מערכות הפעלה מטלה 3

מגיש: חיים חוגי 207686650

הפלט של ריצת התוכנית

- בצילום תוכלו לראות את הפלט לכל input אחר שנכניס לאלגוריתם.
- למשל ב input 1 ניתן לראות שהסכום הממוצע של ה turnaround באלגוריתם FCFS הוא 17.25 מחזורי שעון.

```
FCFS: mean turnaround 17.25                    main.js:23
LCFS (NP): mean turnaround 19.25               main.js:24
LCFS (P): mean turnaround 20                   main.js:25
RR: mean turnaround 23.5                       main.js:26
SJF: mean turnaround 16.5                      main.js:27
```

Input 1

```
FCFS: mean turnaround 28.125                   main.js:23
LCFS (NP): mean turnaround 22.75               main.js:24
LCFS (P): mean turnaround 24.625               main.js:25
RR: mean turnaround 36.75                      main.js:26
SJF: mean turnaround 21.625                    main.js:27
```

Input 2

```
FCFS: mean turnaround 11.5                     main.js:23
LCFS (NP): mean turnaround 11.5                main.js:24
LCFS (P): mean turnaround 11.5                 main.js:25
RR: mean turnaround 11.5                       main.js:26
SJF: mean turnaround 11.5                      main.js:27
```

Input 3

```
FCFS: mean turnaround 14                       main.js:23
LCFS (NP): mean turnaround 15                  main.js:24
LCFS (P): mean turnaround 17.5                 main.js:25
RR: mean turnaround 20.5                       main.js:26
SJF: mean turnaround 14                        main.js:27
```

Input 4

```
FCFS: mean turnaround 6                        main.js:23
LCFS (NP): mean turnaround 6                   main.js:24
LCFS (P): mean turnaround 7                    main.js:25
RR: mean turnaround 8                          main.js:26
SJF: mean turnaround 6.5                       main.js:27
```

Input 5

```javascript
for (let index = 1; index < 6; index++) {
    var text = fs.readFileSync("./input" + index + ".txt").toString('utf-8');
    let inputArray = text.split('\n').toString().replace(/\n|\r/g, "").trim();
    inputArray = inputArray.split(',');
    for (let index = 0; index < inputArray.length; index++) {
        if (!isNumeric(inputArray[index])) {
            inputArray.splice(index);
        }
    }

    const answerFCFS = FCFS(inputArray); // V
    const answerLCFSNP = LCFSNP(inputArray); // V
    const answerLCFSP = LCFSP(inputArray); // V
    const answerRoundRobin = RoundRobin(inputArray); // V
    const answerSJF = SJF(inputArray); // V

    console.log(`FCFS: mean turnaround ${answerFCFS}`);
    console.log(`LCFS (NP): mean turnaround ${answerLCFSNP}`);
    console.log(`LCFS (P): mean turnaround ${answerLCFSP}`);
    console.log(`RR: mean turnaround ${answerRoundRobin}`);
    console.log(`SJF: mean turnaround ${answerSJF}` + "\n");
}
```

בצילום הבא ניתן לראות איך אנחנו רצים בלולאה על כל קלטים השונים ומפעילים את האלגוריתמים השונים על כל קלט בנפרד ומדפיסים את התוצאה

FCFS

```
33 function FCFS(inputArray) {
34     let numberOfProccess = inputArray[0];
35     const enterTimeArray = [];
36     const workTimeArray = [];
37     for (let i = 1; i < inputArray.length; i++) {
38         const element = inputArray[i];
39         if (i % 2 === 1) {
40             enterTimeArray.push(element);
41         }
42         if (i % 2 === 0) {
43             workTimeArray.push(element);
44         }
45     }
46     let whereNow = 1;
47     let SumKolelTurnAround = 0;
48     for (let i = 0; i < numberOfProccess; i++) {
49         let waitingTimeCounter;
50         let runningTimeCounter;
51         let turnAroundSum;
52
53         let min = Math.min(...enterTimeArray);
54         if (whereNow < min) {
55             whereNow = min;
56         }
57         let indexOfMinNumber = enterTimeArray.indexOf(min.toString());
58         waitingTimeCounter = whereNow - parseInt(enterTimeArray[indexOfMinNumber]);
59         whereNow += parseInt(workTimeArray[indexOfMinNumber]);
60         runningTimeCounter = parseInt(workTimeArray[indexOfMinNumber]);
61         turnAroundSum = parseInt(waitingTimeCounter) + runningTimeCounter;
62         SumKolelTurnAround += turnAroundSum;
63         enterTimeArray.splice(indexOfMinNumber, 1);
64         workTimeArray.splice(indexOfMinNumber, 1);
65     }
66     return SumKolelTurnAround / numberOfProccess;
```

- בתחילת האלגוריתם אנחנו מעבירים את הקלט שקיבלנו לתוך מערכים שונים (זמני כניסה וזמני עבודה של כל תהליך).
- יש לנו משתנה שמחזיק היכן אנחנו נמצאים בזמן מחזור השעון.
- נרוץ בלולאה ונראה מי האיבר שזמן כניסתו הוא המינימלי (הראשון שמגיע הוא זה שיקבל עדיפות גבוהה מאחרים שהגיעו אחריו) , נבדוק האם האיבר נכנס ויכול להתחיל לקבל זמן מעבד או שהמחזור שעון עדיין לא הגיע אליו והוא צריך להמתין לאחר מכן נעשה את החישובים של זמני עבודה של אותו איבר ונגדיל את המצביע לאיפה שאני עכשיו להצביע לאחר סיום האיבר ונוסיף לסכום הכולל את הזמן ריצה וזמן המתנה של אותו איבר ונוציא אותו מהתור.
- לאחר ריצת כל האיברים נקבל את הזמן הכולל של כולם ונחלק במספר התהליכים כדי לקבל זמן ממוצע כולל

LCFS - NP

```
function LCFSNP(inputArray) {
    let numberOfProccess = inputArray[0];
    const enterTimeArray = [];
    const workTimeArray = [];
    for (let i = 1; i < inputArray.length; i++) {
        const element = inputArray[i];
        if (i % 2 === 1) {
            enterTimeArray.push(element);
        }
        if (i % 2 === 0) {
            workTimeArray.push(element);
        }
    }
    let CopyArray;
    let whereNow = Math.min(...enterTimeArray);
    let SumKolelTurnAround = 0;
    for (let i = 0; i < numberOfProccess; i++) {
        if (CopyArray != undefined) {
            if (whereNow < Math.min(...enterTimeArray))
                whereNow = Math.min(...enterTimeArray);
        }
        let waitingTimeCounter;
        let runningTimeCounter;
        let turnAroundSum;

        let max = Math.max(...enterTimeArray);
        let indexOfMaxNumber = enterTimeArray.indexOf(max.toString());
        let maxtemp = max;
        CopyArray = [...enterTimeArray];
        while (whereNow < maxtemp) {
            indexOfMaxNumber = CopyArray.indexOf(maxtemp.toString());
            CopyArray.splice(indexOfMaxNumber, 1);
            maxtemp = Math.max(...CopyArray);
        }
        indexOfMaxNumber = enterTimeArray.lastIndexOf(maxtemp.toString());
        waitingTimeCounter = whereNow - parseInt(enterTimeArray[indexOfMaxNumber]);
        whereNow += parseInt(workTimeArray[indexOfMaxNumber]);
        runningTimeCounter = parseInt(workTimeArray[indexOfMaxNumber]);
        turnAroundSum = parseInt(waitingTimeCounter) + runningTimeCounter;
        SumKolelTurnAround += turnAroundSum;
        enterTimeArray.splice(indexOfMaxNumber, 1);
        workTimeArray.splice(indexOfMaxNumber, 1);
    }
    return SumKolelTurnAround / numberOfProccess;

}
```

- בתחילת האלגוריתם אנחנו מעבירים את הקלט שקיבלנו לתוך מערכים שונים (זמני כניסה וזמני עבודה של כל תהליך).
- יש לנו משתנה שמחזיק היכן אנחנו נמצאים בזמן מחזור השעון.
- נרוץ מחזור שעון אחרי מחזור שעון ונראה מי יכול לרוץ כעת בכל מחזור שעון, כמובן עם עדיפות לתהליך שהגיע אחרון, כאשר תהליך מתחיל לרוץ אף תהליך אחר לא יוכל להפריע לו וכולם יצטרכו לחכות שהתהליך יסיים למרות שיש להם עדיפות גבוהה יותר.
- האלגוריתם שכתבתי ימצא את האיבר שנכנס בזמן המקסימלי ועומד בתנאי שכניסתו תהיה קטנה או שווה למשתנה הזמן שאנחנו נמצאים עכשיו.
- העתקנו את מערך זמן הכניסה נעבור עליו ונמחק ממנו כל איבר שלא מתאים לקריטריונים וככה נמצא את האיבר המקסימלי שעומד בתנאים , במידה ויש כמה שנכנסו באותו זמן ניקח את האיבר האחרון לפי הנחת המטלה
- לאחר ריצת כל האיברים נקבל את הזמן הכולל של כולם ונחלק במספר התהליכים כדי לקבל זמן ממוצע כולל

LCFS - P

```javascript
while (1) {

    counter = 0;
    for (let index = 0; index < workTimeArray.length; index++) {
        const element = workTimeArray[index];
        if (element == 0) {
            for (let z = 0; z < activeProcesstTor.length; z++) {
                const element1 = activeProcesstTor[z];
                if (element1 == index) {
                    activeProcesstTor[z] = -1;
                }

            }
        }

    }

    flag = 1;
    if (indexOfActive == numberOfProccess) {
        indexOfActive = 0;
    }
    //Check If i on -1 in activeproccess if yes then i skip to the next
    while (activeProcesstTor[indexOfActive] === -1) {
        indexOfActive++;
        if (indexOfActive == numberOfProccess) {
            indexOfActive = 0;
        }
        if (counter == numberOfProccess - 1) {
            counter = 0;
            flag = 0;
            break;
        }
        counter++;
    }
```

```javascript
if (flag != 0) {
    //active tor not empty
    let CopyArray = [...enterTimeArray];
    let max = Math.max(...enterTimeArray);
    let indexOfMaxNumber = enterTimeArray.indexOf(max.toString());
    let maxtemp = max;

    while (whereNow < maxtemp + 1) {
        indexOfMaxNumber = CopyArray.indexOf(maxtemp.toString());
        CopyArray.splice(indexOfMaxNumber, 1);
        maxtemp = Math.max(...CopyArray);
    }

    indexOfActive = enterTimeArray.lastIndexOf(maxtemp.toString());

    if (indexOfActive != undefined) {
        if (workTimeArray[indexOfActive] == 0) {
            SumKolelTurnAround += whereNow - parseInt(enterTimeArray[indexOfActive]);
            enterTimeArray[indexOfActive] = -1;
            for (let j = 0; j < activeProcesstTor.length; j++) {
                const element = activeProcesstTor[j];
                if (element == indexOfActive) {
                    activeProcesstTor[j] = -1;
                }
            }
        }

        workTimeArray[indexOfActive]--;
        if (workTimeArray[indexOfActive] == 0) {
            SumKolelTurnAround += whereNow - parseInt(enterTimeArray[indexOfActive]);
            enterTimeArray[indexOfActive] = -1;
            for (let j = 0; j < activeProcesstTor.length; j++) {
                const element = activeProcesstTor[j];
                if (element == indexOfActive) {
                    activeProcesstTor[j] = -1;
                }
            }
        }
    }
}
```

- בתחילת האלגוריתם אנחנו מעבירים את הקלט שקיבלנו לתוך מערכים שונים (זמני כניסה וזמני עבודה של כל תהליך).
- יש לנו משתנה שמחזיק היכן אנחנו נמצאים בזמן מחזור השעון.
- בניגוד לשאר האלגוריתמים שהיו לנו עד כה האלגוריתם הזה צריך להיות preemptive, מה שאומר שבכל ריצת מחזור שעון מי שיש לו סדר עדיפות גבוה (פה מי שנכנס אחרון) יקבל זמן מעבד.
- לכן נבדוק כל מחזור שעון למי מגיע זמן ריצה לפי העדיפות מי מגיע אחרון והוא יקבל זמן מעבד , נשמור את מי שיכול לקבל זמן מעבד במערך ונאתחל את הערכים להיות -1 במידה ומישהו נכנס לתור הוא מקבל את הערך של המיקום שלו במערך הכניסה ומערך העבודה שיש לנו.
- נרוץ ונראה אם במערך שבנינו יש תהליך שיכול לרוץ ונריץ אותו מחזור אחד ונסכום ונוריד המערך של הזמן עבודה, במידה ויש איבר במערך של הזמן עבודה שהוא 0 נסיר אותו גם מהמערך שיצרנו ונעבור לאיבר הבא שזכאי לפי הסדר עדיפויות לרוץ עד שנסיים עם כולם

RR

```
function RoundRobin(inputArray) {
    let numberOfProccess = inputArray[0];
    const enterTimeArray = [];
    const workTimeArray = [];
    for (let i = 1; i < inputArray.length; i++) {
        const element = inputArray[i];
        if (i % 2 === 1) {
            enterTimeArray.push(element);
        }
        if (i % 2 === 0) {
            workTimeArray.push(element);
        }
    }


    let timeQuantom = 2;
    let whereNow = 1;
    let SumKolelTurnAround = 0;

    // let sumWorkingTimeArray = [...workTimeArray];
    let waitingTimeArray = [];
    for (let index = 0; index < numberOfProccess; index++) {
        waitingTimeArray[index] = 0;
    }
    let indexOfActive = 0;

    let activeProcesstTor = new Array(parseInt(numberOfProccess)).fill(-1);
    let counter = 0;
    let flag = 1;
```

```
    if (flag != 0) {
        //active tor not empty
        if (workTimeArray[activeProcesstTor[indexOfActive]] == 0) {
            SumKolelTurnAround += whereNow - enterTimeArray[activeProcesstTor[indexOfActive]];
            activeProcesstTor[indexOfActive] = -1;
            timeQuantom = 2;
            indexOfActive++;
            whereNow++;
            continue;
        }
        workTimeArray[activeProcesstTor[indexOfActive]]--;
        if (workTimeArray[activeProcesstTor[indexOfActive]] == 0) {
            SumKolelTurnAround += whereNow - enterTimeArray[activeProcesstTor[indexOfActive]];
            activeProcesstTor[indexOfActive] = -1;
            timeQuantom = 2;
            indexOfActive++;
            whereNow++;
            continue;
        }


        timeQuantom--;


    }
```

- בתחילת האלגוריתם אנחנו מעבירים את הקלט שקיבלנו לתוך מערכים שונים (זמני כניסה וזמני עבודה של כל תהליך).
- יש לנו משתנה שמחזיק היכן אנחנו נמצאים בזמן מחזור השעון.
- האלגוריתם נותן זמן שווה לכל תהליך לפי ההגדרה 2 מחזורי שעון לכל תהליך כל תהליך שמצטרף יצטרף לסוף התור ויחכה לתורו.
- בקוד הכנו תור שבו נדע מי מחכה לקבל זמן מעבד וכל אחד בתורו ירוץ 2 מחזורי שעון, נעבור מחזור מחזור ונצרף כל תהליך שהגיע זמנו לסוף התור.
- נחזיק מצביע בתור שכל פעם יתקדם לאיבר הבא במידה ונגמר 2 מחזורי שעון.
- נסכום כל איבר שסיים לרוץ ונמחק מהתורים ובסוף נחלק במספר התהליכים את הסכום הכולל של כולם כדי לקבל ממוצע.

SJF

```javascript
function SJF(inputArray) {

    let numberOfProccess = inputArray[0];
    const enterTimeArray = [];
    const workTimeArray = [];
    for (let i = 1; i < inputArray.length; i++) {
        const element = inputArray[i];
        if (i % 2 === 1) {
            enterTimeArray.push(element);
        }
        if (i % 2 === 0) {
            workTimeArray.push(element);
        }
    }


    let whereNow = 1;
    let SumKolelTurnAround = 0;

    let waitingTimeArray = [];
    for (let index = 0; index < numberOfProccess; index++) {
        waitingTimeArray[index] = 0;
    }
    let indexOfActive = 0;
    let activeProcesstTor = new Array(parseInt(numberOfProccess)).fill(-1);
    let counter = 0;
    let flag = 1;
    let activeNow;
```

```javascript
    if (flag != 0) {
        //active tor not empty

        let minTemp = Infinity;
        let indexOfMinWorking;
        for (let i = 0; i < workTimeArray.length; i++) {
            const element = parseInt(workTimeArray[i]);
            if (element == 0) {
                continue;
            }
            if (element < minTemp) {
                activeNow = 0;

                for (let index = 0; index < activeProcesstTor.length; index++) {
                    const element = activeProcesstTor[index];
                    if (i == element) {
                        activeNow = 1;
                        break;
                    }
                }

                if (activeNow) {
                    minTemp = element
                    indexOfMinWorking = i;
                }
            }
            if (element == minTemp) {
                //check who came in first
                if (enterTimeArray[i] < enterTimeArray[indexOfMinWorking]) {
                    minTemp = element
                    indexOfMinWorking = i;
                }
            }
```

- בתחילת האלגוריתם אנחנו מעבירים את הקלט שקיבלנו לתוך מערכים שונים (זמני כניסה וזמני עבודה של כל תהליך).
- יש לנו משתנה שמחזיק היכן אנחנו נמצאים בזמן מחזור השעון.
- האלגוריתם נותן סדר עדיפות למי שיש לו את זמן הריצה הקצר ביותר. האלגוריתם הזה גם preemptive שזה אומר שבכל ריצת מחזור שעון מי שיש לו סדר עדיפות גבוה (פה מי שזמן העבודה שלו קטן יותר) יקבל זמן מעבד.
- נמצא את האיבר המינימלי במערך זמני עבודה ונבדוק האם הוא נכנס כבר מבחינת הזמן כניסה שלו , וניתן לו זמן מעבד למחזור אחד, כל מחזור נבדוק מחדש למי מגיע לרוץ על המעבד וניתן לו זמן לרוץ.
- לאחר ריצת כל האיברים נקבל את הזמן הכולל של כולם ונחלק במספר התהליכים כדי לקבל זמן ממוצע כולל

דוגמה לריצות של אלגוריתם שיצרתי בטבלה

4	Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
3,5	P1			0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5
5,8	P2					0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8					
1,10	P3	0	1	2	3	4	5	6	7	8	9	10																						
6,9	P4						0	0	0	0	0	0	1	2	3	4	5	6	7	8	9													

PERIMTIVE

4	Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
3,5	P1			0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	4	5								
5,8	P2					0	1	1	1	1	1	1	1	1	1	1	2	3	4	5	6	7	8											
1,10	P3	0	1	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	3	4	5	6	7	8	9	10
6,9	P4						0	1	2	3	4	5	6	7	8	9																		

4	Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
3,5	P1			0	1	2	2	2	2	2	3	4	4	4	4	4	4	4	5															
5,8	P2					0	0	0	1	2	2	2	2	2	2	2	3	4	4	4	4	4	4	5	6	6	6	6	6	7	8			
1,10	P3	0	1	2	2	2	3	4	4	4	4	4	4	4	5	6	6	6	6	6	6	7	8	8	8	8	8	9	9					
6,9	P4						0	0	0	0	0	0	1	2	2	2	2	2	2	3	4	4	4	4	4	5	6	6	6	6	6	7	8	9

sjf premtive

4	Time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33
3,5	P1			0	1	2	3	4	5																									
5,8	P2					0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8									
1,10	P3	0	1	2	2	2	2	2	2	3	4	5	6	7	8	9	10																	
6,9	P4						0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	3	4	5	6	7	8	9

תודה רבה!

חיים חוגי

207686650