HUST

ĐẠI HỌC BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

APPLIED ALGORITHMS

ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

APPLIED ALGORITHMS

Data structures and advanced techniques
Cumulative array, 2-pointing technique

ONE LOVE. ONE FUTURE.

CONTENTS

- Cumulative array
- 2-pointing technique

Cumulative array

- **Illustrative exercise (P.02.02.01).** Given the sequence $a_1, a_2, \ldots, a_n$. Execute Q queries, each query is characterized by a pair of indices (i, j) in which we need to calculate the sum $a_i + a_{i+1} + \ldots + a_j$.
- Direct algorithm:
  - For each query, we traverse the array from a_i to a_j to calculate the sum of the elements in this subrange.
  - The worst-case complexity of each query is O(n).
  - The worst-case complexity of Q queries is O(Qn).

```
sum(i, j){
    T = 0;
    for k = i to j do
       T = T + a_k;
    return T;
}
```

Cumulative array

- **Illustrative exercise (P.02.02.01).** Given the sequence $a_1, a_2, \ldots, a_n$. Execute Q queries, each query is characterized by a pair of indices (i, j) in which we need to calculate the sum $a_i + a_{i+1} + \ldots + a_j$.
- Algorithm uses cumulative array:
  - Calculate the sum of elements in cumulative array $S_k = a_1 + a_2 + \ldots + a_k$ ($k = 1, 2, \ldots, n$), $S_0 = 0$
    - Complexity $O(n)$
  - The query (i, j) has the value of $S_j - S_{i-1}$
    - Complexity of each query: O(1)
    - Complexity of Q queries: $O(n) + O(Q)$

```
Input a1, a2, …, an and Q;
S0 = 0;
for k = 1 to n do
   Sk = Sk-1 + ak;

for q = 1 to Q do {
   Input i, j;
   res = Sj – Si-1;
   output(res);
}
```

Cumulative array

- Given 2D array $a[1..n, 1..m]$, cumulative array $S[1..n, 1..m]$ is defined as following:
  - $S[0, j] = 0$, $S[i, 0] = 0$, $i = 0, 1, \ldots, n$ and $j = 0, 1, \ldots, m$
  - $S[i, j] = \sum_{k=1}^{i} \sum_{q=1}^{j} a[k, q]$, $i = 1, \ldots, n$ and $j = 1, \ldots, m$
  - Recursive formula : $S[i, j] = S[i-1,j] + S[i, j-1] - S[i-1, j-1] + a[i,j]$
- Algorithm to calculate cumulative array
  - Complexity $O(nm)$

```
for i = 0 to n do S[i,0] = 0;
for j = 0 to m do S[0,j] = 0;
for i = 1 to n do {
   for j = 1 to n do {
       S[i,j] = S[i-1,j] + S[i,j-1] –
             S[i-1,j-1] + a[i,j];
   }
}
```

Cumulative array

- **Illustrative exercise (P.02.02.02).** Given the 2D array $a[1..n, 1..m]$. Execute Q queries, each query is characterized by a set of indices (a, b, c, d) and defined as follows:

$$\text{query}[a, b, c, d] = \sum_{k=a}^{c} \sum_{q=b}^{d} a[k, q]$$

- Direct algorithm:
  - Each query, we perform 2 nested loops to traverse all elements and calculate the sum
  - The worst-case complexity of each query is O(nm)

Cumulative array

- **Illustrative exercise (P.02.02.02).** Given the 2D array $a[1..n, 1..m]$. Execute Q queries, each query is characterized by a set of indices (a, b, c, d) and defined as follows:

$$\text{query}[a, b, c, d] = \sum_{k=a}^{c} \sum_{q=b}^{d} a[k, q]$$

- Algorithm that uses cumulative algorithm:
  - Formular: query[a, b, c, d] = S[c, d] – S[c, b-1] – S[a-1, d] + S[a-1, b-1]
  - The worst-case complexity of each query is O(1)

2-pointing technique

- In many problems, we have to traverse a sequence a_1, a_2, ..., a_n to search for objects characterized by 2 indices (i, j) on the sequence (for example, a subsequence consists of consecutive elements or pairs of 2 elements of a sequence) that satisfies some certain properties.
  - Use 2 nested loops to traverse through all pairs of 2 indices (i, j): complexity O(n^2)
  - Use 2 pointers to move in 1 direction or to move in 2 opposite directions: complexity O(n)

2-pointing technique

- **Illustrative exercise 2.1 (P.02.02.03).** Given the sequence a[1], a[2], . . ., a[n] is sorted in ascending order (distinct elements: no elements with the same value). Given the value Q, count the number of pairs of 2 indices i and j such that a[i] + a[j] = Q.

- Direct algorithm
  - Use 2 nested loops to browse through all pairs (i, j) and check the condition a[i] + a[j] = Q
  - Complexity $O(n^2)$

```
res = 0;
for i = 1 to n do {
  for j = i+1 to n do {
     if a[i] + a[j] = Q then
        res = res + 1;
  }
}
Output(res);
```

2-pointing technique

- **Illustrative exercise 2.1 (P.02.02.03).** Given the sequence a[1], a[2], . . ., a[n] is sorted in ascending order (distinct elements: no elements with the same value). Given the value Q, count the number of pairs of 2 indices i and j such that a[i] + a[j] = Q.

- Algorithm that uses 2-pointing technique
  - Variable *i* moves from left to right and variable *j* moves from right to left through the sequence
  - Complexity O(*n*)

```
res = 0;
i = 1; j = n;
while i < j do {
  if a[i] + a[j] = Q then {
     res = res + 1; i = i + 1; j = j – 1;
  }else if a[i] + a[j] < Q then
     i = i + 1;
  else
     j = j – 1;
}
Output(res);
```

2-pointing technique

- **Illustrative exercise 2.2 (P.02.02.04).** Given a sequence of non-negative numbers a[1], a[2], . . ., a[n]. Given the value Q, find the longest subsequence (consisting of a number of consecutive elements: a[i], a[i+1],…,a[j]) whose sum is less than or equal to Q.

- Direct algorithm
  - Use 2 nested loops to consider all starting and ending positions of a subsequence and check whether the sum is less than or equal to Q?
  - Complexity O(n^2)

```
res = 0;
for i = 1 to n do {
  S = 0;
  for j = i to n do {
    S = S + a[j];
    if S <= Q then {
       res = max(res, j – i + 1);
    }
  }
}
Output(res);
```

2-pointing technique

- **Illustrative exercise 2.2 (P.02.02.04).** Given a sequence of non-negative numbers a[1], a[2], . . ., a[n]. Given the value Q, find the longest subsequence (consisting of a number of consecutive elements: a[i], a[i+1],…,a[j]) whose sum is less than or equal to Q.

- Algorithm that uses 2-pointing technique
  - Variable *L* moves from left to right and variable *R* moves from right to left through the sequence
  - Complexity O(*n*)

```
res = 0; S = 0;
L = 1;
for R = 1 to n do {
   S = S + a[R];
   while S > Q do {
      S = S - a[L]; L = L + 1;
   }
   res = max(res, R - L + 1);
}
Output(res);
```

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !