Difference Array & Prefix Sum Array

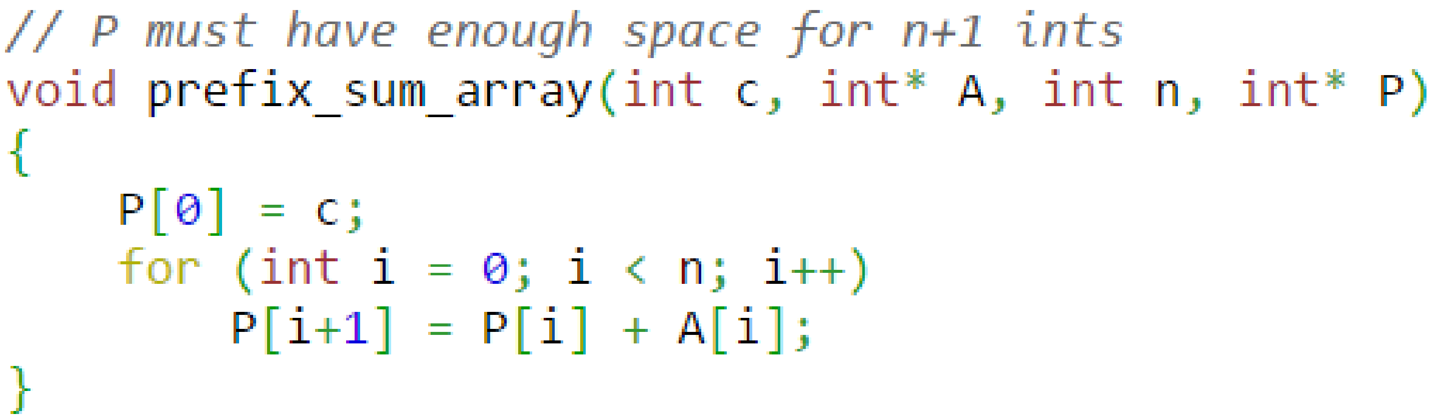
Bruce Nan

Prefix Sum Array

- **Definition**: Given an array of numbers *A*, we replace each element with the sum of itself and all the elements preceding it.
- Example:
  - A=[3, 5, 4, 1, 2]
  - P(A)=[3, 8, 12, 13, 15]

Time complexity

- Linear time O(N)



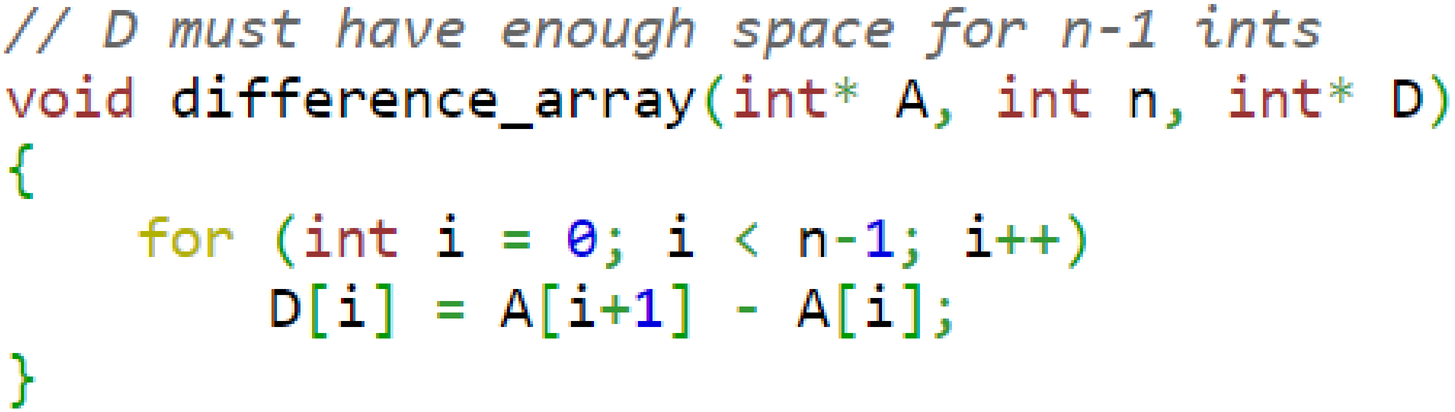
```
// P must have enough space for n+1 ints
void prefix_sum_array(int c, int* A, int n, int* P)
{
    P[0] = c;
    for (int i = 0; i < n; i++)
        P[i+1] = P[i] + A[i];
}
```

Difference Array

- **Definition**: Given an array of numbers, we can construct a new array by replacing each element by the difference between itself and the previous element, except for the first element, which we simply ignore.
- Example:
- A=[3, 5, 4, 1, 2]
- D(A)=[5-3, 4-5, 1-4, 2-1] = [2, -1, -3, 1]
- D(A) = [3, 2, -1, -3, 1]

Time complexity

- Linear time O(N)



```
// D must have enough space for n-1 ints
void difference_array(int* A, int n, int* D)
{
    for (int i = 0; i < n-1; i++)
        D[i] = A[i+1] - A[i];
}
```

Analysis

- Difference array and Prefix Sum array carry out **reverse process**
- Given an array A
  - D(P(A)) = A
    - A=[9,2,6,3,1,5,0,7]
    - P(A)=[9, 11, 17, 20, 21, 26, 26, 33]
    - D(P(A)) = [9,2,6,3,1,5,0,7]
  - P(A0, D(A)) = A

Application

- Prefix Sum Array
  - Used for frequent reference to range sums
  - Example: http://www.spoj.com/problems/SUBSEQ/

Application(2)

- Difference array is used to keep track of an array when ranges of said array can be updated all at once.

Application(3)

- If we have array A and add an increment k to elements A_i,A_{i+1},...,A_{j-1}, then notice that D_0,D_1,...,D_{i-2} are not affected; that $D_{i-1} = A_i - A_{i-1}$ is increased by k; that D_i,D_{i+1},...,D_{j-2} are not affected; that $D_{j-1} = A_j - A_{j-1}$ is decreased by k; and that D_j,D_{j+1},... are unaffected.
- If we are required to update many ranges of an array in this manner, we should keep track of D rather than A itself, and then integrate at the end to reconstruct A.

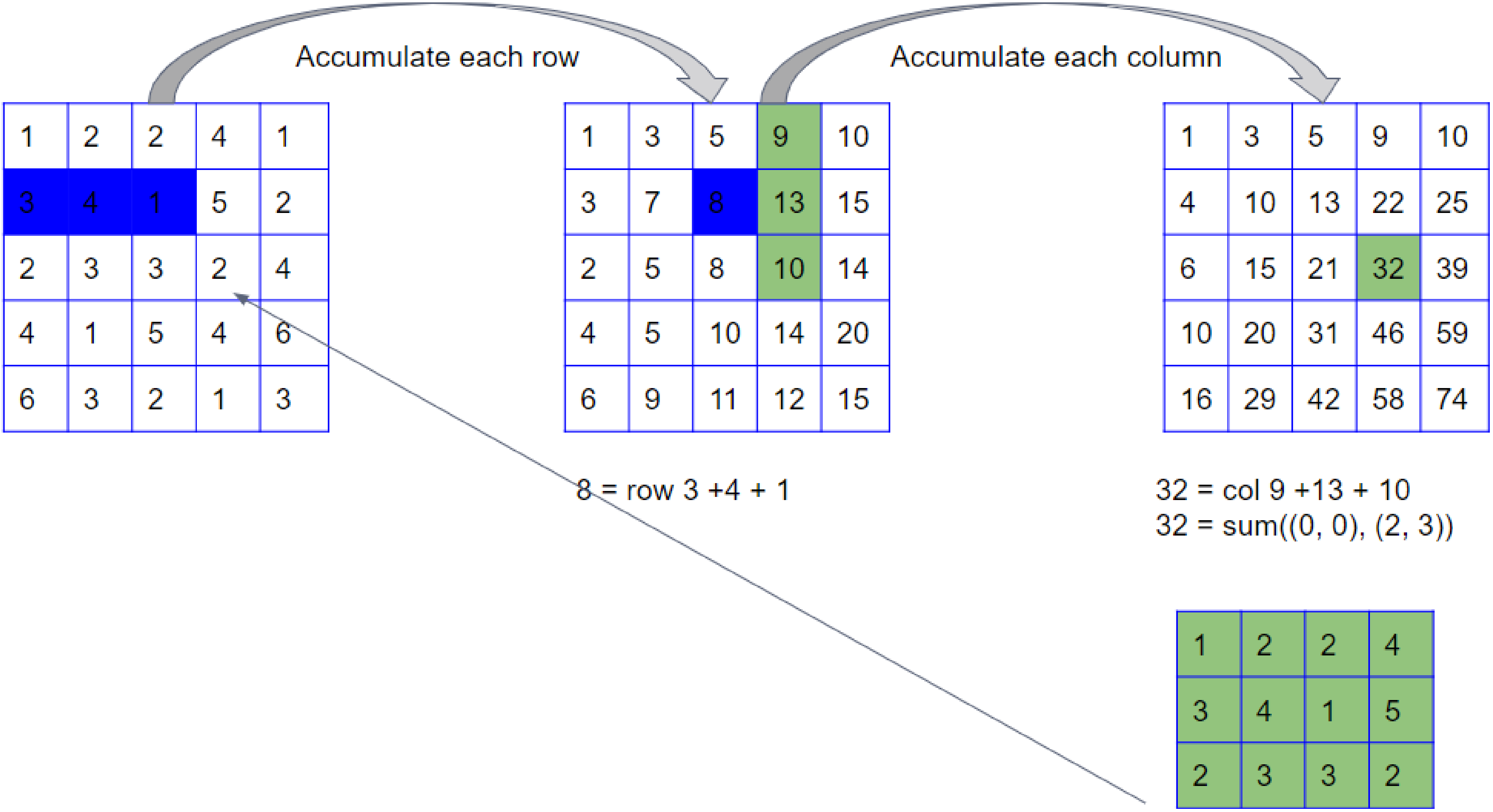
Application(3)

- Difference Array Examples:
  - Battle Positions (https://dmoj.ca/problem/seed3)
  - Wireless (https://dmoj.ca/problem/ccc09s5)

How about 2D sum query

- PSA from 1D to 2D
  - What if instead 1D array, we have 2D one
- Query is to find a rectangle sum
  - E.g. Sum((2, 4), (5, 7)) where (2, 4) is the top left corner of a sub-matrix?
- Can we still use prefix sum array?
  - Create new Array S.
  - For each row in A, create its cumulative sum in S
  - In S, in-place, create cumulative sum for each column
  - Now S(i, j) = Sum((0, 0), (i, j))

2D prefix sum array



Accumulate each row

Accumulate each column

1	2	2	4	1
3	4	1	5	2
2	3	3	2	4
4	1	5	4	6
6	3	2	1	3

1	3	5	9	10
3	7	8	13	15
2	5	8	10	14
4	5	10	14	20
6	9	11	12	15

8 = row 3 +4 + 1

1	3	5	9	10
4	10	13	22	25
6	15	21	32	39
10	20	31	46	59
16	29	42	58	74

32 = col 9 +13 + 10
32 = sum((0, 0), (2, 3))

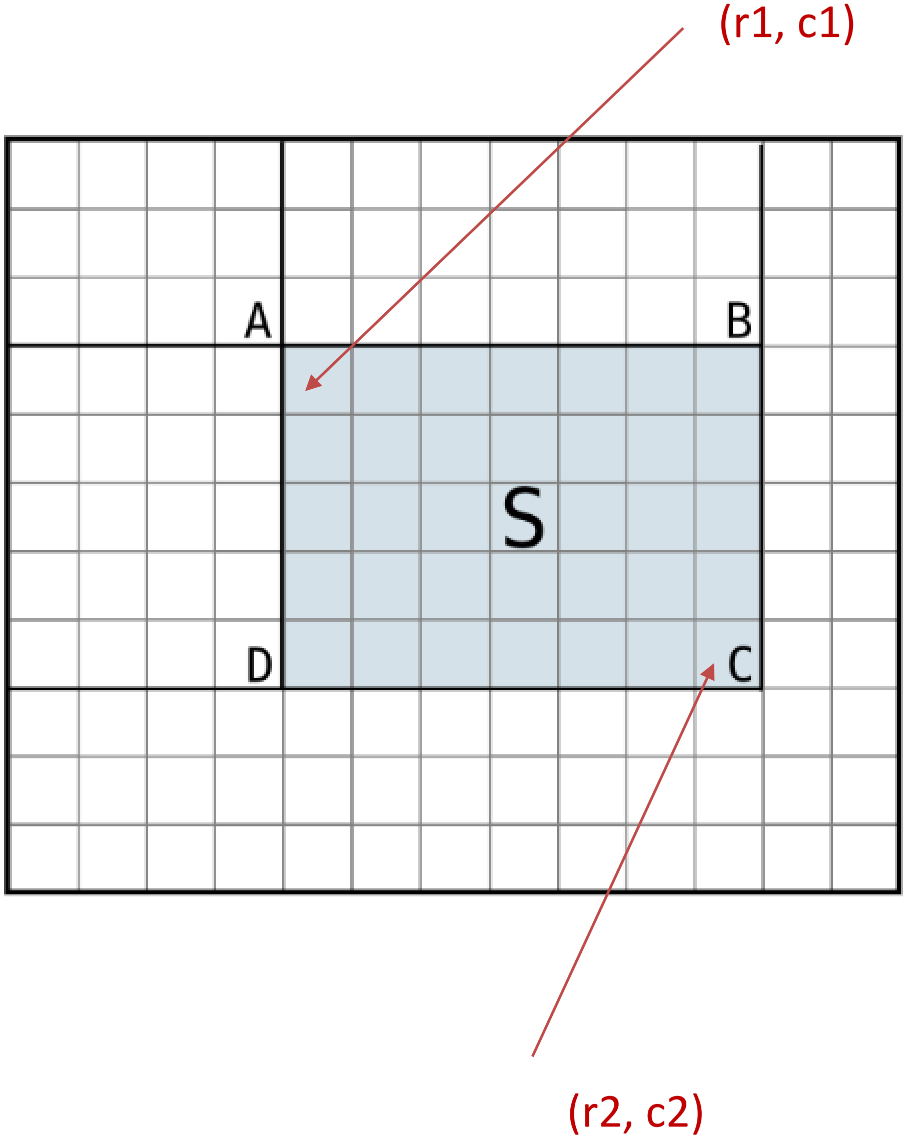
1	2	2	4
3	4	1	5
2	3	3	2

2D sum query

How to compute the sum in submatrix S?

- Sum of C is the sum from top-left corner to C
- Let's remove sum of B
- Let's remove sum of D
- Sum of A is removed twice
- Let's add A back
- S = C – B – D + A

- A: psa[r1-1][c1-1]
- B: psa[r1-1][c2]
- C: psa[r2][c2]
- D: psa[r2][c1-1]
- S = psa[r2][c2] – psa[r1-1][c2] – psa[r2][c1-1] + psa[r1-1][c1-1]



How about 3D?

- Same idea to calculate psa[i][j][k]
  - Accumulate sum over i
  - Accumulate sum over j
  - Accumulate sum over k
- How about 3D sum query?

  Sum of (i1, j1, k1) to (i2, j2, k2) (inclusive)
  = psa[i2][j2][k2] – psa[i1-1][j2][k2] – psa[i2][j1-1][k2] – psa[i2][j2][k1-1] + psa[i1-1][j1-1][k2] + psa[i1-1][j2][k1-1] + psa[i2][j1-1][k1-1] – psa[i1-1][j1-1][k1-1]

How about 2D update

- DA from 1D to 2D
  - What if instead 1D array update, we have 2D update
- Update is to add/subtract value in a submatrix
- Can we still use difference array?
  - Yes
  - Update 4 corners:

    ```
    dif[r1][c1] += val      dif[r2+1][c1] -= val
    dif[r1][c2+1] -= val   dif[r2+1][c2+1] += val
    ```