**파일처리론 리포트**

**replacement selection sorting**

**이름: 윤일준**

**학과: 컴퓨터공학과**

**학번: 2343457**

**구현한 프로그램의 실행방법**

**파이썬 버전 :** **3.11.4**

**변수**

**num은 정렬 횟수**

**n은 정렬할 값의 개수**

**all\_List에는 정렬할 값들이 공백 없이 리스트로 들어간다.**

**run에는 정렬 할 값을 넣어주며 runs에는 지금까지의 모든 run을 저장한다.**

**freeze\_List는 freeze된 값들을 넣어주는 리스트이다.**

**list1은 버퍼 같은 개념으로 all\_List에서 5개의 값을 가져와 최소값을 찾는다.**

**메커니즘**

**Input 파일에서 각각 정렬횟수 정렬 값의 개수 정렬할 값을 받아 변수**

**num,n,all\_List에 삽입한다.**

**list1에 all\_List의 값 5개를 가져온 후 가져온 값을 all\_List에서 삭제한다.**

**list1의 값 중 최소값(min\_element)을 찾아 run리스트에 삽입한다.**

**list1의 최소값 인덱스(min\_index)에 all\_List의 값을 삽입한다.**

**가져온 새로운 값과 run에 들어간 값을 비교하여**

**1.run 값이 큰경우**

**새로운 값을 포함하여 list1의 최소값을 찾아**

**list1이 다 빌때까지 위와 같이 반복한다.**

**2.새로운 값이 큰 경우**

**새로운 값을 freeze\_List에 넣고 list1은 채우지 않는다.**

**list1이 다 빌때까지 위와 같이 반복한다.**

**list1이 다 비었을 경우(값들이 모두 freeze 된 경우)**

**현재 run에 있는 값을 runs에 저장하고**

**freeze\_List에서 버퍼 크기 만큼 정렬을 위해 list1에 채운다.**

**all\_list가 다 비었을 경우**

**freeze\_List의 값을 list1에 채우며 위와 같이 반복한다**

**all\_List와 list1이 비어 run에 숫자가 남아있을 경우**

**runs에 남은 run을 저장한다**

**output.txt 에 출력 방법**

**for r in runs:**

**f.write(" ".join(map(str, r)) + "\n")**

**이 코드로 runs에 저장된 run 리스트의 갯수 만큼 반복하여 각 한줄씩**

**문자열로 출력한다.**

**구현한 프로그램의 실행 결과 스크린샷**

**프로그램 실행 화면**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**Input**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**Output**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

자동 생성된 설명**

**프로그램 구현중 어려웠던 점과 해결 방법**

**1. Freeze된 값의 구현 방법**

**고려했던 방법**

* 1. **Freeze된 값에 임의의 큰 수(예: 1000)를 더하여 min 함수에서 제외시키고, 모든 값이 1000 이상이 되면 freeze로 간주하여 다시 1000을 빼서 재정렬하는 방식**
  2. **Freeze된 값을 freeze\_List로 관리하는 방식**

**해결 방안**

**첫 번째 방법은 정수 범위가 0~100으로 제한되어 있어 구현 자체는 가능했지만, 일반 값과 freeze된 값이 섞이면서 복잡성이 증가할 우려가 있었습니다.**

**따라서 두 번째 방법을 선택하여 freeze된 값을 freeze\_List에 따로 저장하도록 구현했습니다.**

**이를 통해 원래 값에 영향을 주지 않으면서**

**디버깅과 관리가 더 명확해졌습니다.**

**2. all\_List에서 list1으로 값을 옮긴 후 처리 방법**

**all\_List에서 list1로 값을 옮긴 후, all\_List의 값을 삭제할지 유지할지를 고민했습니다.**

**초기에는 all\_List의 값을 유지한 채로, list1에서 all\_List로 값을 던져줄 때 인덱스를 증가시키며 관리하려고 했습니다.**

**하지만 다음과 같은 문제로 인해 복잡성이 증가했습니다.**

* 1. **디버깅이 어려움**
  2. **조건에 따라 값 처리 방식이 달라져 코드가 지나치게 복잡해짐**
  3. **예를 들어, all\_List에서 버퍼 크기(5개)의 값을 가져오고 min\_element와 next\_value를 비교해 가져올지 말지를 판단해야 했는데 인덱스 관리와 조건문이 많아 가독성과 유지보수성이 저하됨**

**해결 방안**

**all\_List에서 list1으로 값을 옮긴 후, 해당 값을 all\_List에서 삭제하도록 결정했습니다.**

**이를 위해 all\_List.pop(0)을 사용해 리스트의 첫 번째 값을 제거하면서 list1로 옮겼습니다.**

**이 방식은 불필요한 조건문을 제거하고 값을 가져올 때 all\_List의 맨 앞값만 처리하도록 명료하게 작성할 수 있었습니다.**