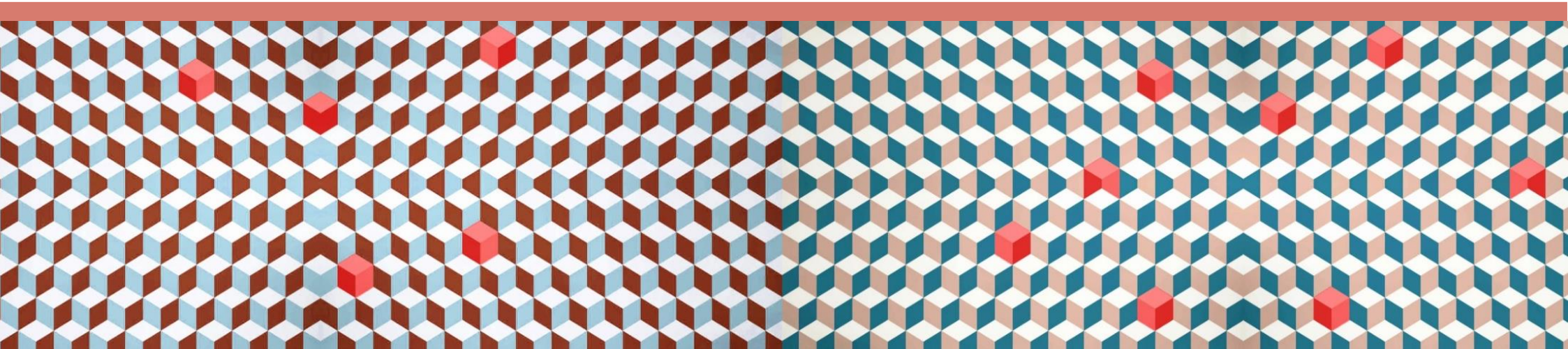
Scheduling Disco, Sincronizzazione:

Esercitazione e concetti

Tutor: Giovanni Hauber



Scheduling del disco

Si supponga che nella coda delle richieste di un'unità disco si trovano, nell'ordine, le richieste dei dati alle seguenti tracce:

0 - 0 - 0 - 3 - 29 - 29 - 31 – 30

Si assuma inoltre che:

- l'unità disco sia composta da 400 tracce (0 a 399)
- il tempo di ricerca, T_s, è pari ad 0,1 ms
- la latenza rotazionale, T_r, è pari ad 11,6 ms
- il tempo di trasferimento, T_t, è pari ad 0,08 ms

Se la testina ha eseguito l'ultimo movimento portandosi dalla traccia 120 alla traccia 150, si calcoli il tempo necessario per l'accesso alle tracce secondo:

- SSTF
- C-LOOK nel caso che arrivi al tempo 40ms la richiesta di lettura per la traccia 7.

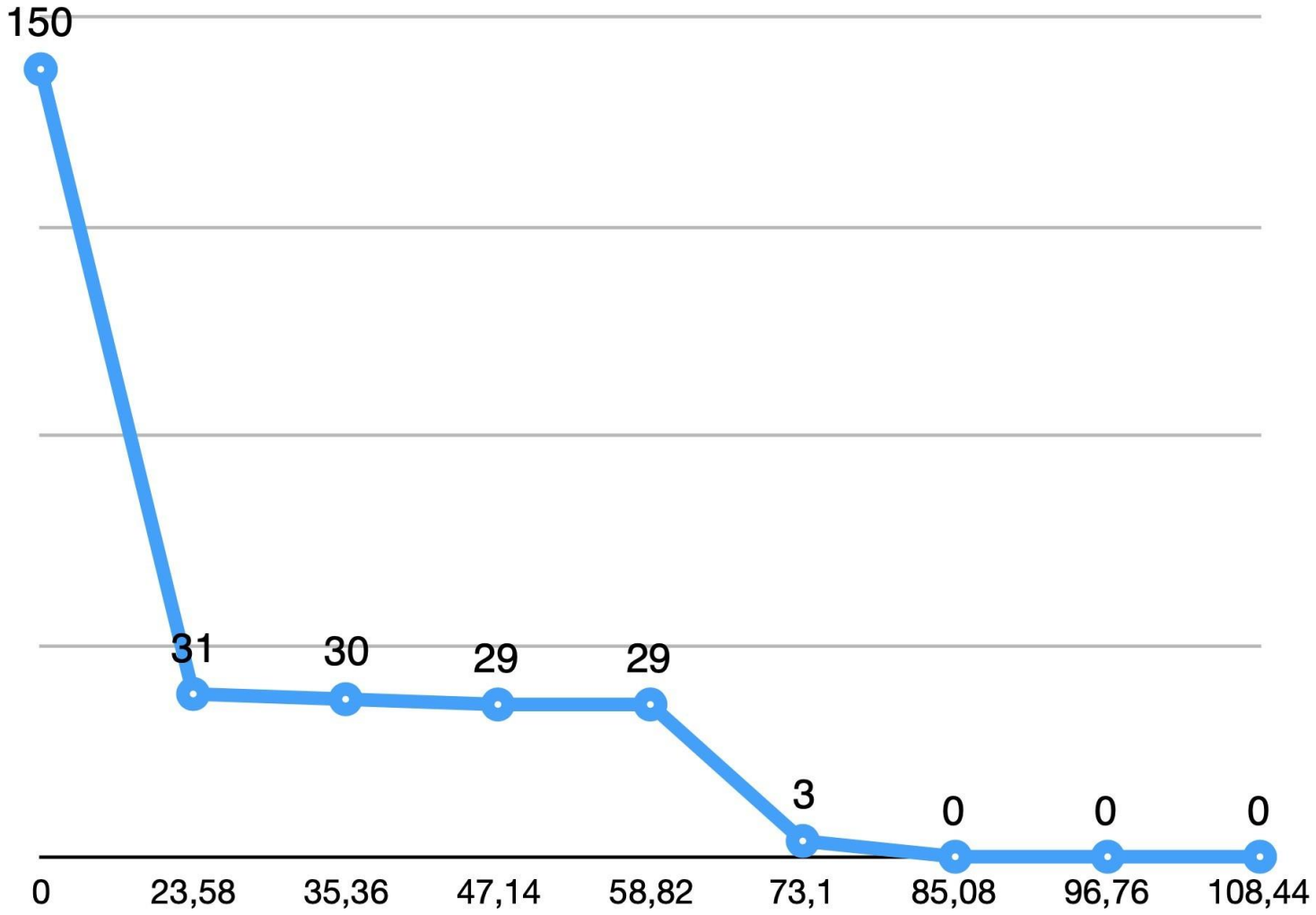
Tempo di accesso ad una traccia

- Il tempo totale di accesso ad una traccia è ottenuto come:
  - $T_a = T_s + T_r + T_t$, con:
    - T_s = Tempo di ricerca, ovvero il tempo necessario per posizionare la testina sulla traccia richiesta;
    - T_r = Latenza rotazionale, ovvero il tempo per accedere al record selezionato sulla traccia;
    - T_t = Tempo di trasferimento, ovvero il tempo necessario per trasferire un numero di blocchi da leggere/scrivere.

Quando ci spostiamo da una traccia all'altra, il tempo di accesso diventa:
Ta = (Traccia1-Traccia2)*Ts + Tr + Tt!

Politiche di scheduling: SSTF

- SSTF: Seleziona l'operazione di I/O con il più breve tempo di ricerca rispetto alla posizione corrente delle testine sul disco.

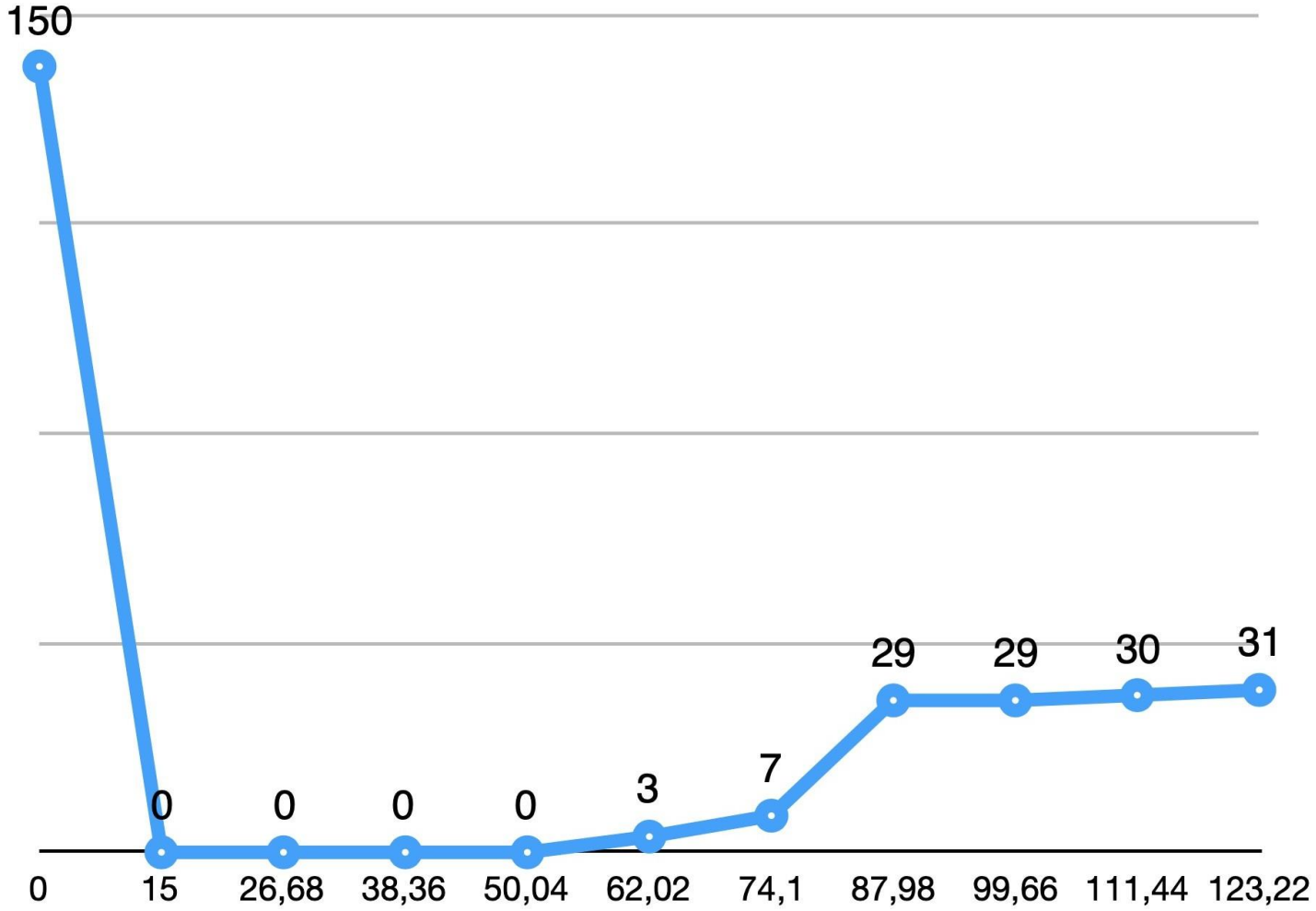


Tempi di accesso totali SSTF

Tempo di accesso alla traccia	Tempo di accesso totale
[150→31]=(150−31)∗0,1ms+11,6 ms+0,08ms=23,58ms	23,58ms
[31→30]=(31−30)∗0,1ms+11,6ms+0,08ms=11,78ms	35,36ms
[30→29]=(30−29)∗0,1ms+11,6 ms+0,08ms=11,78 ms	47,14ms
[29→29]=(29−29)∗0,1ms+11,6ms+0,08ms=11,68ms	58,82ms
[29→3]=(29−3)∗0,1ms+11,6ms+0,08ms=14,28ms	73,1ms
[3→0]=3∗0,1ms+11,6 ms+0,08ms=11,98ms	85,08ms
[0→0]=0∗0,1ms+11,6 ms+0,08ms=11,68ms	96,76ms
[0→0]=0∗0,1ms+11,6 ms+0,08ms=11,68ms	108,44ms

Politiche di scheduling: C-LOCK

- C-LOOK: sposta le testine nella sua direzione, che in questo caso è in senso crescente, perché si muove inizialmente dalla traccia 120 alla traccia 150, finché ci sono richieste da eseguire per poi riposizionarsi sulla estremità opposta e ripartire con la schedulazione. Come richiesto dobbiamo tener conto che a 40ms arriva una richiesta di accesso alla traccia 7.



Tempi di accesso totali C-LOCK

Tempo di accesso alla traccia	Tempo di accesso totale
[150→0]=(150−0)∗0,1ms=15ms;	15ms
[0→0]=0ms+11,6 ms+0,08ms=11,68ms;	26,68ms
[0→0]=0ms+11,6ms+0,08 ms=11,68ms	38,36ms
[0→0]=0ms+11,6ms+0,08 ms=11,68ms	50,04ms
[0→3]=3∗0,1ms+11,6 ms+0,08ms=11,98ms	62,02ms
[3→7]=4∗0,1ms+11,6ms+0,08ms=12,08ms	74,1
[7→29]=(29−7)∗0,1ms+11,6ms+0,08ms=13.88ms	87,98ms
[29→29]=(29−29)∗0,1ms+11,6ms+0,08 ms=11,68ms	99,66ms
[29→30]=(30−29)∗0,1ms+11,6 ms+0,08ms=11,78 ms	111,44ms
[30→31]=(31−30)∗0,1ms+11,6ms+0,08ms=11,78ms	123,22ms

Esercizio 2

Si supponga che nella coda delle richieste di un'unità disco si trovano, nell'ordine, le richieste dei dati alle seguenti tracce:

0 - 0 - 0 - 4 - 15 - 71 - 34 – 48

Si assuma inoltre che:

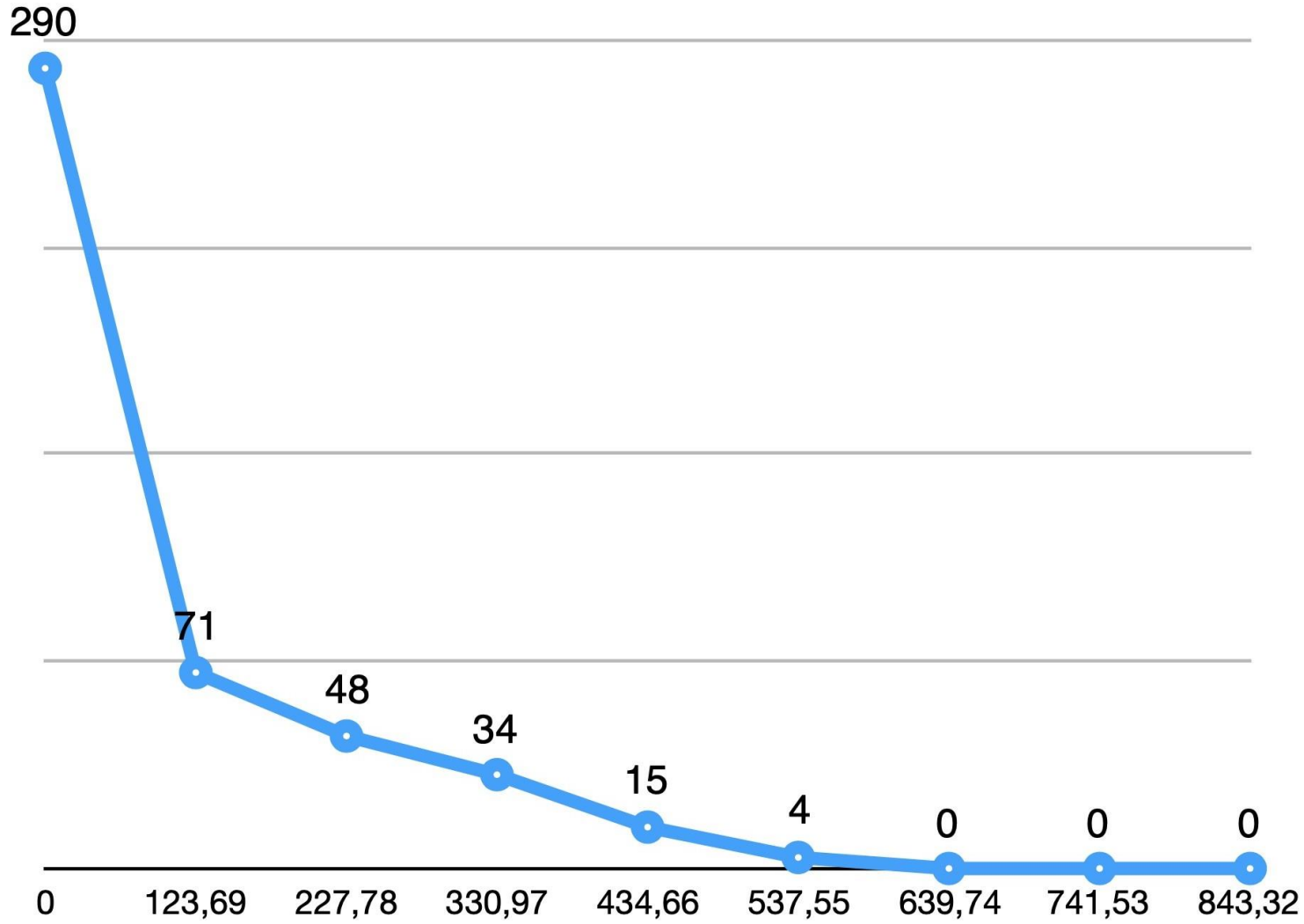
- l'unità disco sia composta da 400 tracce (0 a 399)
- il tempo di ricerca, T_s, è pari ad 0,1 ms
- la latenza rotazionale, T_r, è pari ad 101,4 ms
- il tempo di trasferimento, T_t, è pari ad 0,39 ms

Se la testina ha eseguito l'ultimo movimento portandosi dalla traccia 230 alla traccia 290, si calcoli il tempo necessario per l'accesso alle tracce secondo:

- SSTF
- C-SCAN nel caso che arrivi al tempo 10ms la richiesta di lettura per la traccia 8.

Politiche di scheduling: SSTF

- SSTF: Seleziona l'operazione di I/O con il più breve tempo di ricerca rispetto alla posizione corrente delle testine sul disco.

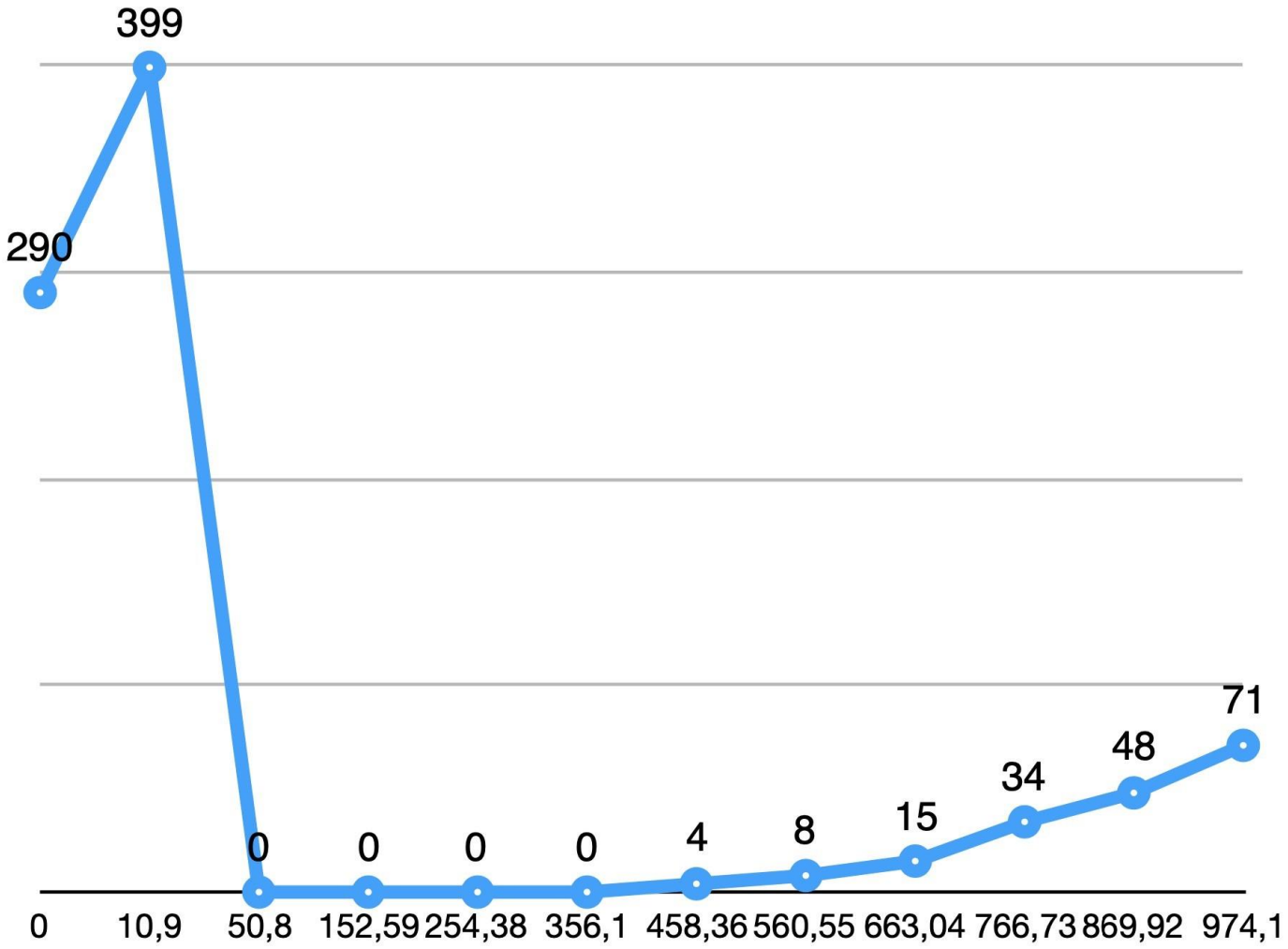


Tempi di accesso totali SSTF

Tempo di accesso alla traccia	Tempo di accesso totale
[290→71]=(290−71)∗0,1ms+101,4ms+0,39ms=123,69ms	123,69ms
[71→48]=(71−48)∗0,1ms+101,4ms+0,39ms=104,09ms	227,78ms
[48→34]=(48−34)∗0,1ms+ 101,4ms+0,39ms=103,19ms	330,97ms
[34→15]=(34−15)∗0,1ms+101,4ms+0,39ms=103,69ms	434,66ms
[15→4]=(15−4)∗0,1ms+101,4ms+0,39ms=102,89ms	537,55ms
[4→0]=(4-0)∗0,1ms+101,4ms+0,39ms=102,19ms	639,74ms
[0→0]=0∗0,1ms+101,4ms+0,39ms=101,79ms	741,53ms
[0→0]=0∗0,1ms+101,4ms+0,39ms=101,79ms	843,32ms

Politiche di scheduling: C-SCAN

C-SCAN: Questa politica esegue la scansione come nello scheduling SCAN. Tuttavia non esegue mai la scansione inversa; invece sposta le testine nella posizione di partenza sul piatto e inizia un'altra scansione. Come richiesto dobbiamo tener conto che a 10ms arriva una richiesta di accesso alla traccia 8.



Tempi di accesso totali C-SCAN

Tempo di accesso alla traccia	Tempo di accesso totale
[290→399]=(399–290)∗0,1ms=10,9ms	10,9ms
[399→0]=(399–0)∗0,1ms=39,9ms	50,8ms
[0→0]=(0–0)∗0,1ms+ 101,4ms+0,39ms=101,79ms	152,59ms
[0→0]=(0–0)∗0,1ms+101,4ms+0,39ms=101,79ms	254,38ms
[0→0]=(0–0)∗0,1ms+101,4ms+0,39ms=101,79ms	356,17ms
[0→4]=(4–0)∗0,1ms+101,4ms+0,39ms=102,19ms	458,36ms
[4→8]=(8-4)∗0,1ms+101,4ms+0,39ms=102,19ms	560,55ms
[8→15]=(15-8)∗0,1ms+101,4ms+0,39ms=102,49ms	663,04ms
[15→34]=(34-15)∗0,1ms+101,4ms+0,39ms=103,69ms	766,73ms
[34→48]=(48-34)∗0,1ms+101,4ms+0,39ms=103,19ms	869,92ms
[48→71]=(71-48)∗0,1ms+101,4ms+0,39ms=104,09ms	974,01ms

Sincronizzazione

Jurassic Park è costituito da un museo dei dinosauri e da un parco safari.
Nel parco ci sono M visitatori ed N autovetture monoposto (M>N).
I visitatori vagano per il museo per un po', quindi si mettono in fila per salire in un'auto da safari.
Quando un'auto è disponibile, carica l'unico visitatore che può contenere e gira per il parco per un periodo di tempo casuale.
Se le N auto sono tutte fuori a portare i visitatori in giro, allora un visitatore che intende fare il safari in auto aspetta;
se un'auto è pronta per iniziare un nuovo giro ma non ci sono visitatori in attesa, l'auto attende.

Sincronizzazione: breakdown

Jurassic Park è costituito da un museo dei dinosauri e da un parco safari.

- Nel parco ci sono M visitatori ed N autovetture monoposto (M>N).
- I visitatori vagano per il museo per un po', quindi si mettono in fila per salire in un'auto da safari.
- Quando un'auto è disponibile, carica l'unico visitatore che può contenere e gira per il parco per un periodo di tempo casuale.
- Se le N auto sono tutte fuori a portare i visitatori in giro, allora un visitatore che intende fare il safari in auto aspetta; se un'auto è pronta per iniziare un nuovo giro ma non ci sono visitatori in attesa, l'auto attende.

Usare i semafori per sincronizzare i visitatori e le auto. Si ricorda che l'esercizio deve essere svolto in pseudocodice e NON in codice C.

Soluzione: main e init

```
semaforo contatore: clienti = 0
semaforo contatore: auto_disponibili = N
array semaforo binario: sali_auto[N] = 0
array semaforo binario: finisci_visita[N] = 0
mutex: accesso_risorse = 1


main() {

  for i=0 to N:
    pthread_create(autovettura, i)

  for i=0 to M:
    pthread_create(visitatore)

}
```

Soluzione: Visitatore

```
semaforo contatore: clienti = 0
semaforo contatore: auto_disponibili = N
array semaforo binario: sali_auto[N] = 0
array semaforo binario: finisci_visita[N] = 0
mutex: accesso_risorse = 1
```

```
visitatore() {

  // vago per museo
  sleep(5)

  // controllo se ci sono macchine disponibili in mutua esclusione,
  // quindi mi prendo il valore della macchina scelta
  wait(accesso_risorse)
  wait(auto_disponibili)
  macchina = sem_getvalue(auto_disponibili) + 1
  signal(accesso_risorse)

  // avverto presenza
  signal(clienti)

  // Segnalo al guidatore che sto salendo in macchina; se arrivo a questo punto,
  // vuol dire che ho già preso un accesso ad auto disponibili in precedenza.
  signal(sali_auto[macchina])

  // Aspetto di finire la visita con questa macchina
  wait(finisci_visita[macchina])

  // esco

}
```

Soluzione: Autovettura

```
semaforo contatore: clienti = 0
semaforo contatore: auto_disponibili = N
array semaforo binario: sali_auto[N] = 0
array semaforo binario: finisci_visita[N] = 0
mutex: accesso_risorse = 1
```

```
autovettura(i){

  while(true) {

    // Aspetto visitatori
    wait(clienti)

    // Aspetto che il visitatore salga a bordo
    wait(sali_auto[i])

    // faccio fare la visita
    fai_giro_parco()

    // Aggiorno le risorse; comunico che la visita è finita, e incremento
    // auto disponibili
    signal(finisci_visita[i])
    signal(auto_disponibili)

  }

}
```