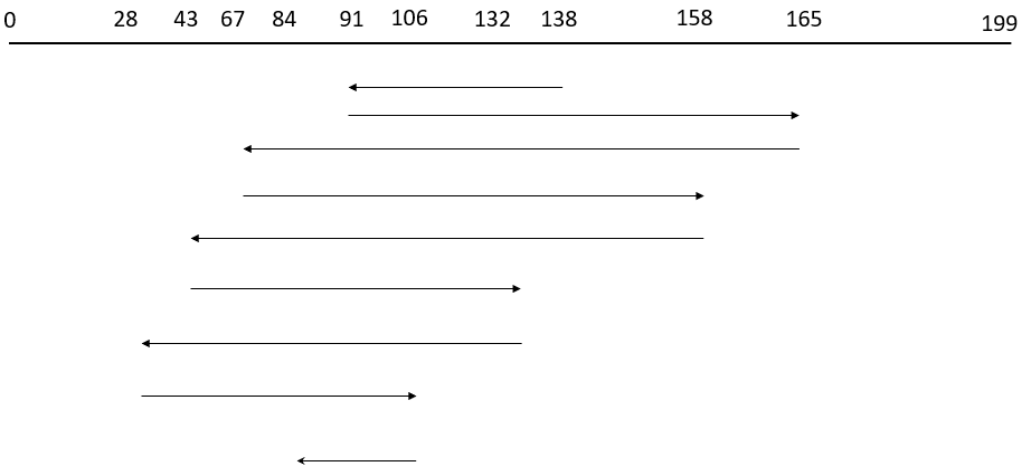
Systèmes d'exploitation II
TD 4 : Gestion du disque

Exercice I:

Soit un disque contenant 200 pistes, numérotées de 0 à 199, et que la dernière requête concernait la piste 112, et une requête pour la piste 138 est en cours.

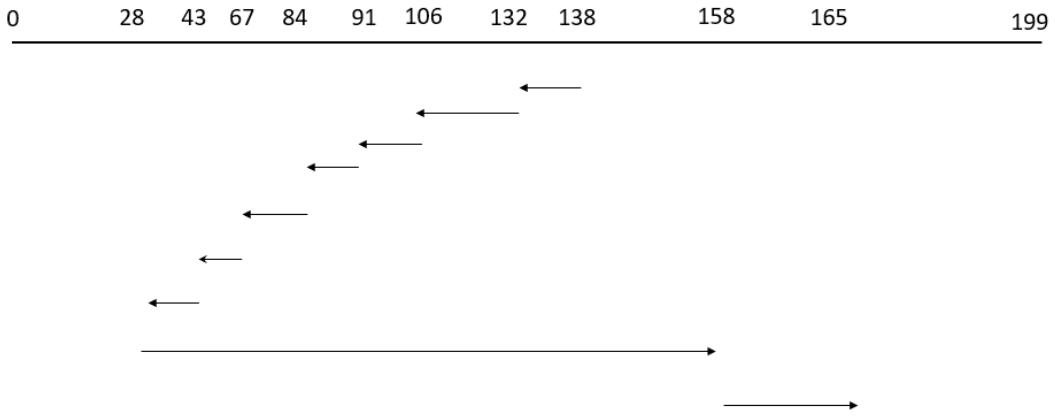
La liste des nouvelles requêtes dans l'ordre d'arrivée est la suivante: 91,165, 67, 158, 43, 132, 28, 106, 84. Représenter le déplacement et calculer le déplacement pour les algorithmes suivants:

1) FIFO



Dep (FIFO) =|138-91|+ |91-165|+ |165-67|+ |67-158|+ |158-43|+ |43-132|+ |132-28|+ |28-106|+ |106-84|

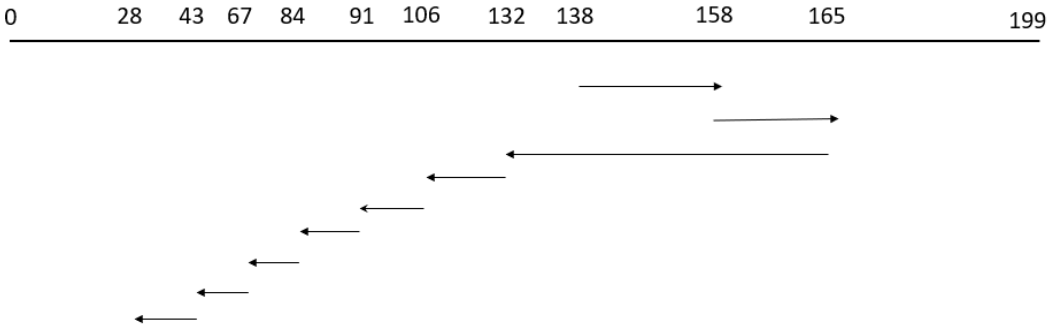
2) PCTR



Dep (PCTR) =|138-106|+ |106-91|+ |91-84|+ |84-67|+ |67-43|+ |43-28|+ |28-158|+ |158-165|

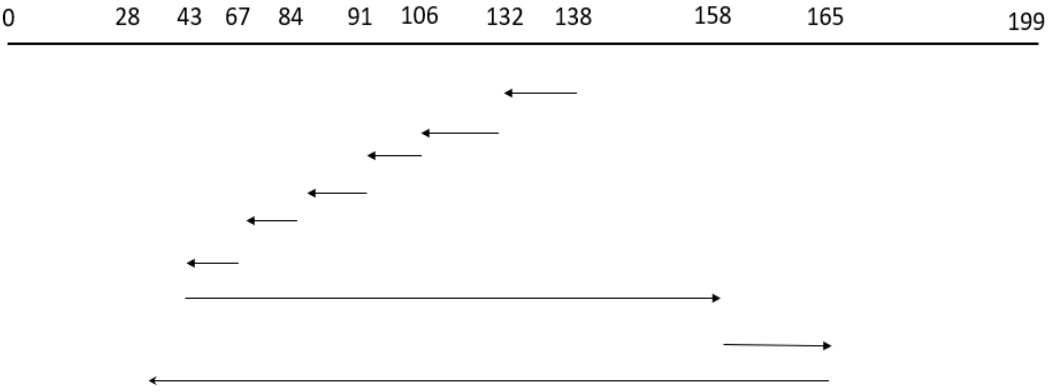
3) SCAN

Selon l'énoncé la dernière requête est 112 et maintenant la requête courante est 138. Ainsi e sens du traitement est : →



Dep (SCAN) =|138-158|+ |158-165|+ |165-132|+ |132-106|+ |106-91|+ |91-84|+ |84-67|+ |67-43| + |43-28|

4) LOOK



Dep (LOOK) =|138-132|+ |132-106|+ |106-91|+ |91-84|+ |84-67|+ |67-43|+ |43-28|+ |28-158| + |158-165|

5) Expliquer ce qu'est l'équité pour un algorithme d'ordonnancement.

L'étiqueté est d'assurer la visite de toute les pistes dans un temps optimale en évitant la famine.

6) Pourquoi SCAN est plus équitable que PCTR?

Car il évite la famine contrairement à la méthode PCTR.

7) Citer les objectifs de gestion de disque. Expliquer la différence avec les objectifs de la gestion de la mémoire centrale.

L'objectif du gestionnaire du disque est l'optimisation des déplacements, contrairement au gestionnaire de la mémoire centrale qui vise à optimiser l'utilisation de l'espace.

Exercice II (Travail à rendre avant le 18/052020) :

Proposer une implémentation de la technique PCTR et FIFO pour un nombre de requêtes saisies à partir du clavier.

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

void fifo (int requests[], int num_requests, int head) {
    int total_distance = 0;
    int current_position = head;

    printf("Ordre des requêtes traitées: ");
    for (int i = 0; i < num_requests; i++) {
        total_distance += abs(current_position - requests[i]);
        current_position = requests[i];
        printf("%d ", requests[i]);
    }
    printf("\nDistance total de déplacement: %d\n", total_distance);
}

// Function to find the closest request
int proche_demande(int requests[], int num_requests, int current_position, int
processed[]) {
    int min_distance = INT_MAX;
    int closest_index = -1;

    for (int i = 0; i < num_requests; i++) {
        if (processed[i]==-1) {
            int distance = abs(current_position - requests[i]);
            if (distance < min_distance) {
                min_distance = distance;
                closest_index = i;
            }
        }
    }
    return closest_index;
}
```

```c
void pctr(int requests[], int num_requests, int head) {

    int total_distance = 0;
    int current_position = head;
    int processed[num_requests];

    // Initialize processed array
    for (int i = 0; i < num_requests; i++) {
        processed[i] = -1;
    }

    printf("Order of requests processed: ");
    for (int i = 0; i < num_requests; i++) {
        int closest_index = proche_demande(requests, num_requests,
current_position, processed);
        if (closest_index == -1) {
            break; // No more unprocessed requests
        }

        int distance = abs(current_position - requests[closest_index]);
        total_distance += distance;
        current_position = requests[closest_index];
        processed[closest_index] = 1;

        printf("%d ", requests[closest_index]);
    }
    printf("\nDistance total de déplacement: %d\n", total_distance);
}

int main() {

    int num_requests;

    // Input number of requests
    printf("Enter le nombre des demandes de disque: ");
    scanf("%d", &num_requests);

    int requests[num_requests];

    // Input the requests
    for (int i = 0; i < num_requests; i++) {
        printf("Enter request %d: ", i + 1);
        scanf("%d", &requests[i]);
    }

    int head;
```

```
    // La position initiale dans le disque
    printf("Enter the initial position of the disk head: ");
    scanf("%d", &head);

    // Perform FCFS disk scheduling
    fifo(requests, num_requests, head);

    return 0;
}
```