|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4주차 - A 문제해결기법 보고서 | | | | | |
| 문제 | 빠진 숫자 찾기 | | | | |
| 분반 | 002 | 학번 | 12141540 | 이름 | 박영창 |
| 알고리즘 개요 | * Testcase의 개수 T를 입력 받는다. (1<=T<=100) * 정수 N을 입력 받고 각 수가 30000000 이하인 크기 N의 집합A, B를 입력 받는다. (1<=N<=1000000) * 숫자가 너무 크므로 완전탐색 또는 이진탐색으로 메모리 초과 또는 시간 초과의 결과가 나타났다. 그래서 사용한 방법은 비트마스킹이다. * 크기 1000001의 배열 arr을 선언하고, 이 배열에 비트마스킹을 사용하여 입력 받은 수가 입력 되었다는 정보를 위치에 맞게 저장해 두었다. 먼저 A 집합의 수를 입력 받을 때, 입력 받은 수를 num이라고 한다. 그 다음 arr 배열의 num을 30으로 나눈 값에 해당하는 인덱스 부분에 num을 30으로 나눈 나머지 만큼 비트연산자 중 shift(<<)를 사용해 연산을 진행한다.(31이 입력받으면 arr[1]의 값에 (1 << 1 = 10 - 이진수)를 넣게 된다.)그 다음 원래 그 인덱스 값에 들어있던 값에 더해준다. (값이 중복된다면 OR연산을 사용하겠지만 문제에서 수는 중복되지 않는다는 조건이 있으므로) * A 집합의 모든 수가 이 기법을 통해 입력된 뒤, B 집합의 수를 하나씩 입력 받고, 같은 방식으로 해당 인덱스의 이진법 표기상 위치가 같다면(&연산자 사용) arr 배열에 입력 받은 수가 A 집합의 수로써 존재한다는 뜻이므로 출력하지 않는다. 이런 식으로 arr 배열에 들어 있지 않은 것으로 확인되는 B 집합의 모든 수를 입력 받은 순서대로 출력한다. 출력이 진행될 때 마다 cnt라는 int 변수를 증가시키게 되는데, B 집합의 수를 모두 입력 받는 연산이 끝난 뒤에 cnt가 0이라면, 0을 출력한다. * T를 입력 받는 것을 제외한 위 과정을 T-1번 반복한다. | | | | |
| 성능분석 | A, B 집합의 숫자의 개수 : N  시간 복잡도 : O(N) | | | | |
| 실험결과 | 소요시간 : 2.75s  사용된 메모리 공간 : 419.4304KB | | | | |
| 비고 | 입력과 출력이 너무 많아서 std::ios::sync\_with\_stdio(false); 문장을 사용해도 cin, cout을 사용하는 것보다 scanf, print를 사용하는 것이 훨씬 빨랐습니다… | | | | |