Important: Be clear and precise. It is important to be able to spot logical mistakes especially in what you write. Please make sure that what you write is correct. Logically incorrect statements may attract negative marks.

Individual effort only. Please report any accidental sources of information that helped you solve any problems.

We model the two headed disk system as follows. There are n tracks and two heads on a line. The heads cannot cross. Let the tracks be at positions $i; 1 \leq i \leq n$.

A request sequence is of the form $i_1, \ldots, i_k$. For a request, one of the heads has to move to serve the request. The cost is the cost of head movement. For instance if the head is at position i on the line and it moves to position j, the cost of movement is $|j - i|$.

Our objective as earlier is to compare online algorithms to optimal offline ones.

1. Consider the algorithm: move the closest head. Show that there are request sequences where the cost of this algorithm is arbitrarily large while the cost of an optimal offline algorithm is a finite number.

2. Consider the following algorithm. Let the heads of the algorithm be denoted by H_1 and H_2. We will also refer to the position of this head by H_1. Say H_1 is the head on the left. Our algorithm is the following: If the request is to the left of H_1 move the head at H_1 to serve the request. If it is to the right of H_2 serve with the head at H_2. If it is in the middle of the two, serve with the closest head and *move the other head the same distance towards the request.*

   Note that this algorithm does something non obvious. When the request is in the middle, the algorithm does twice the work. However you will show below that the total cost of this algorithm is within twice of the optimal offline algorithm.

   Assume that the optimal offline algorithm also has its heads: say O_1 and O_2 serving each request. O_1 is the one on the left. We will also assume that for each request it moves only one head. Initially all heads are at 1.

   For two points on the line, let $d(x, y)$ denote the distance between them. For instance $d(H_1, O_1)$ is the distance between the head H_1 and O_1. Consider the following function $\Phi = d(H_1, H_2) + 2d(O_1, H_1) + 2d(O_2, H_2)$.

   Consider any sequence of requests $r_1, \ldots, r_t$. We will consider this sequence of requests one by one and first the optimal will move a head to serve the request and then the algorithm mentioned above will. With each move you will figure out what happens to Φ.

   Suppose on a request r_i, one of the optimal heads moves a distance d.

   Q1 What is the maximum increase in Φ ? In other words, bound the increase in Φ from above by the best function of d that you can.

   If now the request is to the right of H_2 and the algorithm moves a distance d'.

   Q2 Prove that Φ must decrease. What is the minimum decrease? In other words, bound the decrease in Φ from below by a function of d'. Do a similar analysis (no need to write it) when the request is to the left of H_1.

Q3 If the request is between H_2 and H_1 show that again Φ must decrease. By how much?

Q4 Prove using what you have done so far that the cost of the online algorithm is at most twice the cost of the optimal offline algorithm.

Hint: For each request prove that the cost of the online algorithm plus the change in Φ is at most twice the cost of the offline algorithm.