

코딩 테스트 with알고리즘

CMD 세션 4주차



01.
기업별
코딩테스트

02.
금쪽같은
인터뷰

04.
알고리즘 공부
사이트

05.
잘 나오는
알고리즘들

03.
추천 언어
소개



01. 기업별 코딩테스트

NAVER

kakao

배달의민족

Google

SAMSUNG

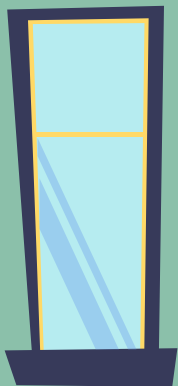
당근마켓

LINE

toss

NETFLIX

coupang



코딩테스트란?

- 기업/기관에서 직원이나 연수생을 선발하기 위한 목적으로 시행되는 일종의 문제 풀이 시험
- 공개채용을 하는 기업에서 주로 이용

코딩 테스트의 유형

온라인 코딩 테스트

- 인터넷을 활용해 프로그래밍 역량을 평가하여 응시자를 선별
- 대체로 라인과 문제풀이를 공유하지 않는 선에서 인터넷 검색을 허용

오프라인 코딩 테스트

- 시험장에 방문하여 치르는 시험
- 대체로 인터넷 검색이 허용되지 않으며 회사에서 제공하는 컴퓨터 환경을 이용



구글 코리아



구글 채용 과정

1

지원

2

**Phone
Interview**

45분간
인터뷰

3

Onsite

4회의
코딩인터뷰

4

**Team
Matching**

5

**Hiring
Committee**




구글 채용 과정의 특징

- 구체적인 요구사항
- Google Docs에서의 코딩
- Hiring committee
- Grading scale (1~4 사이의 등급)
- 4개의 채용 기준
(cognitive ability, googleyness, leadership skills, technical skills)




Software Engineer, Site Reliability Engineering

 Share

 Save

 Google

 San Francisco, CA, USA

 Apply

Minimum qualifications:

- Bachelor's degree in Computer Science, Software or System Engineering, a related technical field, or equivalent practical experience.
- Experience programming in at least one of the following languages: C, C++, Java, Python, or Go.
- Experience with algorithms and data structures.

Preferred qualifications:

- Experience in designing, analyzing, and troubleshooting large-scale distributed systems.
- Understanding of Unix/Linux operating systems.
- Ability to debug and optimize code and automate routine tasks.
- Systematic problem-solving approach, coupled with effective communication skills and a sense of drive.

About the job



Site Reliability Engineering (SRE) combines software and systems engineering to build and run large-scale, massively distributed, fault-tolerant systems. SRE ensures that Google's services—both our internally critical and our externally-visible systems—have reliability, uptime appropriate to users' needs and a fast rate of improvement. Additionally SRE's will keep an ever-watchful eye on our systems capacity and performance. Much of our software development focuses on optimizing

출처: <https://careers.google.com/jobs/results/89459978537968326-software-engineer-site-reliability-engineering/?distance=50&q=Software%20Engineering>

코딩 테스트 3가지 유형

- System design questions:
high-level system design with scalability
- Coding interview challenges:
data structures and algorithms
- General analysis questions:
thought process through mathematical or opinion-based problem

코딩 테스트 예제

Find the kth largest element in a number stream

Problem Statement: Design a class to efficiently find the Kth largest element in a stream of numbers. The class should have the following two things:

- The constructor of the class should accept an integer array containing initial numbers from the stream and an integer 'K'.
- The class should expose a function add(int num) which will store the given number and return the Kth largest number.

Find 'k' closest numbers

Problem Statement: Given a sorted number array and two integers 'K' and 'X', find 'K' closest numbers to 'X' in the array. Return the numbers in the sorted order. 'X' is not necessarily present in the array.

Delete node with given key

Problem statement: You are given the head of a linked list and a key. You have to delete the node that contains this given key.

출처: <https://www.educative.io/blog/google-coding-interview-questions>

코딩 테스트 예제

Implement a LRU cache

Problem statement: Least Recently Used (LRU) is a common caching strategy. It defines the policy to evict elements from the cache to make room for new elements when the cache is full, meaning it discards the least recently used items first.

Merge overlapping intervals

Problem statement: You are given an array (list) of interval pairs as input where each interval has a start and end timestamp. The input array is sorted by starting timestamps. You are required to merge overlapping intervals and return output array (list).

Print balanced brace combinations

Problem statement: Print all braces combinations for a given value 'N' so that they are balanced.



출처: <https://www.educative.io/blog/google-coding-interview-questions>

후기

- 말하면서 문제를 풀이하는 연습이 필요함
- 온사이트 인터뷰에서는 영어로 진행되기도 함
- LeetCode: Hard 문제
- Topcode SRM: Div1 250점 문제
- GeeksforGeeks: 코딩 인터뷰 알고리즘
- HackerRank

출처: <https://jeinalog.tistory.com/30>

출처: <http://mrkimkim.com/category/diary/test-interview-review/>

kakao
카카오



카카오 공개 채용

영입절차

01
서류전형

02
1차 인터뷰

03
2차 인터뷰

04
입사확정

실무진과 직무별
전문 인터뷰어가 참석하여
직무역량을 심층 검증

리더 및 인사부서가
참석하여 조직적합도와
잠재력을 심층 검증

위는 카카오 영입의 기본 프로세스이며, 경우에 따라 추가 절차가 포함될 수 있습니다.

- 기술직군 : 서류전형 합격 후 코딩테스트, 원격 인터뷰 진행
- 디자인직군 : 서류전형 합격 후 정규직에 한해 과제전형 진행
- 비기술직군(서비스사입/스텝/디자인) : 포지션에 따라 3차 인터뷰가 진행될 수 있음

인터뷰에서 불합격하신 경우, 동일직무 재지원은 입사지원 시점으로부터 1년 후 가능합니다.
장애/보훈 대상자는 관계 법령에 의거하여 우대합니다.

카카오 인턴 (Tech developers)

전형방법 및 일정

지원 접수	> 코딩 테스트 >	서류 전형 >	인터뷰 >	인턴 합격자 발표 >	인턴십
04. 15 (목) ~ 05. 03 (월) 오후 5시	05. 08 (토)	5월 중순	6월 초	6월 둘째 주	06. 28 (월) ~ 08. 31 (화) 예정 (2개월)

- * 세부분야 선택에 도움드리고자, 서류전형 전 코딩테스트 합격자 대상으로 설명회를 진행합니다. (5.15 예정)
- * 세부분야로 Statistics를 선택할 경우, 추가 코딩테스트를 진행합니다. (5.19 예정)



신입공채 1차 온라인 코테

- 2020년 9월 12일 오후 2시부터 7시까지 **5시간 동안** 진행
- 총 7개의 문제 출제
- C++, Java, Javascript, Kotlin, Python, Swift 총 6가지 언어 제공
- 부분 점수 x
- 일부 문제는 정확성 테스트 외에 효율성 테스트도 진행돼서 추가 점수 부여되도록 설계



• 2021 카카오 신입공채 1차 코테 정답률 •

	정확성	효율성
1. 아이디 추천	57.50%	-
2. 메뉴 리뉴얼	25.40%	-
3. 순위 검색	44.07%	4.49%
4. 합승 택시 요금	9.60%	7.45%
5. 광고 삽입	1.23%	-
6. 카드 짝 맞추기	0.95%	-
7. 매출 하락 최소화	0.57%	-

역대 카카오 1차 테스트 컷

카카오	1차 (2019-09-07)	5시간	7문제	4문제(예상)	구현, 다이나믹 프로그래밍, 자료구조	온라인
	2차 (2019-09-21)	5시간	1문제	-	추천 시스템 개발	오프라인
카카오	1차 (2018-09-15)	5시간	7문제	3문제	그리디, 구현	온라인
	2차 (2018-10-06)	5시간	1문제	-	시뮬레이션 개발	오프라인

출처: https://www.hanbit.co.kr/channel/category/category_view.html?cms_code=CMS4385594264

신입 공채 2차 오프라인 코테

- 2020년 9월 26일 오후 12시 30분부터 7시까지 총 **6시간 30분** 동안 진행
- CS 문제와 코딩 테스트가 진행
- CS 테스트 (단답형 주관식 1문제, 객관식 9문제)
- 자료구조 3문제, 데이터베이스 2문제, 네트워크 2문제, 운영체제 2문제, 알고리즘 1문제
- CS 기초 지식을 확인하는 수준의 난이도

출처: <https://tech.kakao.com/2021/02/16/2021-kakao-recruitment-round-2/>



신입 공채 2차 오프라인 코테

- 2차 코딩 테스트
- 15분 휴식 이후, 4시간 45분 동안 진행
- 실시간으로 리더보드 갱신(100위까지)
- 80점대가 100위, 78점 합격
- REST API 호출 처리 모듈과 JSON에 대한 이해 필수

출처: <https://tech.kakao.com/2021/02/16/2021-kakao-recruitment-round-2/>



NAVER 네이버



네이버 채용 과정

1

서류
접수

2

온라인
코딩 테스트

3

1차 면접

기술 면접

4

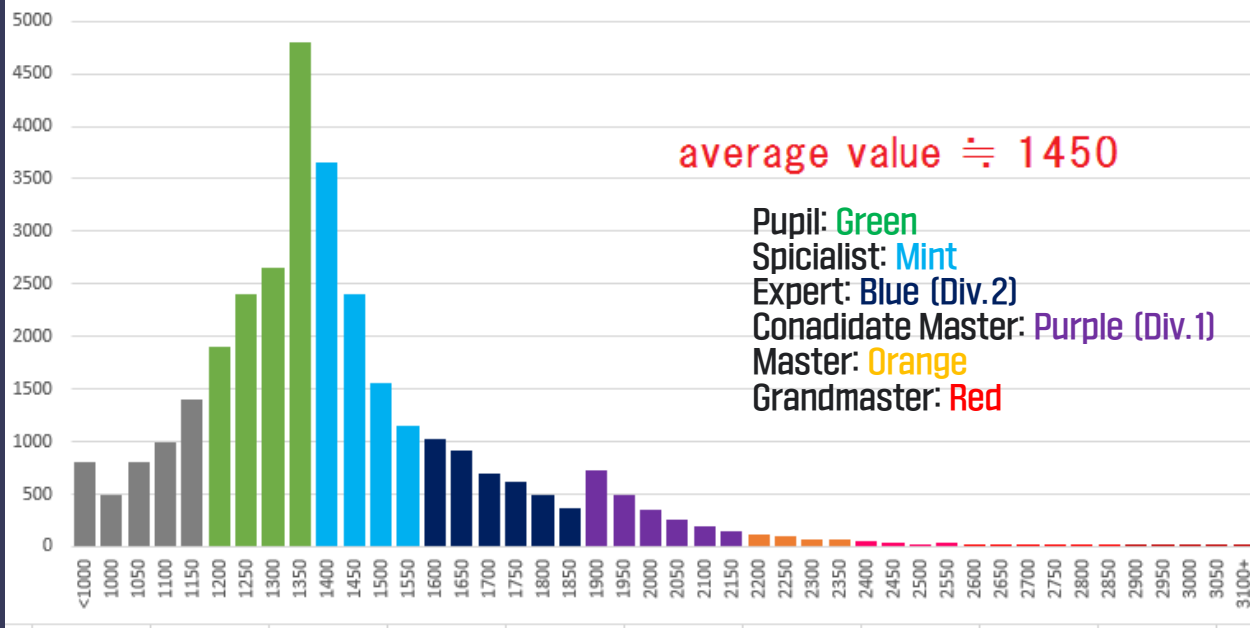
2차 면접

인성 면접

네이버 코딩 테스트

- 2021년 4월 24일 오후 2시부터 4시까지 총 **두시간**동안 진행
- 총 4문제
- C, C++, Java, Javascript, Python3, Swift, Kotlin 사용 가능
- 문자열, 구현, 이분탐색, 조합, 단순구현 등을 사용하는 문제
- 4문제 중 1문제는 효율성도 평가

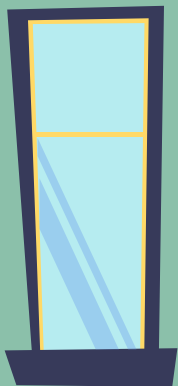
Codeforces Rating Distribution





02. 금쪽같은 인터뷰

CMD 선후배님들의
코딩테스트 경험 공유



//

보다 현실적인 후기가 없을까
실전 팁이 더 없을까 싶어 혼자 고민할 때!

같은 과목들을 들었고,
함께 걸어가고 있는 학우들의 인터뷰





19 최지원 학우

• 인터뷰 •

A

준비했던/봤던 코딩테스트?

프로그래머스 주최
2021 Summer Coding

B

어떻게 준비하셨나요?

학교 알고리즘 수업을 열심히 듣고
백준 알고리즘 사이트에서 알고리즘 별로 모아서
풀었습니다. 따로 행사를 위해 준비하지는 않았고
평소에 공부했고 별로 공부하지도 않았어요 $\pi\pi$
50개 풀고 티어도 실버니까..
SQL문제는 백준 같은 곳에 없어서 **프로그래머스에**
있는 SQL 고득점 키트를 다 풀고 들어갔어요.

인터뷰

C

응시 언어?

파이썬, MySQL

D

문제 유형과 체감 난이도?

난이도는 백준과 비슷!

1. 문자열을 바탕으로 정렬하는 간단한 문제
[백준 기준으로 브론즈 1~실버 5정도]
2. 우선순위 큐와 정렬을 이용하는 문제
[백준 기준으로 실버 1~골드4 정도]
3. BFS/DFS 사용하는 개애애애복잡한 구현 문제...
[백준 기준으로 플래5 이상]
4. Column 1과 Column 2 데이터 쌍의 중복을 없애는 문제

1,2는 쉬웠고 3은 1시간 넘게 고민해도 못풀었고 4는 할만했어요!

• 인터뷰 •

E

CMD를 위한 코테 후기/ 준비나
실전 팁?



1

파이썬은 개애애애꿀이다

2

BFS, DFS, DP는 무조건 나온다

3

옥료구조 옥고리즘 수업이 생각보다
도움이 많이 된다 숙제만 하고
땡치지 말고 공부를 열심히 하자

4

테스트에만 사용되는 잡기술들을
다른 사람의 코드를 보면서 익혀두자



19 윤세린 학우

인터뷰

준비했던/봤던 코테

어떻게 준비하셨나요?

대외활동을 위한 코딩테스트 !

평상시에는 **백준**에서 단계별로 풀고
코테 임박해서는 **프로그래머스**랑 백준을
같이 풀었다!

인터뷰



응시 언어

알고리즘은 python,
sql은 postgresql

문제유형과 체감난이도?

시간 너무 없고 어려웠다.
유형은 **그래프가 가장 많이** 나온다.
(dfs/bfs, dp, backtracking 등)
출제 유형은 거의 정해져 있는 거 같다.

• 인터뷰 •

Q

CMD를 위한 코테 후기/ 준비나 실전 팁?

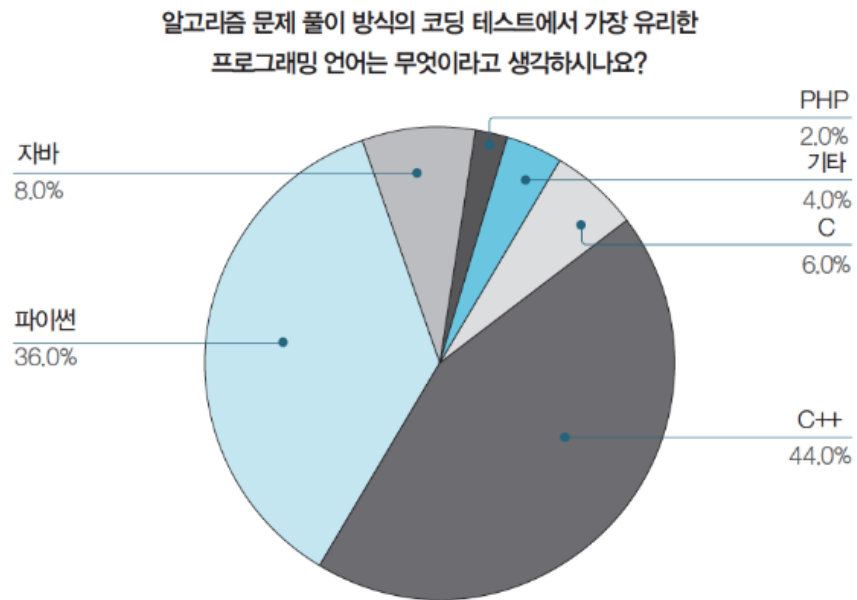
- ① 일단 초반에는 백준에서 단계별로 차근차근 풀며 연습해보는 게 좋은 거 같고,
- ② python을 사용한다면 나동빈의 <이것이 코딩테스트다> 책을 추천!
내가 쓴 코드보다 효율이 엄청나게 좋은 코드들이 많기 때문에 비교해보는 것도 좋은 거 같다.
- ③ 구현에만 집중하다가 수학적 계산을 잘 못하는 경우가 있는데 짧은 시간 내에 머리를 잘 굴려야 한다,,
문제를 많이 풀면서 연습해보자!



코딩테스트 준비 - 언어

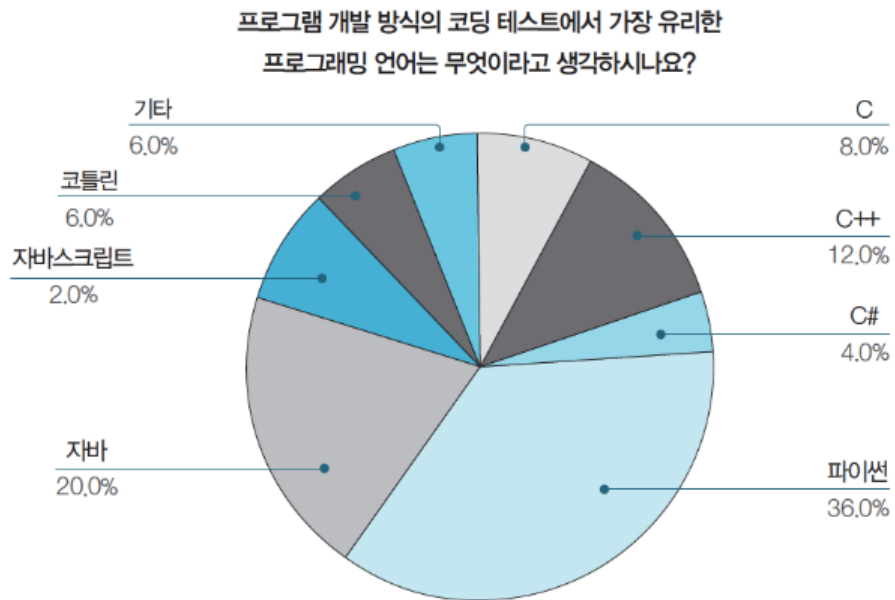
보다 인기있거나 유리한 언어는 무엇일까

언어들 간 비교

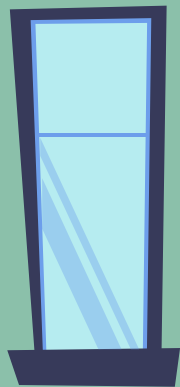


[출처: 이것이 취업을 위한 코딩 테스트다 with 파이썬(책)]

언어들 간 비교



[출처: 이것이 취업을 위한 코딩 테스트다 with 파이썬(책)]



C++ ?

기프와 자구, 알고리즘에서 C 사용



Python?

컴사코, 문알, 자구, 알고리즘에서 사용

C++

특징

- 거의 모든 코테에서 지원
- 파이썬보다 하드웨어에 가까운 언어

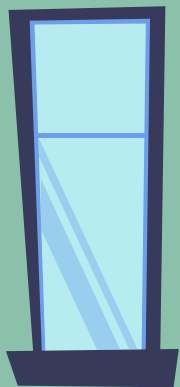
장점

- 실행 시간에 유리
- 알고리즘 대회/문제풀이 코테에서 주로 사용
- 다양한 알고리즘에서 우수 코드 자료 多

단점

- 파이썬보다 어렵다
(포인터의 잘못된 사용, 배열 인덱스 초과, integer overflow 등등)
- 문자열 처리가 까다로움





Python

특징

- 거의 모든 코테에서 지원
- 최근 응시자 증가 및 최근 출제 경향에 유리한 경우 多

장점

- C++ 보다 쉬움, 짧고 직관적
- 기본 라이브러리가 많은 기능 제공
- 문자열, 리스트, 큰 수 등 처리에서 훨씬 유리
- 프로그램 개발 방식의 코테에 유리
- API 개발 문제들에 유리

단점

- 느림 느림 느림 (수행시간, Pypy3)
- C++에 비해 자료 적음 (물론 빠른 속도로 많아지고 있음)

그 외의 언어들 비교

JavaScript

- IT Big3에서 모두 가능
- 문제 풀기에 적합은.....
- 웹프론트의 경우 JS only 경우 0



Kotlin

- IT Big3 모두 지원
- Java보다 덜 불편하지만
Java를 잘 다루면 굳이 코틀린을?



Swift

- IT Big3 모두 지원
- iOS 직무 지원 시
- 활성화된 오픈카톡방 0



코딩 테스트 준비 - 언어

정답은 없지만

파이썬이 주라면 단점을 C++로, C++이 주라면 단점을 파이썬으로 커버할 수 있는 능력을 갖추자



- 아직 코테나 알고리즘 베이비라면 파이썬으로 시작
- 지원 직무가 있으면 직무에 맞는 언어
- C++을 주력언어로 : 문자열처리 등에 파이썬 보조
- 파이썬을 주력언어로 : 시간복잡도 애매 C++로 보조
- 둘다 중요한 언어이니 무엇으로든 실력을 키우는게 우선ㅎㅎ

03. Algorithm Learning Site

문제풀이 순서, 장단점, 사용법





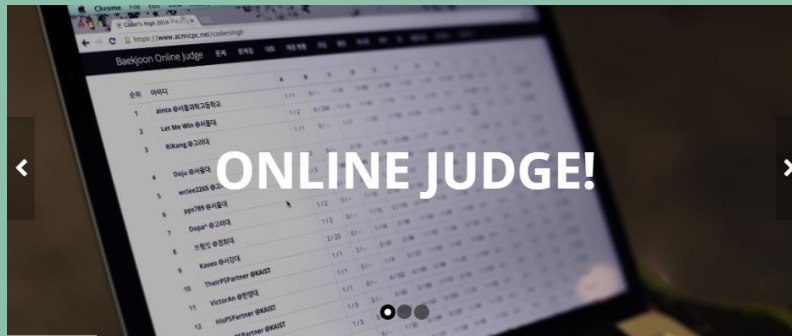
각 사이트들을 직접 들어가
클릭하면서 소개하려 합니다!!

한국정보올림피아드 대비 외에도

프로그래밍 초보자가 프로그래밍

입문을 쉽게 할 수 있도록 도와줌

Baekjoon Online Judge



Baekjoon Online Judge

프로그래밍 문제를 풀고 온라인으로 채점받을 수 있는 곳입니다.

20767

전체 문제

19194

채점 가능한 문제

16066

풀린 문제

75

채점 가능한 언어

내 파일

정보 수정

비밀번호 변경

solved.ac

보기 설정

알림 설정

테마 설정

계정 연동

링크 설정

학교/회사 정보

언어 설정

소스 코드 공개 설정

문의하기 / 탈퇴하기

solved.ac

SOLVED. AC

solved.ac는 BOJ 유저 shiftpsh님이 만든 서비스입니다.

이 서비스는 Baekjoon Online Judge의 문제의 난이도를 보고, 유저의 티어를 볼 수 있습니다.

다음 데이터를 solved.ac에 제공해 soyeon0130님의 티어를 계산 및 공개하려면 사용하기를 눌러주세요.

- 아이디
- 모든 채점 결과

제출한 소스 코드는 solved.ac에 제공하지 않습니다.

사용 중

그만 사용하기

5

soyeon0130

마스터킹

Baekjoon Online Judge

- 나의 수준을 티어를 통해 확인
- 다양한 언어 지원
- 삼성전자 소프트웨어 역량 테스트

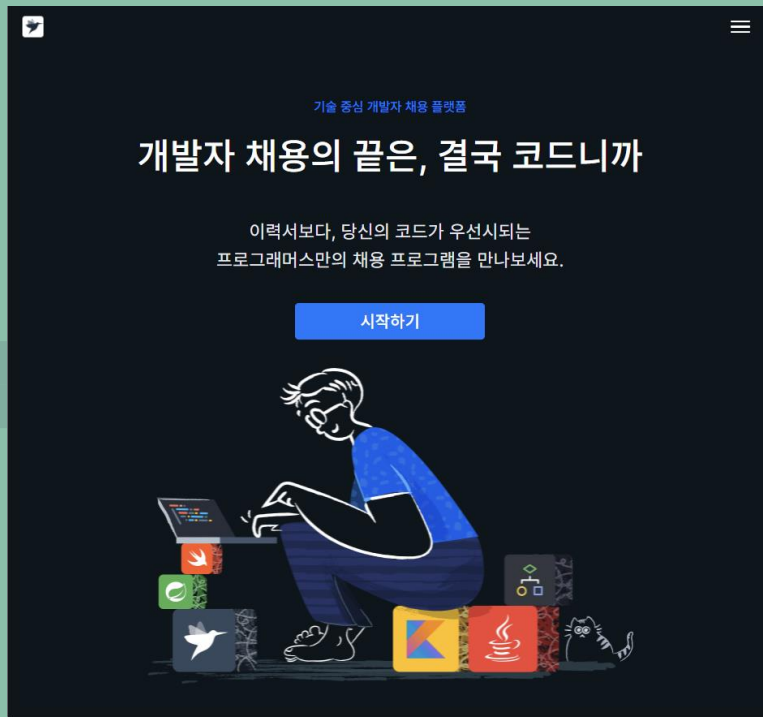


장점

- 오역으로 원문을 읽지 않으면 풀 수 없는 경우 존재
- 난이도의 함정

단점

• Programmers •



취업 및 이직을 희망하는 개발자에게

학습부터 구직

맞춤 커리어 서비스 제공

• Programmers •

- 실제 코테와 유사한 UI
- 기업 문제 모음(카카오 코테)
- 구직자와 회사를 연결
- 추천문제 제공
- 난이도 분류 잘됨

장점

- 가끔 틀린 알고리즘도 통과됨
- 부족한 문항 수
- 전체적으로 난이도가 높음(초심자 비추)



단점

Codeforces

경쟁적 프로그래밍 대회를

주기적으로 개최, 거대한 규모



Sponsored by Telegram

Enter

HOME TOP CONTESTS GYM PROBLEMSET GROUPS RATING EDU API CALENDAR HELP RAIF ML 🏆

★ Rating changes for the last round are temporarily rolled back. They will be returned soon.

Codeforces Raif ML Round 1

By raiffeisen, 3 weeks ago, translation, 

Codeforces, welcomel

Registration for the ML competition from Raiffeisenbank that starts on May 17 is now open. The prize fund of the championship is 700,000 rubles. **The competition is unrated.**



Codeforces Raif ML Round 1

Райффайзен карьера

All terms of participation were developed by the Raiffeisenbank eFX trading team in collaboration with Codeforces. Huge thanks to **geranazavr555** for the testing and great

→ Pay attention

Before contest
[Codeforces Round #721 \(Div. 2\)](#)
46:23:00
[Register now »](#)
*has extra registration

 Like 17 people like this. Sign Up to see your friends like.

→ Streams

[Hard Atcoder Upsolve](#)
By Errichto
Stream is running

→ Top rated

#	User
1	Benq
2	Radewoosh
3	tourist
4	ecnerwala
5	Um_nik
6	Petr
7	ksun48
8	liandyl

Rating Bounds	Color	Title	Division	Number	Number (by color)
2900+	Red	Legendary Grandmaster	1	4	183
2600 — 2899	Red	International Grandmaster	1	46	
2400 — 2599	Red	Grandmaster	1	133	
2300 — 2399	Orange	International Master	1	163	380
2200 — 2299	Orange	Master	1	217	
1900 — 2199	Violet	Candidate Master	1	1253	1253
1600 — 1899	Blue	Expert	2	5095	5095
1400 — 1599	Cyan	Specialist	2	8202	8202
1200 — 1399	Green	Pupil	2	5736	5736
0 — 1199	Gray	Newbie	2	2319	2319

Codeforces

- 세계적인 사이트
- 주마다 대회 실시
- 블루레벨? 합격~~
- Hack!

장점

- 영어와 러시아어만 지원

단점



• Codeforces •

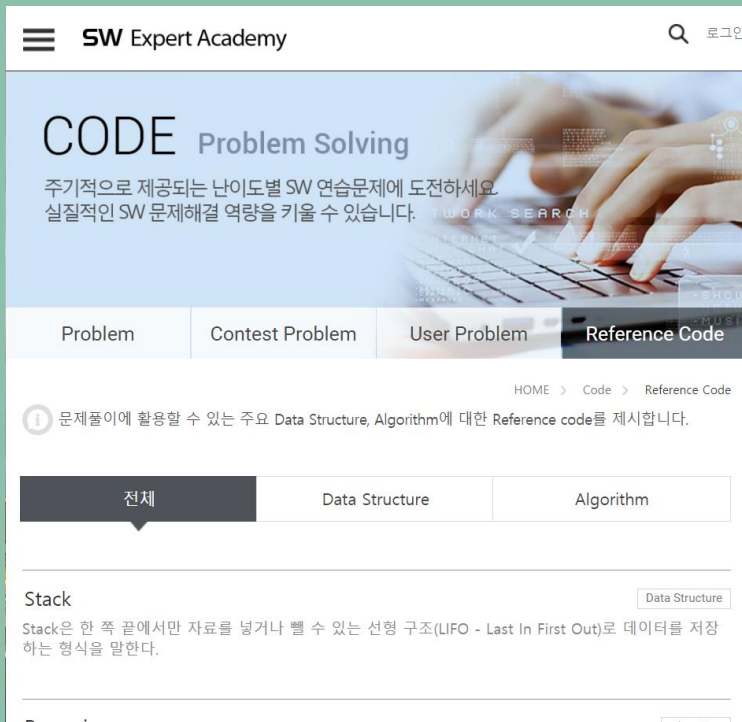
다른 사람의 코드에서 내가 오류가 날 만한 **입력 값을 넣었는데...**

오류가 맞다면 내가 100점 획득! **VS** 오류가 아니라면 내가 50점 감점ㅠ

“단,

hack하기 위해서는 자신의 코드를 lock시켜야 함”

• Samsung Expert Academy •



SW개발자에게 기본적으로 필요한
프로그래밍 역량을 키울 수 있는
다양한 학습 콘텐츠를 제공



• Samsung Expert Academy •

- 삼성에서 운영
- 상시테스트 진행, 그에 따른 혜택
- 영상 학습 콘텐츠 등 다양
- 자료구조 수업에도 유용

장점

- 지원하는 언어가 제한적

단점



04. 알고리즘 기법 및 팁!

기업별 가장 많이 출제된
유형과 풀이방법!

01. 시간 복잡도?

알고리즘을
사용하는 이유!

03. 알고리즘 유형별 기법

Top-down(Memorization),
Bottom-up, BFS/DFS,
Two Pointer

02. 코딩 테스트에서 자주 출제되는 유형

BFS/DFS, 구현,
DP, 그리디



행렬 곱셈 순서

Gold III

난이도 제공: solved.ac

시간 제한

1 초

메모리 제한

256 MB

제출

13404

정답

6086

맞은 사람

4343

정답 비율

43.838%

25630875

a01152a

1946

시간 초과

C++17 / 수
정

904 B

시간 초과

SOLUTION

1. 시간 복잡도

복잡도는 알고리즘의 성능을 나타내는
척도

특정 크기 입력에 대하여 알고리즘의
수행 시간 분석

빅오 표기법(Big-O Notation)

빅오 표기법	명칭
$O(N)$	선형 시간
$O(N^2)$	이차 시간
$O(\log M)$	로그 시간

1. 시간 복잡도

$O(N^3)$

n, m, s의 값이 100이면 몇번?
n, m, s의 값이 1000이면 몇번?

```
for (int i = 0; i < n; i++) {  
    for (int j = 0; j < m; j++) {  
        for (int k = 0; k < s; k++) {  
            cout << "내방 내주고 말아야!" << endl;  
        }  
    }  
}
```

일반적으로 CPU
기반의 개인 컴퓨터나
채점용 컴퓨터에서
연산 횟수가 5억을
넘어가는 경우



1~3초



5~15초

$O(N^3)$ 의 알고리즘을
설계한 경우 N의 값이
50000이 넘는다면
천이백오십억이
연산횟수



약 500초



약 2500초

1. 시간 복잡도

코딩 테스트
문제에서
시간제한은 통상
1~5초가량

[혹여 문제에 명시되어 있지 않은 경우
대략 5초 정도라고 생각하고 문제를 푸는
것이 합리적임]

N의 범위	설계
500	$O(N^3)$
2,000	$O(N^2)$
100,000	$O(N \log N)$
10,000,000	$O(N)$

유형 분석

- 구현: 33%
- BFS/DFS: 20.9%
- 그리디: 19.8%
- 정렬: 8.2%
- **다이나믹 프로그래밍: 8.2%**
 - 이진 탐색: 3.8%
 - 최단 경로: 3.3%
 - 그래프 이론: 2.7%

유형 분석



완전 탐색, 시뮬레이션, 구현,
DFS/BFS

완전 탐색, 구현, DFS/BFS,
시뮬레이션



구현, 이진 탐색, 자료구조
추천 시스템 개발

그리디, 구현, 자료구조,
시뮬레이션 개발



탐색, 구현, 문자열, 다이나믹
프로그래밍, 자료구조, 완전 탐색,
구현

탐색, 그리디, 다이나믹
프로그래밍, 구현, 탐색, 그리디,
구현, 문자열



DP

(Dynamic Programming)

동적 계획법

점화식

최적과 중복



1	1	2	3	5	8	13	21
---	---	---	---	---	---	----	----

$$F(n) = F(n-1) + F(n-2)$$

$F(1), f(2)=1$



DP

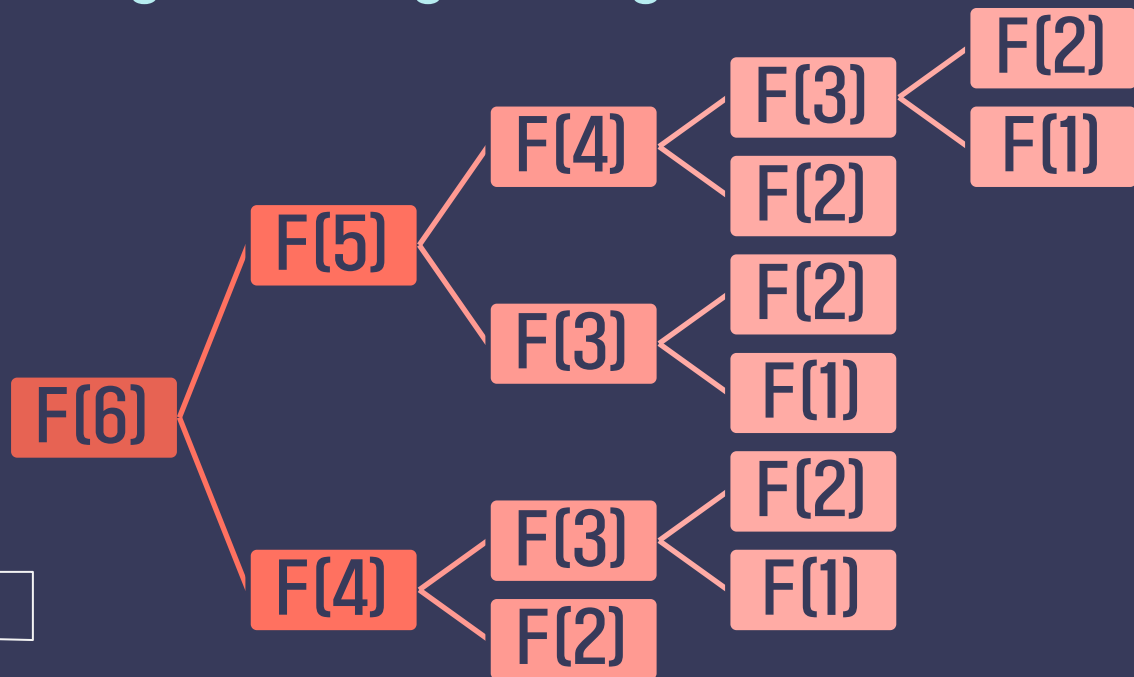
(Dynamic Programming)

동적 계획법

점화식

최적과 중복

$O(2^n)$

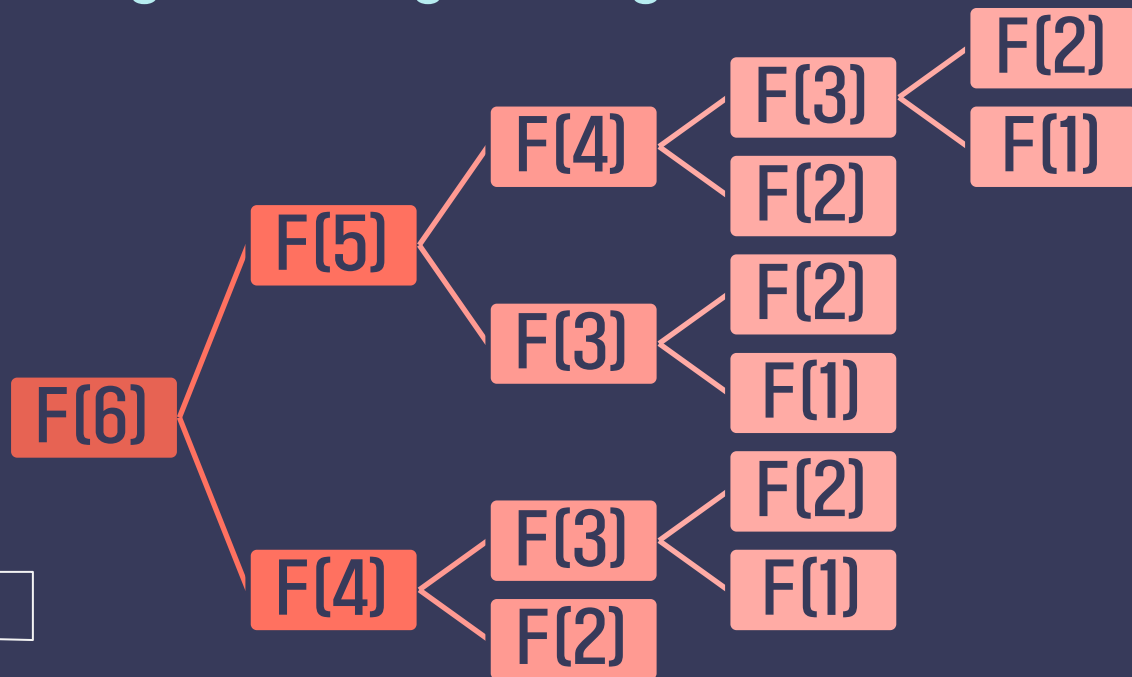


```
def fibo(x):  
    if x == 1 or x == 2:  
        return 1  
    return fibo(x - 1) + fibo(x - 2)  
  
print(fibo(4))
```

DP

(Dynamic Programming)

Memorization



F(3)	F(4)	F(5)	F(6)

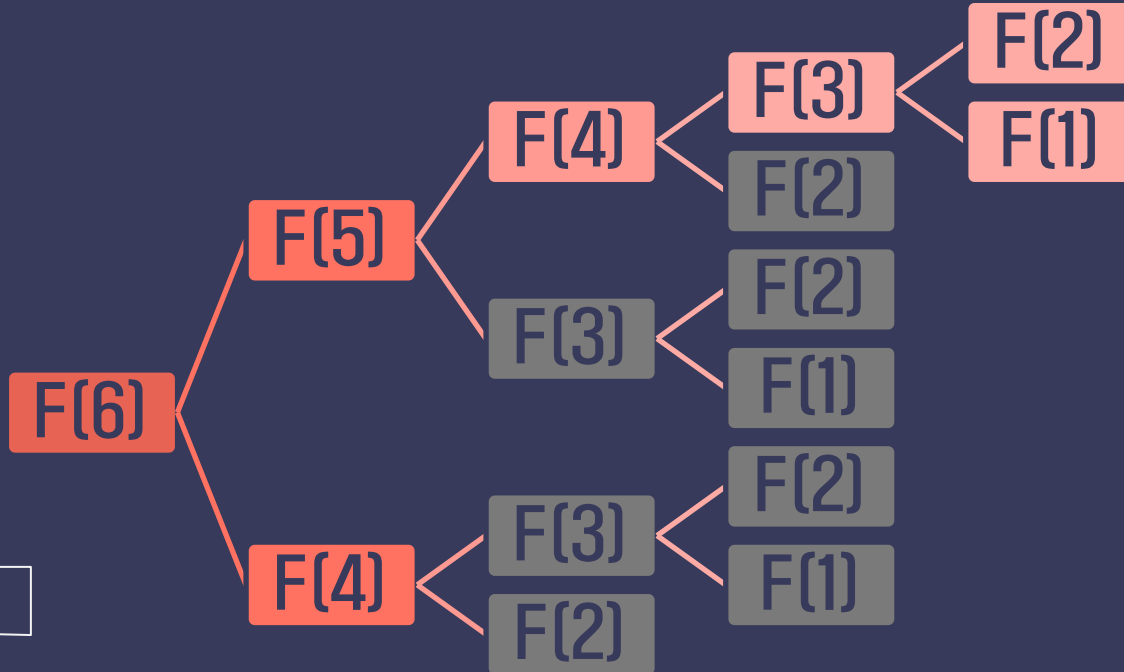
F(3)	F(4)	F(5)	F(6)
2	3		

F(3)	F(4)	F(5)	F(6)
2	3	5	8

DP

(Dynamic Programming)

Memorization



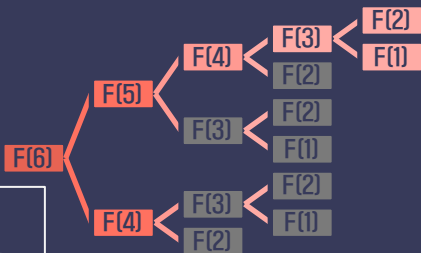
F(3)	F(4)	F(5)	F(6)

F(3)	F(4)	F(5)	F(6)
2	3		

F(3)	F(4)	F(5)	F(6)
2	3	5	8

DP (Dynamic Programming)

Memorization Top - Down



```
# 메모이제이션하기 위한 리스트 초기화
memoization = [0] * 100

# 피보나치 함수를 재귀함수로 구현 (Top-down DP)
def fibo(x):
    if x == 1 or x == 2:
        return 1
    # 이미 계산한 적 있으면 그대로 반환
    if memoization[x] != 0:
        return memoization[x]
    # 계산한 적 없으면 점화식에 따라 피보나치 결과 반환
    memoization[x] = fibo(x - 1) + fibo(x - 2)
    return memoization[x]

print(fibo(6))
```

$$O(2^n) \\ \downarrow \\ O(N)$$

DP

(Dynamic Programming)

Bottom-up

F(1)	F(2)	F(3)	F(4)	F(5)	F(6)	F(7)	F(8)
1	1	2	3	5	8	13	21



```
# 앞서 계산된 결과를 저장하기 위한 DP 테이블 초기화
dp = [0] * 100
dp[1] = 1
dp[2] = 1
n = 99

# 피보나치 수열 반복문으로 구현 (Bottom-Up DP)
for i in range(3, n + 1):
    dp[i] = dp[i - 1] + dp[i - 2]

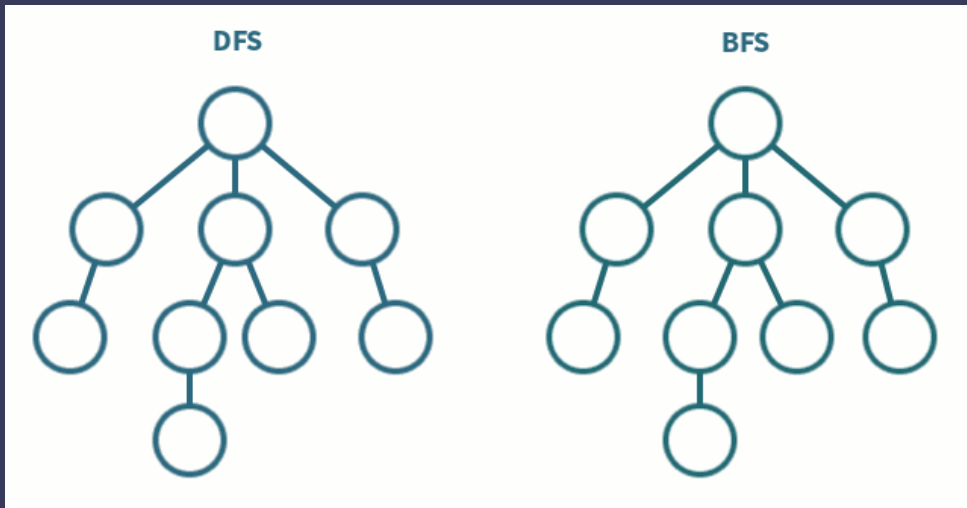
print(dp[n])
```

DFS

[깊이 우선 탐색]

경로를 특징을 저장하는 경우
검색 대상 그래프가 아주 큰 경우

DFS
Stack
재귀



BFS
Queue

[illegible]

© 2006 The Authors


```

int n, min; // 맵의 크기와 최소값을 나타내는 변수
int map[10][10]; // 맵을 나타내는 2차원 배열

void DFS(int x, int y, int l)
{
    // 도착 지점에 도착했을 경우
    if (x == n - 1 && y == n - 1)
    {
        // 현재 최소값이 l보다 크면, l이 작은 것이므로 l를 최소값으로 지정
        if (min > l) min = l;
        return;
    }
    map[y][x] = 0; // 방문했음을 표시하기 위해 0을 대입

    // 위로 이동할 수 있다면 이동!
    if (y > 0 && map[y - 1][x] != 0)
        DFS(x, y - 1, l + 1);
    // 왼쪽으로 이동할 수 있다면 이동!
    if (x > 0 && map[y][x - 1] != 0)
        DFS(x - 1, y, l + 1);
    // 아래로 이동할 수 있다면 이동!
    if (y < n - 1 && map[y + 1][x] != 0)
        DFS(x, y + 1, l + 1);
    // 오른쪽으로 이동할 수 있다면 이동!
    if (x < n - 1 && map[y][x + 1] != 0)
        DFS(x + 1, y, l + 1);

    map[y][x] = 1; // 지나간 자리를 원상태로 되돌리기 위해 1을 대입
}

```

1	2	3	4	5
				6
11	10	9	8	7
12		10		
13	14	15/ 11	16/ 12	17/ 13



```
void BFS(int x, int y) {
```

```
    //출발지 좌표 X,Y를 큐에 넣어준다.
```

```
    q.push(make_pair(y, x));
```

```
    //방문했으니 출발지 1로, 결과는 시작 1로
```

```
    visited[y][x] = 1;
```

```
    result[y][x] = 1;
```

```
    //2. 반복문
```

```
    while (!q.empty()) {
```

```
        //큐 맨앞 좌표 저장후 큐에서 꺼냄
```

```
        y = q.front().first;
```

```
        x = q.front().second;
```

```
        q.pop();
```

```
        //위,아래,알,뒤
```

```
        for (int i = 0; i < 4; i++) {
```

```
            //다음 경로 지정
```

```
            int ny = y + dy[i];
```

```
            int nx = x + dx[i];
```

```
            //만약 다음 경로가 미로 밖으로 안나가면
```

```
            if (nx >= 0 && nx < n && ny >= 0 && ny < n) {
```

```
                //만약 막힌길도 아니고 방문한적도 없다면
```

```
                if (mat[nx][ny] == 1 && visited[nx][ny] == 0) {
```

```
                    q.push(make_pair(nx, ny));
```

```
                    //방문처리
```

```
                    visited[nx][ny] = 1;
```

```
                    //전도도에 +1
```

```
                    result[nx][ny] += result[x][y] + 1;
```

```
                }
```

```
            }
```

```
        }
```

```
    }
```

```
}
```

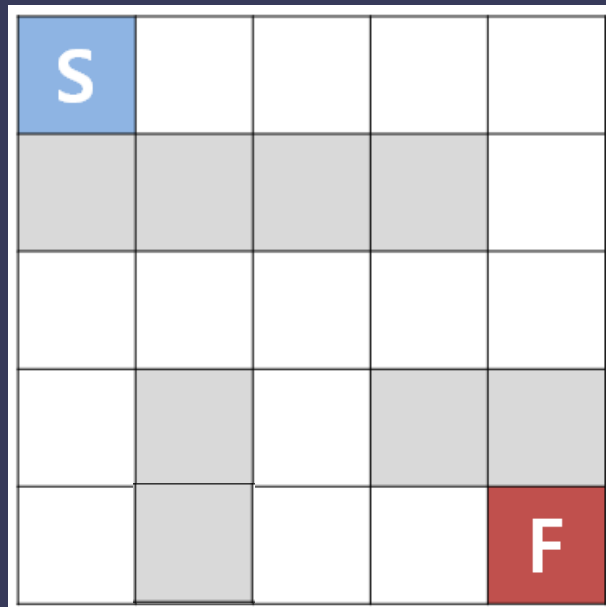
X	0	1	2	2	3	2	4	1	5	0	5	0	5	0	4	1	4	2	4	2	4	3	3	5
Y	0	0	0	1	0	2	0	2	0	2	1	3	2	4	2	4	3	4	4	5	5	5	5	5
L	1	2	3	4	4	5	5	6	6	7	7	8	8	9	9	10	10	11	11	12	12	13	13	13

1	2	3	4	5	6
		4			7
7	6	5		9	8
8				10	
9	10	11		11	
		12	13	12/ 14	13/ 15

DFS vs BFS

DFS

1. 모든 노드를 방문하고자 하는 경우에 이 방법을 선택함
2. 깊이 우선 탐색(DFS)이 너비 우선 탐색(BFS)보다 좀 더 간단함
3. 검색 속도 자체는 너비 우선 탐색(BFS)에 비해서 느림



DFS vs BFS

그래프의 모든 정점을 방문하는 문제

DFS

경로의 특징을 저장해줘야 하는 문제

예를 들면 각 정점에 숫자가 적혀 있고 a부터 b까지 가는 경로를 구하는데 경로에 같은 숫자가 있으면 안 된다는 문제 등, 각각의

경로마다 특징을 저장해줘야 할 때

(BFS는 경로의 특징을 가지지 못함)

0	1	1	0	1	0	0
0	1	1	0	1	0	1
1	1	1	0	1	0	1
0	0	0	0	1	1	1
0	1	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	0	0	0

<그림 1>

0	1	1	0	2	0	0
0	1	1	0	2	0	2
1	1	1	0	2	0	2
0	0	0	0	2	2	2
0	3	0	0	0	0	0
0	3	3	3	3	3	0
0	3	3	3	0	0	0

<그림 2>

탐색 시 먼저 찾아지는 해답이 곧 최단거리기 때문입니다.