SAL
Software Analysis Lab.

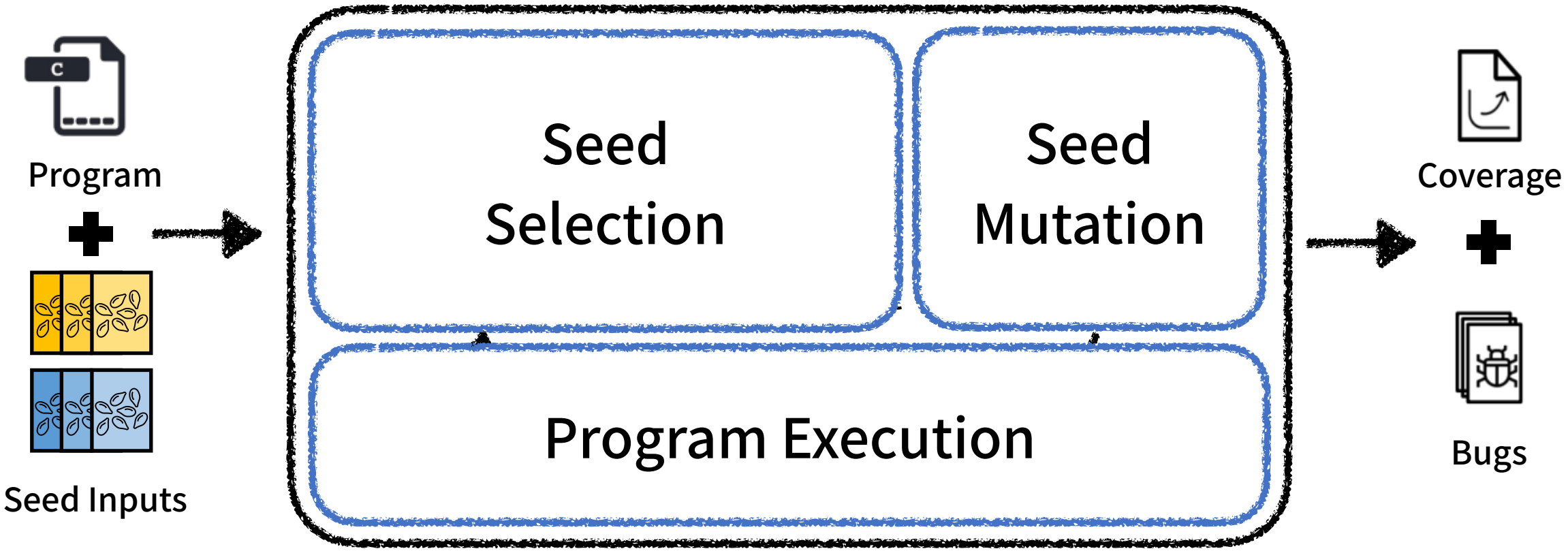
Learning Seed-Adaptive Mutation Strategies for Greybox Fuzzing


이 명 호[1] 차 수 영[2] 오 학 주[1]

[1]고려대학교

[2]성균관 대학교

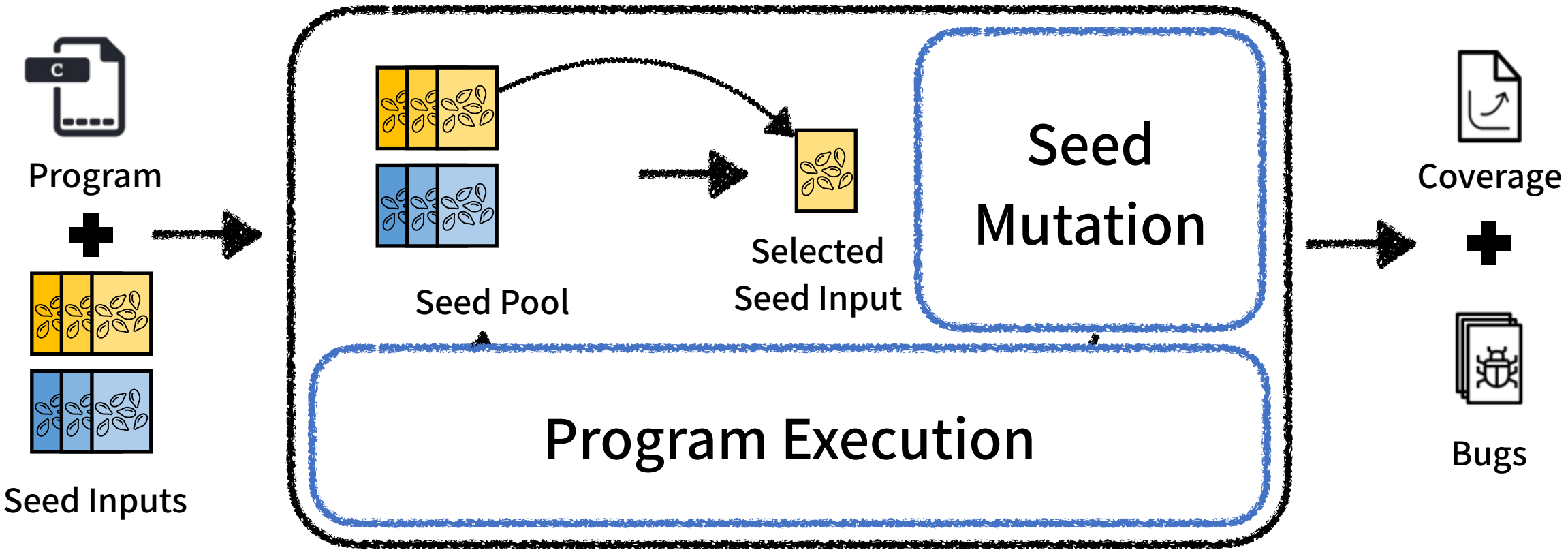
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

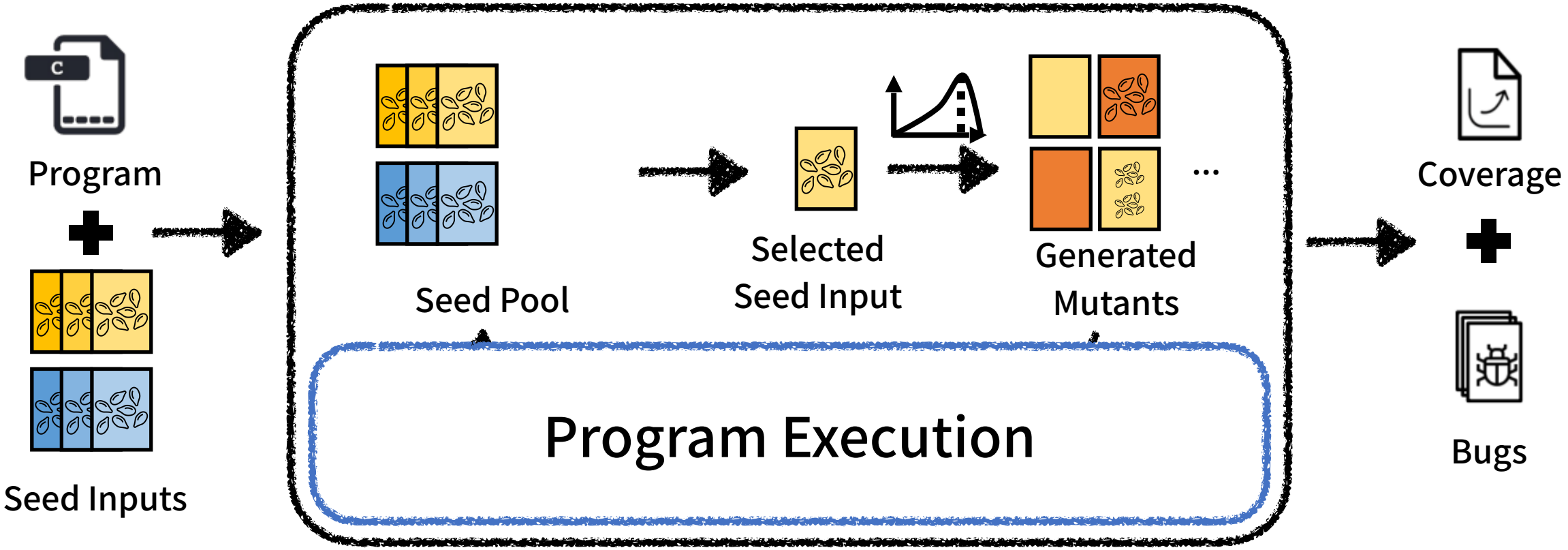
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

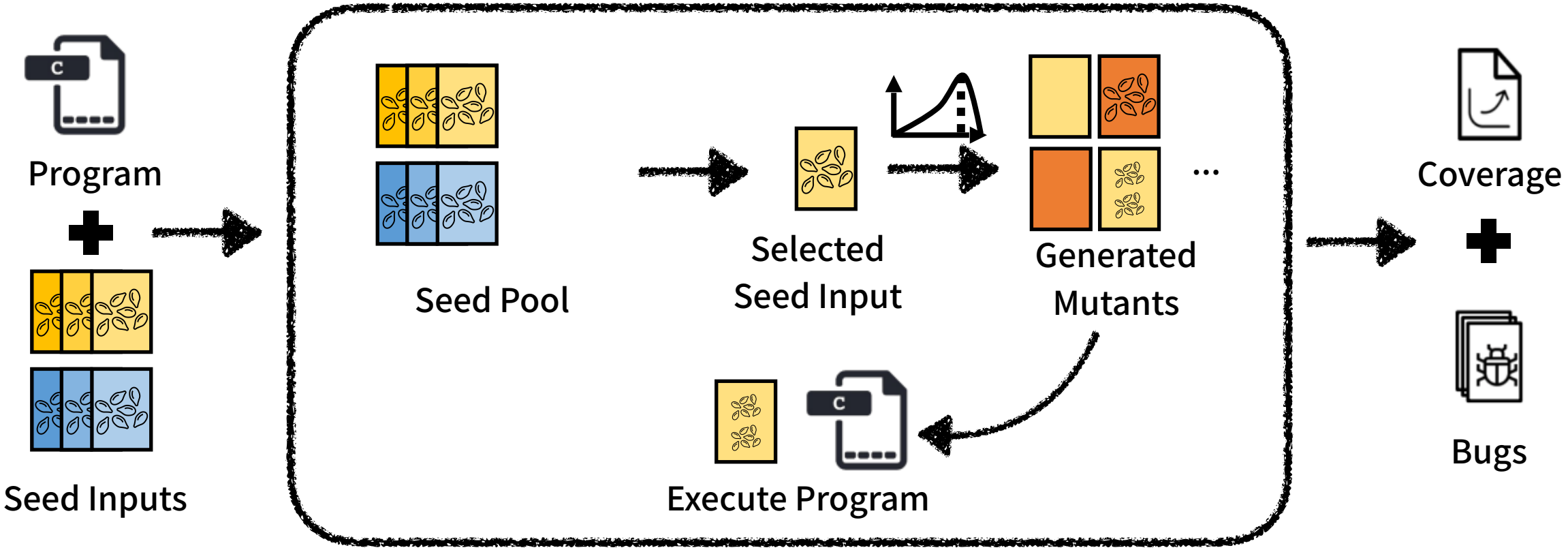
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

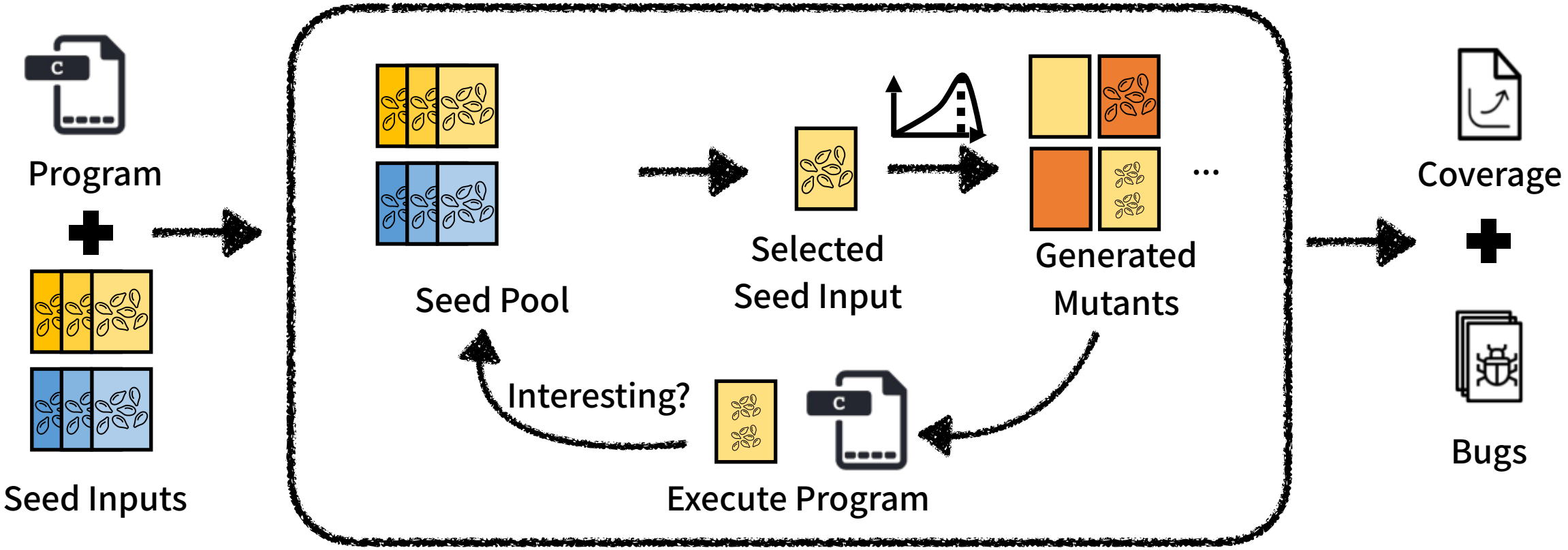
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

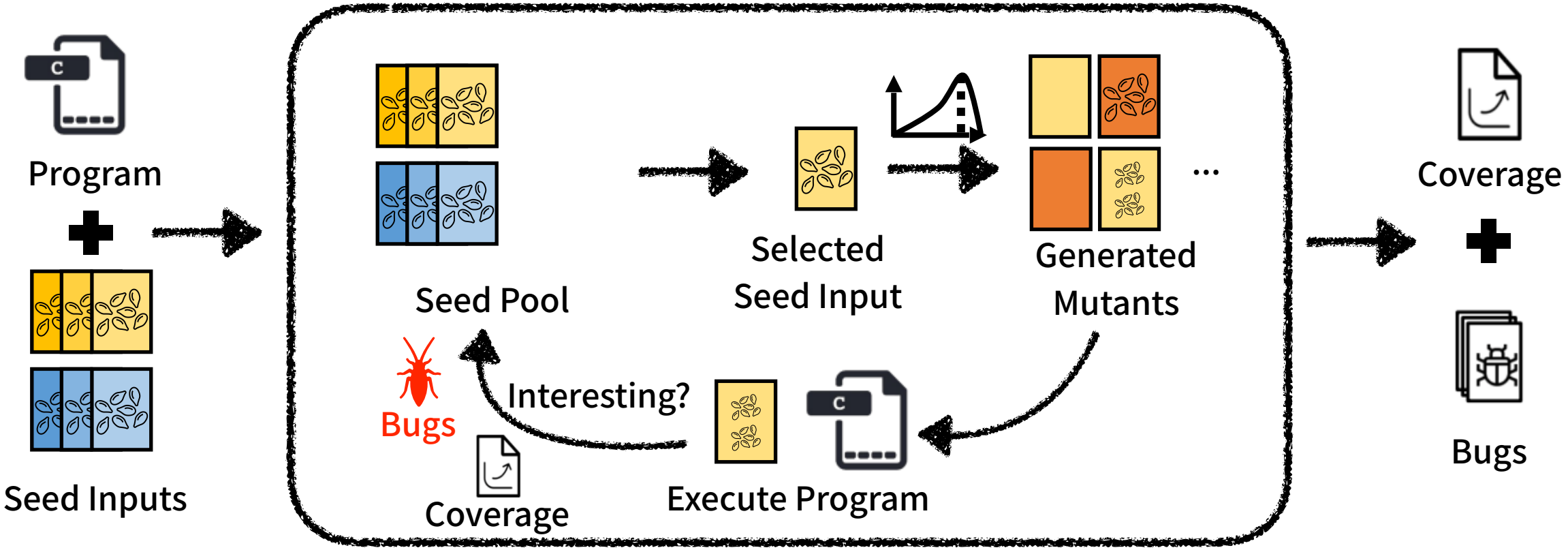
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

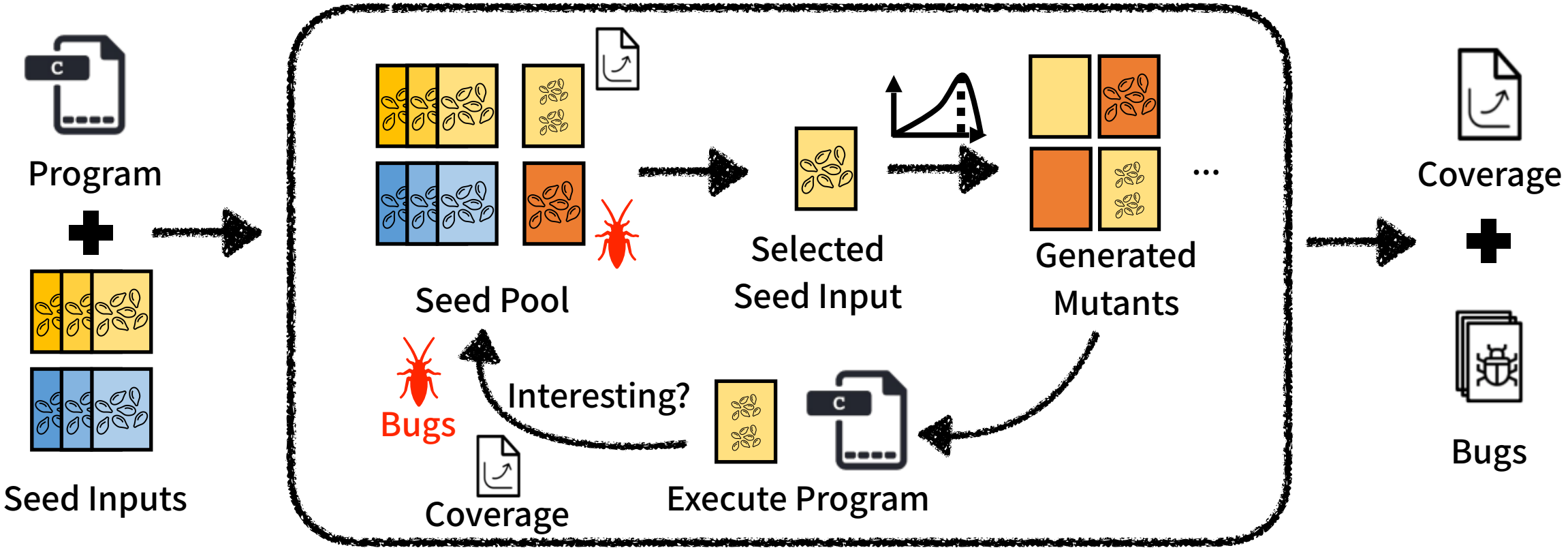
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

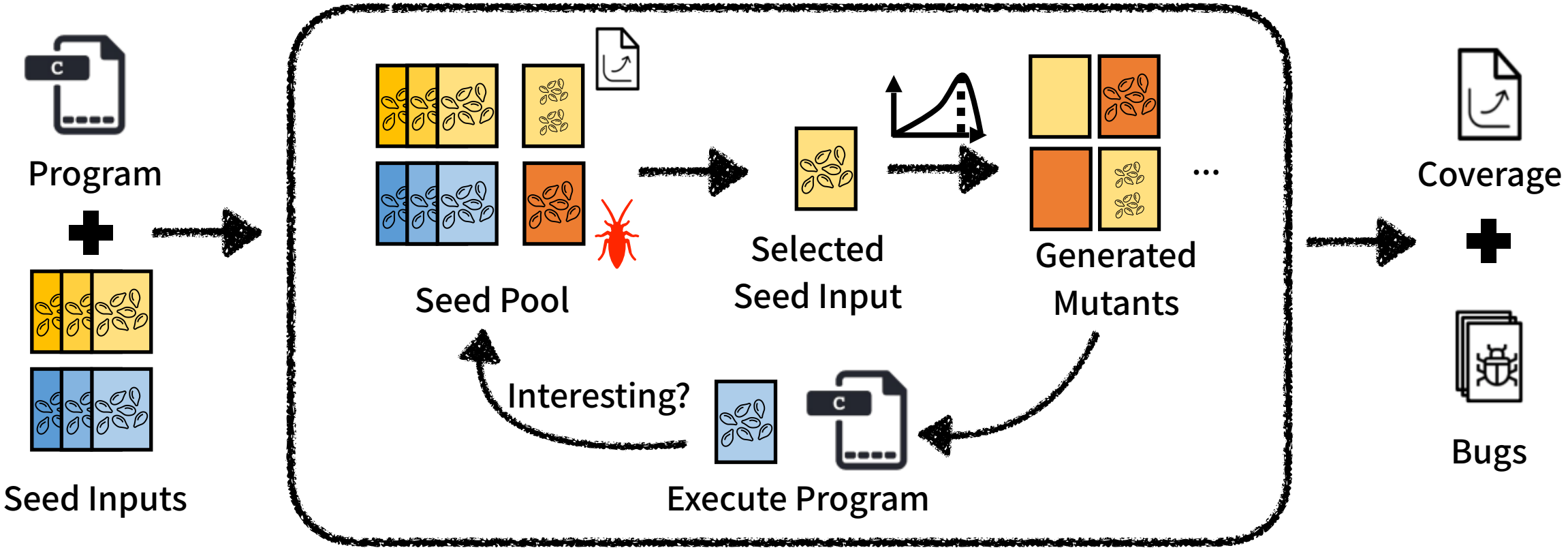
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

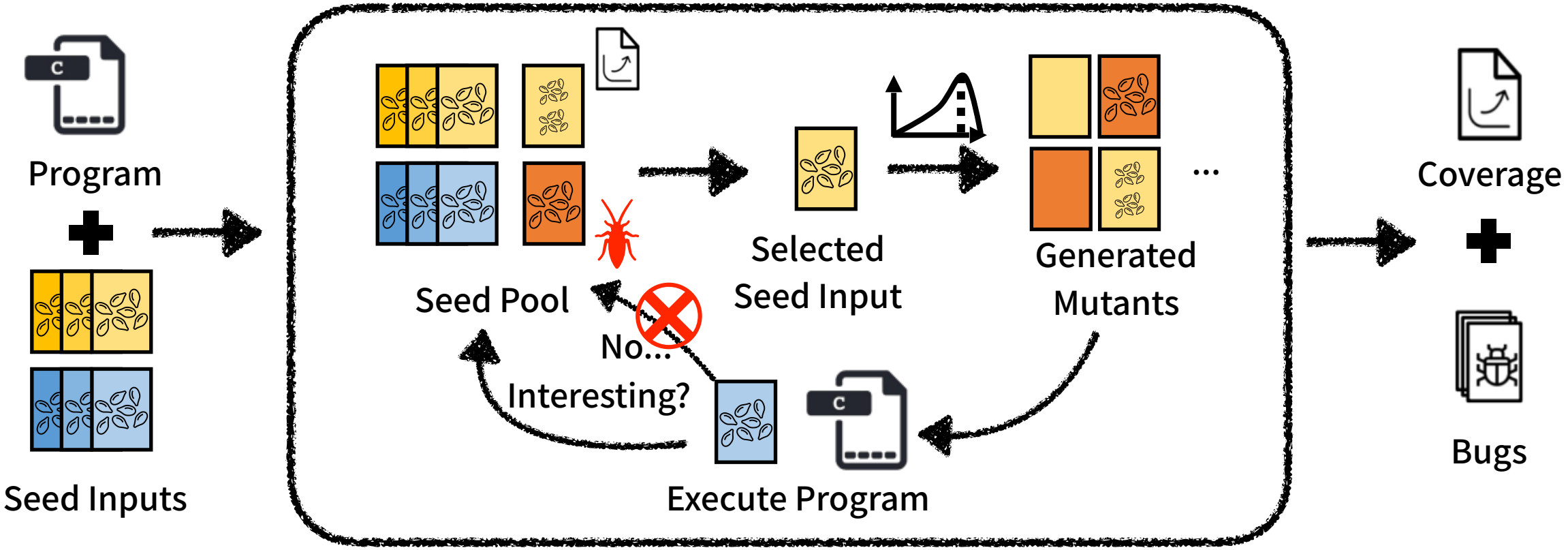
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

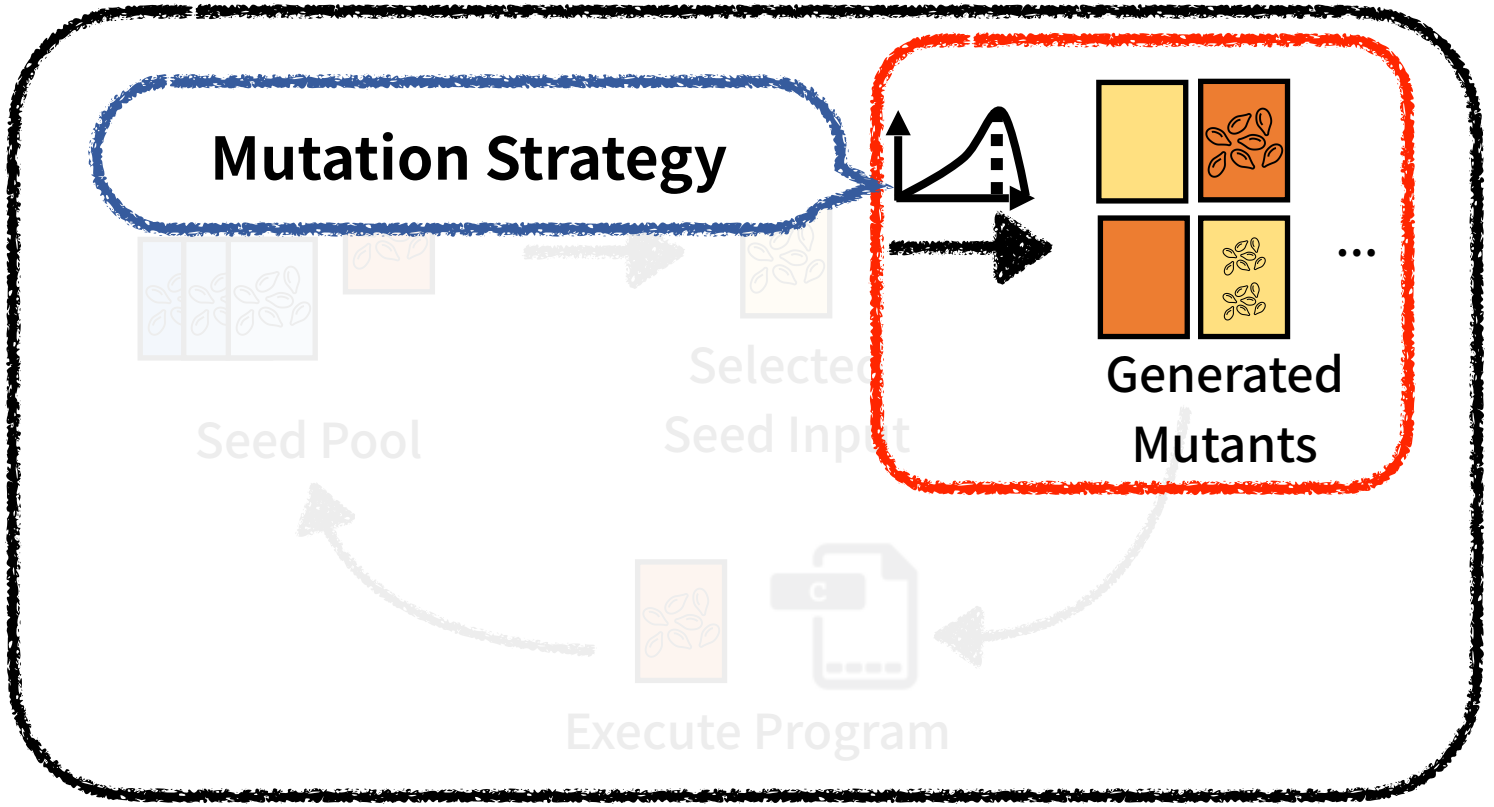
Mutation-based Greybox Fuzzer



소프트웨어 테스팅에서 가장 많이 쓰이는 테스팅 기법 중 하나

- 쓰기 쉬운데 효과적
- 세 가지 스탭 활용 (**Seed Selection, Seed Mutation, Program Execution**)

Mutation-based Greybox Fuzzer



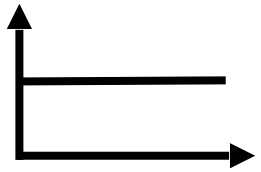
Mutation Strategy:

- Seed Input을 **어떻게** 바꿀 것인가에 대한 방법론

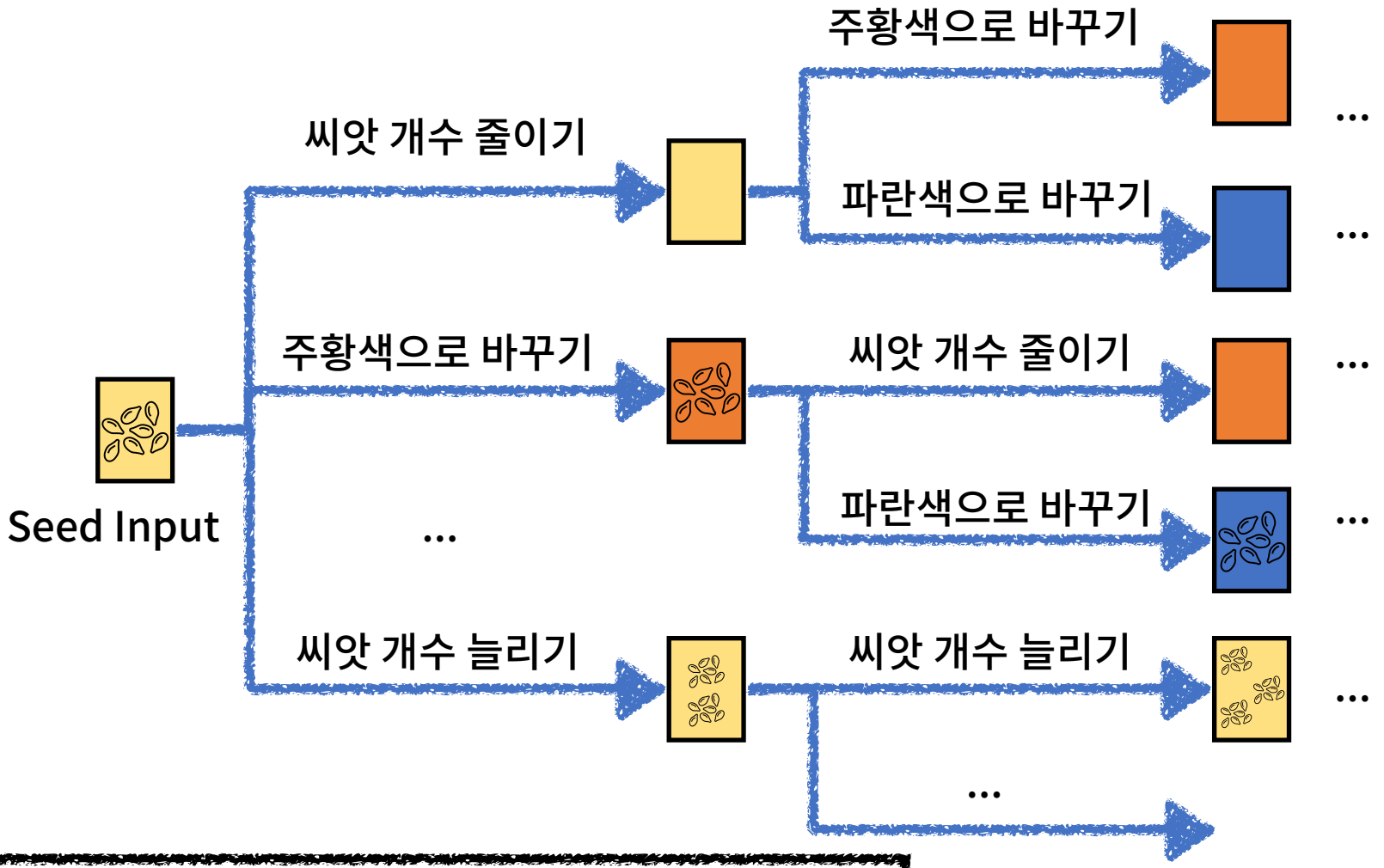
Mutation Strategy

A: 씨앗 개수 줄이기
B: 씨앗 개수 늘리기
C: 주황색으로 바꾸기
D: 파란색으로 바꾸기
...

Mutation Operators (M_{op})

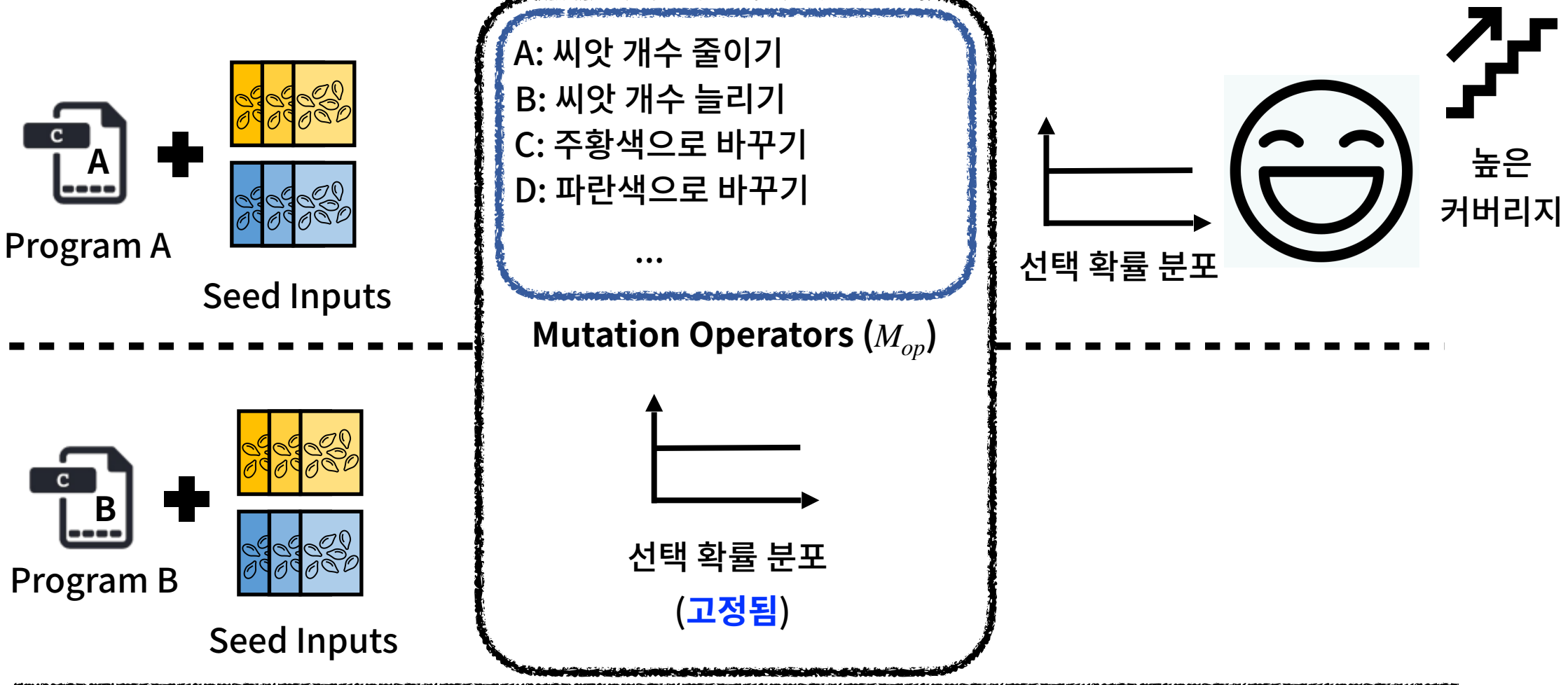


선택 확률 분포



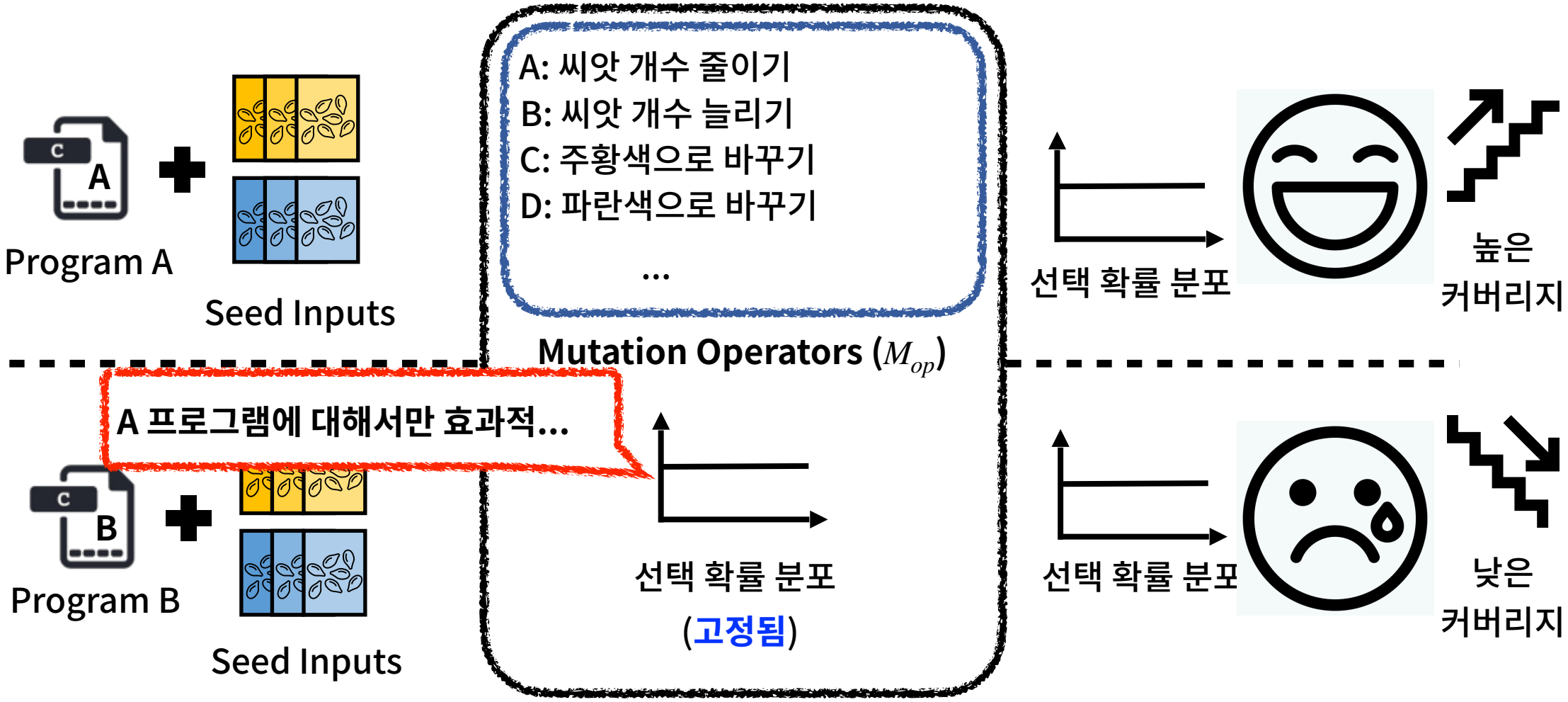
주어진 입력값을 어떻게 바꿀까??

Mutation Strategy: Program **Agnostic**



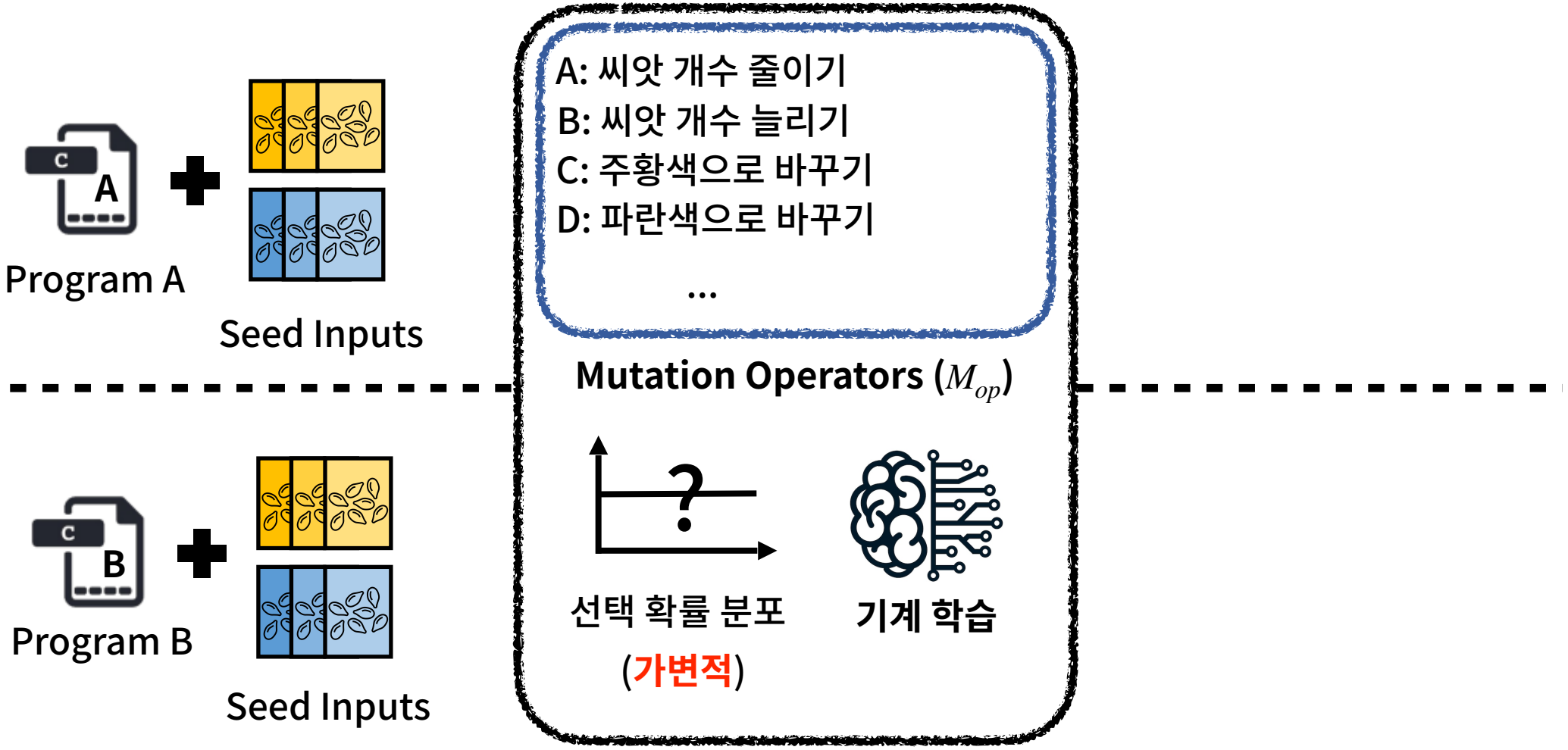
사전에 확정되어 고정된 선택 확률 분포 활용 - 프로그램 상관 X

Mutation Strategy: Program **Agnostic**



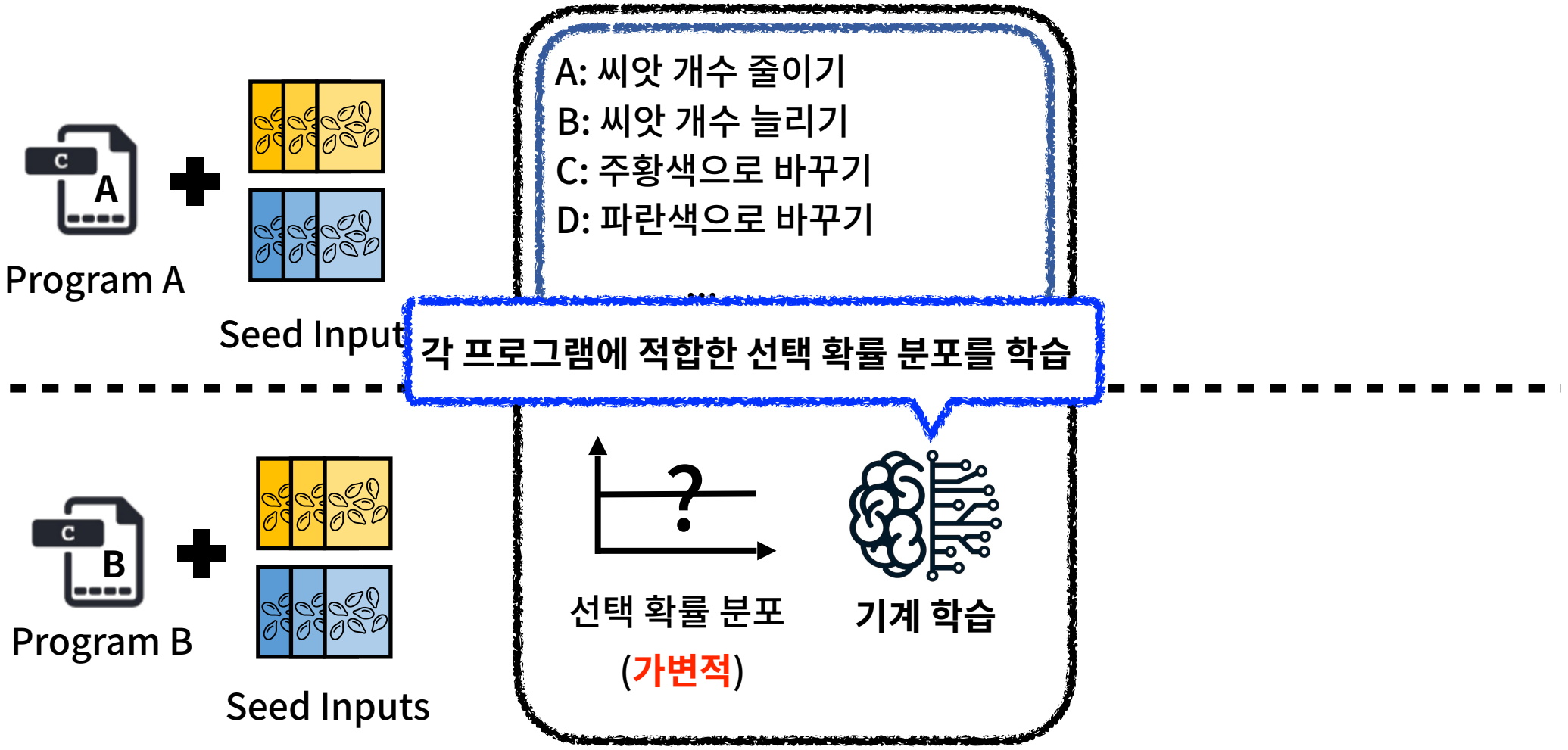
항상 좋은 결과를 가질 수 있다고 말하기 힘듦

Mutation Strategy: Program **Adaptive**



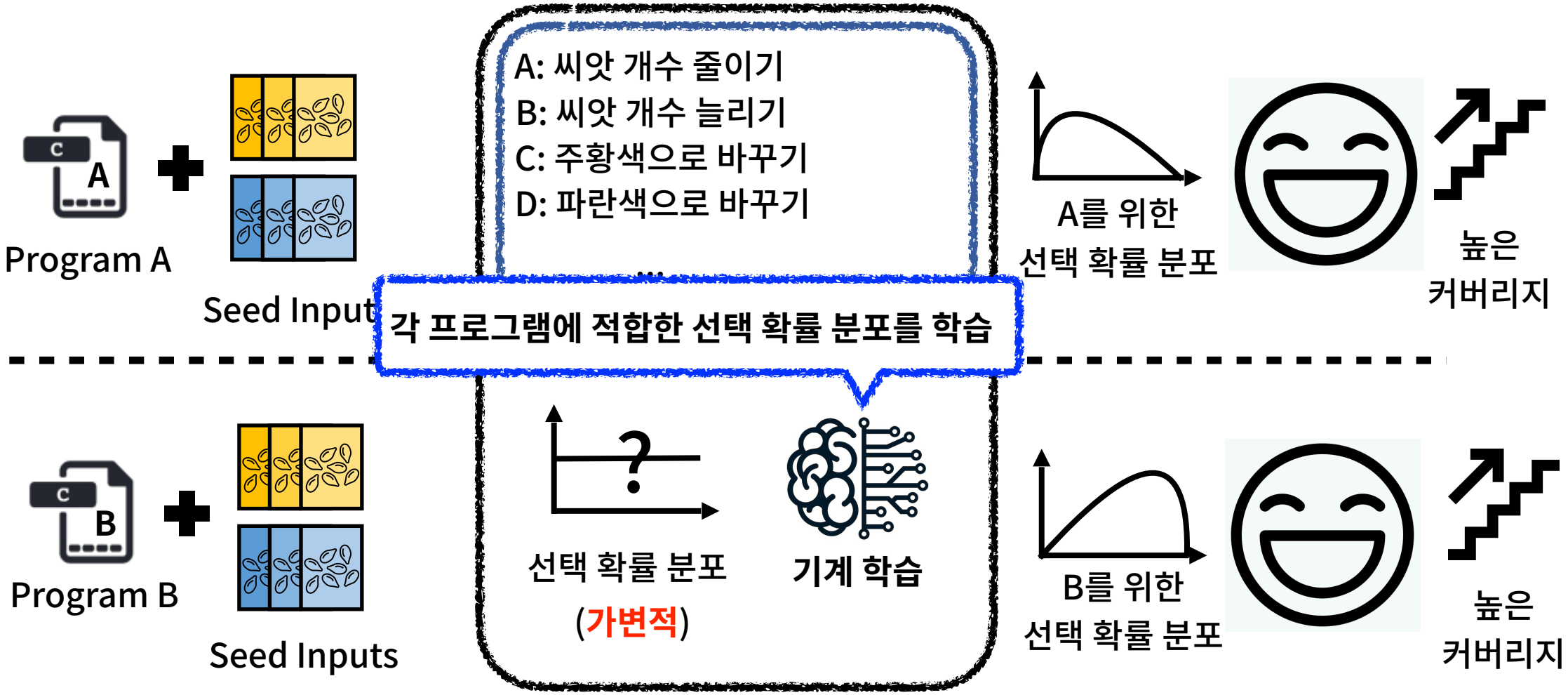
가변적인 선택 확률 분포 활용 - 프로그램마다 적합한 방식

Mutation Strategy: Program **Adaptive**



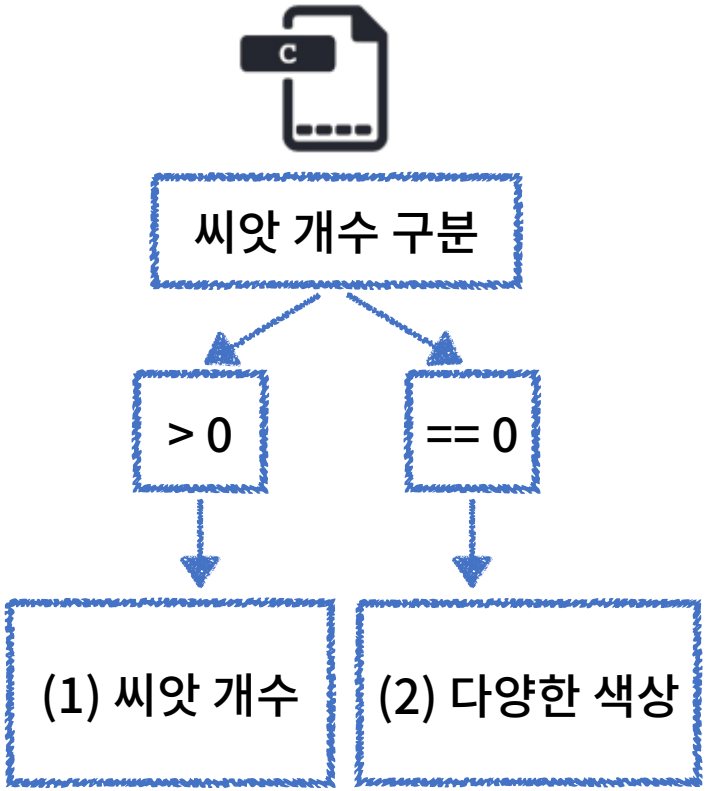
가변적인 선택 확률 분포 활용 - 프로그램마다 적합한 방식

Mutation Strategy: Program **Adaptive**

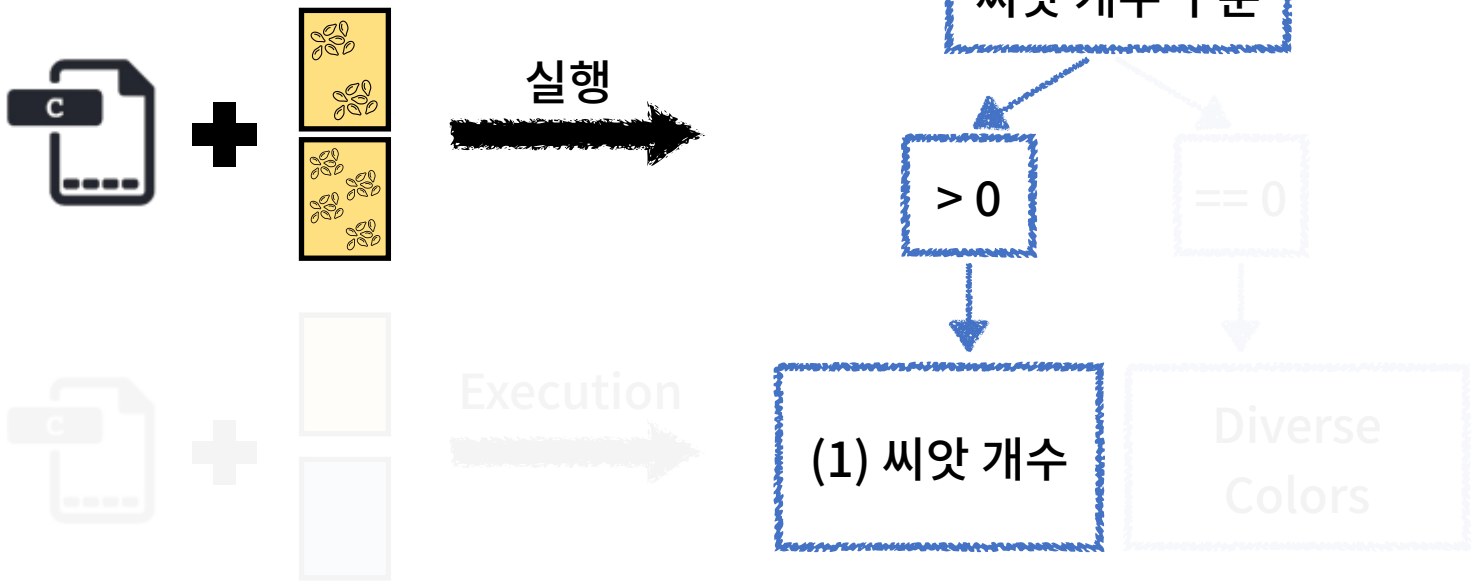


항상 좋은 결과를 가질 수 있을거라 기대할 수 있음

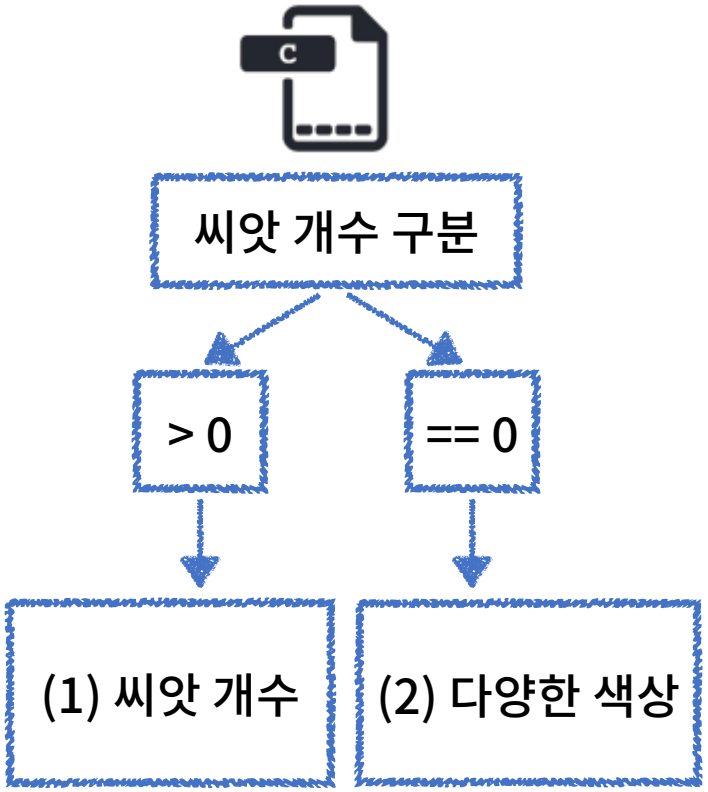
하지만...



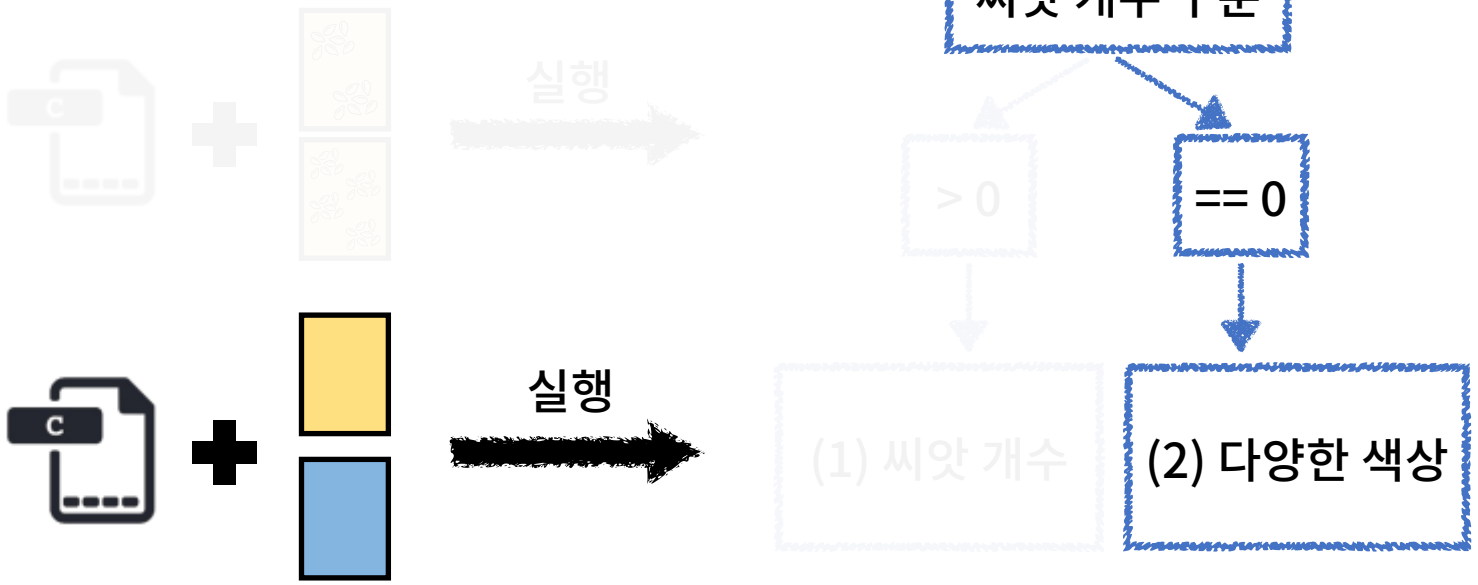
프로그램 동작 방식



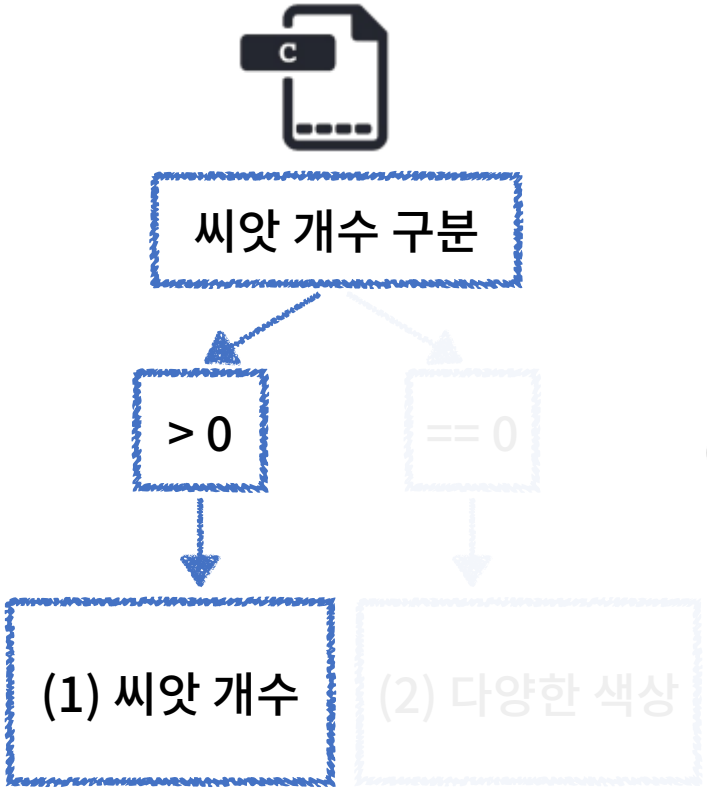
하지만...



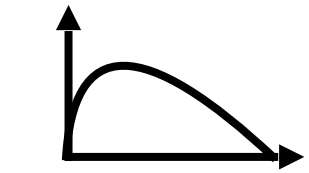
프로그램 동작 방식



하지만...

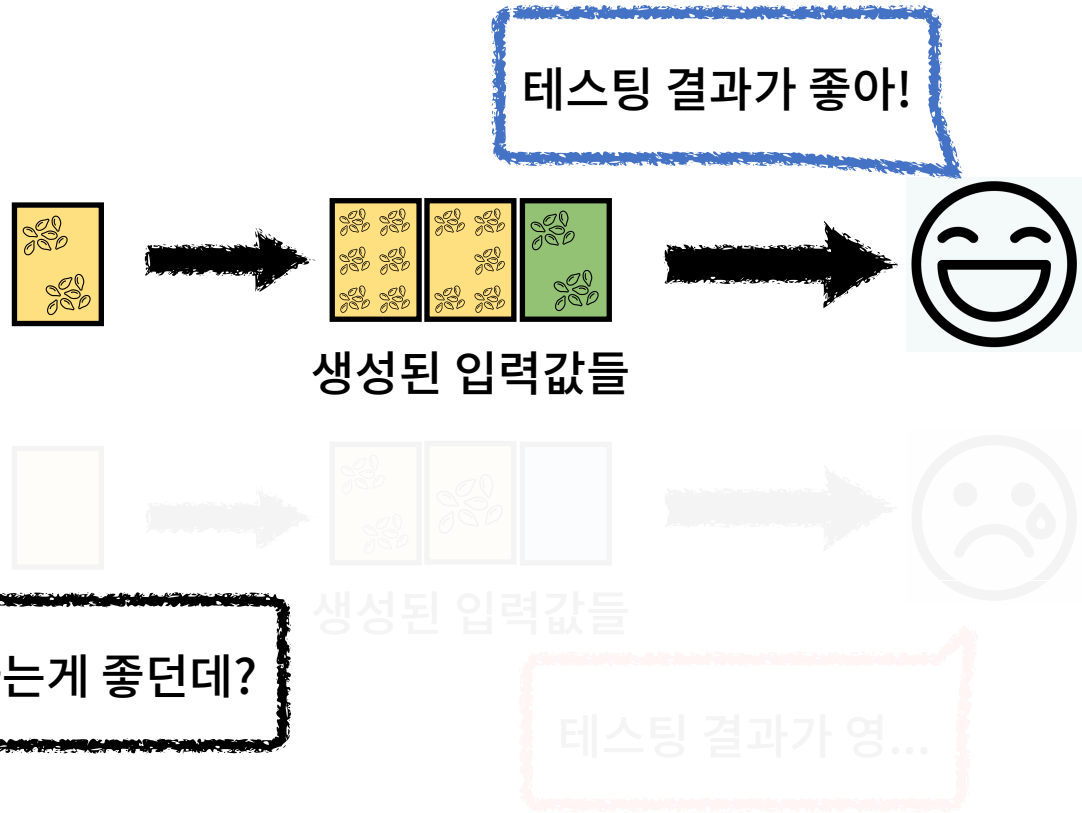


프로그램 동작 방식

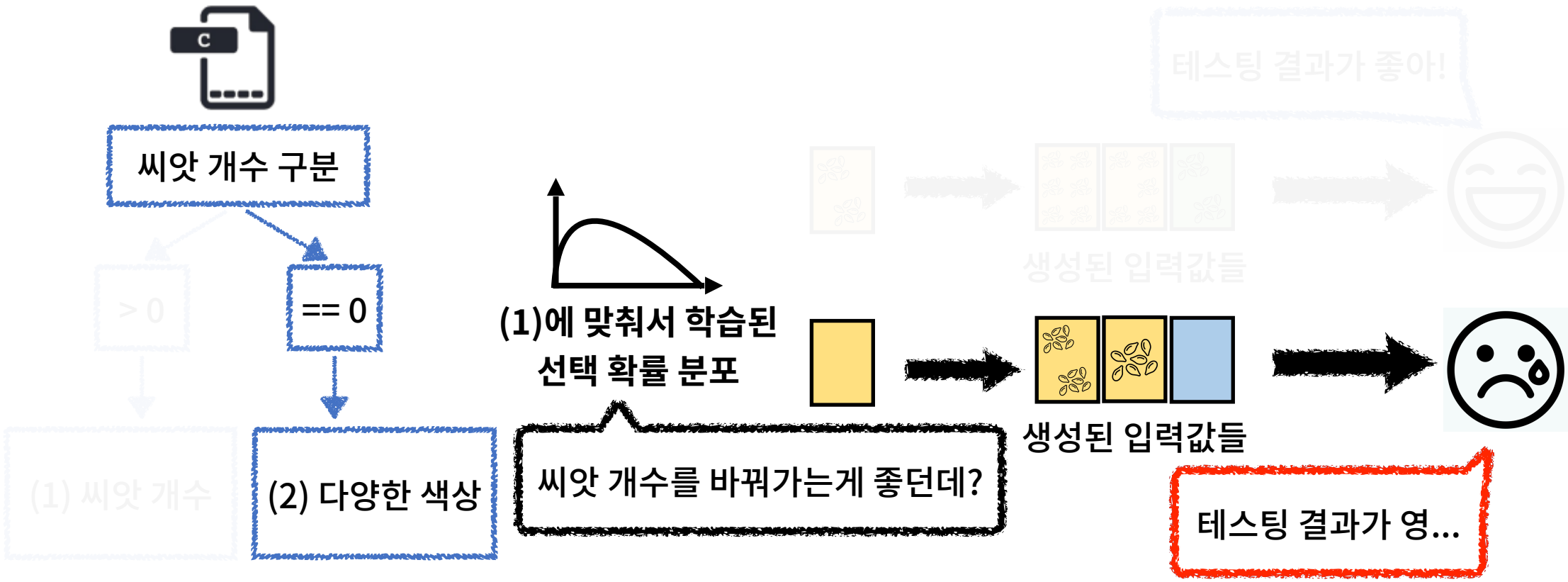


(1)에 맞춰서 학습된
선택 확률 분포

씨앗 개수를 바꿔가는게 좋던데?

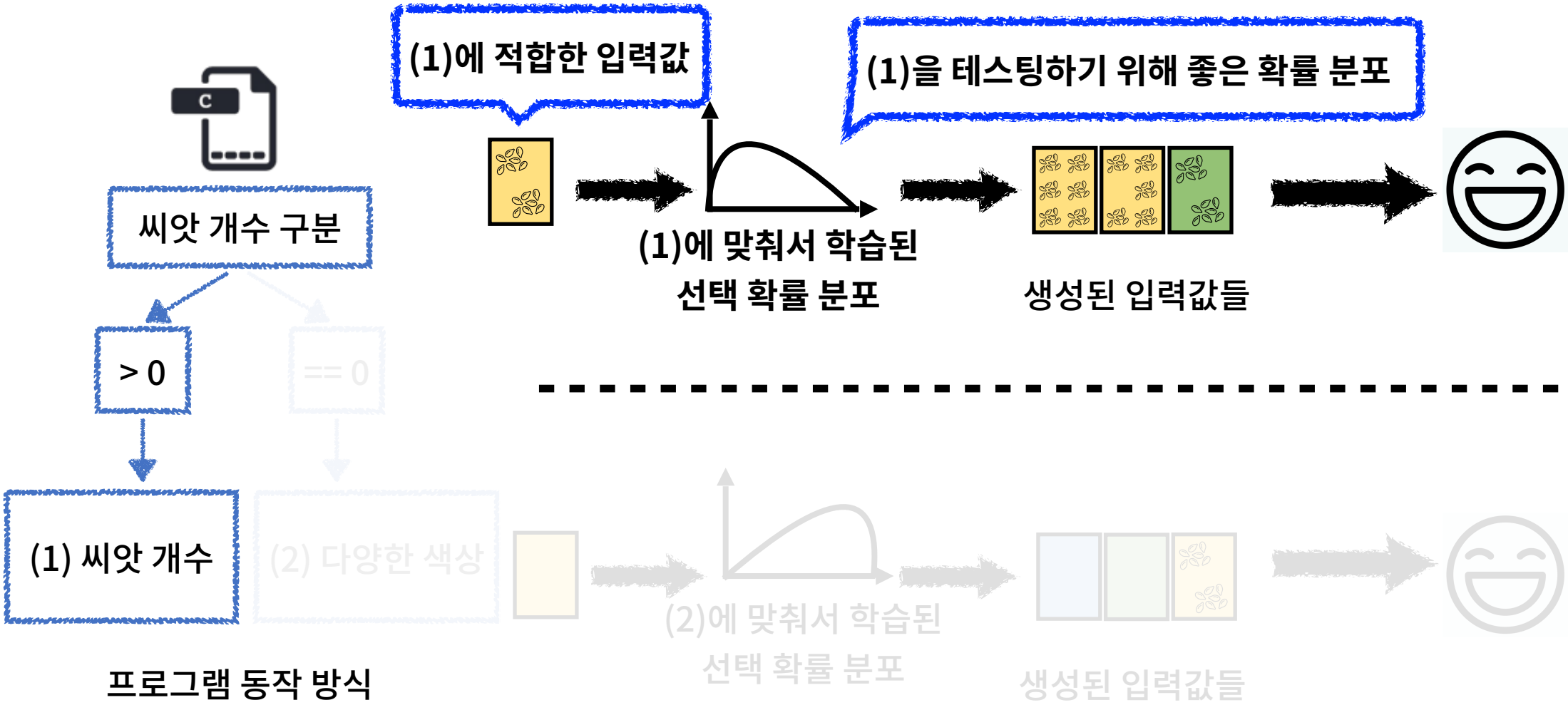


하지만...

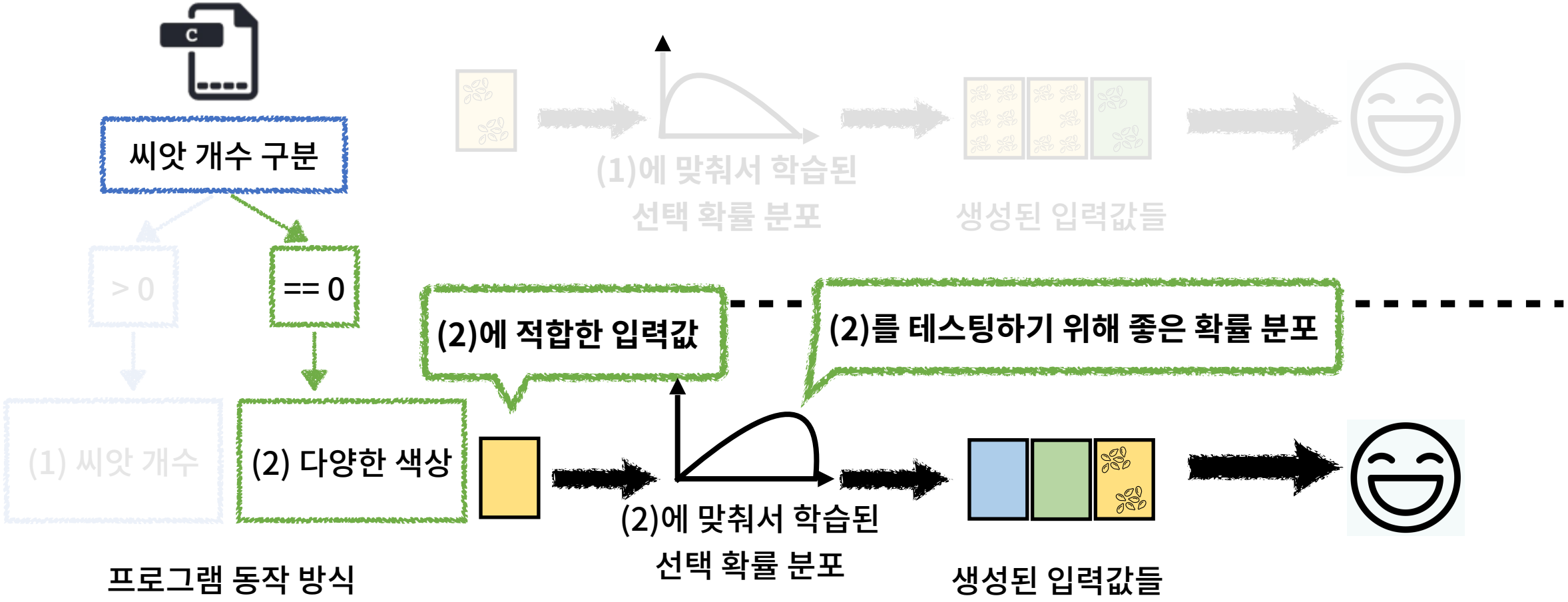


프로그램 동작 방식

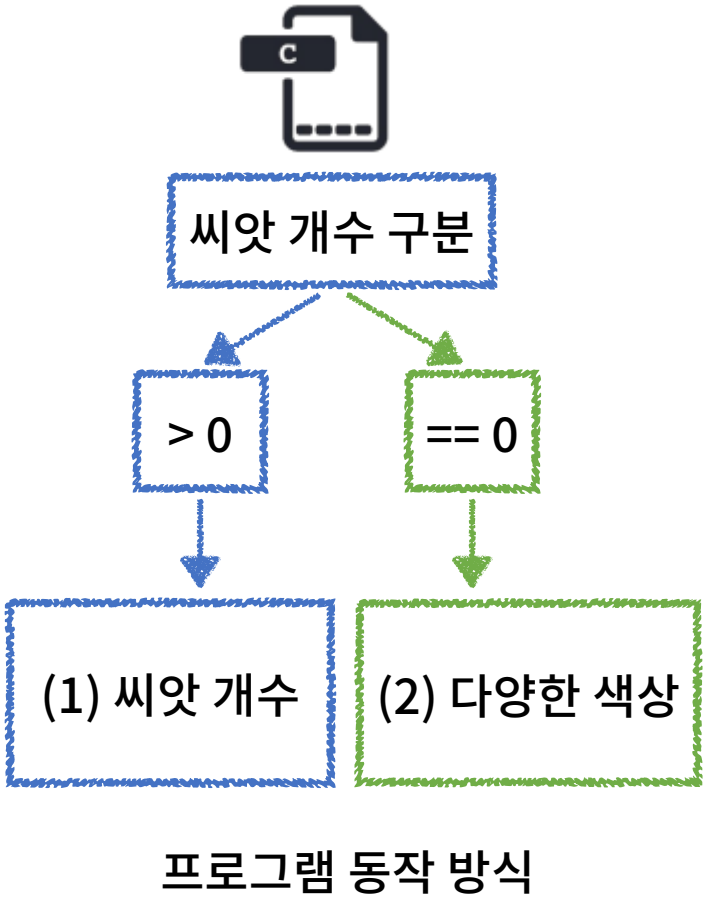
Mutation Strategy: **Seed** Adaptive



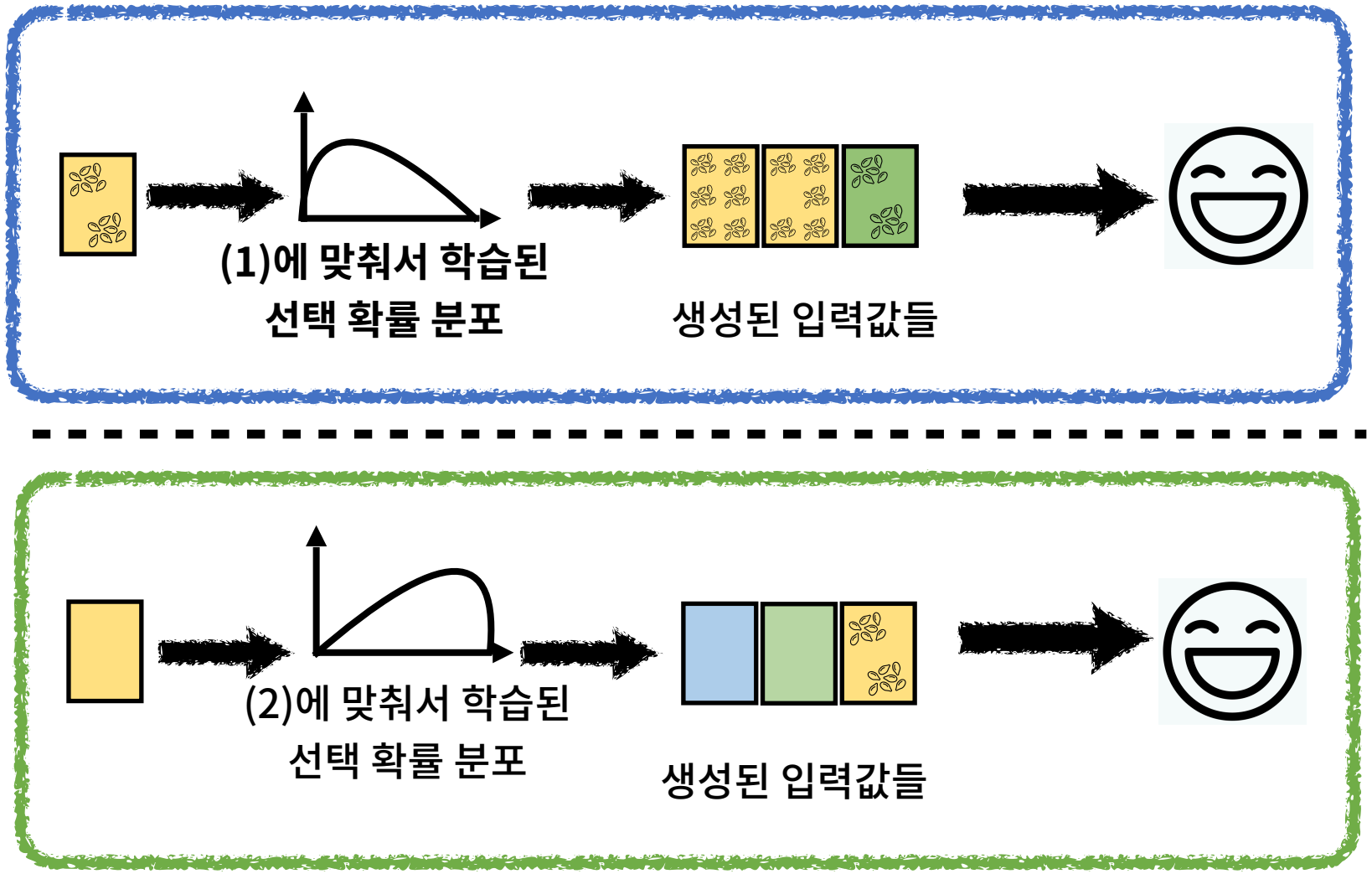
Mutation Strategy: **Seed** Adaptive



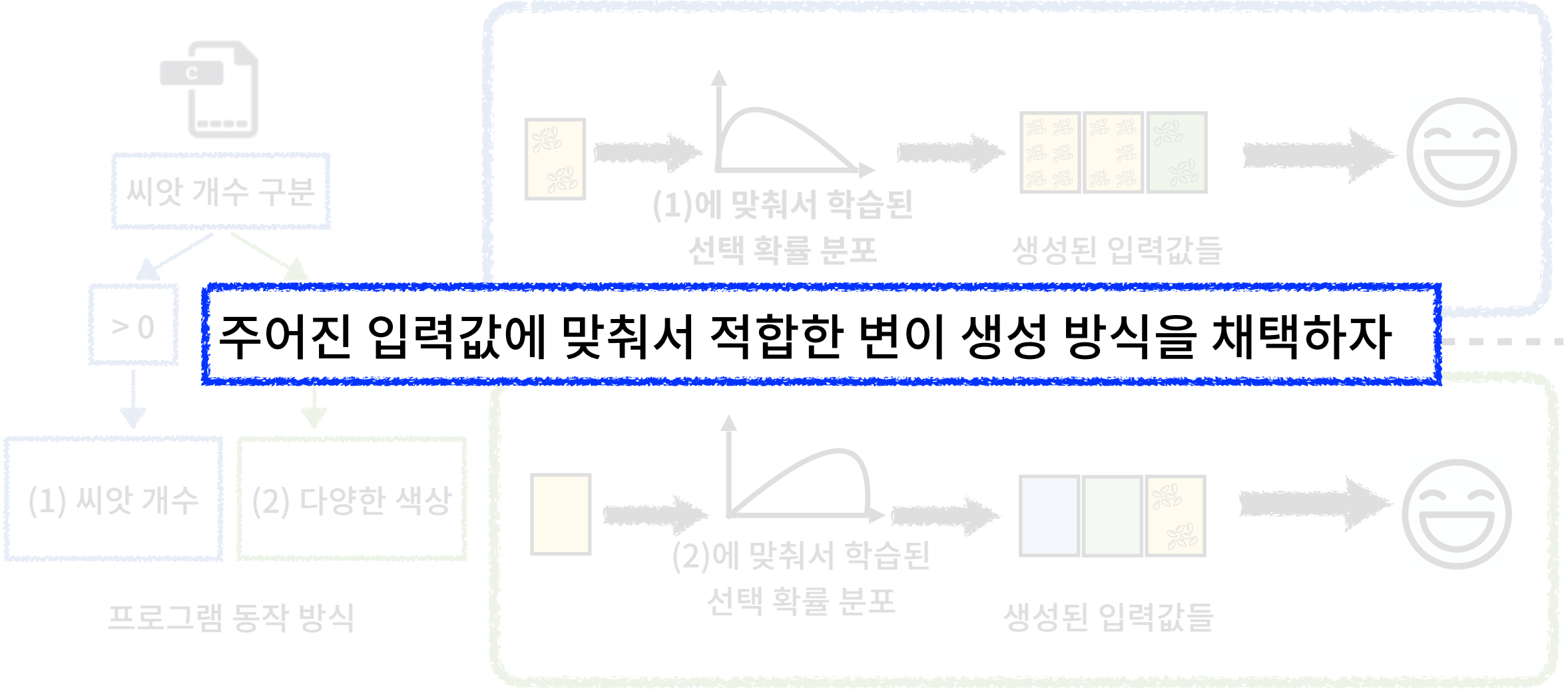
Mutation Strategy: **Seed** Adaptive



프로그램 동작 방식

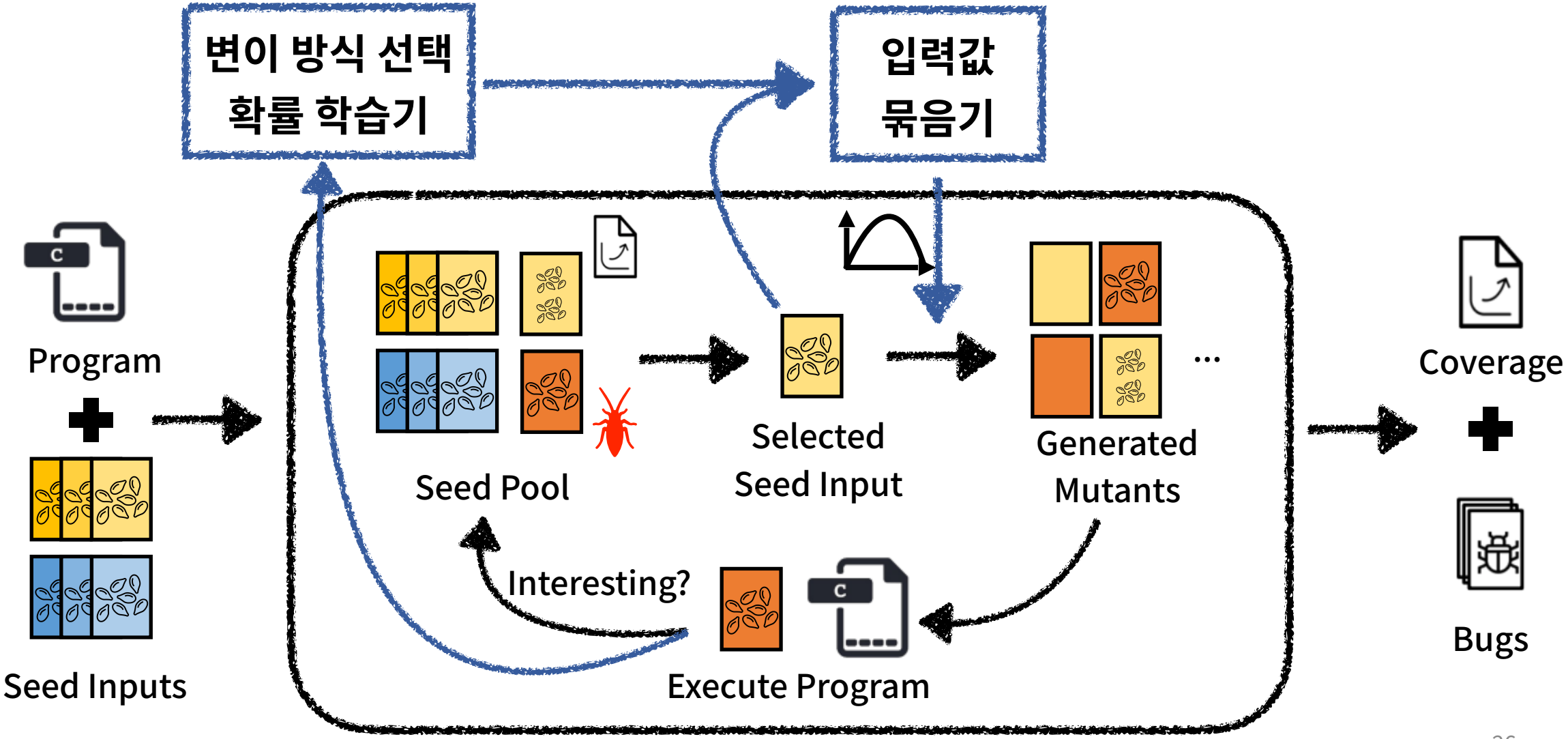


Mutation Strategy: **Seed** Adaptive

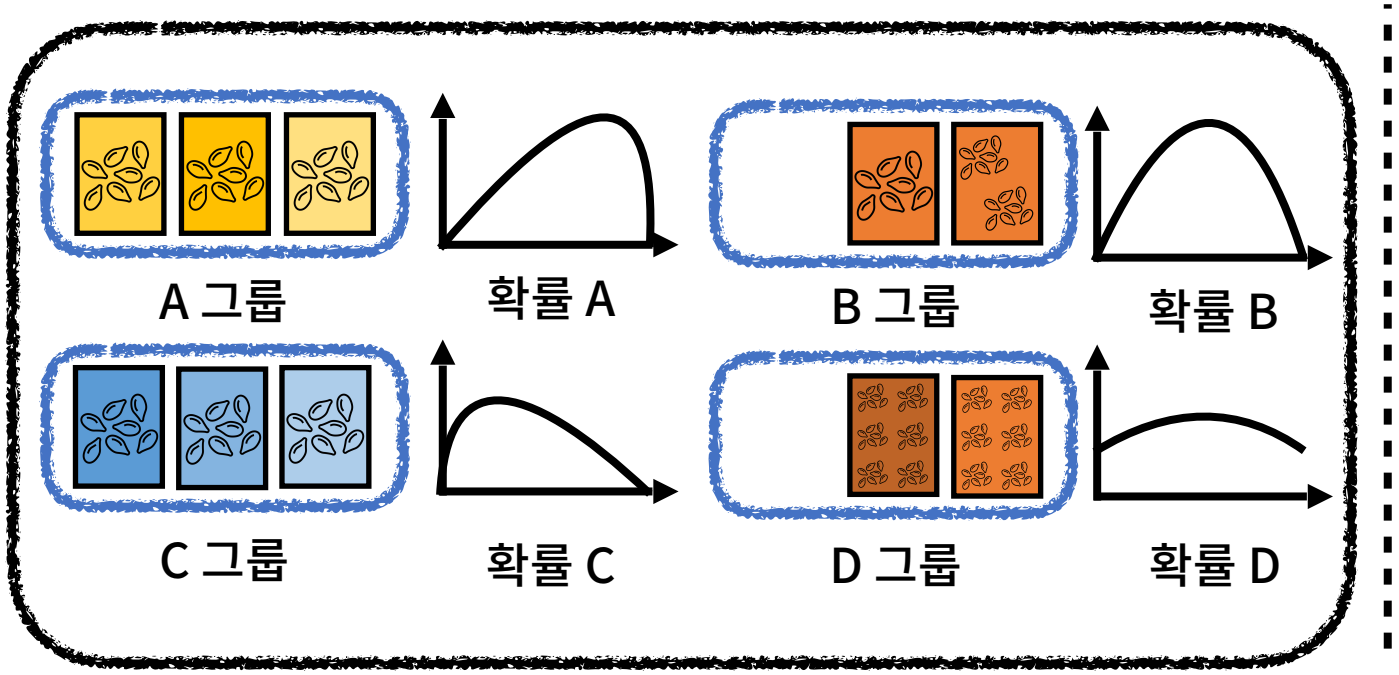


주어진 입력값에 맞춰서 적합한 변이 생성 방식을 채택하자

SeamFuzz: Seed-Adaptive Mutation-based Fuzzer



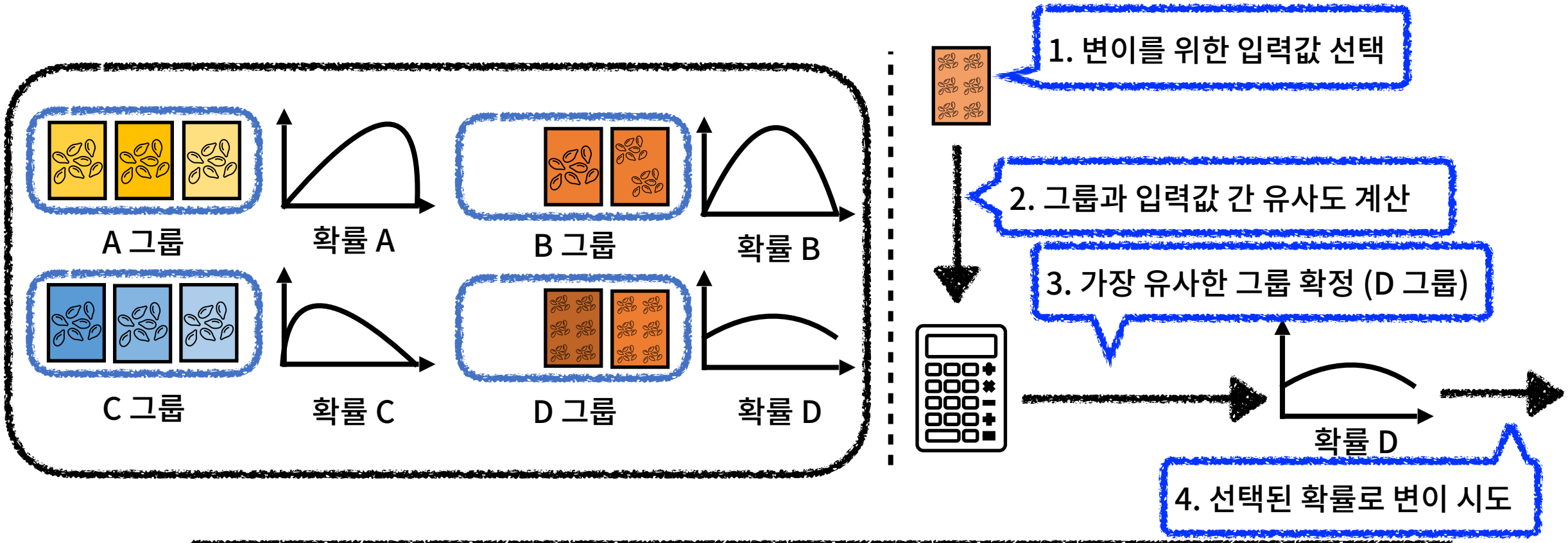
SeamFuzz: **Seed Cluster**



Seed Cluster: 비슷한 **특성**을 지닌 입력값들의 집합 만들기

구조적(Syntactic) + **의미적(Semantic)** 유사도

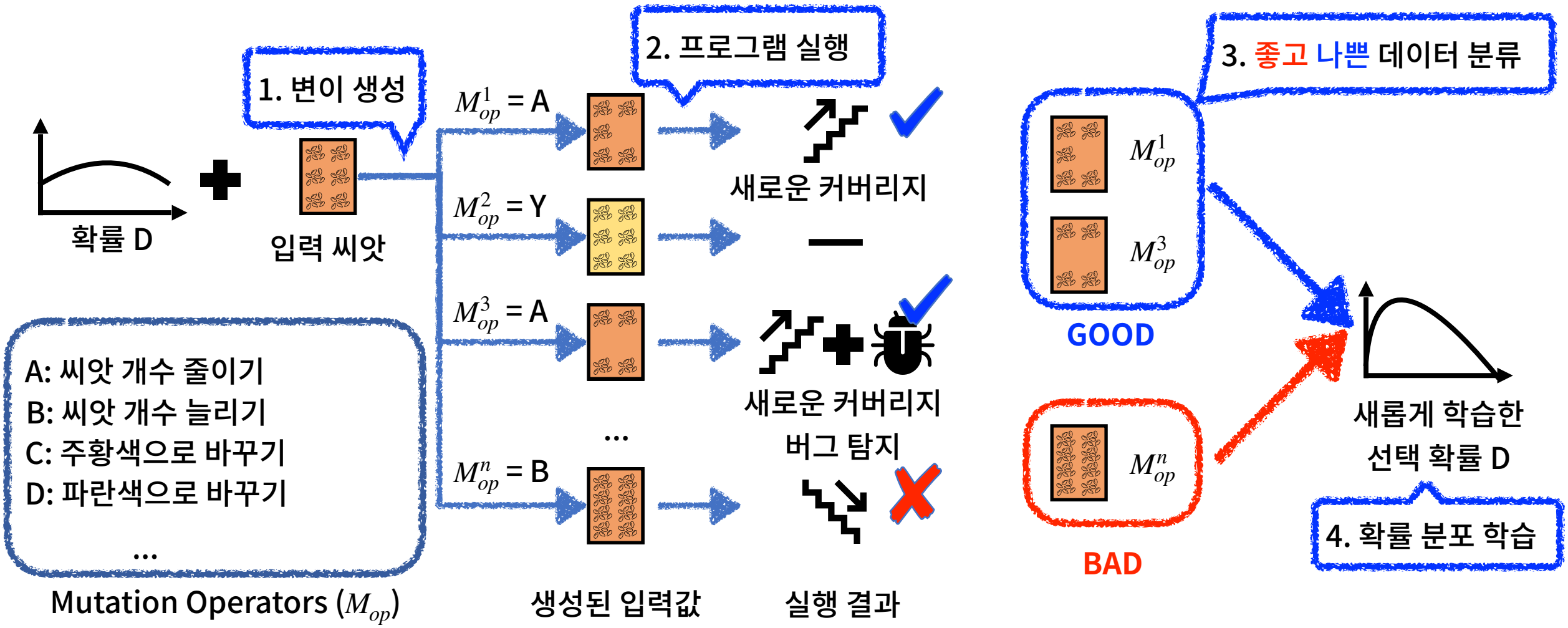
SeamFuzz: **Seed Cluster**



Seed Cluster: 비슷한 **특성**을 지닌 입력값들의 집합 만들기

구조적(Syntactic) + **의미적(Semantic)** 유사도

SeamFuzz: **Probability Learner**



좋은 변이 방법은 더 많이, 나쁜 변이 방법은 더 적게 선택

실험: 실험 세팅

- Google의 FuzzBench를 활용해서 모든 실험을 진행
  - FuzzBench: Google에서 제공하는 Fuzzer 성능 측정을 위한 프레임워크
  - 각 벤치마크 프로그램 당 **24 시간** X **20 회** (**480 시간**)

실험: 실험 세팅

- Google의 FuzzBench를 활용해서 모든 실험을 진행
  - FuzzBench: Google에서 제공하는 Fuzzer 성능 측정을 위한 프레임워크
  - 각 벤치마크 프로그램 당 **24 시간** X **20 회** (**480 시간**)

- 벤치마크 프로그램
  - “Fuzzbench”, “MOpt[USENIX`19]”, “Magma[POMACS`20]”에서 총 14개 프로그램 선정

Programs	LoC	Source	Programs	LoC	Source
objdump-2.37	1,654K	[28]	php-parser	1,500K	[17], [30]
arrow-6.0.1	959K	[30]	openssl	695K	[17], [30]
libxml2-2.9.9	451K	[17], [30]	sqlite3	319K	[17], [30]
proj4-9.0.0	279K	[30]	grok-9.7.1	240K	[30]
poppler	196K	[30]	libarchive	164K	[30]
zstd	107K	[30]	infotocap-6.2	83K	[28]
libpng	76K	[30]	podofopdfinfo	62K	[28]

실험: 실험 세팅

- Google의 FuzzBench를 활용해서 모든 실험을 진행
  - FuzzBench: Google에서 제공하는 Fuzzer 성능 측정을 위한 프레임워크
  - 각 벤치마크 프로그램 당 **24 시간** X **20 회** (**480 시간**)

- 벤치마크 프로그램
  - “Fuzzbench”, “MOpt[USENIX`19]”, “Magma[POMACS`20]”에서 총 14개 프로그램 선정

- 2개의 퍼저와 비교 진행
  - **AFL++**: **Program Agnostic**한 mutation-based greybox Fuzzer (AFL의 업그레이드 버전)
  - **MOpt**[USENIX`19]: **Program Adaptive**한 mutation-based greybox fuzzer

RQ1: SeamFuzz의 성능

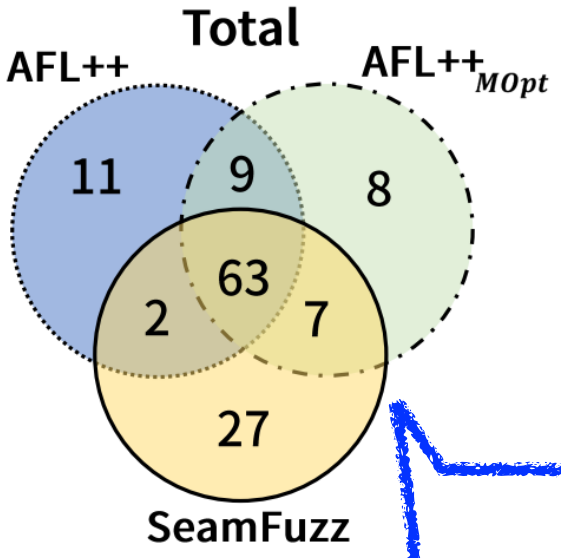
Program	LoC	AFL++ [12]		AFL++$_{\text{MOPT}}$ [28]				SeamFuzz			
		Cover	*Crash*	*Cover*	*Crash*	R_{Cov}	R_{Crash}	*Cover*	*Crash*	R_{Cov}	R_{Crash}
objdump	1,653,733	4,969	74.6	4,663	102.6	-6.2%	37.5%	**5,759**	**146.8**	15.9%	96.8%
php-parser	1,500,422	17,010	0.6	**17,161**	0.9	0.9%	50%	17,072	**2.0**	0.4%	233.3%
arrow	958,895	2,430	101.3	2,454	**111.8**	1.0%	10.4%	**2,466**	101.7	1.5%	0.4%
openssl	695,059	5,831	0	5,832	0	0%	0%	**5,843**	0	0.2%	0%
libxml2	450,545	7,608	113.1	6,698	47.8	-12.0%	-57.7%	**7,888**	**176.5**	3.7%	56.1%
sqlite3	318,961	11,581	0	11,222	0	-3.1%	0%	**12,259**	0	5.9%	0%
proj4	279,128	2,572	15.6	2,292	21.3	-10.9%	36.5%	**2,658**	**37.4**	3.4%	139.7%
grok	240,034	5,234	0.0	5,315	0	1.5%	NaN	**5,360**	**21.3**	2.4%	NaN
poppler	195,849	17,436	16.4	18,084	27.6	3.7%	68.3%	**19,588**	**28.2**	12.3%	72.0%
libarchive	164,242	4,781	0.2	4,203	0.3	-12.1%	33.3%	**5,262**	**0.5**	3.7%	150%
zstd	107,194	5,208							0	0.4%	0%
libpng	94,561	1,888							0	0.4%	0%
infotocap	83,054	1,805							**66.2**	12.6%	22.4%
podofopdfinfo	62,000	1,552							**25.6**	5.6%	118.8%
total	6,803,677	89,905	387.6	88,156	385.9	-1.9%	-0.4%	94,950	606.2	5.6%	56.4%

더 높은 branch coverage 달성

더 많은 크래시 유발 입력값 생성

- **Cover**: 커버된 브랜치의 평균 개수
- **Crash**: 생성된 크래시를 유발하는 입력값들의 평균 개수

RQ2: 버그 탐지 능력



FuzzBench에서 제공하는 리포트 활용

Integer-overflow:xmlParse3986Port xmlParse3986Authority xmlParse3986HierPart	Heap-buffer-overflow READ:xmlParseXMLDecl xmlParseDocument xmlDoRead	Index-out-of-bounds:xmlAddDefAttrs xmlParseAttributeListDecl xmlParseMarkupDecl

(libxml2)

SeamFuzz (99) > MOpt (87) > AFL++(85)

Project	Fuzz Target Program	AFL++	AFL++$_{MOPT}$	SEAMFUZZ
php	exif	PHP004, PHP009, PHP011	PHP004, PHP009, PHP011	PHP004, PHP009, PHP011
libxml2	read_memory_fuzzer	XML001, XML002, XML003, XML009, XML017	XML001, XML003, XML009, XML017	XML001, XML002, XML003, XML009, XML017
openssl	client server x509	SSL002 SSL002 SSL009	SSL002 SSL002, SSL020 SSL009	SSL002 SSL002 SSL009
sqlite3	sqlite3_fuzz	SQL002, SQL014, SQL015, SQL018, SQL020	SQL002, SQL014, SQL015, SQL018, SQL020	SQL002, SQL014, SQL015, SQL018, SQL020

MAGMA에 등록된 버그들

마치며

- SeamFuzz: 입력값들에 맞춰서 변이 방식을 학습하는 greybox fuzzer
  - 각 입력 값들의 **특성을 고려**하여 fuzzer의 전반적인 성능을 높이고자 함.
  - **주요 특징**: **입력값 묶음** 및 **변이 방식 선택 확률 학습 알고리즘 제안**

감사합니다!