17.

부분 합

부분 합

- **1. INTRO**
  - Prefix sum 구하기
  - 분산 구하기
  - 2차원으로의 확장
- **2. 예제 : 합이 0에 가장 가까운 구간**
- **3. 문제 : 크리스마스 인형**
  - 생략
- **4. BOJ 문제**

INTRO

- 단일 문제로는 잘 나오지 않음
- 주로 DP등 더 큰 문제를 풀기 전에 pre-calculate
- 시간복잡도의 차수를 한 단계 낮춰주는 효과

INTRO – prefix sum 구하기 (1)

- **Prefix sum (접두사 합)**
  - https://en.wikipedia.org/wiki/Prefix_sum

$$y_0 = x_0$$
$$y_1 = x_0 + x_1$$
$$y_2 = x_0 + x_1 + x_2$$
$$\dots$$

input numbers	1	2	3	4	5	6	...
prefix sums	1	3	6	10	15	21	...

- $psum[i] = \sum_{j=0}^{i} arr[j]$ // psum[i] : 0부터 i까지의 구간 합
- psum[i] = psum[i-1] + arr[i]
- arr[a] + arr[a+1] + … + arr[b-1] + arr[b] = psum[b] - psum[a-1]
- 어떤 구간 [a,b] (0<=a<=b<=N) 에서의 배열의 합을 계산할 때 complexity
  : O(b-a) = O(n)
- Prefix sum를 구해두면 구간 합 쿼리를 O(1) 에 계산 가능

INTRO – prefix sum 구하기 (2)

- https://www.acmicpc.net/problem/11441
  - 매번 구하기 vs 구해두고 쓰기
  - prefix sum을 한번 구해두면 계속 사용 가능 (일종의 DP)
  - https://github.com/taej0127/ps2016/blob/master/prefix%20sum.cpp
  - psum[b] - psum[a-1]
  - a 가 0일 때 ?? Ex) ([0, 3] 같은 쿼리)

INTRO – prefix sum 구하기 (3)

- **해결책**
  - 1. psum[]을 index가 1부터 시작하도록 구현
  - Psum[0] = 0, psum[i] = [1,i] 에서의 합
  - 문제 입력에서 배열 인덱스가 1부터 시작할 때 유용
  - https://github.com/taej0127/ps2016/blob/master/prefix%20sum2.cpp
  - 2. index가 0일때만 예외 처리 (교재)
  - [a,b] 에서 a가 0일 때는 psum[b]
  - 3. C++ STL partial_sum (in <numeric>) 을 이용한 prefix sum
  - 링크 https://github.com/taej0127/ps2016/blob/master/prefix%20sum3.cpp

INTRO - 분산 구하기

- 합을 구했으면 평균은 쉬움
- 분산?? A[i]*A[i] 에 대한 prefix sum (prefix square sum) 을 미리 구함
- sqpsum[i] = sqpsum[i-1] + a[i]*a[i]

$$v = \frac{1}{b-a+1} \cdot \sum_{i=a}^{b} (A[i] - m_{a,b})^2$$

$$= \frac{1}{b-a+1} \cdot \sum_{i=a}^{b} (A[i]^2 - 2A[i] \cdot m_{a,b} + {m_{a,b}}^2)$$

$$= \frac{1}{b-a+1} \cdot \left(\sum_{i=a}^{b} A[i]^2 - 2m_{a,b} \cdot \sum_{i=a}^{b} A[i] + (b-a+1){m_{a,b}}^2 \right)$$

INTRO – 2차원으로의 확장

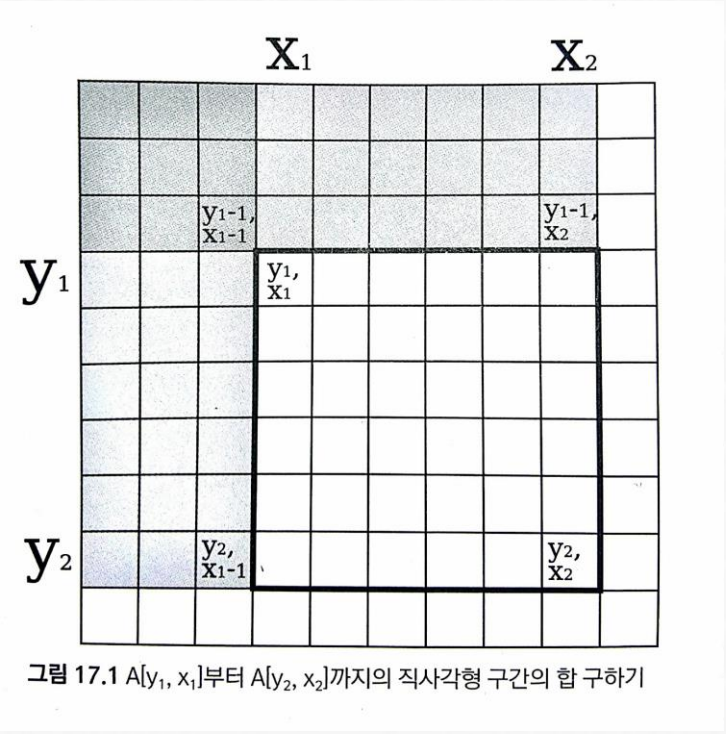

그림 17.1 A[y_1, x_1]부터 A[y_2, x_2]까지의 직사각형 구간의 합 구하기

sum(y1,x1,y2,x2) = psum[y2 x2]
- psum[y2, x1-1]
- psum[y1-1 , x2]
+ psum[y1-1,x1-1]

```
// 어떤 2차원 배열 A[][] 의 부분합 psum[][] 이 주어질 때,
// A[y1,x1] 과 A[y2,x2] 을 양 끝으로 갖는 부분 배열의 합을 반환한다.
int gridSum(const vector<vector<int> >& psum, int y1, int x1, int y2, int x2)
{
int ret = psum[y2][x2];
if(y1 > 0) ret -= psum[y1-1][x2];
if(x1 > 0) ret -= psum[y2][x1-1];
if(y1 > 0 && x1 > 0) ret += psum[y1-1][x1-1];
return ret;
}
```

INTRO

- 구간 곱 … 등등으로 확장 가능

- 주의점 : psum 에서의 overflow 주의

예제 : 합이 0에 가장 가까운 구간

i	0	1	2	3	4	5	6	7	8	9
A[i]	-14	7	2	3	-8	4	-6	8	9	11

- **양수 음수가 모두 포함된 배열 A[] 에서 구간 합이 0에 가장 가까운 구간 찾기**
  - arr[a] + arr[a+1] + … + arr[b-1] + arr[b] = psum[b] - psum[a-1]
  - 좌변 값이 가장 작다는 것은 psum 간의 차이가 가장 적다는 뜻
- **psum[ ]을 구한 후 정렬**
- **인접 구간에서 최소 차이가 답 - Complexity : O(nlgn) (정렬)**
- https://github.com/taej0127/ps2016/blob/master/%ED%95%A9%EC%9D%B4%200%EC%97%90%20%EA%B0%80%EC%9E%A5%20%EA%B0%80%EA%B9%8C%EC%9A%B4%20%EA%B5%AC%EA%B0%84.cpp

문제 : 크리스마스 인형

- 부분 합 + DP

BOJ 문제

- https://www.acmicpc.net/problem/11659
- https://www.acmicpc.net/problem/10986
- https://www.acmicpc.net/problem/3673
- https://www.acmicpc.net/problem/1912 (+ DP?)
- https://www.acmicpc.net/problem/1806

더 공부할 거리

구간 쿼리(최소값, 최대값, 최대 출연빈도 ….) 에 대한 일반적인 해결책을 제공하는 자료구조

=> 구간 트리(Segment Tree) (24장)