DEKKER VA BANKIR ALGORITMLARI: ZAMONAVIY OPERATSION TIZIMLAR VA PARALLEL DASTURLASH KONTEKSTIDA ILMIY TAHLIL

Umarov Bekzod Azizovich
Farg'ona davlat universiteti amaliy matematika va informatika kafedrasi katta o'qituvchisi
ubaumarov@gmail.com
Sobirov Asadbek Nodirjon o'g'li
Farg'ona Davlat Universiteti talabasi
asadbeksobirov@yandex.ru

Annotatsiya. Maqolada zamonaviy operatsion tizimlar va parallel dasturlash sharoitida Dekker va Bankir algoritmlarining mohiyati, ishlash usullari, farqlari, afzalliklari hamda cheklovlari tahlil qilinadi. Dekker algoritmi ikki jarayon (ip) o‘rtasida o‘zaro eksklyuziya (mutual exclusion) ta’minlashga xizmat qilsa, Bankir algoritmi ko‘p jarayonli tizimda resurslarni xavfsiz taqsimlash orqali deadlock (o‘lik blok) holatining oldini olishga qaratilgan.

Kalit so‘zlar: O‘zaro eksklyuziya (mutual exclusion), O‘lik blokdan qochish (deadlock avoidance), Parallel dasturlash, Ko‘p yadrolik arxitektura (multi-core architecture), Dekker algoritmi, Bankir algoritmi.

Abstract. This article analyzes the Dekker and Banker algorithms in the context of modern operating systems and parallel programming. The Dekker algorithm ensures mutual exclusion between two processes, while the Banker algorithm focuses on preventing deadlock through safe resource allocation in multiprocess systems. The paper discusses the working principles, advantages, limitations, and applicability of both algorithms.

Keywords: Mutual exclusion, deadlock avoidance, parallel programming, multi-core architecture, Dekker’s algorithm, Banker’s algorithm.

Аннотация. В статье проводится анализ алгоритмов Деккера и банкира в контексте современных операционных систем и параллельного программирования. Алгоритм Деккера обеспечивает взаимное исключение между двумя процессами, тогда как алгоритм банкира предотвращает взаимоблокировки за счёт безопасного распределения ресурсов в многопроцессных системах. Рассматриваются принципы работы, преимущества, ограничения и применимость обоих алгоритмов.

Ключевые слова: Взаимное исключение, предотвращение взаимоблокировок, параллельное программирование, многоядерная архитектура, алгоритм Деккера, алгоритм банкира.

Kirish. Bugungi kunda kompyuter tizimlari keng ko‘lamda parallel ishlashni qo‘llaydi va ko‘plab jarayonlar bitta tizim resurslariga bir vaqtning o‘zida murojaat qilishi mumkin. Bunday holatlarda kritik bo‘lim (critical section)ga kirayotgan jarayonlar o‘rtasida ziddiyat yuzaga kelmasligi (o‘zaro eksklyuziya) va o‘lik blok (deadlock) holatlari sodir bo‘lishining oldi olinishi muhimdir. Ushbu muammolarni hal qilish uchun turli algoritmik yondashuvlar ishlab chiqilgan. Ular orasida Th.J. Dekker tomonidan taklif qilingan ikki jarayonlik o‘zaro eksklyuziya algoritmi muhim ahamiyatga ega bo‘lib, u mutlaq to‘g‘ri ishlashga ega bo‘lgan birinchi algoritm sifatida tarixga kirgan. Shuningdek, Edsger Deykstraning «Bankir» (Banker’s) algoritmi resurslarni xavfsiz taqsimlash yo‘li bilan deadlockdan qochishni ta’minlashga mo‘ljallangan. Ushbu maqolada yuqoridagi algoritmlarning ishlash mexanizmi, ularning o‘ziga xos afzalliklari va cheklovlari, shuningdek, zamonaviy operatsion tizimlar hamda ko‘p yadrolik arxitekturalarda ularning qo‘llanilish imkoniyatlari batafsil tahlil qilinadi.

Dekker algoritmi. Dekker algoritmi — ikki jarayon (ip) o‘rtasidagi o‘zaro eksklyuziya muammosini hal qilish uchun ishlab chiqilgan dasturiy yechimdir. Har bir jarayon (p0 va p1) kritik bo‘limga kirishni istashi bilan tegishli *wants_to_enter[i]* bayroq o‘rnatiladi va *turn* («navbat») o‘zgaruvchisi orqali prioritet belgilanadi. Rasmda Quyida Dekker algoritmining jarayonlar oqimi berilgan:

Dekker algoritmining ikki jarayon uchun ishlash oqimi: har bir jarayon kritik bo‘limga kirishdan oldin flag va turn tekshiradi.

Dekker algoritmida birinchi bo‘lib kirish uchun navbat belgilanadi; agar ikki jarayon ham kritik bo‘limga kirmoqchi bo‘lsa, *turn* o‘zgaruvchisi qaysi jarayonga ustunlik berishini belgilaydi. Boshqaruv siklli kutish (busy-wait) yordamida amalga oshiriladi: agar ikkinchi jarayon kritik bo‘limda bo‘lsa va *turn* unga tegishli bo‘lmasa, birinchi jarayon flagini «o‘chirib», *turn* o‘zgaruvchisi ustun bo‘lganicha kutadi, so‘ng flagni qayta yoqib, kritik bo‘limga kiradi. Dekker algoritmi bajarilgach, jarayon o‘z navbatini ikkinchiga o‘tkazadi va flagini o‘chiradi. Natijada, bu algoritm *o‘zaro eksklyuziya* holatini kafolatlaydi va holatdagi o‘lik blok hamda jarayonlarning cheklangan darajada kutib qolishini oldini oladi.

Dekker algoritmining asosiy afzalligi shundaki, u faqat umumiy xotira va bayroqlar (flag) yordamida ishlaydi va mexanik signalizatsiya vositalarini talab qilmaydi. U doimiy navbatga asoslangan oddiy konstruktsiyaga ega bo‘lib, to‘g‘ri ishlashini matematik jihatdan isbot qilish mumkin. Ammo uning bir qator cheklovlari ham mavjud. Avvalo, algoritm faqat ikkita jarayon uchun mo‘ljallangan: uch yoki undan ortiq jarayon bilan ishlash talab qilinsa, uni bevosita kengaytirib bo‘lmaydi. Shuningdek, Dekker algoritmida jarayonlar faqat siklli kutish orqali (busy-wait) ishlaydi, bu esa ko‘p jarayonli muhitda samaradorlikni pasaytiradi. Yana bir jiddiy cheklov: bugungi kunda ko‘p yadrolik protsessorlar instruktsiyalarni ketma-ket emas, tartibsiz bajarishi mumkin. Shuning uchun Dekker algoritmi SMP (Symmetric Multi-Processing) muhitida xotira to‘sig‘i (memory barrier) ishlatilmasa, noto‘g‘ri natijalar beradi. Zamonaviy C/C++ kabi dasturlash tillarida *volatile* yoki atomik o‘zgaruvchilardan foydalanib va kerakli xotira to‘siqlarini qo‘llash orqali Dekker algoritmi to‘g‘ri ishlashi ta’minlanishi mumkin.

Dekker algoritmining afzalliklari va cheklovlari:

- O‘zaro eksklyuziya, o‘lik blok va starvation holatlarini kafolatlaydi.
- Ikki jarayonli muhit uchun oddiy va ishonchli algoritm bo‘lib, mexanik sinxronizatsiya talab qilmaydi.

- Cheklovlari: faqat ikki jarayon uchun ishlaydi, busy-waiting usuli ishlatiladi, katta yadrolik tizimlarda xotira tartibini ta’minlash uchun maxsus vositalar talab etiladi.

Bankir algoritmi. Bankir algoritmi resurslarga nisbatan deadlock (o‘lik blok) oldini olish uchun mo‘ljallangan yondashuvdir. Bu algoritm har bir jarayon tomonidan talab qilinishi mumkin bo‘lgan maksimal resurslar sonini (Max), hozirgacha jarayonga berilgan resurslar sonini (Allocated) va tizimda mavjud bo‘lgan bo‘sh resurslar sonini (Available) kuzatib boradi. Har bir resurs so‘rovi paytida Bankir algoritmi vaqtincha resurslarni ajratish taqlidini (emulyatsiyasini) o‘tkazib, tizim holatining xavfsiz (safe) ekanligini tekshiradi. Agar so‘rov resurslarni ajratgandan so‘ng tizimning xavfsiz holatda bo‘lishi ta’minlansa, so‘rov bajariladi, aks holda rad etiladi. Bu holatda «xavfsiz holat» deganda tizimda mavjud bo‘lgan resurslar yordamida har bir jarayon ketma-ket to‘liq bajarilishi mumkin bo‘lgan ketma-ketlik topilishi tushuniladi.

Bankir algoritmi komponentlari quyidagicha: *Available* (mavjud resurslar), *Max* (jarayonlarning maksimal ehtiyoji), *Allocation* (ajratilgan resurslar) va *Need* (jarayonlarning qo‘shimcha ehtiyoji). Har bir yangi jarayon tizimga kirganda, u o‘z maksimal ehtiyojini deklaratsiya qilishi kerak (shuningdek, bu son tizimdagi jami resurslar sonidan oshmasligi lozim). Keyin resurs so‘rov bo‘lganda, Bankir algoritmi ushbu so‘rovni vaqtincha ajratib, qolgan jarayonlar oxirigacha yetishi uchun yetarliligi (safe state)ni tekshiradi. Agar xavfsiz ketma-ketlik mavjud bo‘lsa, algoritm resurslarni ajratishni amalga oshiradi; aks holda so‘rovni rad etadi. Bankir algoritmi shu tariqa tizimni hech qachon deadlockga tushmaydigan xavfsiz holatda saqlashga harakat qiladi.

Bankir algoritmining afzalliklari va cheklovlari:

- Afzalliklari: Nazariy jihatdan to‘g‘ri ishlashi mumkin va tizim o‘lik blokga tushmaydigan holatda qoladi.
- Cheklovlari: Har bir jarayon o‘z maksimal ehtiyojini oldindan bildirishini talab qiladi, bu amaliy tizimlarda ko‘pincha noma’lum bo‘ladi, algoritmni amalda qo‘llashni imkonsiz qiladi. Jarayonlar sonining dinamik o‘zgarishi va jarayonlar yakunlanishini uzoq

vaqt kutish kabi talablar esa haqiqiy tizimlarga mos kelmaydi. Shuningdek, operatsion tizimlarda jarayonlar va resurslar juda katta bo‘lgani sababli, Bankir algoritmini har bir resurs so‘rovi uchun bajarish katta hisob-kitob yukini keltirib chiqaradi. Shu bois an’anaviy operatsion tizimlarda deadlockdan qochish uchun odatda resurslarga qat’iy tartib berish yoki deadlockni aniqlab tiklash usullari afzalroq qo‘llaniladi.

Algoritmlarning farqlari. *Dekker* va *Bankir* algoritmlari yechishga urinayotgan muammolari jihatidan farq qiladi. Dekker algoritmi ikki jarayon o‘rtasida yagona resursni o‘zaro tolmay ishlatishni (kritik bo‘limni almashib ishlashni) ta’minlaydi. Bankir algoritmi esa ko‘p jarayonli muhitda resurslarni xavfsiz holatda taqsimlashga yo‘naltirilgan va uning maqsadi tizimni o‘lik blokka tushib qolmasligidir. Dekker oddiy bo‘lak (bit) bayroqlar va navbat o‘zgaruvchisi orqali ishlasa, Bankir resurs ajratish emulyatsiyasi yordamida xavfsiz holat tekshiruviga tayangan. Boshqa farqlar: Dekker faqat ikki jarayon bilan ishlaydi va bir xil turdagi yagona resursga nisbatan mo‘ljallangan, Bankir esa N ta jarayon va M xil resurslarni hisobga oladi.

Tahlil: Zamonaviy operatsion tizimlar va ko‘p yadrolik arxitekturalarda qo‘llanilishi. Zamonaviy kompyuterlar deyarli har doim **ko‘p yadrolik arxitekturaga** ega bo‘lib, bitta chipda bir nechta protsessor yadrolari birlashgan. Har bir yadro alohida ijro bloki sifatida ishlasa-da, ular barchasi umumiy tizim xotirasini bo‘lishadi. Shu bilan birga, ko‘p yadrolik tizimlarda xotira ketma-ketligini (memory ordering) ta’minlash muammosi dolzarbdir: amaldagi protsessorlar instruktsiyalarni tartibsiz bajarishi tufayli, oddiy Dekker algoritmi SMP muhitda xatolarga olib kelishi mumkin. Shu sababli zamonaviy C/C++ dasturlash tillarida Dekker algoritmidagi bayroq va navbat o‘zgaruvchilarini *atomik* (atomic) qilib belgilash yoki kerakli xotira to‘siqlarini o‘rnatish tavsiya etiladi; C++11 standartida atomik o‘zgaruvchilar ketma-ketlikni kafolatlaydi, shuning uchun ularni qo‘llash kodni to‘g‘ri ishlashini taminlaydi. Amaliy operatsion tizimlarda esa o‘zaro eksklyuziya uchun odatda Dekker kabi dasturiy algoritmlar o‘rniga apparat yondashuvlari (test-and-set, compare-and-swap) va sinxronizatsion primitivlar (mutex, semaphor) ishlatiladi.

Bankir algoritmi zamonaviy operatsion tizimlarda kam qo‘llaniladi. Chunki u ishlashi uchun har bir jarayon o‘z maksimal resurs ehtiyojini oldindan bilishi va jarayonlar soni statik bo‘lishi lozim; bu talablar real tizimlarda deyarli hech qachon bajarilmaydi. Tarmoq uskunalarida yoki ba’zi maxsus maqsadli tizimlarda esa Bankir algoritmi g‘oyasi ishlatilishi mumkinligi qayd qilingan: masalan, tarmoq jihozlarining ba’zi protokollarida resurslarni ajratishda shu kabi nazariy yondashuvga murojaat qilinishi mumkin. Biroq keng tarqalgan operatsion tizimlarda deadlockni bartaraf etish uchun ko‘proq deadlock deteksiyasi yoki oldini olishning statik strategiyalari (resurslarga qat’iy ketma-ketlik berish) qo‘llanadi.

Shuni ta’kidlash kerakki, ilmiy jamoatchilik ham Bankir algoritmini optimallashtirish ustida ishlamoqda. Masalan, ko‘p protsessorli tizimlar uchun Bankir algoritmini apparat moslamasi ko‘rinishida amalga oshirib, eng yaxshi holatda O(1) vaqt ishini ta’minlash mumkinligi ko‘rsatilgan. Bu kabi tadqiqotlar nazariy jihatdan algoritm samaradorligini oshirishga xizmat qiladi, ammo amaliy OS-lar hali ham bunday yondashuvni keng qo‘llamaydi.

Xulosa. Dekker va Bankir algoritmlari nafaqat parallel dasturlash nazariyasida, balki operatsion tizimlar mo‘him qismida ham boshqaruv strukturalarining poydevori hisoblanadi. Dekker algoritmi birinchi to‘g‘ri mutual exclusion yechimlaridan bo‘lib, ikki jarayon o‘rtasida o‘zaro bekorga yo‘l qo‘ymaslikni kafolatlaydi. Biroq zamonaviy ko‘p yadrolik arxitekturalarda xotira tartibini nazorat qilish muammolari tufayli, u odatda til darajasidagi atomik o‘zgaruvchilar yoki apparat mexanizmlar orqali qo‘llaniladi. Bankir algoritmi esa resurslar xavfsizligi konseptini birinchi marta taklif qilgan bo‘lib, nazariy jihatdan tizimni o‘lik blokdan himoya qiladi. Ammo u uchun jarayonlar maksimal ehtiyojini bilish kabi shartlar haqiqiy tizimga to‘g‘ri kelmaydi, shuning uchun amaliy OS-larda aksariyat hollarda ishlatilmaydi. Natijada, amaliy dasturlash va OS-larda resurslarni boshqarish uchun odatda yuqori darajadagi sinxronizatsiya vositalari va deadlockni aniqlash mexanizmlari afzallik beriladi. Shu bilan birga, Dekker va Bankir algoritmlarini o‘rganish o‘quvchilarni dasturlash va tizim bilimlari asoslarini mustahkamlashda katta foyda beradi.

Adabiyotlar ro‘yxati

1. Umarov B. RAQAMLI TEXNOLOGIYALAR VOSITASIDA PEDAGOGLARNING PROFESSIONAL KOMPETENTLIGINI RIVOJLANTIRISH MAZMUNI //Евразийский журнал математической теории и компьютерных наук. – 2023. – Т. 3. – №. 5. – С. 87-93.

2. Azizovich U. B. PRINCIPLES OF FORMING TEACHER COMPETENCE THROUGH INNOVATIVE TECHNOLOGIES. Finland International Scientific Journal of Education //Social Science & Humanities. – 2023. – Т. 11. – №. 5. – С. 823-828.

3. Azizovich U. B. PEDAGOGICAL-PSYCHOLOGICAL PRINCIPLES OF THE FORMATION OF PROFESSIONAL COMPETENCE //Confrencea. – 2023. – Т. 6. – №. 6. – С. 204-212.

4. Azizovich U. B., Zarifjon o’g’li X. N. BULUT TEXNOLOGIYALARINING AFZALLIKLARI VA KAMCHILIKLARI //TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI. – 2024. – Т. 1. – №. 1. – С. 46-54.

5. Azizovich U. B., Rustamjon o‘g‘li R. Z. MA’LUMOTLARNI SHIRFLASH TENALOGIYALARI VA XAVFSIZLIK STANDARTLARI //TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI. – 2024. – Т. 1. – №. 1. – С. 105-108.

6. Azizovich U. B. et al. OLAP TIZIMLARINING ASOSIY PRINSIPLARI //TA’LIM, TARBIYA VA INNOVATSIYALAR JURNALI. – 2024. – Т. 1. – №. 1. – С. 81-86.

7. Azizovich U. B. THE DEVELOPMENT OF PROFESSIONAL COMPETENCY OF TEACHERS IN EDUCATIONAL TECHNOLOGY BASED ON DIGITAL TECHNOLOGIES //Eurasian Journal of Mathematical Theory and Computer Sciences. – 2024. – Т. 4. – №. 7. – С. 11-14.

8. Azizovich U. B. et al. MASHINALI O ‘QITISHDA REGRESSIYA ENG KICHIK KVADRATLAR USULINI QO ‘LLASH //INNOVATION IN THE MODERN EDUCATION SYSTEM. – 2024. – Т. 5. – №. 46. – С. 266-270.

9. *Wikipedia*, “Dekker’s algorithm”.

10. *Wikipedia*, “Banker’s algorithm”.

11. *Wikipedia*, “Multi-core processor”.

12. GeeksforGeeks, *“Banker’s Algorithm in Operating System”*.

13. Reddit, *r/computerscience*, *“Is the Banker's algorithm by Dijkstra used practically in any operating systems?”*.