گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

سیستم عامل به لطف الگوریتم های قدرتمند و کارای خود در هسته خود، دارای توانایی های زیادی است.

این الگوریتم ها به زبان قدرتمند "C" طراحی شده اند که گاه دسترسی عموم به آنان ممکن نیست!

این ها همچنین از پیچیدگی والایی برای بسیار کارا بودن برخوردارند که به همین سبب،

طراحی دقیق آنها عملی دشوار است.

در این راستا به دو الگوریتم زمان بندی فرآیند های قابل اجرا به نام های "FCFS" و "SJF" می پردازیم:

الگوریتم FCFS :

تعدادی فرآیند به همراه مدت زمان اجرای هر یک از آنها داده شده اند.

می خواهیم همه آنها را توسط یک ماشین که تنها یک فرآیند را می تواند در لحظه مورد بررسی قرار دهد، اجرا کنیم. این الگوریتم هر یک از فرآیند ها را با رعایت ترتیب در ورودی به ماشین داده و متوسط زمان انتظار فرآیند ها و گاه متوسط زمان اجرای فرآیند هارا در زمان چند جمله ای، در اختیار می گذارد.

FCFS Example

Process	Duration	Order	Arrival Time
P1	24	1	0
P2	3	2	0
P3	4	3	0

Wikitechy

The final schedule (Gantt chart):

P1 waiting time: 0
P2 waiting time: 24
P3 waiting time: 27

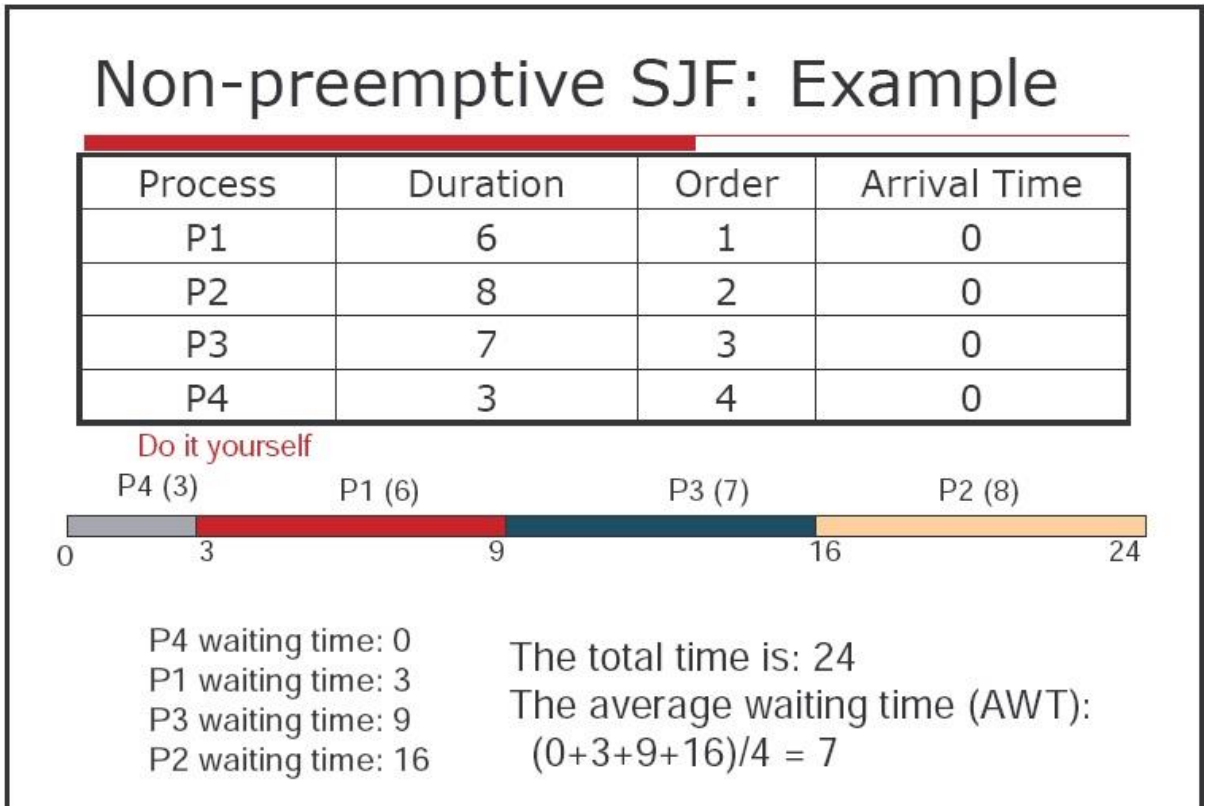
The average waiting time:
(0+24+27)/3 = 17

الگوریتم SJF :

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

تعدادی فرآیند به همراه مدت زمان اجرای هر یک از آنها داده شده اند.

می خواهیم همه آنها را توسط یک ماشین که تنها یک فرآیند را می تواند در لحظه مورد بررسی قرار دهد، اجرا کنیم. این الگوریتم مشابه الگوریتم "FCFS" عمل کرده و متوسط زمان انتظار فرآیند ها و گاه متوسط زمان اجرای فرآیند هارا در زمان چند جمله ای محاسبه می کند. متوسط زمان انتظار فرآیند های محاسبه شده توسط این الگوریتم در بهینه ترین حالت خود قرار دارد؛ زیرا این الگوریتم اجرای فرآیند های با زمان کمتر نسبت به سایر را مقدم می داند. در نتیجه این اولویت بندی، الگوریتم "SJF"با روش حریصانه و انتخاب فرآیند با کمترین زمان اجرا از فرآیند های اجرا نشده در هر مرحله، متوسط زمان انتظار فرآیند ها برای اجرا را کمینه می کند. بدیهی است این ایده حریصانه روی متوسط زمان اجرای فرآیند ها اثری نداشته و مقدار این کمیت را می توان با الگوریتم "FCFS" نیز به درستی محاسبه کرد.



در ادامه این دو الگوریتم را به صورت شهودی و تقریبی بر روی زبان برنامه نویسی "Python" پیاده سازی کرده و آن را در سه مرحله کلی شرح می دهیم:

مرحله اول (الگوریتم ها به صورت توابع)

با توجه به ساختار دو الگوریتم "FCFS" و "SJF"، توابعی به همین نام ها تعبیه شده اند.

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

تابع FCFS(BurstTimes, Time) ، لیستی از زمان اجرای فرآیند هارا با ترتیب دلخواه و مقدار زمانی که کاربر می خواهد در اختیار ماشین قرار دهد تا فرآیند ها را به طور ترتیبی به پایان رساند، دریافت کرده و لیستی سه تایی به ترتیب شامل مجموع کل زمان انتظار فرآیند ها، مجموع کل زمان اجرای فرآیند ها، و نمایی گرافیکی از ترتیب، مدت زمان اجرا، مدت زمان انتظار فرآیند ها را با توجه به ترتیب اعمالی از سوی کاربر را برمی گرداند.

در درون این تابع، متغیر TimeNeeded میزان زمان مورد نیاز برای اجرای کلیه فرآیند ها را در خود ذخیره می کند. حال این مقدار با مدت زمانی که کاربر تعیین کرده است مقایسه می شود. اگر این مقدار از مقدار مجاز تعیین شده توسط کاربر تجاوز کند، الگوریتم زمان مورد نیاز را در متغیر TimeLoss ذخیره کرده و این مقدار را از زمان اجرای آخرین فرآیندی که قرار است اجرا شود، کم می کند. اگر مقدار TimeLossچنان بزرگ باشد که حداقل مانع اجرای آخرین فرآیند شود، زمان اجرای فرآیند مذکور را از لیست زمان اجرای فرآیند ها خارج کرده و از زمان اضافی باقی مانده در متغیر TimeLoss ، که اختلاف مقدار زمان اضافی اولیه و مقدار زمان مورد نیاز برای اجرای فرآیند حذف شده است (که در متغیر Reduction ذخیره می شود)، برای به حد نصاب رساندن زمان کل اجرای فرآیند ها مطابق با مقدار زمانی که کاربر تعیین کرده است استفاده می کنیم. توجه شود این کار (شبیه به حالت بازگشتی) بر روی لیست زمان اجرای فرآیند ها برای آخرین زمان اجرای فرآیند موجود در آن تا آنجایی تکرار می شود که مقدار TimeLoss به مقدار نابیشتر از آخرین زمان اجرای فرآیند موجود در لیست زمان اجرای فرآیند ها برسد.

حال با گذری بر روی لیست زمان اجرای فرآیند ها (که ممکن است تغییر یافته باشد) توسط آخرین حلقه for موجود در تابع الگوریتم، مجموع زمان انتظار فرآیند ها را توسط متغیر WaitingTimes ذخیره کرده و همچنین نمایی گرافیکی را برای نمایش ترتیب و چگونگی اجرای فرآیند ها (البته به صورت غیر داینامیکی!) با متغیر GanttChart طراحی می شود.

در آخر لیستی سه تایی به ترتیب شامل مجموع کل زمان های انتظار فرآیند ها، مجموع کل زمان های اجرای فرآیند ها، و نمایی گرافیکی از جدول، متوسط مدت زمان اجرا، و متوسط مدت زمان انتظار فرآیند هارا با توجه به ترتیب اعمالی از سوی کاربر را نمایش داده می شود.

تابع SJF(BurstTimes, Time) ، مشابه تابع FCFS(BurstTimes, Time) عمل می کند؛ با این تفاوت که لیست زمان اجرای فرآیند ها را مرتب شده به کار می گیرد تا بتواند به صورت حریصانه، متوسط زمان انتظار فرآیند ها را کمینه کند. باقی جزئیات ساختار این تابع بنا بر تعریف، همان ساختار تابعFCFS(BurstTimes, Time) بوده و به همین دلیل از توضیح بیشتر صرف نظر می شود.

در تصویر زیر، روند تعریف این توابع را با جزئیات تکمیلی مشاهده می کنید.

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

```python
from random import randint
from statistics import mean, stdev, median, mode
from scipy.stats import chi2_contingency

def FCFS(BurstTimes, Time):
    TimeNeeded = sum(BurstTimes)
    WaitingTimes = 0
    GanttChart = "\n0"
    if TimeNeeded > Time:
        TimeLoss = TimeNeeded - Time
        for Index in reversed(range(len(BurstTimes))):
            Reduction = BurstTimes[Index]
            BurstTimes[Index] -= TimeLoss
            if BurstTimes[Index]>=0:
                if BurstTimes[Index]==0:
                    BurstTimes = BurstTimes[:Index]
                else:
                    BurstTimes = BurstTimes[:Index+1]
                break
            else:
                TimeLoss -= Reduction

    if sum(BurstTimes)== TimeNeeded:
        Note = "IMPRESSIVE!\n"
    else:
        Note = "END IT FORCELY!\n"

    for Index in range(len(BurstTimes)):
        WaitingTimes += sum(BurstTimes[:Index])
        GanttChart += "-"* BurstTimes[Index] + str(sum(BurstTimes[:Index+1]))

    return [WaitingTimes, sum(BurstTimes), Note+GanttChart+"\n\nAverage Times \n\nAverage waiting time: "+str(WaitingTimes/len(BurstTimes))+"\nAverage burst time: "+str(sum(BurstTimes)/len(

def SJF(BurstTimes, Time):
    return FCFS(sorted(BurstTimes), Time)
```

پس از تعریف کردن توابع الگوریتمی، برنامه شبیه ساز مقدار زمان مورد تایید کاربر توسط او تعیین شده را در متغیر Time ذخیره می کند. اگر این مقدار نامثبت باشد،

برنامه بدون اجرای هر گونه الگوریتمی پایان می پذیرد.

در ادامه از کاربر خواسته می شود تا تعیین کند آیا مایل است برنامه شبیه ساز

از فرآیند هایی تصادفی به منظور انجام چند برآورد آماری استفاده کند،

یا با نمونه ای مشخص از فرآیند ها که توسط او تعیین می شود؟

جواب کاربر بایستی "N" (به منظور خیر)، و یا "Y" (به منظور بلی) باشد که این جواب ها،

در متغیر YouOrAI ذخیره شده و ادامه کار برنامه شبیه ساز را به ترتیب در حالت دوم یا سوم

قرار می دهد که در ادامه هر یک را به ترتیب شرح می دهیم.

مرحله دوم (هر چه خان بگوید!)

در این مرحله، از کاربر خواسته شده تا تعداد فرآیند های مورد بررسی برنامه

(ذخیره شده در متغیر NumberOfProcesses) و زمان اجرای هر یک از آنها را تعیین کند.

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

برنامه شبیه ساز تمامی مقادیر زمان اجراهای وارده را در متغیر لیست زمان اجرای فرآیند ها به نام BurstTimes به ترتیب ورود آنها قرار می دهد و سپس، هر یک از الگوریتم های FCFS و SJF به وسیله توابع معرف آنها بر روی لیست زمان اجرای فرآیند های تشکیل شده و زمان اعمال شده توسط کاربر،

اعمال گشته، و تنها نمایش آنها به همراه متوسط زمان انتظار فرآیند ها و متوسط زمان اجرای فرآیند ها

(با توجه به مقدار زمان معین Time) برگردانده می شوند.

در تصویر زیر، روند کار در این مرحله را با جزئیات تکمیلی مشاهده می کنید.

```
Time = int(input("How long can you wait for processes to be done? "))
if Time <= 0:
    print("Go Home, KID!")

else:
    YouOrAI = input("\nDo you want to work with unknown processes? (Y, N) ")
    while YouOrAI not in ["Y", "N"]:
        YouOrAI = input("Do you want to work with unknown processes? (Y, N) ")

    if YouOrAI == "N":
        print("You decide what would happen\n")
        NumberOfProcesses = int(input("How many processes are there for being burst? "))
        BurstTimes = [0] * NumberOfProcesses
        for Index in range(NumberOfProcesses):
            BurstTimes[Index] = int(input("How long does "+str(Index+1)+ "th process takes to be done? "))
        print("\nBy FCFS algorithm we have:")
        print(FCFS(BurstTimes, Time)[2])
        print("By SJF algorithm we have:")
        print(SJF(BurstTimes, Time)[2])
```

مرحله سوم (ده، بیست، سه پونزده، هزار و شصت و شونزده...)

نزدیک ترین شبیه سازی یک الگوریتم رایانه ای به نسخه های اصلی آن، زمانی است که الگوریتم،

تمامی اعضای مجموعه حالات را پوشش داده و حل نماید.

بدین منظور به برنامه شبیه ساز قابلیت این را دارد که روند اجرای الگوریتم های FCFS و SJF را به طور تمام کمال بر عهده گیرد و در تعداد مشخصی کارآزمایی که توسط کاربر تعیین شده در متغیر NumberOfExperiences ذخیره می شود، این روند اجرا را توسط یک حلقه for تکرار کند.

مقدار NumberOfExperiences باید بین 30 الی 1000 باشد.

در این صورت برنامه شبیه ساز در هر کارآزمایی، لیستی از زمان اجرای تعداد دلخواهی فرآیند

(بین 1 الی 1000 عدد) با زمان اجراهای دلخواه (بین 1 الی 1000 واحد زمان) که در متغیر BurstTimes ذخیره می شود، تعبیه کرده و در اثر اعمال دو الگوریتم FCFS و SJF بر روی آن،

مجموع کل زمان انتظار فرآیند ها توسط هر یک از آنها و مجموع کل زمان اجرای فرآیند ها

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

(با توجه به مقدار زمان معین Time) را به ترتیب به متغیر از جنس لیست
TotalWaitingTimesFCFS، TotalWaitingTimesSJF، و TotalBurstTimes

به منظور جمع آوری کلیه نتایج در اثر گذراندن تعداد کارآزمایی های الحاقی برای محاسبه

چهار کمیت آماری میانگین (mean())، میانه (median())، مد (mode())، و انحراف معیار (stdev()) برای کلیه محتویات آنها در آخر کار، می افزاید.

در تصویر زیر، روند کار در این مرحله را با جزئیات تکمیلی مشاهده می کنید.

```
    else:
        print("Some unknown processes are coming..!\n")
        NumberOfExperiences = int(input("How many times do you want to run the algorithm? Choose from range (30, 1000) "))
        while 30 > NumberOfExperiences or NumberOfExperiences > 1000:
            NumberOfExperiences = int(input("I Told you choose from range (30, 1000) "))
        TotalWaitingTimesFCFS=[]
        TotalWaitingTimesSJF=[]
        TotalBurstTimes=[]
        for Experience in range(NumberOfExperiences):
            NumberOfProcesses = randint(1, 1000)
            BurstTimes = []
            for Index in range(NumberOfProcesses):
                BurstTimes.append(randint(1, 1000))
            OutputDatas=FCFS(BurstTimes, Time)[:2]
            TotalWaitingTimesFCFS.append(OutputDatas[0])
            TotalBurstTimes.append(OutputDatas[1])
            TotalWaitingTimesSJF.append(SJF(BurstTimes, Time)[0])

        print("\nHere's the statistical quantities 👇")
        print("Waiting Times:")
        print("\nMean of the whole waiting times of the random processes obtained of FCFS algorithm is", mean(TotalWaitingTimesFCFS))
        print("\nMedian of the whole waiting times of the random processes obtained of FCFS algorithm is", median(TotalWaitingTimesFCFS))
        print("\nMode Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is", mode(TotalWaitingTimesFCFS))
        print("\nStandard Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is", stdev(TotalWaitingTimesFCFS))
        print("\nMean of the whole waiting times of the random processes obtained of SJF algorithm is ", mean(TotalWaitingTimesSJF))
        print("\nMedian of the whole waiting times of the random processes obtained of SJF algorithm is ", median(TotalWaitingTimesSJF))
        print("\nMode Deviation of the whole waiting times of the random processes obtained of SJF algorithm is", mode(TotalWaitingTimesSJF))
        print("\nStandard Deviation of the whole waiting times of the random processes obtained of SJF algorithm is", stdev(TotalWaitingTimesSJF))
        print("\nBurst Times:")
        print("\nMean of the whole burst times of the random processes obtained of both algorithms is", mean(TotalBurstTimes))
        print("\nMedian of the whole burst times of the random processes obtained of both algorithms is", median(TotalBurstTimes))
        print("\nMode Deviation of the whole burst times of the random processes obtained of both algorithms is", mode(TotalBurstTimes))
        print("\nStandard Deviation of the whole burst times of the random processes obtained of both algorithms is", stdev(TotalBurstTimes))
print("\nHave a nice day😊")
```

برای دو نمونه تصادفی، برآورد های آماری آنها به صورت زیر گزارش شده اند:

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Sepehr\Desktop\Abbaspour - 610398147 - FCFS & SJF Scheduling - OS.py
How long can you wait for processes to be done? 3000

Do you want to work with unknown processes? (Y, N) Y
Some unknown processes are coming..!

How many times do you want to run the algorithm? Choose from range (30, 1000) 1000

Here's the statistical quantities 👇
Waiting Times:

Mean of the whole waiting times of the random processes obtained of FCFS algorithm is 8964.895

Median of the whole waiting times of the random processes obtained of FCFS algorithm is 8649.0

Mode Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is 8750

Standard Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is 2710.6558544264035

Mean of the whole waiting times of the random processes obtained of SJF algorithm is  -41325300346.619

Median of the whole waiting times of the random processes obtained of SJF algorithm is  -22335577515.5

Mode Deviation of the whole waiting times of the random processes obtained of SJF algorithm is -7849728964

Standard Deviation of the whole waiting times of the random processes obtained of SJF algorithm is 45790595339.733925

Burst Times:

Mean of the whole burst times of the random processes obtained of both algorithms is 2994.678

Median of the whole burst times of the random processes obtained of both algorithms is 3000.0

Mode Deviation of the whole burst times of the random processes obtained of both algorithms is 3000

Standard Deviation of the whole burst times of the random processes obtained of both algorithms is 93.88637263717757

Have a nice day😊
```

گزارش شبیه سازی برخی از الگوریتم های زمان بندی فرآیندها

```
Python 3.8.5 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 3.8.5 (tags/v3.8.5:580fbb0, Jul 20 2020, 15:57:54) [MSC v.1924 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: C:\Users\Sepehr\Desktop\Abbaspour - 610398147 - FCFS & SJF Scheduling - OS.py
How long can you wait for processes to be done? 99999

Do you want to work with unknown processes? (Y, N) Y
Some unknown processes are coming..!

How many times do you want to run the algorithm? Choose from range (30, 1000) 10000
I Told you choose from range (30, 1000) 1000

Here's the statistical quantities 👇
Waiting Times:

Mean of the whole waiting times of the random processes obtained of FCFS algorithm is  8683285.696

Median of the whole waiting times of the random processes obtained of FCFS algorithm is  9845958.0

Mode Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is  10483957

Standard Deviation of the whole waiting times of the random processes obtained of FCFS algorithm is  2980780.542222651

Mean of the whole waiting times of the random processes obtained of SJF algorithm is  -24861081647.482

Median of the whole waiting times of the random processes obtained of SJF algorithm is  -9822454231.5

Mode Deviation of the whole waiting times of the random processes obtained of SJF algorithm is  -46236435860

Standard Deviation of the whole waiting times of the random processes obtained of SJF algorithm is  31395695844.00452

Burst Times:

Mean of the whole burst times of the random processes obtained of both algorithms is  90301.428

Median of the whole burst times of the random processes obtained of both algorithms is  99999.0

Mode Deviation of the whole burst times of the random processes obtained of both algorithms is  99999

Standard Deviation of the whole burst times of the random processes obtained of both algorithms is  23095.071113390157

Have a nice day😊
```

تفسیر نتایج

بر اساس مطالعات آماری انجام شده و نتایج آماری بدست آمده، میانه و مد مجموع های زمان های اجرا مورد نیاز فرآیند های تصادفی برابر زمان مطلوب کاربر است. اگر زمان مطلوب کاربر به اندازه کافی بزرگ نباشد، میانگین مجموع های زمان های اجرای فرآیند ها برابر زمان مطلوب کاربر بوده و انحراف معیار آنها

و همچنین میانه و مد مجموع های زمان انتظار فرآیند ها (چون مقدار آنها صفر است)، برابر صفر خواهد بود.

باقی کمیت های آماری در حالات مختلف آماری، مقادیر متغیری دارند که علم آمار صحت آنها

را تضمین می کند.

پایان