정렬 감식 과제 — 학생 안내

서버에 정체를 숨긴 정렬 알고리즘 세 개(A, B, C)가 있습니다. 코드는 볼 수 없고, 입력 리스트를 보내면 **비교 횟수와 쓰기 횟수**만 알려 줍니다. 이 숫자를 보고 각 알고리즘이 아래 후보 9종 중 무엇인지 알아낸 뒤, 왜 그렇게 판단했는지 근거와 함께 제출하는 과제입니다.

입력을 한 번 보내고 횟수를 돌려받는 것을 **질의**라고 부릅니다.

- 배정된 알고리즘도, A·B·C의 순서도 학생마다 다릅니다. 친구와 답을 맞춰 볼 수 없습니다.
- 시작: **Algo Tutor(https://algorithm.chansu.app) 시험 탭의 「정렬 감식 과제」**. 가입이나 비밀번호는 따로 필요 없습니다.
- 기간: 그 카드에 표시된 오픈·마감 시각(한국 시간)이 기준입니다. 마감 뒤에는 질의도 제출도 할 수 없습니다.

1. 시작하기

1. **과제 화면 열기**: Algo Tutor에 평소처럼 이메일 코드로 로그인하고 **시험** 탭에서 「과제 열기」를 누르세요. 카드가 보이지 않거나 「이름이 명단과 다르다」고 나오면 조교에게 알리세요.
2. **연습하기**: 먼저 연습 알고리즘 P에 입력을 보내 보세요. 정체가 공개되어 있고, A·B·C의 질의 횟수를 쓰지 않습니다.
3. **A·B·C에 질의 보내기**: 아래 두 방법 중 편한 쪽을 쓰세요.
4. **답안 쓰고 제출하기**: 과제 화면의 「답안 쓰기」에서 작성하고 마지막에 꼭 「제출」을 누르세요.

질의를 보내는 두 가지 방법

	방법 1. 과제 화면에서 보내기	방법 2. 파이썬 코드로 보내기
준비	없음	`sortlab.py`와 접속 키 (파이썬 3.9 이상, 따로 설치할 것 없음)
보내는 법	「질의 보내기」에서 알고리즘을 고르고, 숫자를 적어 「보내기」	코드에서 `lab.query(...)` 실행
이럴 때 편합니다	입력이 짧을 때, 손으로 세어 보며 확인할 때	긴 입력을 만들 때, 여러 입력을 차례로 보낼 때

두 방법은 결과도 횟수도 같습니다. 어느 쪽으로 보내든 같은 기록에 남고 같은 횟수에서 빠지므로, 섞어 써도 됩니다.

방법 2를 쓰는 순서

1. 시험 탭 카드에서 `sortlab.py`를 내려받아 코드를 작성할 폴더에 넣습니다.
2. 카드의 「파이썬 접속 키」를 눌러 키를 받습니다. **키는 비밀번호와 같습니다.** 코드를 남에게 보여 줄 때는 키를 빼세요.
3. 같은 폴더에서 아래처럼 코드를 작성해 실행합니다.

```
import sortlab
lab = sortlab.Lab("slk_여기에_접속_키")

print(lab.query("P", [5, 4, 3, 2, 1]))   # 연습: 선택 정렬
# {'query_id': 'P0001', 'label': 'P', 'n': 5, 'comparisons': 10, 'writes': 4, 'remaining': None}

r = lab.query("A", list(range(1000)))    # 알고리즘 A에 보내기 — 질의 1번 사용
print(r["query_id"], r["comparisons"], r["writes"], "남은 질의", r["remaining"])

lab.status()                             # 알고리즘별 남은 질의, 제출 상태
rows = lab.history("A", with_data=True)  # 지금까지 보낸 질의(입력 포함)
```

결과는 같은 폴더의 `sortlab_log.csv`에도 자동으로 쌓입니다.

2. 후보 9종

이름	설명
`insertion`	삽입 정렬(강의 코드: 인접 교환)
`selection`	선택 정렬(최댓값을 뒤로, 같은 자리면 교환 안 함)
`bubble`	버블 정렬(조기 종료 없음)
`quick_first`	퀵 정렬, 피벗 = 구간의 첫 값 (강의 기본형, 2-way partition)
`quick_random`	퀵 정렬, 피벗 = 구간에서 무작위로 고른 값
`merge_topdown`	합병 정렬, 반씩 나누는 재귀형 (강의 기본형)
`heap`	힙 정렬 (강의 make_heap + heap_sort)
`hybrid_quick_insertion`	퀵 정렬(첫 값 피벗) + 구간 크기 ≤ k이면 삽입 정렬
`hybrid_merge_insertion`	합병 정렬(재귀형) + 구간 크기 ≤ k이면 삽입 정렬

- 하이브리드라면 **k도 알아내야 합니다.** k는 5 이상 40 이하의 정수입니다.
- 알고리즘 A·B·C는 서로 다른 후보입니다.

3. 서버가 세어 주는 연산

- **비교(comparisons)**: 원소 두 개(또는 원소와 피벗 값)의 크기를 비교한 횟수입니다. 인덱스 비교는 세지 않습니다.
- **쓰기(writes)**: 값 하나를 리스트(임시 리스트 포함)의 한 칸에 써 넣은 횟수입니다.
  - 교환(swap) 한 번은 쓰기 2번입니다. 자기 자신과의 교환은 세지 않습니다.
  - 합병에서는 임시 리스트에 넣을 때 1번, 원래 리스트로 되돌려 쓸 때 1번입니다.

예: 연습 알고리즘 P(선택 정렬)에 `5 4 3 2 1`을 보내면 비교 10 · 쓰기 4가 나옵니다. 비교는 4 + 3 + 2 + 1 = 10이고, 쓰기는 교환 2번 × 2 = 4입니다(5와 1, 4와 2만 자리를 바꿉니다).

4. 질의 규칙

	알고리즘 A·B·C	**연습 알고리즘 P**
정체	비공개	**공개:** `selection`
질의 횟수	**알고리즘마다 30번**	하루 300번
입력	정수 리스트. 개수 n은 1개 이상 2000개 이하, 값의 절댓값은 10^9 이하	같음

- 쓴 횟수는 되돌릴 수 없습니다. **무엇을 확인할지 먼저 생각하고** 보내세요.
- 보낸 질의는 모두 서버에 기록되고, **채점은 이 기록만 보고 합니다.** 같은 입력을 두 번 보내도 됩니다.
- 질의마다 `A001`, `A002` … 처럼 번호가 붙습니다. 답안에서 근거를 댈 때 이 번호를 고릅니다.
- 강의 노트의 기본형과 같은 알고리즘이라도 구현의 세부(피벗을 어디로 옮기는지, 중앙값을 고르는 비교 순서 등)는 공개하지 않습니다. 그래서 **숫자를 한 자리까지 똑같이 맞힐 필요는 없습니다.** 입력의 크기와 모양(정렬된 입력, 역순, 같은 값이 많은 입력 등)에 따라 횟수가 어떻게 달라지는지를 보고 가려내세요.

5. 답안 쓰기와 제출

과제 화면의 **답안 쓰기**에서 작성합니다. 쓰는 내용은 자동으로 저장되어 다른 컴퓨터에서도 이어 쓸 수 있습니다. **저장만 해서는 채점되지 않습니다.** 제출하지 않은 답안은 마감 때 자동으로 제출되지 않으니, 마지막에 꼭 「제출」을 누르세요. 마감 전까지 몇 번이든 다시 낼 수 있고 **마지막 제출만** 채점합니다.

1. **정체 고르기**: 알고리즘 A·B·C 탭마다 정체(후보 이름)와 확신도를 고릅니다. 하이브리드를 고르면 k 칸이 나타납니다.
2. **후보 기각하기**: 기각은 「이 후보는 아니다」라고 근거를 대는 것입니다. 정체를 고르면 꼭 기각해야 할 후보 칸이 나타납니다. 다른 후보도 기각하려면 「후보 추가」를 누르세요. 칸마다 다음을 채웁니다.
   - **근거 질의**: 그 알고리즘에 보낸 질의 중 **1~2개**를 고릅니다. 근거 질의를 고르지 않은 칸은 제출에 들어가지 않습니다.
   - **예측 항목과 예측값**: **그 후보였다면 첫 번째 근거 질의에서 비교나 쓰기가 몇 번 나왔을지** 적습니다. 정확하지 않아도 됩니다(아래 「기각이 인정되는 조건」 참고). 같은 입력을 두 번 보내 무작위인지 가르는 경우에는 비워 둡니다.
   - **근거 설명**: 그 질의로 왜 이 후보가 아니라고 판단했는지 적습니다.
3. **판별 과정과 LLM 사용 메모 쓰기**: 판별 과정은 알고리즘마다, LLM 사용 메모는 전체에 한 번 적습니다.
4. **검사하고 제출하기**: 「형식 검사」로 빠진 곳을 확인한 뒤 「제출」을 누릅니다. 형식 오류가 있으면 제출되지 않고 해당 칸에 이유가 표시됩니다. 맞았는지 틀렸는지는 마감 전에 알려 주지 않습니다.

파이썬으로 내려면

웹 폼만 쓴다면 몰라도 됩니다. 답안 쓰기 아래의 「answers.json 내려받기」나 `lab.template("answers.json")`으로 파일을 받아 채운 뒤 `lab.check("answers.json")`으로 검사하고 `lab.submit("answers.json")`으로 냅니다. 이렇게 낸 답안도 과제 화면에 나타나 이어서 고칠 수 있습니다.

```
{
  "functions": {
    "A": {
      "algorithm": "heap",
      "k": null,
      "confidence": "상",
      "rejections": [
        {"candidate": "quick_first",
         "queries": ["A004"],
         "predicted": {"metric": "comparisons", "value": 500000},
         "reason": "A004는 정렬된 n=1000. quick_first라면 매번 한쪽이 비어 비교가 약 n²/2이어야 하는데 실
제는 훨씬 적다"},
        {"candidate": "quick_random",
         "queries": ["A001", "A002"],
         "predicted": null,
         "reason": "같은 무작위 입력을 두 번 보냈는데 비교·쓰기가 같았다 → 무작위 피벗 아님"}
      ],
      "reasoning": "판별 과정 요약(어떤 순서로 무엇을 확인했는지)"
    },
    "B": { "...": "..." },
    "C": { "...": "..." }
  },
  "llm_use": "LLM을 어디에 썼는지 몇 줄(점수 없음)"
}
```

`algorithm`은 후보 이름, `k`는 정수 또는 `null`, `confidence`는 상·중·하입니다. `queries`를 비워 둔 칸은 제출에 들어가지 않고, 예측이 필요 없는 기각의 `predicted`는 `null`입니다. `metric`은 `comparisons`(비교) 또는 `writes`(쓰기)입니다.

기각이 인정되는 조건

기각은「내 알고리즘은 이 후보가 아니다」라는 주장입니다. 서버는 근거 질의 하나를 놓고 **숫자 세 개**를 비교해 인정할지 정합니다.

예를 들어 정렬된 입력(n=1000)을 보낸 질의 `A004`로 `quick_first`를 기각한다고 합시다.

숫자	어디서 나오나	예: 비교 횟수
① 내 예측값	내가 답안에 적은 값	500,000
② 후보의 값	서버가 같은 입력으로 그 후보를 직접 실행한 값	500,499
③ 내 알고리즘의 값	질의 `A004`의 결과	10,000

다음 두 가지를 **모두** 만족하면 인정됩니다. 비교는 내가 고른 예측 항목(비교 또는 쓰기)으로 합니다.

1. **①과 ②가 비슷하다(차이 25% 이내)**: 그 후보가 어떻게 동작하는지 알고 있다는 뜻입니다. 정확히 맞힐 필요는 없습니다.
2. **②와 ③이 다르다(차이 10% 이상)**: 그 입력에서 후보와 내 알고리즘이 확실히 달랐다는 뜻입니다.

경우	판정	이유
위의 예	**인정**	①과 ②는 거의 같고, ②와 ③은 크게 다릅니다.
①을 300,000으로 적음	인정 안 됨	①과 ②가 40% 다릅니다.
③이 480,000으로 나옴	인정 안 됨	②와 ③이 4%밖에 다르지 않아, 이 질의로는 가를 수 없습니다.

따로 정해 둔 경우가 세 가지 있습니다.

- **k가 있는 후보**: ①은 k = 5~40 중 어느 한 값의 결과와 비슷하면 됩니다. 하지만 ②와 ③의 차이는 k = 5~40 **모든 값**에서 나야 합니다. 한 질의로 어려우면 근거 질의 2개로 나눠 보여도 됩니다.
- **무작위로 섞은 입력**: 섞은 입력에서는 후보끼리 우연히 10% 넘게 차이 날 수 있어 기준이 더 엄격합니다. 서버가 같은 값들을 20번 섞어 후보를 실행하고, ③이 그때 나온 값의 범위를 벗어나야 인정합니다. 그래서 **정렬된 입력, 역순, 같은 값 반복처럼 특징이 뚜렷한 입력**이 근거로 훨씬 좋습니다.
- **무작위인지 가를 때**: 같은 입력을 두 번 보내 비교합니다. 예측값은 적지 않아도 됩니다. 두 결과가 같으면(n ≥ 16) 내 알고리즘은 무작위가 아니므로, 그 질의로 `quick_random`을 기각할 수 있습니다. 두 결과가 다르면 내 알고리즘이 무작위이므로, 그 질의로 나머지 후보를 기각할 수 있습니다.

꼭 기각해야 하는「헷갈리기 쉬운 후보」

근거 점수는 **실제 정답의 행**에 있는 후보들을 제대로 기각했는지로 매깁니다. 그 밖의 후보를 기각한 것은 점수에 들어가지 않습니다. 제출하면 내가 고른 답의 행에서 빠진 후보를 경고로 알려 줍니다.

정답이 이것이면	기각해야 할 후보
`insertion`	selection, bubble
`selection`	insertion, bubble
`bubble`	insertion, selection
`quick_first`	hybrid_quick_insertion, quick_random
`quick_random`	hybrid_quick_insertion, quick_first
`merge_topdown`	heap, hybrid_merge_insertion
`heap`	merge_topdown, quick_random
`hybrid_quick_insertion`	quick_first, hybrid_merge_insertion, insertion
`hybrid_merge_insertion`	merge_topdown, hybrid_quick_insertion, insertion

6. 배점 (100점)

항목	점수
정답 — k가 없는 알고리즘	하나당 20점
정답 — 하이브리드	알고리즘 15점 + k 정확 5점 (±1이면 2.5점)
근거	하나당 40/3점(약 13.3점) × 헷갈리기 쉬운 후보 중 제대로 기각한 비율

판별 과정과 근거 설명은 자동 채점에 들어가지 않습니다. 대신 **수업 중 현장 확인에서 여러분이 보낸 질의와 답안에 대해 질문합니다.** 직접 이해하지 못한 답안은 거기서 드러납니다.

7. LLM 사용

허용합니다. 코드 작성, 후보별 예상 횟수 계산 등 어디에 써도 됩니다. 대신 답안의「LLM 사용 메모」에 **어디에 썼는지** 적어 주세요(점수와 무관). 채점은 여러분이 보낸 질의 기록만 보고 하므로, 어떤 입력을 보낼지 정하고 숫자를 해석하는 일은 결국 직접 해야 합니다.

8. 자주 묻는 질문

- **규칙이 헷갈려요.** 과제 화면 오른쪽 아래 「과제 질문」에 물어보세요. 안내문 기준으로 답하며 하루 30번까지입니다. 알고리즘의 정체, 예측값 계산, 어떤 입력을 보낼지는 답하지 않습니다.
- **웹과 파이썬을 섞어 써도 되나요?** 됩니다. 질의 기록과 횟수는 하나로 합쳐집니다.
- **질의를 다 썼어요.** 원칙적으로 더 드리지 않습니다. 서버 오류로 날린 경우에만 조교에게 질의 번호와 함께 알리세요.
- **저장만 하고 제출을 안 했어요.** 마감 전이면 지금 「제출」을 누르세요. 마감 뒤에는 임시 저장한 답안을 채점하지 않습니다.
- **접속 키를 잃어버렸거나 남에게 보였어요.** 시험 탭 카드의 「파이썬 접속 키」를 다시 누르세요. 이전 키는 바로 멈추고, 질의 기록은 그대로 남습니다.
- **「요청이 너무 잦습니다」가 떠요.** 1분에 보낼 수 있는 질의 수를 넘은 것입니다. 잠시 뒤 다시 보내세요. `sortlab.py`는 두 번까지 자동으로 다시 시도합니다.
- **입력을 몇 개로 해야 하나요?** 정해진 답은 없습니다. 손으로 셀 수 있는 작은 입력과 증가율이 보이는 큰 입력이 각각 무엇을 알려 주는지 생각해 보세요.