guruh 026-18 Bahronov Khusan

Fanning nomi: Algortimlarni loihalash

Yakuniy Nazorat

Variant -15

Savollar:

1.Demon oqimlari nima?

2.NP-toliqlik masalasi nima?

Testga Savollar:

1. Static void sleeep nima qiladi?

Savob: (A)- Soriy oqimni kamida millisekundlar-da bajarishni tuhtatadi

2. Void start nima qiladi?

Savob: (B)- oqimni bajarishni boshlaydi

3.Agar takrorlanuvchi algoritmlar bir nechta parametrlarga bog'liq bo'lsa ular qanday nom-

lanadi?

Javob: (C)- ichma-ich joylashgan siklik algoritmlar

4. Algoritm deb, ...

Javob: (A)- qo'yilgan masalani yechish uchun ma'lum qoidaga binoan bajariladigan amallarning chekli qadamlar ketma-ketligiga aytiladi

5. Algoritmning diskretlilik xossasi –

Javob: (D)- algoritmlarni chekli qadamlardan tashkil qilib bo'laklash imkoniyati bo'lishi.

Yozma savollarga javoblar:

1.Demon thread-bu dastur tugagach, JVM chiqishining oldini olmaydigan oqim, lekin oqim hali ham ishlaydi. Demon oqimi uchun namuna-axlat yig'ishOqim boshlanishidan oldin jinlarning xususiyatlarini o'zgartirish uchun setDaemon(boolean) usulidan foydalanishingiz mumkin.

Yana bir nechta fikrlar (havola: amaliyotda Java kontseptsiyasi) Yangi oqim yaratilganda, u ota-onasining jin maqomini egallaydi. JVM to'xtatilganda va jinlarning qolgan barcha oqimlari o'chirilganda, jinlarning barcha oqimlari tugaydi: nihoyat, bloklar bajarilmaydi, stacklar ochilmaydi-JVM faqat tashqariga chiqadi. Shu sababli, jinlarning oqimlari ehtiyotkorlik bilan ishlatilishi kerak va har qanday i / u operatsiyalarini bajarishi mumkin bo'lgan vazifalar uchun foydalanish xavfli.

```
public class DaemonTest{
    public static void main(String[] args)
{
    new WorkerThread().start();
try {
    Thread.sleep(7500);
    } catch (InterruptedException e)
{
    // handle here exception
```

```
}
        System.out.println("Main Thread
ending") ;}}
    class WorkerThread extends Thread{
  public WorkerThread() {
      // When false, (i.e. when it a user
thread),
      // the Worker thread continues to
run.
      // When true, (i.e. when it a da-
emon thread),
      // the Worker thread terminates
when the main
      // thread terminates.
setDaemon(true);}
    public void run() {    int count = 0;
while (true){
        System.out.println("Hello
from Worker "+count++);
        try {          sleep(5000);
```

}

catch (InterruptedException e) { //

handle exception here } }}}

2. nchisiga javob

Polinomial vaqt samaradorlik ko'rsatkichi sifatida. Tabiiy kombinatorial masalalarda qidirish vaqti, kirish ma'lumotlari N hajmiga nisbatan eksponensional o'sishga moyildir; agar o'lcham bittaga ko'paysa, unda imkoniyatlar xajmi bir necha marta ko'payadi. Bunday masalalarni yechish uchun yaxshi algoritm yanada samarali miqyoslash modeliga ega bo'lishi kerak; kirish ma'lumotlarining kattalashib borishi bilan o'zgarmas ko'paytuvchiga(aytaylik, ikki baravar) oshishi bilan algoritmning bajarilish vaqti ham qandaydir o'zgarmas S ko'paytuvchiga ko'payishi kerak.

Matematik nuqtai nazardan ushbu masshtablash modelini quyidagicha shakllantirish mumkin. Aytaylik, algoritm quyidagi xususiyatga ega: c> 0 va d> 0 absolyut konstantalar mavjud-

ki, har qanday N xajmli kirish ma'lumotlari to'plami uchun, bajarilish vaqti cN^d sondagi eng sodda operatsiyalar soni bilan chegaralanadi. Boshqacha qilib aytganda, bajarilish vaqti N^d ga proportsionallikdan ko'p emas. Qanday bo'lmasin, ba'zi bir c va d lar uchun bajarilish vaqti ushbu chegaradan oshmaganda, algoritm polinomial bajarilish vaqtini ta'minlaydi deymiz yoki u polinomial vaqtga ega bo'lgan algoritmlar toifasiga kiradi. polinomial vaqtga ega har qanday chegara yuqoridagi masshtablashga ega bo'ladi.

(3)T: Agar algoritm polinomial bajarilish vaqtiga ega bo'lsa, u samarali deb ataladi.

Lekin, polinomial vaqt d ning katta qiymatlarida yaxshi natija bermasligi mumkin, masalan d>=100 holatda bu son juda katta bo'ladi, natijada polinomial bajarilish vaqti kattalashib ketadi. Algoritm ishlayveradi. Bu xolda N^d faqat chegara vazifasini o'taydi.

Polinomial vaqtga ega bo'lgan algoritmlar

mavjud bo'lgan masalalar uchun bu algoritmlar deyarli har doim n, n*log n, n^2 yoki n^3 kabi o'rtacha o'sish sur'ati bilan ko'paytuvchilarga mutanosib ravishda ishlaydi. Va aksincha, Polinomial vaqtga ega bo'lmagan algoritmli masalalar uchun bajarilish vaqti juda ham kattalashib ketadi, uni baholash noma'lum bo'ladi, odatda bunday masalalarni yechish juda murakkab bo'lib chiqadi. Umuman olganda, bunday baholashlar ba'zi masalalarda darajaning yoki koeffitsientlarning kattaligi sababli yaxshi natija bermaydi, algoritm yaxshi bo'lsa ham. Hjallanarlisi shuki, aksariyat hollarda polinomial vaqtni matematik aniqlash algoritmlarning samaradorligi va real hayotdagi masalalarni hal qilish bo'yicha kuzatishlarimiz bilan mos keladi. Bunday masalalarni polinomial yechish mumkin bo'lgan masalalar deyiladi. Demak, 3-ta'rifni ma'lum ma'noda talabga javob beruvchi ta'rif desak boladi. U xolda Polinomial vaqtga ega bo'lmagan algoritmlarni qayta ko'rib

chiqish kerak.

Polinomial vaqtga ega bo'lgan algoritmlarni ishlab chiqish nima uchun zarurligini quyidagi jadval ma'lumotlarini tahlil qilib bilish mumkin.

Polinomial vaqt samaradorlik ko'rsatkichi sifatida. Tabiiy kombinatorial masalalarda qidirish vaqti, kirish ma'lumotlari N hajmiga nisbatan eksponensional o'sishga moyildir; agar o'lcham bittaga ko'paysa, unda imkoniyatlar xajmi bir necha marta ko'payadi. Bunday masalalarni yechish uchun yaxshi algoritm yanada samarali miqyoslash modeliga ega bo'lishi kerak; kirish ma'lumotlarining kattalashib borishi bilan o'zgarmas ko'paytuvchiga(aytaylik, ikki baravar) oshishi bilan algoritmning bajarilish vaqti ham qandaydir o'zgarmas S ko'paytuvchiga ko'payishi kerak.

Matematik nuqtai nazardan ushbu masshtablash modelini quyidagicha shakllantirish mumkin. Aytaylik, algoritm quyidagi xususiyatga

ega: c> 0 va d> 0 absolyut konstantalar mavjudki, har qanday N xajmli kirish ma'lumotlari to'plami uchun, bajarilish vaqti cN^d sondagi eng sodda operatsiyalar soni bilan chegaralanadi. Boshqacha qilib aytganda, bajarilish vaqti N^d ga proportsionallikdan ko'p emas. Qanday bo'lmasin, ba'zi bir c va d lar uchun bajarilish vaqti ushbu chegaradan oshmaganda, algoritm polinomial bajarilish vaqtini ta'minlaydi deymiz yoki u polinomial vaqtga ega bo'lgan algoritmlar toifasiga kiradi. polinomial vaqtga ega har qanday chegara yuqoridagi masshtablashga ega bo'ladi.

(3)T: Agar algoritm polinomial bajarilish vaqtiga ega bo'lsa, u samarali deb ataladi.

Lekin, polinomial vaqt d ning katta qiymatlarida yaxshi natija bermasligi mumkin, masalan d>=100 holatda bu son juda katta bo'ladi, natijada polinomial bajarilish vaqti kattalashib ketadi. Algoritm ishlayveradi. Bu xolda N^d faqat chegara vazifasini o'taydi.

Polinomial vaqtga ega bo'lgan algoritmlar mavjud bo'lgan masalalar uchun bu algoritmlar deyarli har doim n, n*log n, n^2 yoki n^3 kabi o'rtacha o'sish sur'ati bilan ko'paytuvchilarga mutanosib ravishda ishlaydi. Va aksincha, Polinomial vaqtga ega bo'lmagan algoritmli masalalar uchun bajarilish vaqti juda ham kattalashib ketadi, uni baholash noma'lum bo'ladi, odatda bunday masalalarni yechish juda murakkab bo'lib chiqadi. Umuman olganda, bunday baholashlar ba'zi masalalarda darajaning yoki koeffitsientlarning kattaligi sababli yaxshi natija bermaydi, algoritm yaxshi bo'lsa ham. Hjallanarlisi shuki, aksariyat hollarda polinomial vaqtni matematik aniqlash algoritmlarning samaradorligi va real hayotdagi masalalarni hal qilish bo'yicha kuzatishlarimiz bilan mos keladi. Bunday masalalarni polinomial yechish mumkin bo'lgan masalalar deyiladi. Demak, 3-ta'rifni ma'lum ma'noda talabga javob beruvchi ta'rif desak boladi. U xolda Polinomial

vaqtga ega bo'lmagan algoritmlarni qayta ko'rib chiqish kerak.

Polinomial vaqtga ega bo'lgan algoritmlarni ishlab chiqish nima uchun zarurligini quyidagi jadval ma'lumotlarini tahlil qilib bilish mumkin.