|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 4주차 - B 문제해결기법 보고서 | | | | | |
| 문제 | 비룡 베이커리 | | | | |
| 분반 | 002 | 학번 | 12141540 | 이름 | 박영창 |
| 알고리즘 개요 | * Testcase의 수 T를 입력한다 (0<T<=50) * 손님이 구매하고자 하는 케익의 크기를 나타내는 자연수 S를 입력한다.(0<S<=10000) * 케익 A, B의 조각 수인 N, M을 입력한다. (3<N,M<500) * A 케익과 B 케익의 조각의 크기를 나타내는 N개의 수를 입력한다. 입력하는 과정에서 벡터를 사용해 각각의 벡터에 값을 넣는다. 넣어 주면서 각 케익의 전체 조각 수를 구해 totalN, totalM이라는 변수에 저장하고, 두 케익의 전체 조각 수를 total이라는 변수에 저장한다. 입력 과정에서의 시간 복잡도는 O(N+M)이다. * s보다 total이 작다면, 케익을 팔 수 없으므로 바로 0을 출력하고 다음 testcase로 넘어간다. * S가 total보다 크다면, 각 벡터에 우선 케익에서 한 조각도 쓰지 않는 경우인 0을 삽입한다. 그 다음 이중 포문을 사용해 문제 조건에 맞게 각 케익을 팔 수 있는 조각 수를 모두 벡터에 넣어준다. 벡터를 채워주는 부분에서의 시간 복잡도는 O(N^2+M^2)이다. * 그 다음 각 벡터를 정렬한다. 정렬 과정에서의 시간 복잡도는 O(NlogN+MlogM)이다. * 그 다음 B케익의 벡터 사이즈만큼 반복문을 돌면서, A 케이크의 벡터 내에서 cakeM[i]와 합해서 s가 되는 경우가 있는지 lower\_bound와 upper\_bound 함수를 이용해 찾고 그 위치가 존재한다면, ( upper\_bound - lower\_bound = 존재하는 값의 개수 )이므로 문제에서 원하는 결과값(케익을 팔 수 있는 방법의 수)인 int형 변수 cnt를 그만큼 증가시켜준다. 연산을 진행하면서, cakeM[i]가 s보다 크면, 바로 반복문을 탈출한다. 이 과정에서의 시간 복잡도는 O(M\*logN)이다. * 연산이 끝난 뒤에 cnt를 출력해주고 각 벡터를 비워준다. * T를 입력 받는 것을 제외한 위 과정을 T-1번 반복한다. | | | | |
| 성능분석 | 구매하고자 하는 케익 조각 수 : S , A 케익의 조각 수 : N, B 케익의 조각 수 : M  S가 모든 케익 조각 수 합보다 큰 경우 시간 복잡도 : O(N+M)  S개의 케익 조각 수를 만들 수 있는 경우의 시간 복잡도 : O(N^2+M^2) | | | | |
| 실험결과 | 소요시간 : 0.66s  사용된 메모리 공간 : 2.732MB | | | | |
| 비고 |  | | | | |