S222 - Gathering

David Wai {wjx}

2022-01-29

香港電腦奧林匹克競賽
Hong Kong Olympiad in Informatics

Background

Problem idea by David Wai

Preparation by David Wai, Joseph Cheung

Figures by Tony Wong

Problem Restatement

Given N friends living at A_1, A_2, ..., A_N respectively

Choose K of them and select a gathering position x

Cost of each move = number of friends at the front

Find the minimum total cost

Sample Input	Sample output
10 3 2 3 2 5	1
10 5 3 1 1 7 10 4	6
10 5 4 3 6 1 10 9	27

Statistics

Task	Attempts	Max	Mean	Std Dev
S222 - Gathering	53	100	18.641	26.289

Subtasks					
7: 35	10: 25	15: 9	10: 10	21: 7	37: 3

First solved by **s16325** at **1:33:26**

SUBTASKS

For all cases:

$1 \leq N, L \leq 10^6$

$1 \leq K \leq N$

$0 \leq A_i \leq L$

	Points	Constraints
1	7	$K = 2$ $1 \leq N \leq 5000$
2	10	$K = 3$
3	15	$K = N$
4	10	$1 \leq N, L \leq 500$
5	21	$1 \leq N \leq 5000$
6	37	No additional constraints

Subtask 1

Subtask 1 (7%): $K = 2$, $1 \le N \le 5000$

Sample Input 1	Sample output 1
10 3 2 3 2 5	1

- Exhaust which 2 friends to choose
- Suppose we are choosing A_i and A_j $(i \neq j)$
  - Choose any point x between their position
  - Before they both move to x, there is only 1 friend at the front
  - Sum of their cost = abs(A_i - A_j)
- Output the minimum total cost

Time complexity: $O(N^2)$

Subtask 2

Subtask 2 (10%): $K = 3$

- To find the minimum total cost, we just need to exhaust every consecutive K friends after sorting by their living positions
  - Suppose we choose A_l to A_r except A_m ($l < m < r$), replace by A_l to A_{r-1} or A_{l+1} to A_r will get a lower cost

Sample Input 2 (Sorted)	Sample output 2
`10 5 3` `1 1 4 7 10`	`6`

Subtask 2

Subtask 2 (10%): $K = 3$

- To find the minimum total cost, we just need to exhaust every consecutive 3 friends after sorting by their living positions
- The order they move does not affect the total cost
  - When someone moves one unit forward and the number of friends at the front decreases by P, the number of friends in front of the P friends will increase by 1

Sample Input 2 (Sorted)	Sample output 2
10 5 3 1 1 4 7 10	6

Subtask 2

Subtask 2 (10%): $K = 3$

- To find the minimum total cost, we just need to exhaust every consecutive 3 friends after sorting by their living positions
- The order they move does not affect the total cost so we can calculate the total cost easily
  - Let F_i = the number of friends in front of the i^{th} friend
  - Total cost = sum(F_i * abs(A_i - x))

Sample Input 2 (Sorted)	Sample output 2
10 5 3 1 1 4 7 10	6

Subtask 2

Subtask 2 (10%): $K = 3$

- To find the minimum total cost, we just need to exhaust every consecutive 3 friends after sorting by their living positions
- The order they move does not affect the total cost so we can calculate the total cost easily
- Exhaust every position x to find the minimum total cost

Time complexity: $O(N \log N)$

Sample Input 2 (Sorted)	Sample output 2
`10 5 3` `1 1 4 7 10`	`6`

Subtask 4

Subtask 4 (10%): 1≤ N, L ≤ 500

- To find the minimum total cost, we just need to exhaust every consecutive K friends after sorting by their living positions
- The order they move does not affect the total cost so we can calculate the total cost easily
- Exhaust every position x to find the minimum cost
- $O(K)$ time is needed to calculate the total cost for each selected K and x

Time complexity: $O(NKL)$

Subtask 3

Subtask 3 (15%): $K = N$

- We don't need to consider which K friends to select
- Exhaust the position x
- Calculate the total cost by using two pointers
  - Let F_i = the number of friends in front of the i^{th} friend
  - Let $sum1$ = sum(F_i) ($0 \le A_i \le x$), $sum2$ = sum(F_i) ($x < A_i \le L$)
  - When x increases by 1, new total cost = previous total cost + $sum1$ - $sum2$
  - $sum1$ and $sum2$ can be maintained by using two pointers

Time complexity: $O(N \log N)$

Subtask 3 - Solution 2

Subtask 3 (15%): $K = N$

- In subtask 2, you may find that choosing the median of 3 A_i can get the minimum total cost
- Actually this is true for any K
  - Let F_i = the number of friends in front of the i^{th} friend
  - Let $sum1$ = sum(F_i) ($0 \le A_i \le x$), $sum2$ = sum(F_i) ($x < A_i \le L$)
  - When x increases by 1, new total cost = previous total cost + $sum1$ - $sum2$
  - As x increases, $sum1$ increases and $sum2$ decreases → total cost decreases then increases
  - Total cost attains its minimum when $sum1 = sum2$
  - If K is odd, $x = A_{(K+1)/2}$
  - If K is even, x can be any integer between $A_{K/2}$ and $A_{K/2+1}$

Time complexity: $O(N \log N)$

Subtask 5

Subtask 5 (21%): $1 \leq N \leq 5000$

- By the observation in subtask 3 solution 2, we don't need to exhaust the position x
- Exhaust every consecutive K friends after sorting by their living positions
- Calculate the cost in $O(K)$ time

Time complexity: $O(NK)$

Full Solution

Subtask 6 (37%): No additional constraints

- Exhaust every consecutive K friends after sorting by their living positions
- When choosing A_i to A_{i+K-1}
  - Let F_i = the number of friends in front of the i^{th} friend
  - Let $mid = i + (K + 1) / 2 - 1$
  - Total cost

    $= \text{sum}(F_j * \text{abs}(A_j - A_{mid}))\ (i \le j \le i+K-1)$

    $= \text{sum}(F_j * (A_{mid} - A_j))\ (i \le j \le mid) + \text{sum}(F_j * (A_j - A_{mid}))\ (mid < j \le i+K-1)$

    $= \text{sum}(F_j * A_j)\ (mid < j \le i+K-1) - \text{sum}(F_j * A_j)\ (i \le j \le mid) + (K / 2 * A_{mid}$ (if K is odd))

Full Solution

Subtask 6 (37%): No additional constraints

- Exhaust every consecutive K friends after sorting by their living positions
- When choosing A_i to A_{i+K-1}
  - Let F_i = the number of friends in front of the i^{th} friend
  - Let $mid = i + (K + 1) / 2 - 1$
  - Total cost = sum($F_j * A_j$) ($mid < j \leq i+K-1$) - sum($F_j * A_j$) ($i \leq j \leq mid$) + ($K / 2 * A_{mid}$ (if K is odd))
- For the next consecutive K friends A_{i+1} to A_{i+K}, we can make use of the previous cost
- All the calculations can be done by using two pointers

Time complexity: $O(N \log N)$

Full Solution 2

Subtask 6 (37%): No additional constraints

- Exhaust every consecutive K friends after sorting by their living positions
- If you know partial sum
  - Let $sum1_i$ = sum(A_i) ($1 \le i \le N$)
  - Let $sum2_i$ = sum(A_i * i) ($1 \le i \le N$)
  - Let $sum3_i$ = sum(A_i * $(N - i + 1)$) ($1 \le i \le N$)
- When choosing A_i to A_{i+K-1}
  - Let F_i = the number of friends in front of the i^{th} friend
  - Let $mid = i + (K + 1) / 2 - 1$
  - Total cost = sum(F_j * A_j) ($mid < j \le i+K-1$) - sum(F_j * A_j) ($i \le j \le mid$) + (K / 2 * A_{mid} (if K is odd)), which can be calculated in $O(1)$ time by using the above partial sums

Time complexity: $O(N \log N)$