Modalități de planificare a activităților (de I/O)

1. Planificare aleatoare (Cererile de satisfacere a activitatilor sunt satisfăcute într-o ordine aleatoare. Cererile provin de la procese diferite, sunt destinate atât operațiilor de intrare cât și celor de ieșire)
2. Planificare FIFO (Cererile de satisfacere a activitatilor sunt onorate în ordinea din coadă)
3. Planificare bazată pe priorități (Folosirea priorităților satisface politicile sistemului de operare, dar sunt în contradicție cu mecanismele de gestiune a cozilor I/O)
4. Planificare LIFO (Politica de planificare presupune o organizare de tip stivă)
5. Planificarea SSTF (presupune selectarea cererii care este "cea mai apropiată" de poziția curentă a capului de satisfare a cererii, pentru cereri aflate la distanță egală vor fi necesare mecanisme de decizie suplimentare). SSTF (Shortest Seek Time First)
6. Planificare SCAN (deplasarea capului de satisfacere a activitatilor se realizeaza într-o singură direcție și sunt satisfăcute toate cererile întâlnite pe parcurs, după aceasta este inversată direcția de deplasare pentru satisfacerea celorlalte cereri)
7. Planificare C-SCAN (Această politică extinde politica SCAN prin restricționarea scanării la o singură direcție. Odată terminată căutarea, capul se întoarce în poziția inițială -prima pistă- și se execută o nouă scanare)
8. Planificare FSCAN (Se bazează pe o segmentare a cozii de cereri, pentru a evita un număr prea mare de cereri într-o singură zonă. Alternativ, se poate considera SCAN în N pași.)

Se vor considera N activitati codificate prin numerele 1, 2, …, N, fiecare cu o prioritate p_1, p_2, …, p_N. Distantele de la pozitia capului de satisfacere a activitatilor sunt d_1, d_2, …, d_n.

Pentru fiecare dintre problemele urmatoare realizati un program C/C++/Java

Problema 1 (planificare aleatoare)

Planificati satisfacerea a N activitati folosind generatorul de numere aleatoare din limbajul de programare.

Exemplu

Intrare: N=4

Iesire: 3 2 1 4 (sunt 24 de variante posibile)

Problema 2 (planificare FIFO)

Planificati satisfacerea a N activitati folosind strategia FIFO. Ordinea in care apar activitatile este a_1, a_2,…,a_N.

Exemplu

Intrare:

N=4

a: 2 4 1 3

Iesire: 2 4 1 3

Problema 3 (planificare bazată pe priorități)

Planificati satisfacerea a N activitati folosind strategia bazata pe prioritati. Ordinea in care apar activitatile este a_1, a_2, …, a_N, avand prioritatile p_1, p_2, …, p_N.

Exemplu

Intrare:

N=4

a: 2 4 1 3

p: 9 10 20 9

Iesire: 1 4 2 3

Problema 4 (planificare LIFO)

Planificati satisfacerea a N activitati folosind strategia LIFO. Ordinea in care apar activitatile este a1, a2,…,aN.

Exemplu

Intrare:

N=4

a: 2 4 1 3

Iesire: 3 1 4 2

Problema 5 (planificare SSTF)

Planificati satisfacerea a N activitati folosind strategia SSTF. Distantele de la pozitia capului de satisfacere a activitatilor sunt d_1, d_2, …, d_n. La distante egale se va satisface mai intai activitatea cu numarul de ordine mai mic.

Exemplu

Intrare:

N=4

a: 2 4 1 3

d: 10 5 10 9

Iesire: 4 3 1 2

Problema 6 (planificare SCAN)

Planificati satisfacerea a N activitati folosind strategia FIFO. Ordinea in care apar activitatile este a_1, a_2,…,a_N. Pozitia capului de satisfare a activitatilor este in dreptul activitatii p si acesta se indreapta spre dreapta (satisfacand toate activitatile pana la sfarsit, apoi se intoarce in sens invers pentru a satisface celelalte activitati).

Exemplu

Intrare:

N=7

a: 7 2 4 6 1 3 5

p=6

Iesire: 6 1 3 5 4 2 7

Problema 7 (planificare FSCAN)

Planificati satisfacerea a N activitati folosind strategia FIFO. Ordinea in care apar activitatile este a_1, a_2,…,a_N. Pozitia capului de satisfare a activitatilor este pe rand in pozitiile p_1, p_2, …, p_k, acesta putând satisface x_1, x_2, …, x_k activitati in sensurile s_1, s_2, …, s_k (1 – spre dreapta, -1 spre stanga).

Exemplu

Intrare:

N=7

a: 7 2 4 6 1 3 5

k=3

p: 3 6 2

x: 2 2 7

s: 1 -1 1

Iesire: 4 6 3 1 2 5

Problema 1

```
#include <iostream>
#include <stdio.h>
#include <stdlib.h>
using namespace std;

int main()
{
   int x[1000],n,i,j,sw,k;
   cout<<"n=";cin>>n;
   for(i=1;i<=n;i++){
      //determin x[i]
      do{
         k=1+rand()%n;
         sw=0;
         for(j=1;j<i;j++)
            if(x[j]==k)
            sw=1;
         if(sw==0) x[i]=k;
      }while(sw);
   }
   for(i=1;i<=n;i++)
      {
         for(j=1;j<=n;j++)
            if(i==x[j])
             cout<<j<<" ";
      }

   return 0;
}
```