BOJ 1700 : 멀티탭 스케줄링

beberiche · 5분 전

algolithms greedy 백준

algolithms

목록 보기 12/12

문제

1700번 제출 맞힌 사람 숏코딩 재채점 결과 채점 현황 내 제출 난이도 기여 질문 게시판

멀티탭 스케줄링 성공

시간 제한	메모리 제한	제출	정답	맞힌 사람	정답 비율
2 초	128 MB	23042	6000	4515	26.445%

문제

기숙사에서 살고 있는 준규는 한 개의 멀티탭을 이용하고 있다. 준규는 키보드, 헤어드라이기, 핸드폰 충전기, 디지털 카메라 충전기 등 여러 개의 전기용품을 사용하면서 어쩔 수 없이 각종 전기용품의 플러그를 뺐다 꽂았다 하는 불편함을 겪고 있다. 그래서 준규는 자신의 생활 패턴을 분석하여, 자기가 사용하고 있는 전기용품의 사용순서를 알아내었고, 이를 기반으로 플러그를 빼는 횟수를 최소화하는 방법을 고안하여 보다 쾌적한 생활환경을 만들려고 한다.

예를 들어 3 구(구멍이 세 개 달린) 멀티탭을 쓸 때, 전기용품의 사용 순서가 아래와 같이 주어진다면,

1. 키보드
2. 헤어드라이기
3. 핸드폰 충전기
4. 디지털 카메라 충전기
5. 키보드
6. 헤어드라이기

키보드, 헤어드라이기, 핸드폰 충전기의 플러그를 순서대로 멀티탭에 꽂은 다음 디지털 카메라 충전기 플러그를 꽂기 전에 핸드폰충전기 플러그를 빼는 것이 최적일 것이므로 플러그는 한 번만 빼면 된다.

입력

첫 줄에는 멀티탭 구멍의 개수 N (1 ≤ N ≤ 100)과 전기 용품의 총 사용횟수 K (1 ≤ K ≤ 100)가 정수로 주어진다. 두 번째 줄에는 전기용품의 이름이 K 이하의 자연수로 사용 순서대로 주어진다. 각 줄의 모든 정수 사이는 공백문자로 구분되어 있다.

출력

하나씩 플러그를 빼는 최소의 횟수를 출력하시오.

예제 입력 1 복사

```
2 7
2 3 2 3 1 2 7
```

예제 출력 1 복사

```
2
```

0

풀이과정

아직은 낯선 그리디 문제.
처음에는, 카운트배열을 생성하여 가장 많이 사용하는 순으로 문자를 정렬한 다음 초기 콘센트에 가장 많이 사용하는 것들을 꽂아두고 시작하는 식으로 하였으나 실패. (가장 많이 사용하는 코드를 뽑았다가 꽂았다가를 반복하기에 절대 최소가 나올 수 없다.)

문제를 해결한 최적해는 다음과 같다.
먼저 콘센트가 비어있다면, 빈 곳에 콘센트를 꽂는다.
콘센트가 꽂아져 있다면 continue;
어떠한 콘센트를 뽑아야하는 상황이라면, 더이상 사용하지 않거나 가장 나중에 사용되는 콘센트를 뽑는다.

정답

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.util.*;

public class Main {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        StringTokenizer st = new StringTokenizer(br.readLine());
        int N = Integer.parseInt(st.nextToken());
        int K = Integer.parseInt(st.nextToken());

        int[] arr = new int[K];
        st = new StringTokenizer(br.readLine());
        for (int i = 0; i < K; i++) {
            arr[i] = Integer.parseInt(st.nextToken());
        }
        List<Integer> list = new ArrayList<>();
        for(int i=0; i<N; i++) {
            list.add(0);
        }

        int cnt = 0;
        for (int i = 0; i < K; i++) {
            if (list.indexOf(arr[i]) >= 0) continue;

            if (list.indexOf(0) >= 0) {
                list.set(list.indexOf(0),arr[i]);
                continue;
            }

            int maxCnt = -1;
            int currIdx = -1;
            for (int k = 0; k < N; k++) {
                int temp = 0;
                for (int j = i+1; j < K; j++) {
                    if(arr[j] == list.get(k)) break;
                    temp++;
                }
                if(temp >= maxCnt) {
                    maxCnt = temp;
                    currIdx = k;
                }
            }
            list.set(currIdx, arr[i]);
            cnt++;
        }
        System.out.println(cnt);
    }
}
```