-10211 Maximum Subarray-

소프트웨어학과 202021006 김예진

Maximum Subarray 성공 출처

☆

시간 제한	메모리 제한	제출	정답	맞은 사람	정답 비율
1 초	256 MB	5080	2120	1628	42.013%

문제

크기 N인 정수형 배열 X가 있을 때, X의 부분 배열(X의 연속한 일부분) 중 각 원소의 합이 가장 큰 부분 배열을 찾는 Maximum subarray problem(최대 부분배열 문제)은 컴퓨터 과학에서 매우 잘 알려져 있다.

여러분은 N과 배열 X가 주어졌을 때, X의 maximum subarray의 합을 구하자. 즉, $\max_{1 \le i \le j \le N}(X[i]+...+X[j])$를 구하자.

입력

입력 파일의 첫 번째 줄에 테스트 케이스의 수를 의미하는 자연수 T가 주어진다. 그 다음에는 T개의 테스트 케이스가 주어진다.

각 테스트케이스 별로 첫 번째 줄에 배열의 크기 N이 주어진다. ($1 \le N \le 1,000$)

그리고 두 번째 줄에 배열 X의 내용을 나타내는 N개의 정수가 공백으로 구분되어 주어진다. 이때 주어지는 수는 절댓값이 1,000보다 작은 정수이다.

출력

각 테스트케이스 별로 maximum subarray의 합을 줄로 구분하여 출력한다.

예제 입력 1 복사

```
2
5
1 2 3 4 5
5
2 1 -2 3 -5
```

예제 출력 1 복사

```
15
4
```

-> 배열 x에서 부분배열 중 원소의 합이 가장 큰 부분을 찾기

접근 방법

1) 부분합을 나타내는 변수인 sum과 부분합의 최대값을 나타내는 max변수를 nums[0]으로 초기화. (비교할 첫 수를 고정해놓기 위함)

2) Nums배열의 크기 만큼 반복

   1) Sum + nums[i] < nums[i] 이면?

      - 수를 더했지만, 원래의 부분합보다 작아지므로, sum = nums[i]. 이 과정을 하면서 만약에 sum > max이면 max값을 sum으로 갱신해줌.

   2) sum + nums[i] > nums[i] 이면?

      - 수를 더함으로써 부분합이 더 커짐. 따라서 sum 에 nums[i]를 더해주고, 만약 sum이 max보다 크다면 max갱신

3) Max리턴

코드 설명

```cpp
// Baekjoon_10211.cpp
#include <iostream>
#include <vector>
using namespace std;
int maxSubArr(vector<int>& nums){
    int sum = nums[0];
    int max = nums[0];

    for (int i = 1; i < nums.size(); i++){
        if (sum + nums[i] < nums[i]){
            sum = nums[i];
            if (sum > max)
                max = sum;
        }

        else {
            sum += nums[i];
            if (sum > max)
                max = sum;
        }
    }

    return max;
}
```

sum(부분합)과 max(부분합들 중에 최대값)을 나중에 비교할때 초기값을 설정해주기 위해 nums배열의 0번째 값으로 초기화 해준다.

위에 sum과 max를 nums[0]으로 초기화 해줬으므로 for문의 i는 1부터 시작.

1. sum + nums[i] < nums[i]

   - 부분합에 i번째 수를 더한게 i번째 수보다 작다는 것은 i번째 수가 오히려 부분합을 더 작게 만든다는 것을 알 수 있음. 따라서 sum = nums[i]

2. Sum + nums[i] > nums[i]

   - 부분합에 i번째 수를 더했을때 sum이 더 커졌으므로, sum에 nums[i]를 더해준다.

코드 설명

```
26  int main(void){
27      int TestCase;
28      int nums;
29      vector<int> n;
30
31      cin >> TestCase;
32
33      for (int i = 0; i < TestCase; i++){
34          cin >> nums;
35          n.clear();
36          for (int j = 0; j < nums; j++){
37              int temp;
38
39              cin >> temp;
40              n.push_back(temp);
41
42          }
43          cout << maxSubArr(n) <<'\n';
44      }
45
46      return 0;
47
48  }
```

-

1번째 for문 (33~35)
Testcase만큼 반복 (시행 횟수)
- nums배열의 크기를 입력 받음
- 반복을 시작할 때마다 부분합을 초기화 해주기 위해
Clear()이라는 벡터 초기화 함수를 사용.

2번째 for문 (36~42)
- nums의 수만큼 n벡터에다가 수를 입력받음.

1번째 for문(43~44)
- maxSubArr함수에 n벡터를 넣어 결과값 출력

감삼당😎