SAT Solver

N-Queens

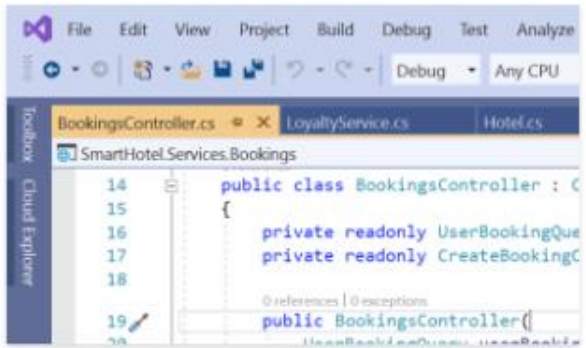
Eunok Paek @ Hanyang univ.

설치

Visual Studio

모든 개발자를 위한 최상의 도구

Visual Studio

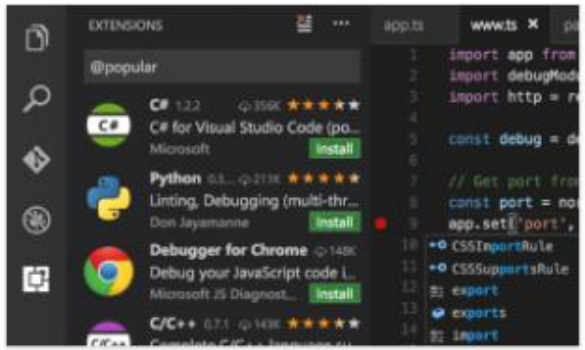


코딩, 디버그, 테스트 및 모든 플랫폼에 배포할 수 있는 완전한 기능을 갖춘 IDE

Visual Studio 다운로드

- Community 2019
- Professional 2019
- Enterprise 2019

Visual Studio Code



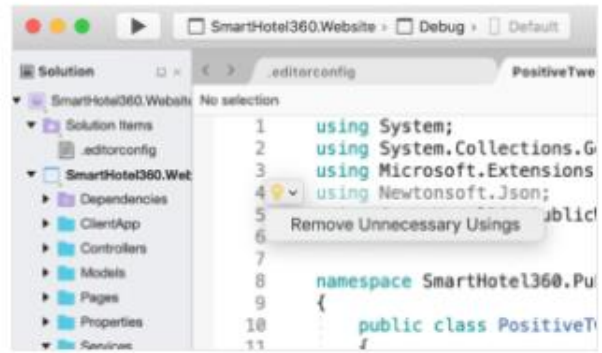
모든 OS 편집 및 디버깅

Visual Studio Code를 사용하면 다음에 동의하는 것입니다. 라이선스 및 개인정보처리방침)

Download Visual Studio Code

자세히 보기 >

Visual Studio for Mac



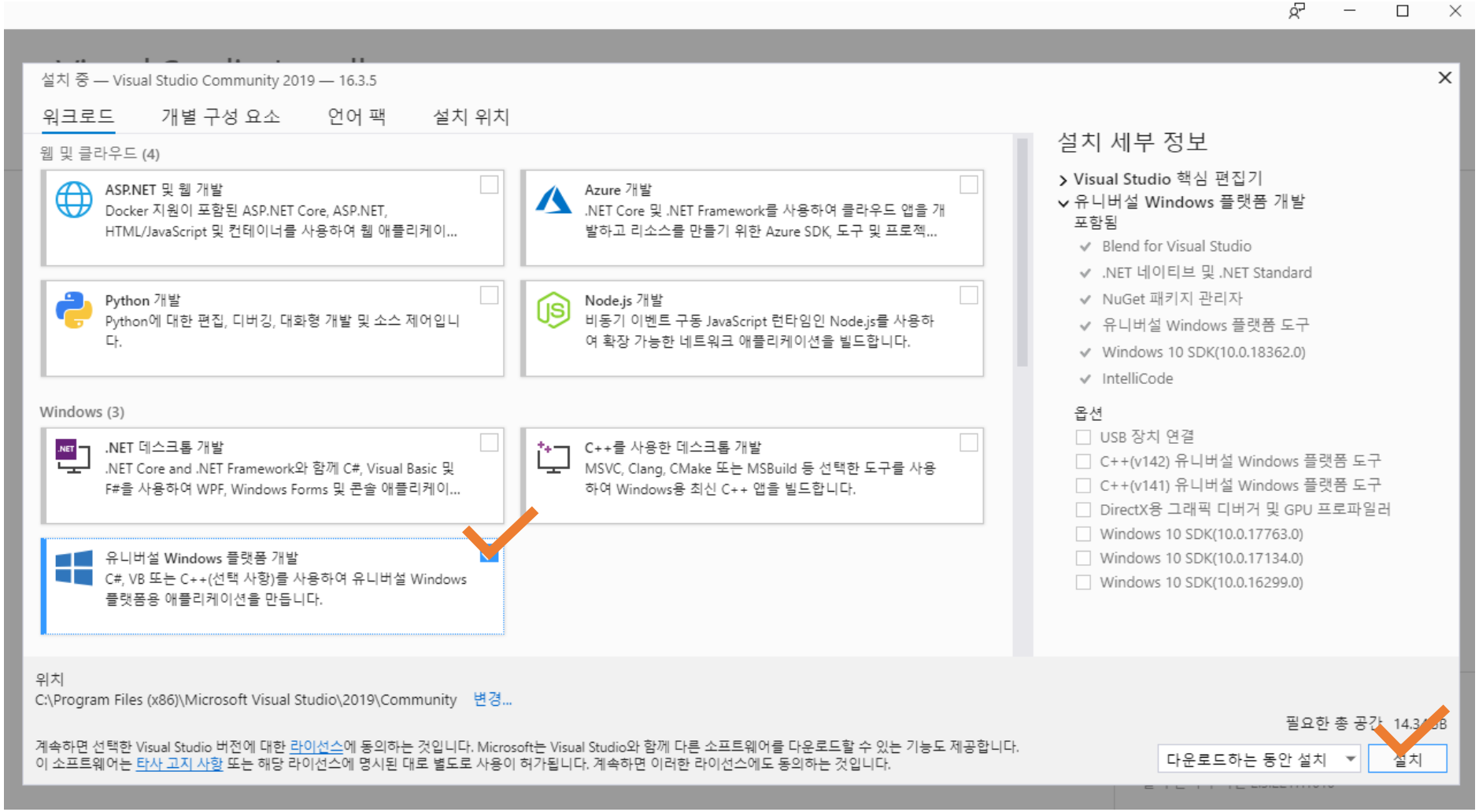
.NET을 사용하여 iOS, Android 및 웹용 앱 및 게임 개발

라이선스 활성화에 대한 자세한 정보

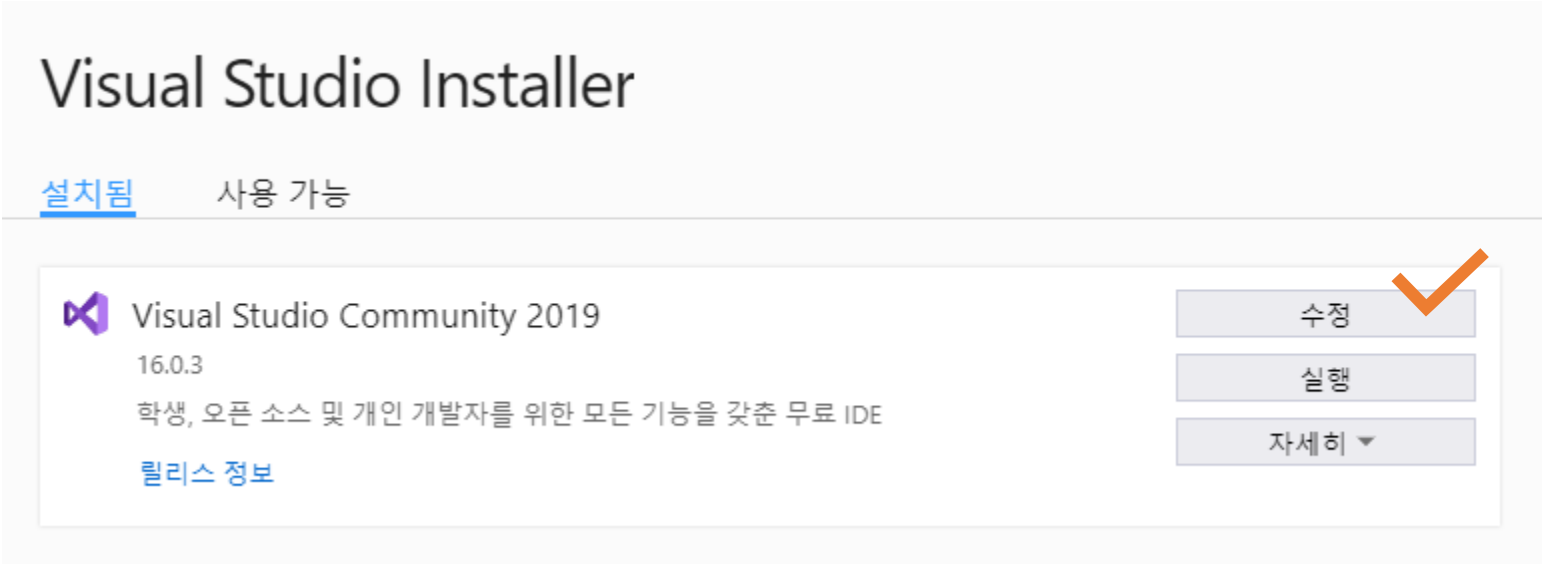
Visual Studio for Mac 다운로드

자세히 보기 >

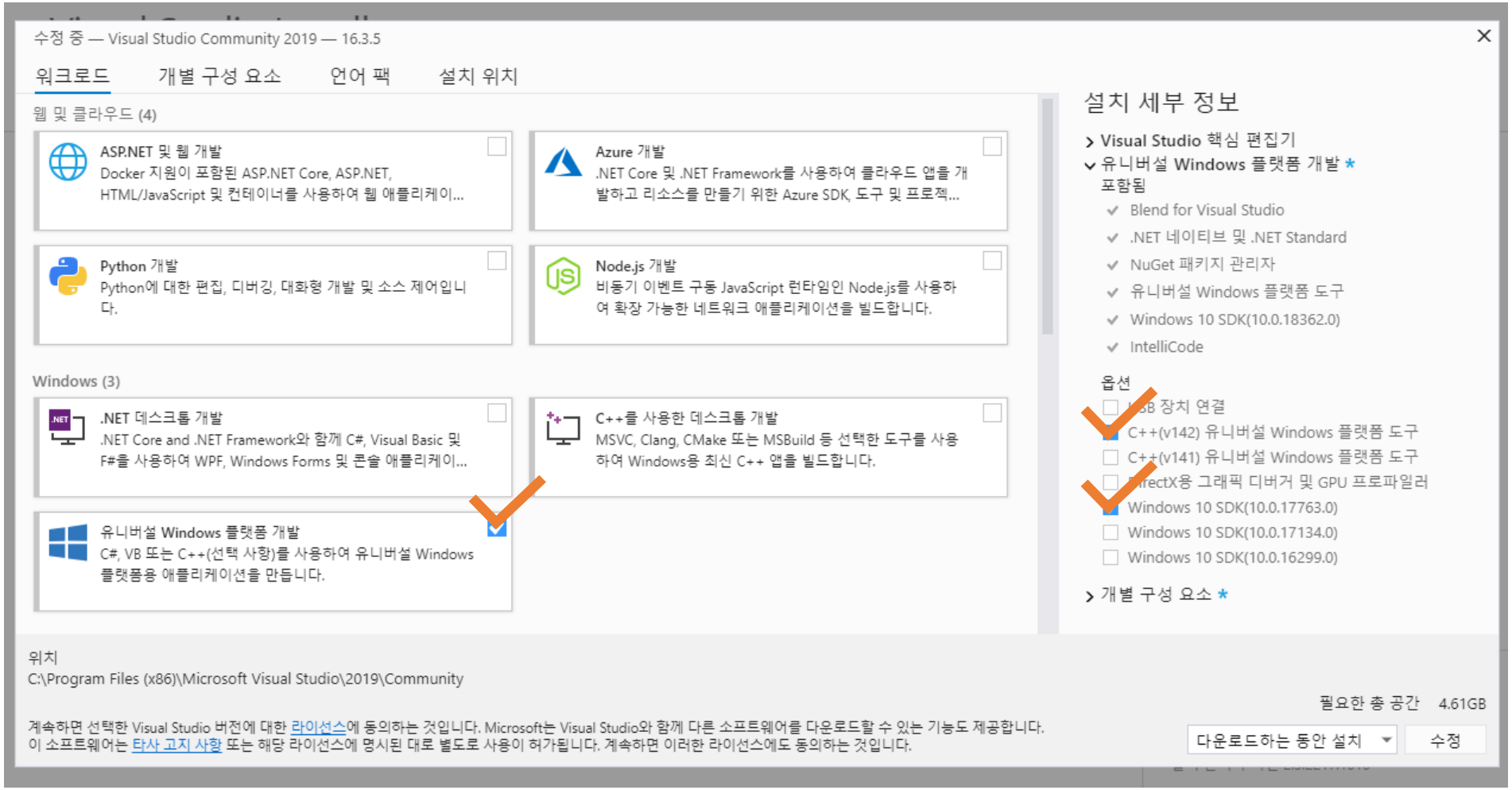
https://visualstudio.microsoft.com/ko/



Windows 항목에 “유니버설 Windows 플랫폼 개발” 을 선택하고 설치를 클릭.

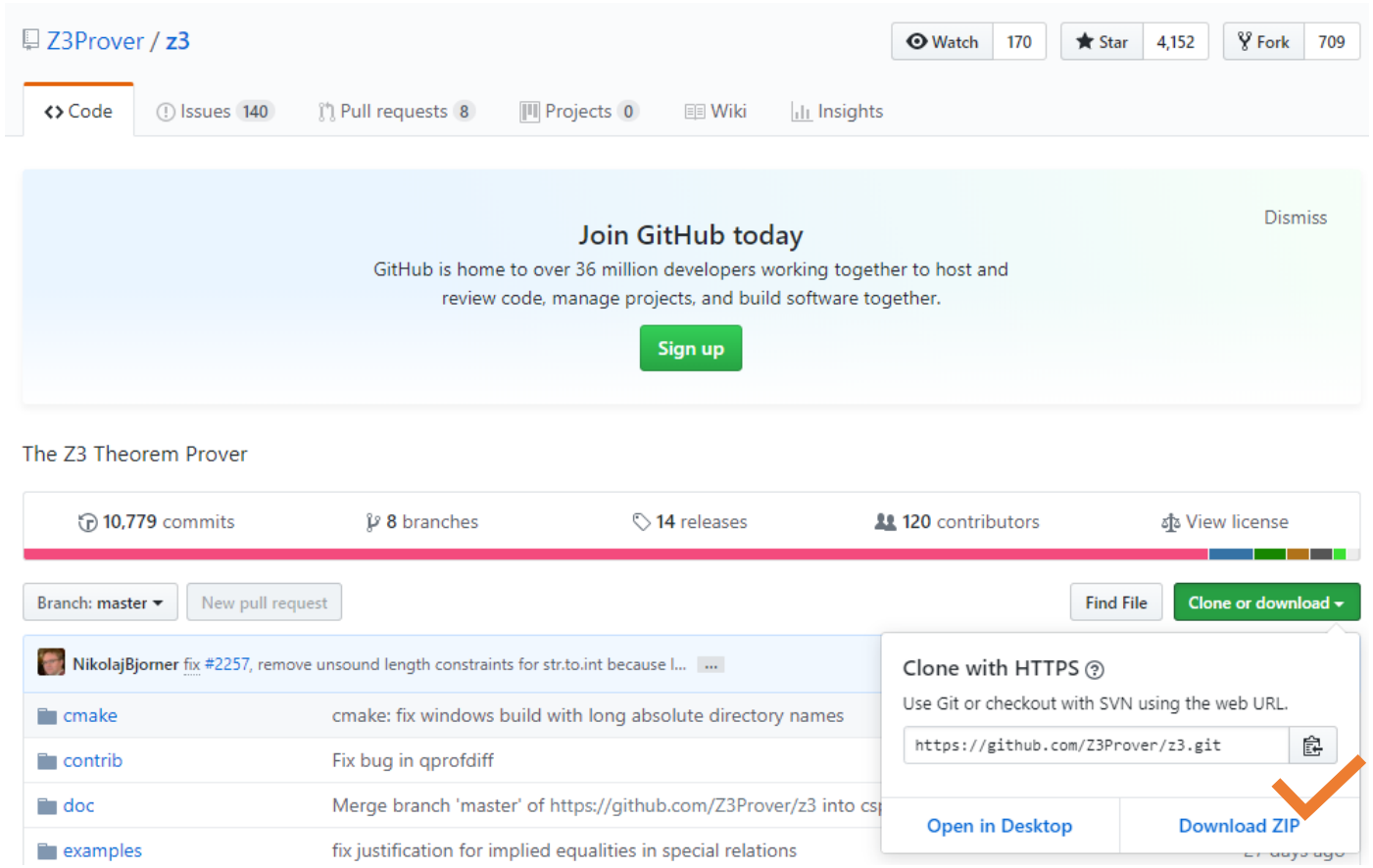


Visual Studio Installer를 실행시키고, Community 2019 버전을 설치.
설치가 완료되면, 수정 버튼 클릭.



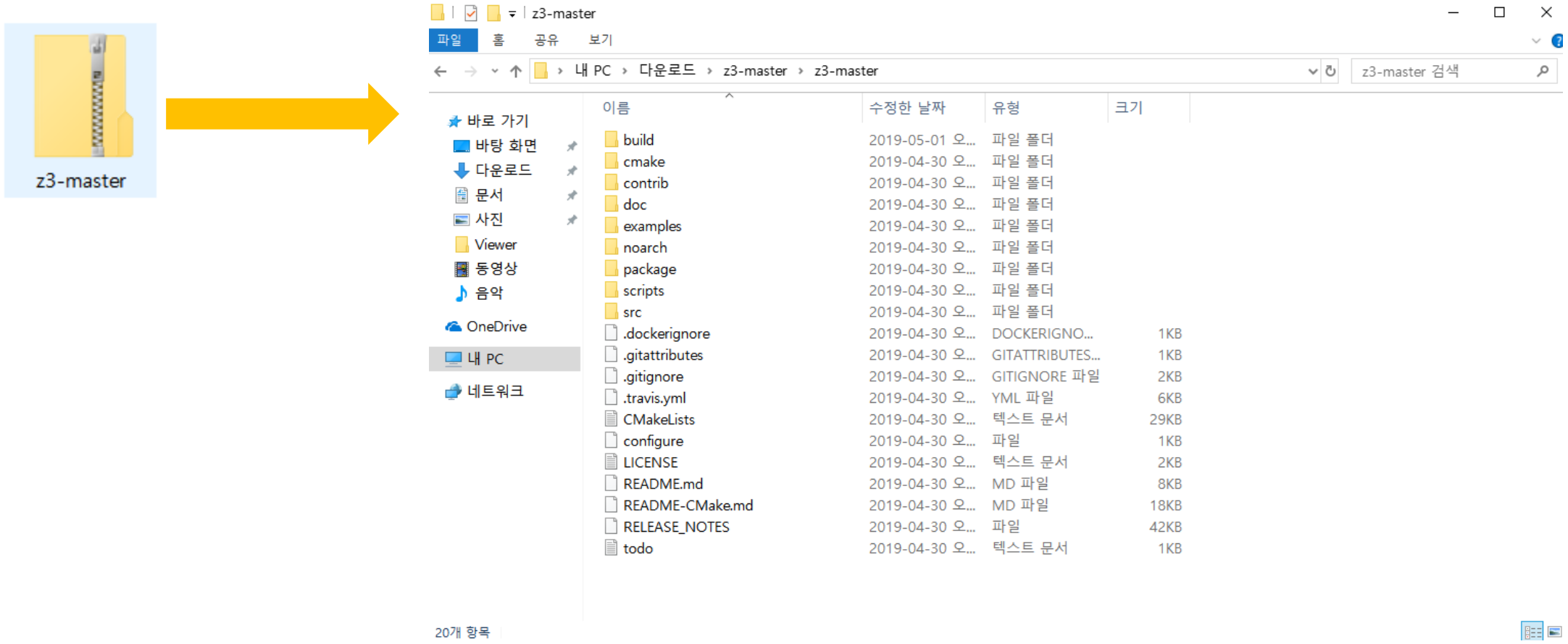
워크로드를 클릭하고, 위의 세 항목을 선택하고 수정을 클릭.

개별 구성 요소를 클릭하고, 위의 세 항목을 선택하고 수정을 클릭.

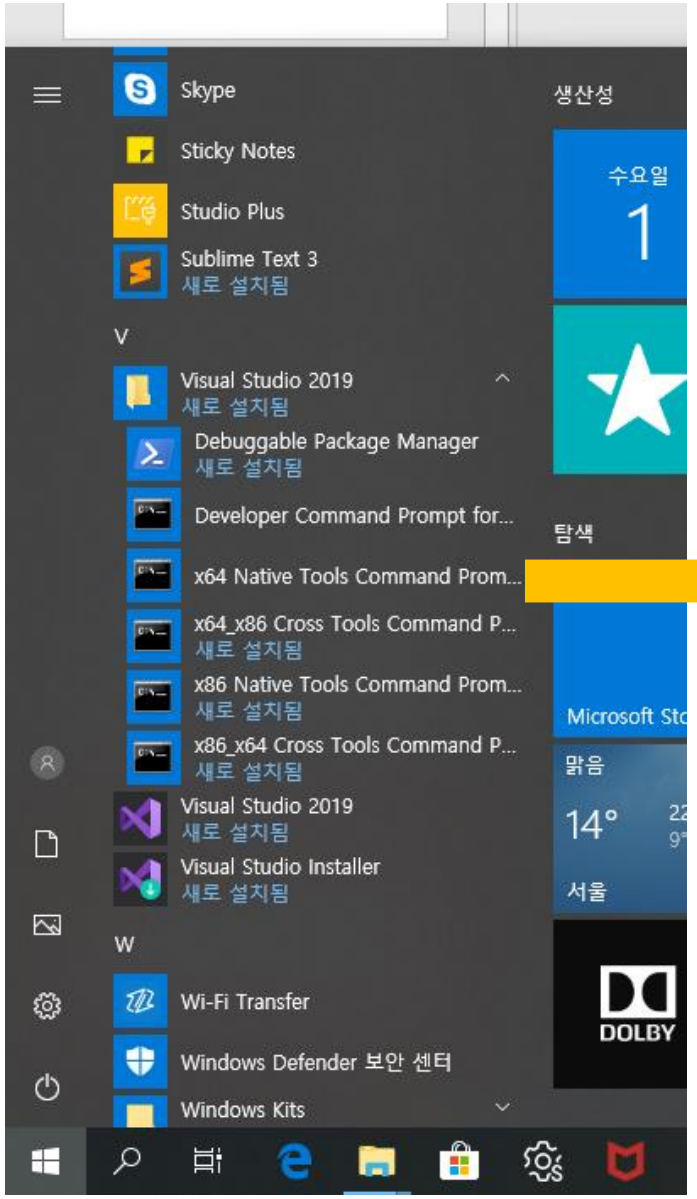


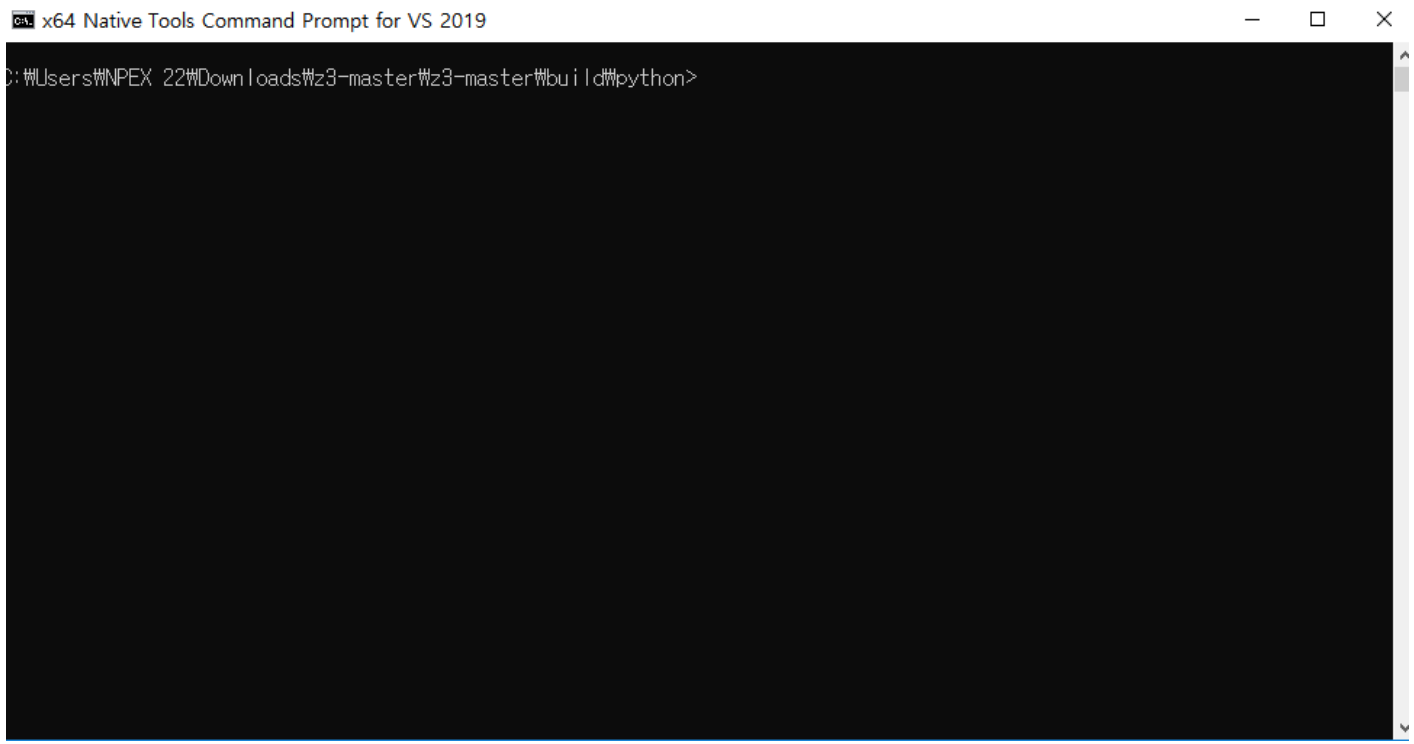
https://github.com/Z3Prover/z3

해당 사이트에 접속해서, “Clone or download”를 클릭 후, Download ZIP 클릭.

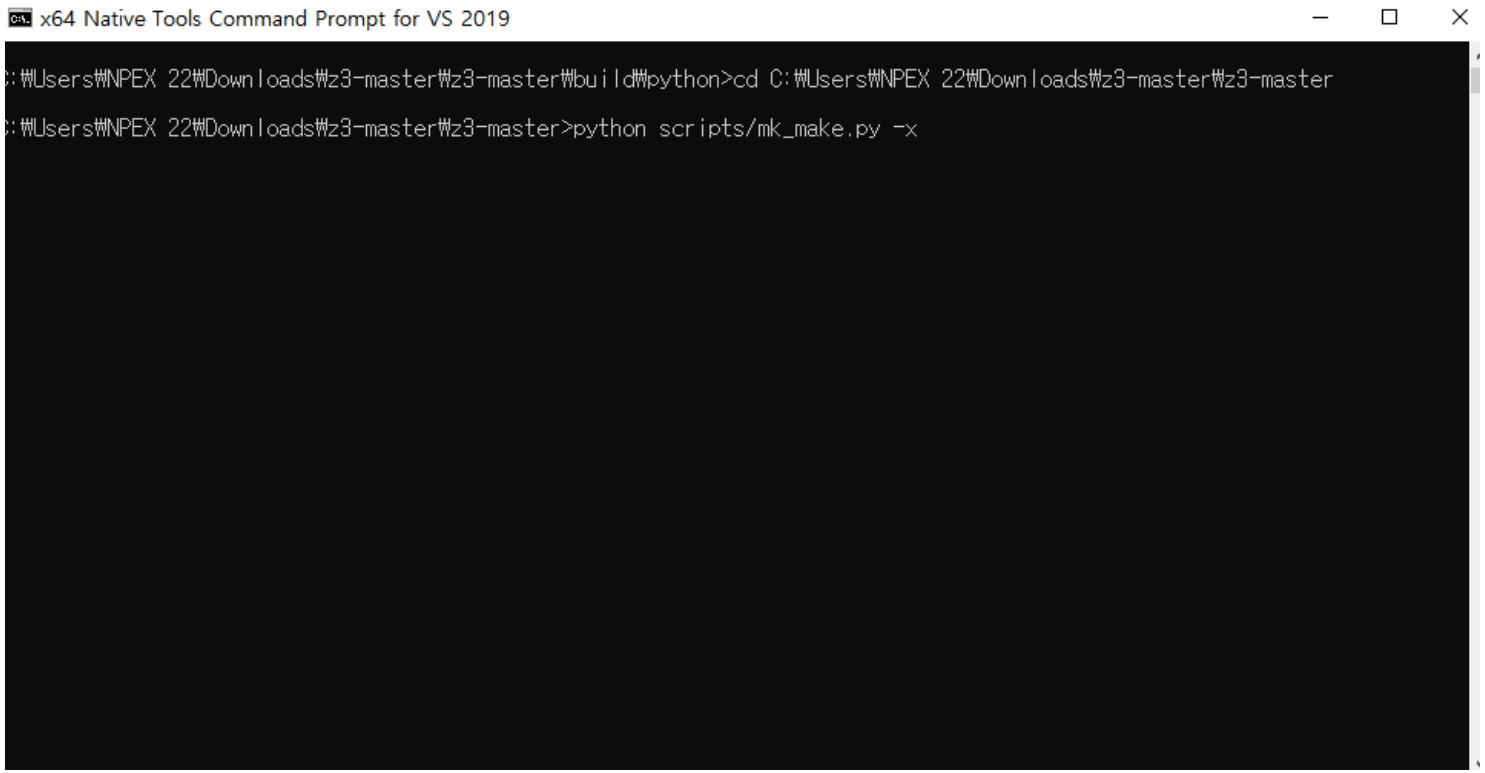


압축을 풀면 다음과 같이 보이게 됨.

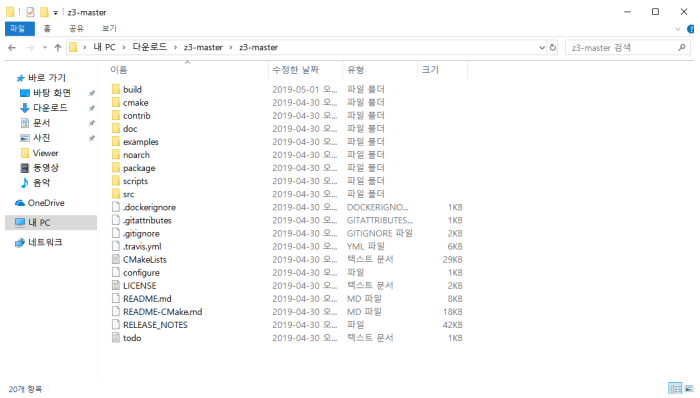




Visual studio 설치가 완료되면, 시작 프로그램에서 “X64 Native Tools Command Prompt for VS 2019” 을 실행.

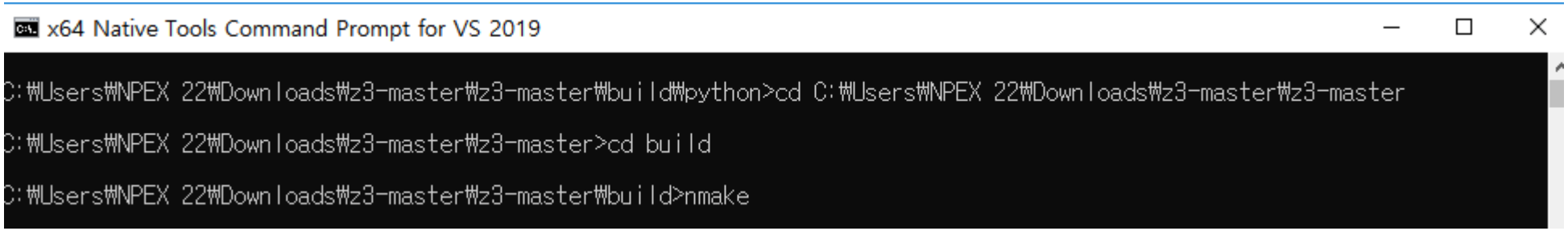


z3-master 폴더에 들어가서, "**python scripts/mk_make.py –x**" 를 입력.

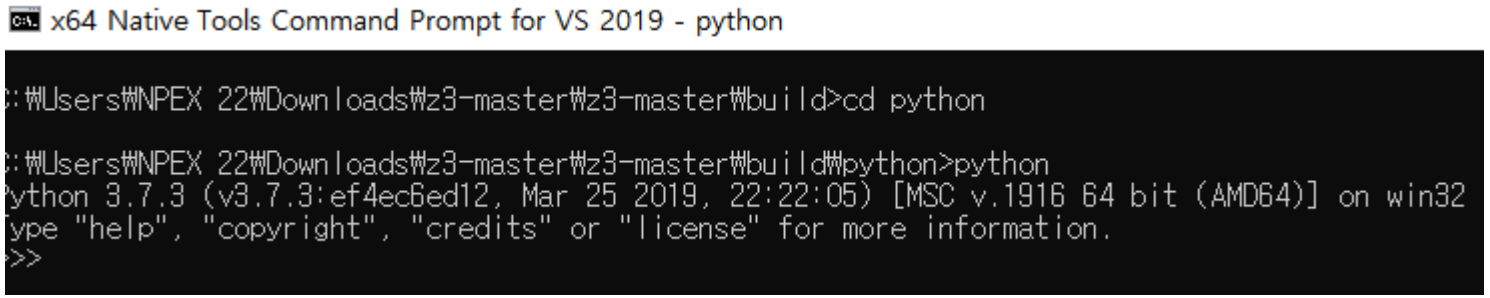


-x 옵션을 반드시 넣어서 입력할 것.

위 명령어가 안될 시: py scripts/mk_make.py -x



z3-master 폴더에서 build 폴더로 이동하고, “nmake”를 입력.



nmake가 완료되면, build 폴더에서 python 폴더로 이동. 해당 위치에서 “python”을 입력.
* build₩python 폴더에서 소스코드를 넣고 테스트할 것.

간단한 명령어

z3 lib 불러오기 및 간단한 변수 선언.

```
from z3 import *
x = Int('x')
X = [Int("x_%s" % (i)) for i in range(4)]
```

다중 배열을 표현하는 변수 선언.

```
X2 = [ [Int("x_%s_%s" % (i,j)) for i in range(4)] for j in range(8)]
```

And Operation

```
andOp = And(x == 3, x == y)
```

Or Operation

```
orOp = Or(x == 3, x == y)
```

Implies Operation

```
impOp = Implies( x == 3, x != y )
```

모델을 생성하고, SAT 판단.

```
sol = Solver()
sol.add(andOp)
sol.check()
```

조건 추가

SAT 판단

```
sol.reset()
```

조건 초기화

생성된 모델 확인

```
model = sol.model()
model
```

sol.check()를 먼저 확인하고, sol.model()을 불러야 함.

출력 => `[y = 3, x = 3]`

다중 operation: Or

```
orOperation = [ Or(X[i] == X[j]) for i in range(4) for j in range(4) ]

s = Solver()
s.add(orOperation)
s.check()
```

다중 operation: Implies

```
    impOperation = [ Implies( i != j, X[i] == 2*X[j]) for i in range(2) for j in range(2) ]
    sol = Solver()
    sol.add(impOperation)
    sol.check()
sat
    sol
[Implies(False, x_0 == 2*x_0),
 Implies(True, x_0 == 2*x_1),
 Implies(True, x_1 == 2*x_0),
 Implies(False, x_1 == 2*x_1)]
```

변수의 합에 대한 operation

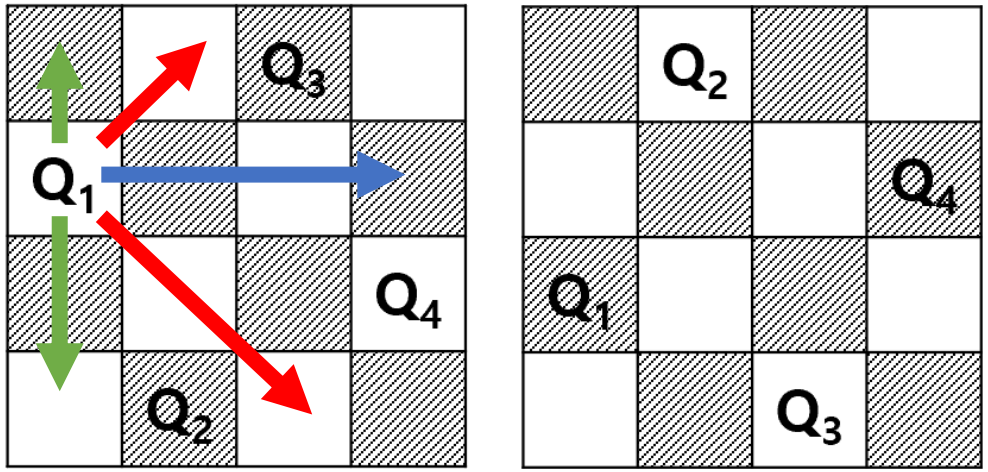
```
    X2 = [[Int("x_%s_%s" % (row, col)) for row in range(4)] for col in range(4) ]
    integrator = [ Sum(X2[i]) == 1 for i in range(4) ]
    sol.reset()
    sol.add(integrator)
    sol.check()
sat
    sol.model()
[x_1_3 = 0,
 x_2_3 = 0,
 x_3_3 = 0,
 x_0_3 = 1,
 x_1_2 = 0,
 x_2_2 = 0,
 x_3_2 = 0,
 x_0_2 = 1,
 x_1_1 = 0,
 x_2_1 = 0,
 x_3_1 = 0,
 x_0_1 = 1,
 x_1_0 = 0,
 x_2_0 = 0,
 x_3_0 = 0,
 x_0_0 = 1]
```

그 외... 수 많은 기능들 (API)

http://z3prover.github.io/api/html/namespacez3py.html

4 – Queens?

4-Queens: Naïve Insight



Variables: All possible positions in the chess board (4x4).
Domain: {0 (No Q), 1 (Yes Q)}
Constraints: {Green, Red, Blue}

How to write in code?
Variables (Symbols): 다중 배열 변수.
Constraints (Formulas): 각 변수에 할당될 수 있는 값. Green, Red, Blue에 대한 로직을 operation으로 표현.
Assignments (Models): Constraints를 모두 만족시키는 상태.

N – Queens?

Naïve Algorithm https://stackoverflow.com/questions/48031462/z3-solve-the-eight-queens-puzzle

```
from z3 import *
import time

# Number of Queens
print("N: ")
N = int(input())

start = time.time()
# Variables
X = [[Int("x_%s_%s" % (row, col)) for row in range(N)] for col in range(N) ]

# Constraints
domain = [Or(X[row][col] == 0, X[row][col] == 1) for row in range(N) for col in range(N) ]

rowConst = [Sum(X[row]) == 1 for row in range(N)]

colConst = [Sum([X[row][col] for row in range(N)]) == 1 for col in range(N) ]

digConst = [Implies(And(X[i][j] == 1, X[k][h] == 1,
            i != k, j != h), abs(k - i) != abs(j - h))
            for i in range(N) for j in range(N)
            for k in range(N) for h in range(N)]

eight_queens_c = domain + rowConst + colConst + digConst

s = Solver()
s.add(eight_queens_c)

if s.check() == sat:
    m = s.model()
    r = [[m.evaluate(X[i][j]) for j in range(N)] for i in range(N)]
    print_matrix(r)

print("elapsed time: ", time.time() - start, " sec")
```

Naïve Algorithm https://stackoverflow.com/questions/48031462/z3-solve-the-eight-queens-puzzle

```
from z3 import *
import time

# Number of Queens
print("N: ")
N = int(input())

start = time.time()
# Variables
X = [[Int("x_%s_%s" % (row, col)) for row in range(N)] for col in range(N) ]

# Constraints
domain = [Or(X[row][col] == 0, X[row][col] == 1) for row in range(N) for col in range(N) ]

rowConst = [Sum(X[row]) == 1 for row in range(N)]

colConst = [Sum([X[row][col] for row in range(N)]) == 1 for col in range(N) ]

digConst = [Implies(And(X[i][j] == 1, X[k][h] == 1,
                i != k, j != h), abs(k - i) != abs(j - h))
                for i in range(N) for j in range(N)
                for k in range(N) for h in range(N)]

eight_queens_c = domain + rowConst + colConst + digConst

s = Solver()
s.add(eight_queens_c)

if s.check() == sat:
    m = s.model()
    r = [[m.evaluate(X[i][j]) for j in range(N)] for i in range(N)]
    print_matrix(r)

print("elapsed time: ", time.time() - start, " sec")
```

N	Naive	Opt
4	0.104	0.025
8	1.125	0.065
12	5.087	0.084
16	34.71	0.264
20	284.686	0.639