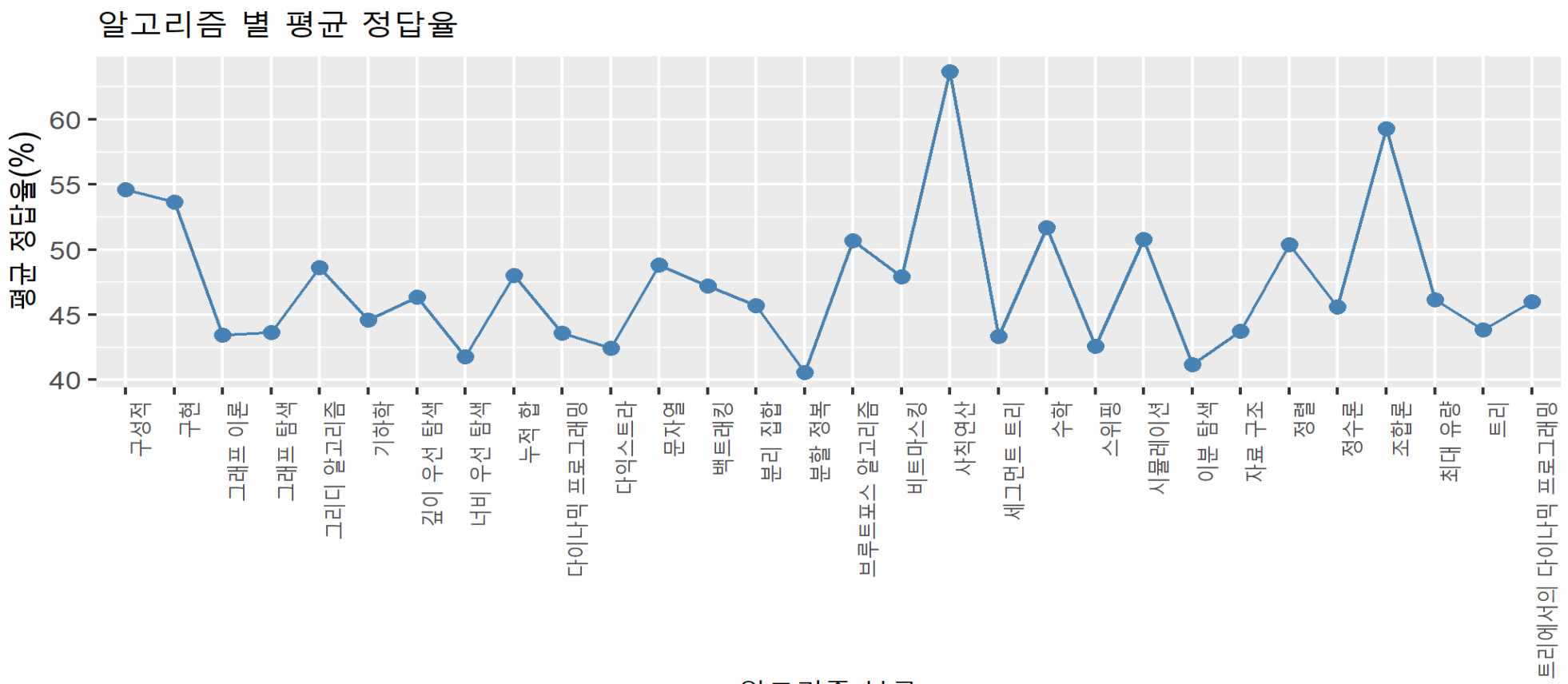
평균 정답비율이

각 알고리즘의 난이도를 뜻할 것이다.

각각의 알고리즘의 평균 정답비율이

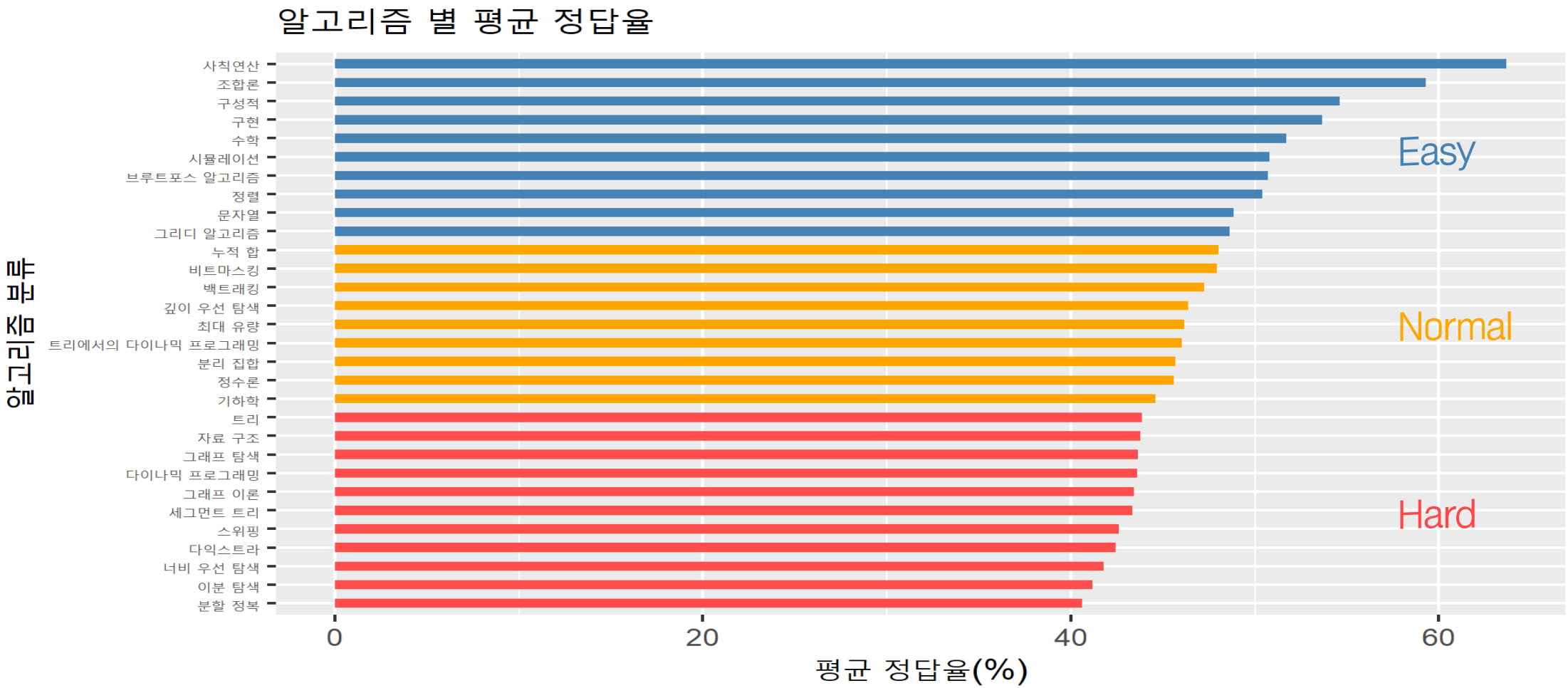
높을수록 쉬운 알고리즘이다.

알고리즘 별 평균 정답비율 그래프



평균 정답율을 구했을 때 40%~65% 까지 평균 정답율이 골고루 분포함을 알 수 있다.

평균 정답률이 높은 순으로 도표



정답비율 간 편차를 반영해 상위 10개는 EASY, 중위 9개는 NORMAL, 하위 11개는 HARD로 분류하였다.

구체적 평균 정답률

순위	algo_title	mean_percent(%)
1	사칙연산	63.67
2	조합론	59.3
3	구성적	54.6
4	구현	53.67
5	수학	51.71
6	시뮬레이션	50.8
7	브루트포스 알고리즘	50.71
8	정렬	50.4
9	문자열	48.83
10	그리디 알고리즘	48.62
11	누적 합	48.01
12	비트마스킹	47.95
13	백트래킹	47.23
14	깊이 우선 탐색	46.37
15	최대 유량	46.15

순위	algo_title	mean_percent(%)
16	트리에서의 다이나믹 프로그래밍	46.03
17	분리 집합	45.7
18	정수론	45.59
19	기하학	44.6
20	트리	43.86
21	자료 구조	43.76
22	그래프 탐색	43.65
23	다이나믹 프로그래밍	43.58
24	그래프 이론	43.43
25	세그먼트 트리	43.33
26	스위핑	42.6
27	다익스트라	42.45
28	너비 우선 탐색	41.79
29	이분 탐색	41.17
30	분할 정복	40.59

평균 정답률 최상위와 최하위 간 편차는 23.08%이다.

평균 정답비율이 제일 높은 알고리즘_사칙연산

10998번 | 제출 | 맞은 사람 | 숏코딩 | 재채점/수정 | 디버그 | 채점 현황 | 강의

A×B 분류

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	80275	61619	57025	77.921%

문제

두 정수 A와 B를 입력받은 다음, A×B를 출력하는 프로그램을 작성하시오.

입력

첫째 줄에 A와 B가 주어진다. (0 < A, B < 10)

출력

첫째 줄에 A×B를 출력한다.

정답 비율 78%에 가까운 문제. 파이썬 기초만 알아도 쉽게 풀 수 있는 문제로 구성되어 있다.

평균 정답비율이 제일 낮은 알고리즘_분할정복

2261번 | 제출 | 맞은 사람 | 숏코딩 | 재채점/수정 | 채점 현황 | 강의

가장 가까운 두 점 분류

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	21262	3421	1702	15.260%

문제

2차원 평면상에 n개의 점이 주어졌을 때, 이 점들 중 가장 가까운 두 점을 구하는 프로그램을 작성하시오.

입력

첫째 줄에 자연수 n(2 ≤ n ≤ 100,000)이 주어진다. 다음 n개의 줄에는 차례로 각 점의 x, y좌표가 주어진다. 각각의 좌표는 절댓값이 10,000을 넘지 않는 정수이다. 같은 점이 여러 번 주어질 수도 있다.

출력

첫째 줄에 가장 가까운 두 점의 거리의 제곱을 출력한다.

예제 입력 1 복사

```
4
0 0
10 10
0 10
10 0
```

예제 출력 1 복사

```
100
```

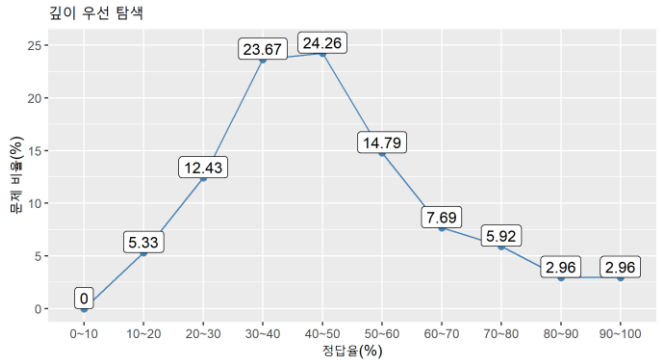
정답 비율 15%의 문제. 문제를 이해하기까지 시간이 걸리고
사칙연산 알고리즘에 비해 많은 생각이 필요하며 난해하다.

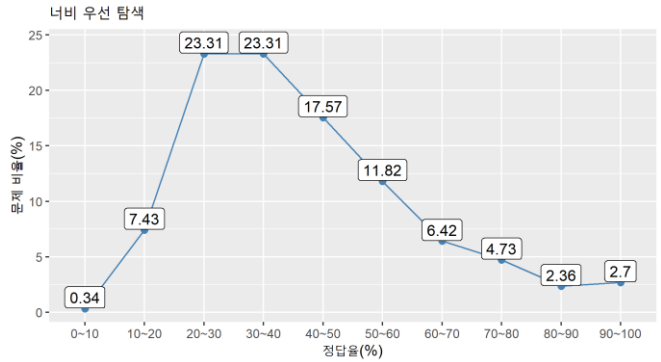
전문가의 답변과 분석 결과의 비교

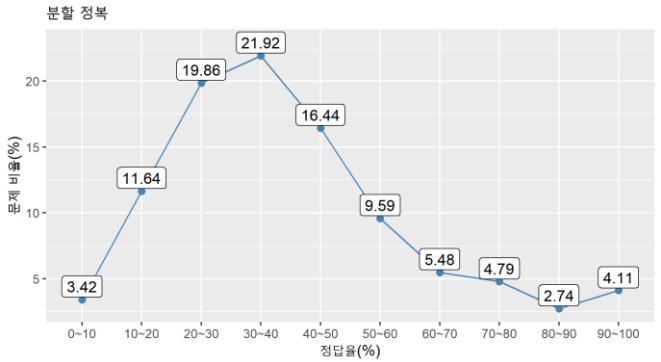
강사님

어려운 알고리즘 : 너비우선탐색, 깊이우선탐색, 분할정복

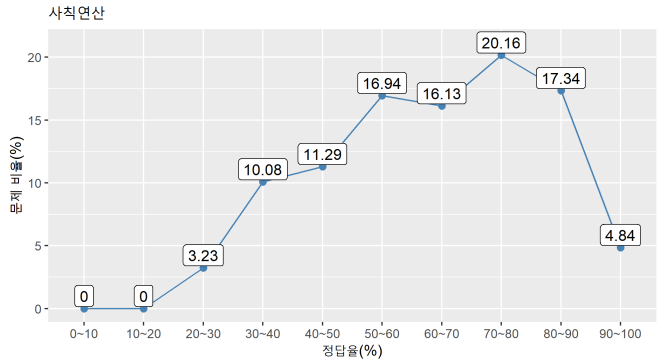
쉬운 알고리즘 : 사칙연산, 정렬, 문자열

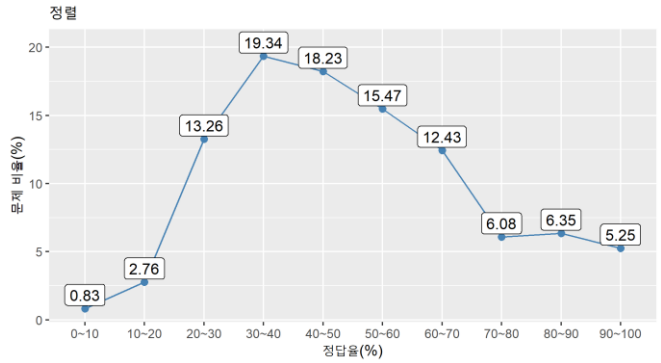


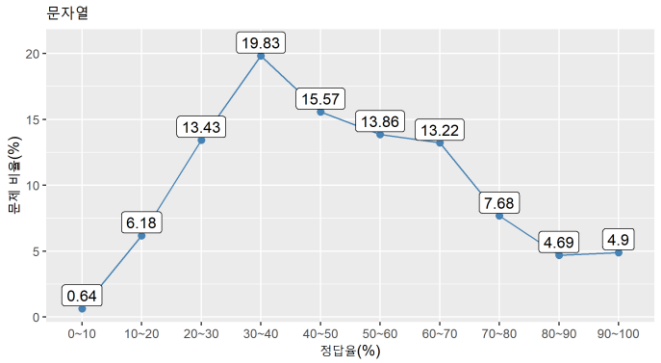




왼쪽으로 치우쳐진 그래프가 그려진다. 정답율 30% 이하 문제의 비율이 현저히 높음을 알 수 있다.







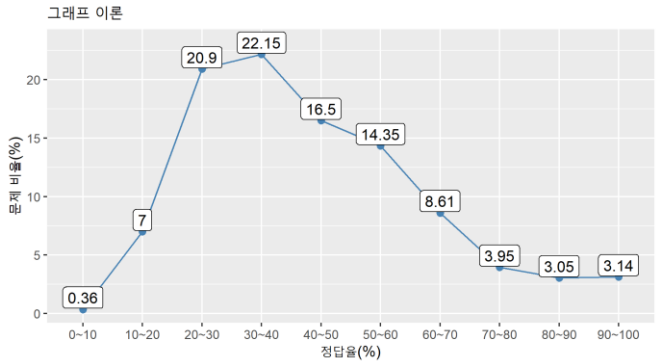
3개의 쉬운 알고리즘 모두 난이도 EASY에 속한다.

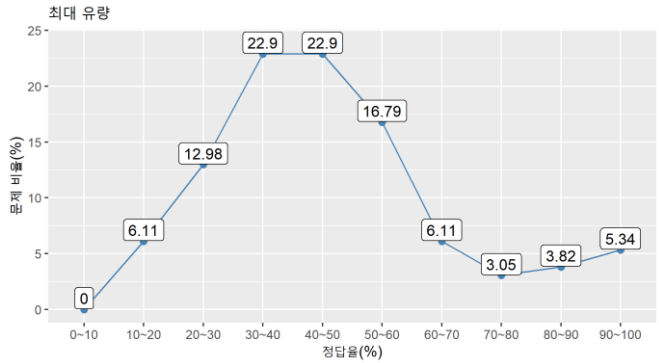
알고리즘을 공부하는 학생의 답변과 분석 결과의 비교

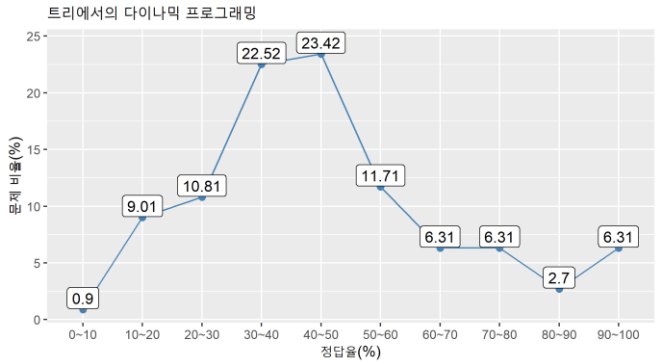
학생

어려운 알고리즘 : 그래프이론, 최대유량, 트리에서의 다이나믹 프로그래밍

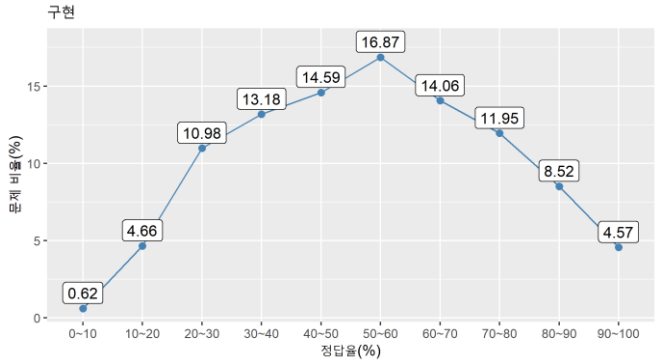
쉬운 알고리즘 : 구현, 사칙연산, 시뮬레이션

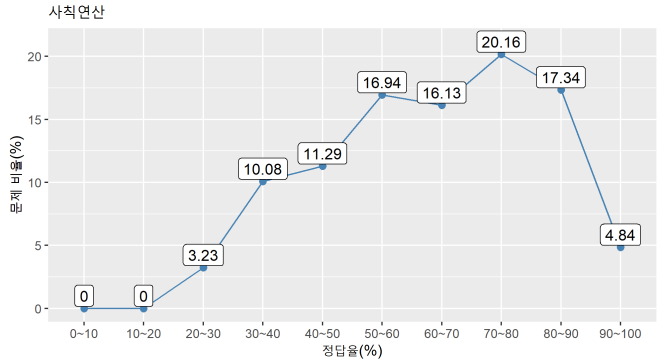


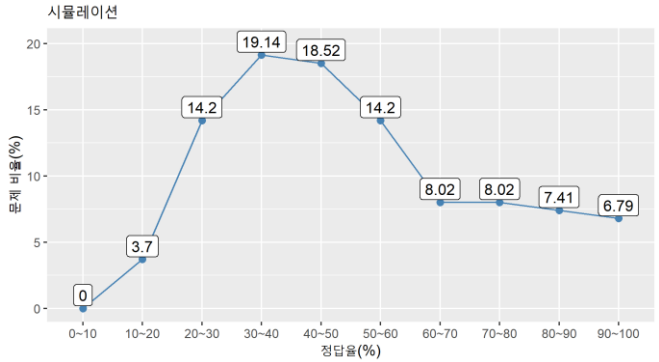




마찬가지로 왼쪽으로 치우쳐진 그래프가 그려진다. 정답율 70% 이상 문제 비율이 10%에 불과하다.







3개의 쉬운 알고리즘 모두 난이도 EASY에 속한다.

우리가 제시하는 공부방안

SOLUTION

난이도 EASY → NORMAL → HARD 에 속하는
알고리즘 순으로 공부를 하는 것이 도움 될 것이다.
평균 정답율이 높은 문제일수록
기초적인 자료구조로 이루어져 있다.

프로젝트를 끝내며..

THE END

당신의 알고리즘 스터디를 응원합니다!!