**OʻZBEKISTON RESPUBLIKASI OLIY TAʼLIM, FAN VA INNOVATSIYALAR VAZIRLIGI**

|  |  |
| --- | --- |
| Logo | Muhammad al-Xorazmiy nomidagi Toshkent Axborot Texnologiyalari Universiteti Samarqand  filiali |

**MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI SAMARQAND FILIALI**

**KOMPYUTER INJINIRINGI FAKULTETI**

**“AXBOROT TEXNOLOGIYALARI” KAFEDRASI**

**MA’LUMOTLAR TUZILMASI VA ALGORITMLAR**

*(O‘zbek tilida)*

**YAKUNIY NAZORAT**

**TEST SAVOLLARI**

Kafedra mudiri: I.Sh. Xujayarov

Fanga masʼul(lar): I.Sh. Xujayarov

R.B. Kudratov

SAMARQAND\_2025

Abstrakt ma’lumotlar tuzilmasi nima?

====

Ma’lumotlar va ular ustida bajariladigan amallar majmuasi

====

Dasturlash tili

====

Faqat massivlar majmuasi

====

Input qurilmalarining to‘plami

++++

Algoritmning asosiy xossalaridan biri bu -…

====

Aniqlik (deterministik)

====

Raqamli ifoda

====

Ma’lumotlar bazasi

====

Kod uzunligi  
++++

Algoritm tahlilida vaqt murakkabligi nimalarni ifodalaydi?

====

Algoritm ishlashi uchun kerakli vaqt miqdori

====

Foydalanuvchi soni

====

Operativ xotira hajmi

====

Natijaning aniqligi  
++++

Ma’lumotlar tuzilmasining nechta asosiy turi mavjud?

====

2 (chiziqli va chiziqsiz)

====

1

====

3

====

Cheksiz  
++++

Qaysi biri ma’lumotlar tuzilmasi hisoblanadi?

====

Graf

====

Algoritm

====

Model

====

Fayl  
++++

Ma’lumotlar qanday bosqichlarda ifodalanadi?

====

Yig‘ish, saqlash, qayta ishlash, uzatish

====

Faqat saqlash

====

Foydalanish, yaratish

====

Chop etish, tahrirlash  
++++

Algoritmlarni ishlab chiqishda birinchi qadam bu -…

====

Masalani aniqlash va tahlil qilish

====

Kod yozish

====

Fayl ochish

====

Chiqarish  
++++

Ma’lumotlar tuzilmasining klassifikatsiyasi qanday amalga oshiriladi?

====

Tashkiliy shakli va ishlov berish usuliga ko‘ra

====

Yaratilgan yiliga ko‘ra

====

Kompyuter modeliga qarab

====

Kod uzunligiga ko‘ra  
++++

Ma’lumotlarning sodda turlariga nimalar kiradi?

====

Butun son, haqiqiy son, belgili, mantiqiy

====

Graf, daraxt

====

Massiv, ro‘yxat

====

Fayl, obyekt  
++++

Abstrakt ma’lumotlar tuzilmalari quyidagilardan iborat bo‘lishi mumkin:

====

Navbat, stek, ro‘yxat, daraxt

====

Faqat fayllar

====

Input qurilmalar

====

Protsessorlar  
++++

Massiv bu -…

====

Bir xil turdagi elementlar to‘plami

====

Turli turdagi qiymatlar jamlanmasi

====

Faqat butun sonlar yig‘indisi

====

Graf shaklidagi tuzilma  
++++

Vektorlar qanday turdagi tuzilmalarga kiradi?

====

Dinamik massivlar

====

Stollar

====

Ko‘rsatkichlar

====

Chiziqli graf  
++++

Yozuv (record) tuzilmasining asosiy xususiyati nima?

====

Turli turdagi ma’lumotlarni bitta birlikda saqlaydi

====

Faqat raqamli qiymatlar saqlaydi

====

Elementlar tartibsiz joylashgan bo‘ladi

====

Faqat o‘qishga mo‘ljallangan  
++++

To‘plam (set) ma’lumot turi quyidagilardan farqlanadi:

====

Takrorlanuvchi qiymatlarni saqlamaydi

====

Elementlar indekslangan bo‘ladi

====

Faqat sonli elementlar bo‘ladi

====

To‘plam ichida boshqa to‘plamlar bo‘ladi  
++++

Ko‘rsatkich (pointer) yordamida nima qilinadi?

====

Xotira manzillari bilan ishlanadi

====

Faqat qiymatlarni saqlaydi

====

Ekranga chiqaradi

====

Matnlarni tahrirlaydi  
++++

Quyidagilardan qaysi biri massiv elementiga murojaat misoli hisoblanadi?

====

A[3]

====

A.3

====

A->3

====

A(3)  
++++

Vektorlar bilan ishlashda afzalliklardan biri -…

====

Hajmi dinamik o‘zgaradi

====

Doim xotirada kam joy egallaydi

====

Faqat o‘qish rejimi mavjud

====

Massivdan sekin ishlaydi  
++++

Record tuzilmasi dasturlashda nima uchun kerak?

====

Murakkab obyektlarni ifodalash uchun

====

Faqat massivlarni almashtirish uchun

====

Fayllarni ajratish uchun

====

Tashqi qurilmalarni ulash uchun  
++++

To‘plamlarda quyidagilardan qaysi amal mavjud?

====

Birlashtirish (union)

====

Index orqali o‘chirish

====

Ko‘rsatkich qo‘shish

====

Tartiblash  
++++

Ko‘rsatkich yordamida qaysi turdagi ma’lumotlarga murojaat qilinadi?

====

Xotirada saqlangan istalgan obyektga

====

Faqat butun sonlarga

====

Faqat massivlarga

====

Matnli fayllargagina  
++++

Rekursiya bu -…

====

Funktsiyaning o‘zini o‘zi chaqirishi

====

Faqat massiv ustida ishlov

====

Input funksiyani ishlatish

====

Dasturga komanda berish  
++++

Rekursiv funksiyaning asosiy elementi -…

====

Bazaviy holat (base case)

====

For sikli

====

Pointer

====

Massiv uzunligi  
++++

Rekursiv algoritmlar qachon to‘xtaydi?

====

Bazaviy holatga yetganda

====

Elementlar tugaganda

====

Funksiya xato berganda

====

Doim davom etadi  
++++

Quyidagi funksiyalardan qaysi biri rekursiv emas?

====

Iterativ factorial

====

Faktorial(n): return n \* faktorial(n-1)

====

Fibonachi(n): return fib(n-1)+fib(n-2)

====

Funksiya: agar(n==0) return 1; else return funksiya(n-1)  
++++

Rekursiv funksiya ishlashida qanday xotira tuzilmasi ishlatiladi?

====

Stack

====

Queue

====

Heap

====

Array  
++++

Faktorial(5) funksiyasi rekursiv hisoblanganda nechta chaqiriq bo‘ladi?

====

6

====

5

====

4

====

1  
++++

Rekursiv yondashuvning iterativ yondashuvga nisbatan kamchiligi nima?

====

Xotira sarfi yuqoriroq

====

Aniqligi past

====

Kutilmagan natija beradi

====

Foydalanish mumkin emas  
++++

Rekursiv algoritmning afzalligi -…

====

Murakkab masalalarni soddalashtiradi

====

Doim tez ishlaydi

====

Faqat matnlar bilan ishlaydi

====

Faqat C++ da yoziladi  
++++

Quyidagilardan qaysi biri rekursiv misol hisoblanadi?

====

Faktorial(n): agar n==0 return 1; aks holda return n\*faktorial(n-1)

====

Foydalanuvchi ma’lumot kiritadi

====

For i=1 to n: print(i)

====

Massivga element qo‘shish  
++++

Fibonachchi sonlar qatori rekursiv usulda qanday ifodalanadi?

====

F(n)=F(n-1)+F(n-2)

====

F(n)=n+1

====

F(n)=n\*2

====

F(n)=n-1  
++++

Chiziqli qidiruv algoritmi qanday ishlaydi?

====

Har bir elementni ketma-ket tekshiradi

====

Elementlar o‘rtasini ikkiga bo‘ladi

====

Faqat eng so‘nggi elementni tekshiradi

====

Faqat tartiblangan massivda ishlaydi  
++++

Binar qidiruv algoritmi qaysi shartda ishlaydi?

====

Tartiblangan massivda

====

Faqat satrli massivda

====

Faqat 10 ta elementdan kam bo‘lsa

====

Faqat raqamli qiymatlar bo‘lsa  
++++

Binar qidiruv qanday ishlaydi?

====

Massivning o‘rtasini tekshirib, chap yoki o‘ng yarmiga o‘tadi

====

Har bir elementni boshidan oxirigacha tekshiradi

====

Faqat birinchi va oxirgi elementni solishtiradi

====

Tasodifiy indeks tanlaydi  
++++

Qidiruv algoritmining asosiy vazifasi nima?

====

Berilgan qiymatni toppish

====

Massivni tartiblash

====

Xatoliklarni tuzatish

====

Grafni chizish  
++++

Chiziqli qidiruvning eng yomon holatdagi ishlash murakkabligi qanday?

====

O(n)

====

O(log n)

====

O(n²)

====

O(1)  
++++

Binar qidiruvda element topilmasa nima bo‘ladi?

====

Qidiruv davomida massiv qismlarga bo‘linib, oxir-oqibat topilmasligi aniqlanadi

====

Element oxirgi indeksda topiladi

====

Dastur xatoga uchraydi

====

Har doim element topiladi  
++++

Qaysi qidiruv algoritmi kichik massivlar uchun samaraliroq bo‘ladi?

====

Chiziqli qidiruv

====

Binar qidiruv

====

Interpolatsion qidiruv

====

Xeshli qidiruv  
++++

Qaysi holatda binar qidiruv ishlamaydi?

====

Massiv tartiblanmagan bo‘lsa

====

Massiv uzun bo‘lsa

====

Qidirilayotgan element mavjud bo‘lsa

====

Elementlar bir xil bo‘lsa  
++++

Qidiruv algoritmlarini optimallashtirish nimani anglatadi?

====

Tezroq va kamroq resurs bilan izlash

====

Elementni o‘chirish

====

Chop etish

====

Yozishni to‘xtatish  
++++

Quyidagilardan qaysi biri qidiruv algoritmi emas?

====

Bubble sort

====

Linear search

====

Binary search

====

Hash search  
++++

Xesh funksiyasining asosiy vazifasi nima?

====

Ma’lumotni o‘zgarmas uzunlikdagi kodga aylantirish

====

Massivni tartiblash

====

Elementlarni tekshirish

====

Xatoliklarni aniqlash  
++++

Xesh jadvali qanday ishlaydi?

====

Xesh funksiyasini ishlatib, ma’lumotni indeksga joylashtiradi

====

Elementlarni tasodifiy tartibda saqlaydi

====

Har bir elementni alohida jadvalga saqlaydi

====

Fayllarni tasvirlash uchun ishlatiladi  
++++

Xesh funksiyasining kolliziyasi nima?

====

Bir nechta ma’lumotning bir xil xesh qiymatiga ega bo‘lishi

====

Xesh jadvalining uzunligi o‘zgarganda yuzaga keladi

====

Xesh funksiyasi faqat raqamlar bilan ishlaganda yuzaga keladi

====

Xesh funktsiyasini optimallashtirishni anglatadi  
++++

Xesh jadvalini amalga oshirishda qanday usullar qo‘llanadi?

====

Zanjirli va ochiq manzil usullari

====

Paskal va C++ dasturlash tillari

====

Graflar va daraxtlar

====

Binar qidiruv va chiziqli qidiruv  
++++

Xesh funksiyasining asosiy afzalligi qanday?

====

Tezkor ma’lumotlarga murojaat qilish imkonini beradi

====

Ma’lumotlarni tartiblashni osonlashtiradi

====

Elementlar bir xil bo‘lishi kerak

====

Faqat raqamli ma’lumotlar bilan ishlaydi  
++++

Saralash algoritmlarining asosiy maqsadi nima?

====

Ma’lumotlarni tartib bilan joylashtirish

====

Massivni kichiklashtirish

====

Ma’lumotlarni takrorlash

====

Elementlarni tasodifiy joylashtirish  
++++

Qaysi algoritm saralashning qat’iy usuliga kiradi?

====

Bubble sort

====

Quick sort

====

Merge sort

====

Heap sort  
++++

Bubble sort algoritmining eng yomon holatdagi ishlash murakkabligi qanday?

====

O(n²)

====

O(n log n)

====

O(n)

====

O(log n)  
++++

Quick sort algoritmi qanday ishlaydi?

====

Pivot elementni tanlab, massivni ikkiga bo‘lib, har bir qismni alohida saralaydi

====

Elementlarni tekshirib, o‘zgaradi

====

Har bir elementni boshidan oxirigacha solishtiradi

====

Har bir elementni o‘zi bilan almashtiradi  
++++

Merge sort algoritmining asosiy afzalligi nima?

====

Massivni bo‘lib, alohida saralaydi va keyin birlashtiradi

====

Tez ishlash va kam resurs sarfi

====

Elementlar bir xil bo‘lishi kerak

====

Massivni tashqi qurilmaga o‘tkazadi  
++++

Saralashning yaxshilangan usullari qaysi algoritmlarni o‘z ichiga oladi?

====

Quick sort, Merge sort

====

Bubble sort, Selection sort

====

Insertion sort

====

Counting sort  
++++

Insertion sort algoritmi qanday ishlaydi?

====

Elementni tartibga keltirib, unga o‘xshash elementlarni joylashtiradi

====

Elementlarni alohida solishtiradi va joyini almashtiradi

====

Massivni ikki bo‘lakka ajratadi

====

Elementlarni faqat oxirida qo‘shadi  
++++

Quyidagilardan qaysi biri saralash algoritmining samaradorligini oshirish usuli?

====

Pivotni to‘g‘ri tanlash

====

Elementlarni faqat oxirida saqlash

====

Chiziqli qidiruvni qo‘llash

====

Massivni bitta bo‘lakka ajratish  
++++

Quyidagi algoritmlardan qaysi biri saralashning yaxshilangan usuliga kiradi?

====

Merge sort

====

Bubble sort

====

Selection sort

====

Insertion sort  
++++

Saralashning qat’iy usullari qanday tavsiflanadi?

====

Resurslarni ko‘proq sarflaydi, ammo aniq natija beradi

====

Samaradorligi past

====

Elementlarni chaqirish orqali amalga oshiriladi

====

Ko‘p vaqt oladi, ammo samarali  
++++

Chiziqli bog‘langan ro‘yxat nima?

====

Elementlar bir-biriga bog‘langan va ma’lumotlar ketma-ket joylashgan ro‘yxat

====

Har bir element boshqa elementlar bilan aloqasiz joylashgan

====

Elementlar bir-biridan mustaqil

====

Elementlar faqat tartiblangan holda joylashgan  
++++

Chiziqli bog‘langan ro‘yxatlarda har bir element qanday tuzilgan?

====

Ma’lumot va keyingi elementga bo‘lgan havoladan iborat

====

Faoliyat va qiymatlardan iborat

====

Indeks va qiymatdan iborat

====

Bir nechta ma’lumotlar to‘plamidan iborat  
++++

Chiziqli bog‘langan ro‘yxatdagi "next" ko‘rsatkichi nima vazifani bajaradi?

====

Keyingi elementga bog‘laydi

====

Oldingi elementni ko‘rsatadi

====

Elementni saqlaydi

====

Elementning qiymatini o‘zgartiradi  
++++

Ikki bog‘lamli ro‘yxat qanday ishlaydi?

====

Har bir element oldingi va keyingi elementga havola qiladi

====

Faqat oldingi elementga havola qiladi

====

Faqat keyingi elementga havola qiladi

====

Elementlar faqat ma’lumotdan iborat  
++++

Chiziqli bog‘langan ro‘yxatdagi "head" nima?

====

Ro‘yxatning boshlanish nuqtasi

====

Ro‘yxatning oxirgi elementini bildiradi

====

Ro‘yxatdagi eng katta element

====

Ro‘yxatni qidirish uchun ishlatiladigan element  
++++

Quyidagi tuzilmalardan qaysi biri chiziqli bog‘langan ro‘yxatni tashkil etadi?

====

Element -> next -> Element -> next

====

Element -> prev -> Element -> next

====

Element -> prev -> Element -> prev

====

Element -> next -> prev -> Element  
++++

Ikki bog‘lamli ro‘yxatning afzalligi nima?

====

Har ikkala yo‘nalishda harakat qilish imkoniyatini beradi

====

Tartiblangan holda saqlaydi

====

Faqat oxirgi elementni ko‘rsatadi

====

Xotira sarfi kamroq  
++++

Chiziqli bog‘langan ro‘yxatni qanday o‘zgartirish mumkin?

====

Element qo‘shish yoki o‘chirish orqali

====

Elementni boshqarish orqali

====

Elementni faqat bir joyda saqlash orqali

====

Foydalanuvchi tomonidan o‘zgartirish mumkin emas  
++++

Chiziqli bog‘langan ro‘yxatni o‘zgartirishda nima muhim?

====

Havolalar va pointerlarni to‘g‘ri o‘zgartirish

====

Elementlarni tartibga solish

====

Faoliyatlarni bajarish

====

Elementlarni faqat oxirida qo‘shish  
++++

Quyidagi amallardan qaysi biri ikki bog‘lamli ro‘yxatda bajarilishi mumkin?

====

Oldingi va keyingi elementlarga o‘tish

====

Faqat oxirgi elementga o‘tish

====

Faoliyatni o‘zgartirish

====

Elementni faqat bitta yo‘nalishda ko‘rsatish  
++++

Stek nima?

====

LIFO (Last In First Out) tizimi bo‘yicha ishlovchi ma’lumotlar tuzilmasi

====

FIFO (First In First Out) tizimi bo‘yicha ishlovchi ma’lumotlar tuzilmasi

====

Keyingi element faqat boshida qo‘shiladigan tuzilma

====

Elementlar tasodifiy tartibda qo‘shiladi va o‘chiriladi  
++++

Navbat nima?

====

FIFO (First In First Out) tizimi bo‘yicha ishlovchi ma’lumotlar tuzilmasi

====

LIFO (Last In First Out) tizimi bo‘yicha ishlovchi ma’lumotlar tuzilmasi

====

Elementlar faqat oxirida qo‘shiladi va boshida o‘chiriladi

====

Tuzilma ichidagi barcha elementlar bir xil qiymatga ega  
++++

Dek (Double-ended queue) nima?

====

Elementlar ikkala uchidan qo‘shilishi va o‘chirilishi mumkin bo‘lgan ma’lumotlar tuzilmasi

====

Elementlar faqat bir uchidan qo‘shilishi mumkin

====

Faqat navbatga o‘xshash ishlaydi

====

Barcha elementlar tartiblangan  
++++

Stekni massiv yordamida qanday tasvirlash mumkin?

====

Tartiblangan massivda oxirgi elementni boshiga qo‘shish va o‘chirish

====

Har bir elementni alohida saqlash

====

Elementlarni faqat oxirida saqlash

====

Boshqa massiv bilan bog‘lash  
++++

Stekni chiziqli bog‘langan ro‘yxat yordamida qanday tasvirlash mumkin?

====

Elementlarni bog‘langan ro‘yxatda faqat boshida qo‘shish va o‘chirish

====

Elementlarni o‘rtasida qo‘shish

====

Elementlarni faqat oxirida qo‘shish

====

Elementlar faqat tasodifiy tartibda qo‘shiladi  
++++

Navbatni massiv yordamida qanday tasvirlash mumkin?

====

Elementlarni tartib bo‘yicha qo‘shish va faqat boshidan o‘chirish

====

Elementlarni faqat boshidan qo‘shish

====

Elementlarni faqat oxirida qo‘shish

====

Har bir elementni tasodifiy joylashtirish  
++++

Ustuvor navbat (Priority Queue) nima?

====

Har bir elementga ustuvorlik berilgan navbat

====

Elementlar faqat tartiblangan holda saqlanadi

====

Elementlar faqat tartiblanmagan holda saqlanadi

====

Elementlar faqat boshida qo‘shiladi  
++++

Quyidagilardan qaysi biri ustuvor navbatni tasvirlaydi?

====

Elementlar ustuvorlik darajasiga qarab joylashtiriladi

====

Elementlar faqat FIFO tizimi bo‘yicha qo‘shiladi

====

Elementlar tasodifiy qo‘shiladi

====

Har bir element o‘zining tartib raqami bo‘yicha qo‘shiladi  
++++

Lug‘atni qanday tuzish mumkin?

====

Kalitlar va qiymatlar juftliklari ko‘rinishida

====

Foydalanuvchi tomonidan kiritilgan elementlar

====

Har bir elementni alohida saqlash

====

Elementlarni tartiblangan holda saqlash  
++++

Lug‘atlarni amalga oshirishda qaysi usul keng tarqalgan?

====

Xesh jadvali yordamida

====

Chiziqli ro‘yxat yordamida

====

Bog‘langan ro‘yxat yordamida

====

Fayl tizimi yordamida  
++++

Daraxtsimon ma’lumotlar tuzilmasi nima?

====

Elementlar ierarxik tuzilishda tashkil etilgan ma’lumotlar tuzilmasi

====

Elementlar ketma-ket joylashgan ma’lumotlar tuzilmasi

====

Elementlar tasodifiy joylashgan ma’lumotlar tuzilmasi

====

Elementlar faqat tartiblangan holda joylashadi  
++++

Daraxtsimon ma’lumotlar tuzilmasining qanday asosiy xususiyati mavjud?

====

Har bir tugun faqat bitta ota tuguniga ega

====

Tugunlar tasodifiy joylashadi

====

Har bir tugun faqat bitta farzand tuguniga ega

====

Tugunlar faqat o‘zaro bog‘lanmagan  
++++

Daraxtning ildizi (root) nima?

====

Daraxtning boshlang‘ich tuguni

====

Daraxtning oxirgi tuguni

====

Barcha tugunlarni o‘z ichiga olgan joy

====

Tugunlar o‘rtasidagi bog‘lanishni anglatadi  
++++

Daraxt klassifikatsiyasida qaysi turlar mavjud?

====

Binar daraxt, AVL daraxti, B-tizim daraxti

====

Faoliyatli daraxtlar, statik daraxtlar

====

Dinamik daraxtlar, statik daraxtlar

====

Maxsus daraxtlar, oddiy daraxtlar  
++++

Binar daraxt nima?

====

Har bir tugun faqat ikkita farzand tuguniga ega bo‘lishi mumkin

====

Har bir tugun bir nechta farzand tuguniga ega bo‘lishi mumkin

====

Har bir tugun faqat bitta ota tuguniga ega

====

Daraxtda faqat ikkita tugun mavjud  
++++

Daraxt ko‘ruvi (Traversal) nima?

====

Daraxtdagi barcha tugunlarni tartib bilan tekshirish jarayoni

====

Daraxtni faqat bir martalik o‘rganish

====

Daraxtni faqat ildizdan boshlash

====

Tugunlarni faqat ma’lum bir tartibda tekshirish  
++++

Quyidagi daraxt ko‘ruvlaridan qaysi biri pre-order ko‘ruvi hisoblanadi?

====

Ildizni, keyin chap farzandni, keyin o‘ng farzandni tekshirish

====

Chap farzandni, ildizni, o‘ng farzandni tekshirish

====

Chap farzandni, o‘ng farzandni, ildizni tekshirish

====

Faoliyatni o‘rganish uchun daraxtni faqat oxirigacha ko‘rish  
++++

Binar daraxtni qanday tasvirlash mumkin?

====

Ildizni boshlang‘ich nuqta qilib, har bir tugunni farzandlari bilan bog‘lash

====

Elementlarni faqat o‘rtada saqlash

====

Elementlar faqat boshqacha tarzda joylashadi

====

Har bir tugun o‘zgaruvchan  
++++

AVL daraxtining xususiyati nima?

====

Har bir tugun uchun balansi ikki bo‘lgan farqni saqlaydi

====

Farzandlar soni cheksiz

====

Daraxtni faqat bir nechta qismga ajratish mumkin

====

Tugunlar faqat bittadan iborat  
++++

B-tizim daraxti nima?

====

Birinchi darajadagi daraxt, unda ko‘p farzandlar bo‘lishi mumkin

====

Daraxtning faqat bir nechta darajasi mavjud

====

Har bir tugun faqat bir farzandga ega

====

Daraxtni faqat 2 qismga bo‘lish mumkin  
++++

Binar qidiruv daraxti qanday ishlaydi?

====

Har bir tugun o‘zining chap farzandidan kichik, o‘ng farzandidan katta qiymatga ega

====

Har bir tugun o‘zining chap farzandidan katta, o‘ng farzandidan kichik qiymatga ega

====

Binar qidiruv daraxti faqat chap farzandga ega

====

Tugunlar faqat tasodifiy qiymatlarga ega  
++++

Binar qidiruv daraxtiga yangi element qo‘shish jarayonida nima qilish kerak?

====

Yangi elementni ildizdan boshlang‘ich nuqtada joylashtirib, qiymatga qarab chap yoki o‘ngga qo‘shish

====

Har doim yangi elementni faqat ildizga qo‘shish

====

Yangi elementni daraxtning oxiriga qo‘shish

====

Yangi elementni faqat o‘ng farzandga qo‘shish  
++++

Binar qidiruv daraxtidan elementni o‘chirishda qanday holatlar mavjud?

====

Elementning hech qanday farzandi yo‘q, faqat bitta farzandi bor yoki ikkita farzandi bor

====

Elementni faqat chap farzand orqali o‘chirish mumkin

====

Elementni faqat o‘ng farzand orqali o‘chirish mumkin

====

Elementni faqat ildizdan o‘chirish mumkin  
++++

Muvozanatlangan binar daraxt nima?

====

Daraxtning har bir tuguni uchun chap va o‘ng farzandlar o‘rtasidagi balansi ma’lum bir chegarada bo‘ladi

====

Binar daraxt bo‘lib, har bir tugun o‘ng farzandini faqat chap farzandidan keyin qo‘shadi

====

Binar daraxt bo‘lib, har bir tugun o‘zining chap farzandini faqat o‘ng farzandidan keyin qo‘shadi

====

Muvozanatlangan binar daraxt faqat ikkita tugundan iborat bo‘ladi  
++++

AVL daraxtining xususiyatlari qanday?

====

Har bir tugunning chap va o‘ng farzandlarining balansi -1, 0 yoki 1 bo‘lishi kerak

====

Har bir tugun faqat bitta farzandga ega bo‘ladi

====

Har bir tugun o‘zining chap farzandi orqali ajratiladi

====

AVL daraxti faqat o‘ng farzandlar bilan ishlaydi  
++++

Binar qidiruv daraxtida element qidirish qanday amalga oshiriladi?

====

Ildizdan boshlanib, kerakli qiymatni topish uchun chap yoki o‘ngga qarab harakat qilinadi

====

Har doim faqat chap farzandda qidiriladi

====

Har doim faqat o‘ng farzandda qidiriladi

====

Foydalanuvchi tomonidan kiritilgan indeks asosida qidiriladi  
++++

AVL daraxtida muvozanatni tiklash uchun qaysi amallarni bajarish kerak?

====

Agar farq 2 bo‘lsa, rotatsiya qilish

====

Farqni faqat 1 ga qisqartirish

====

Farqni 3 ga oshirish

====

Faoliyatni faqat chap farzandda bajarish  
++++

Binar qidiruv daraxtida maksimal elementni qanday topish mumkin?

====

Daraxtning o‘ng farzandlarini bosqichma-bosqich tekshirib, eng oxirgi tugunni toppish

====

Daraxtning chap farzandini bosqichma-bosqich tekshirish

====

Daraxtning ildizini tekshirish

====

Tugunlarni tasodifiy tekshirish  
++++

Muvozanatlashning umumiy algoritmi nima?

====

Muvozanatlash jarayonida rotatsiya amalga oshiriladi

====

Faoliyatni har bir tugunda tekshirish

====

Tugunlarni faqat o‘ngga qo‘shish

====

Daraxtni faqat oxirida tekshirish  
++++

Binar qidiruv daraxtida o‘chirilgan tugunni qanday qayta tiklash mumkin?

====

O‘chirilgan tugunni o‘ng yoki chap farzandning qiymatlarini o‘zgartirish orqali tiklash

====

Tugunni faqat bir martalik o‘zgartirish orqali tiklash

====

Tugunni faqat yangi tugun bilan almashtirish

====

Tugunni faqat ildizdan o‘chirish orqali tiklash  
++++

Graflarni tasvirlashning eng keng tarqalgan usullari qanday?

====

Qo‘shma matritsa va munosabatlar matritsasi

====

Qo‘shnilik ro‘yxati va yoylar ro‘yxati

====

Har bir tugun uchun alohida jadval

====

Tugunlar va ularning farzandlari ro‘yxati  
++++

Graflar tasvirlashda qo‘shma matritsa nima?

====

Grafning barcha tugunlari o‘rtasidagi bog‘lanishlarni ko‘rsatadigan ikki o‘lchovli matritsa

====

Grafning faqat o‘ng tugunlarini ko‘rsatadigan matritsa

====

Grafdagi faqat bog‘lanmagan tugunlarni ko‘rsatuvchi matritsa

====

Har bir tugun uchun alohida matritsa  
++++

Graflarni munosabatlar matritsasi yordamida tasvirlashda qanday xususiyatlar mavjud?

====

Tugunlar o‘rtasidagi bog‘lanishni 1 yoki 0 bilan ko‘rsatadi

====

Faoliyatni har bir tugunda tekshirish kerak

====

Tugunlar faqat alohida ko‘rsatiladi

====

Har bir tugun uchun alohida list kerak bo‘ladi  
++++

Qo‘shnilik ro‘yxati nima?

====

Tugunlarning o‘zaro bog‘lanishlarini saqlovchi ro‘yxat

====

Tugunlar o‘rtasidagi bog‘lanishlarning qo‘shma matritsasini ko‘rsatuvchi ro‘yxat

====

Har bir tugunning faqat o‘ng farzandini ko‘rsatuvchi ro‘yxat

====

Tugunlar o‘rtasidagi faqat bog‘lanmagan elementlarni ko‘rsatuvchi ro‘yxat  
++++

Graflarni yoylar ro‘yxati yordamida qanday tasvirlash mumkin?

====

Tugunlar va ularning orasidagi bog‘lanishlar juftliklar (yo‘ylar) sifatida ro‘yxatlanadi

====

Tugunlar va ular o‘rtasidagi bog‘lanishlar faqat bitta o‘lchamli ro‘yxatda tasvirlanadi

====

Tugunlar o‘rtasidagi bog‘lanishlar faqat alohida ko‘rsatiladi

====

Har bir tugun uchun maxsus jadval tuziladi  
++++

Graflarda ko‘ruv algoritmi nima?

====

Graflarning barcha tugunlarini tekshirish uchun ishlatiladigan algoritm

====

Faqatgina tugunlar orasidagi bog‘lanishlarni tekshirish

====

Grafdagi faqat faollashgan tugunlarni tekshirish

====

Tugunlar o‘rtasidagi bog‘lanishlarni o‘zgartirish  
++++

Graflarda eniga qarab qidiruv (BFS) algoritmi qanday ishlaydi?

====

Boshlang‘ich tugundan boshlanib, barcha qo‘shni tugunlar tekshiriladi

====

Tugunlar faqat chapdan o‘ngga qarab tekshiriladi

====

Faqat oxirgi tugunlar tekshiriladi

====

Tugunlar faqat teskari tartibda tekshiriladi  
++++

Graflarda tubiga qarab qidiruv (DFS) algoritmi qanday ishlaydi?

====

Boshlang‘ich tugundan boshlanib, chuqurroq qarab barcha tugunlar tekshiriladi

====

Tugunlar faqat o‘ngdan chapga qarab tekshiriladi

====

Tugunlar faqat alohida ro‘yxatda tekshiriladi

====

Tugunlar faqat chuqurlikdan qarab tekshiriladi  
++++

Graflarda eng qisqa yo‘lni aniqlash uchun qaysi algoritmlar ishlatiladi?

====

Deykstra, Floyd-Warshall va Bellman-Ford algoritmlari

====

Tugunlarni faqat bir martalik tekshirish

====

Chiziqli qidiruv usullari

====

Tugunlar orasidagi faqat to‘g‘ri yo‘llarni aniqlash  
++++

Graflarda eng qisqa yo‘lni aniqlash algoritmlarining samaradorligini qanday baholash mumkin?

====

Algoritmning vaqt murakkabligi va har bir bosqichda o‘zgartirishlar soni

====

Algoritmning faqat tugunlar soniga qarab samaradorligi

====

Graflarda faqat tugunlar o‘rtasidagi bog‘lanishni tekshirish

====

Faollik darajasiga qarab samaradorlikni baholash  
++++

Floyd-Warshall algoritmi nima?

====

Barcha juft tugunlar orasidagi eng qisqa yo‘lni aniqlash uchun ishlatiladigan algoritm

====

Faqat bir tugundan boshqa barcha tugunlarga eng qisqa yo‘llarni topadi

====

Har bir tugun uchun faqat bir yo‘lni topadi

====

Binar qidiruv usuli yordamida eng qisqa yo‘lni aniqlaydi  
++++

Floyd-Warshall algoritmining asosiy xususiyati nima?

====

Har bir tugun uchun barcha boshqa tugunlarga bo‘lgan eng qisqa yo‘llarni hisoblaydi

====

Faqat boshlang‘ich tugundan oxirgi tugunga bo‘lgan eng qisqa yo‘lni hisoblaydi

====

Tugunlar o‘rtasidagi faqat to‘g‘ri yo‘llarni aniqlaydi

====

Foydalanuvchi tomonidan berilgan maxsus yo‘llar orqali eng qisqa yo‘llarni hisoblaydi  
++++

Deykstra algoritmi qanday ishlaydi?

====

Boshlang‘ich tugundan barcha boshqa tugunlarga eng qisqa yo‘llarni topish uchun ishlatiladi

====

Faqat boshlang‘ich tugundan oxirgi tugunga eng qisqa yo‘lni topadi

====

Barcha tugunlar o‘rtasidagi eng qisqa yo‘llarni hisoblaydi

====

Tugunlar o‘rtasidagi bog‘lanishni teskari tartibda tekshiradi  
++++

Ford-Bellman algoritmi nima uchun ishlatiladi?

====

Manfiy og‘irliklarga ega graflarda eng qisqa yo‘lni topish uchun ishlatiladi

====

Faoliyatni faqat manfiy tugunlar o‘rtasida bajaradi

====

Faqat bitta tugunlar o‘rtasidagi eng qisqa yo‘lni topadi

====

Foydalanuvchi tomonidan kiritilgan maxsus yo‘llar uchun ishlatiladi  
++++

Deykstra algoritmining cheklovlari qanday?

====

Manfiy og‘irliklarga ega bo‘lgan graflarda ishlamaydi

====

Faoliyatni faqat manfiy tugunlar o‘rtasida bajaradi

====

Tugunlar o‘rtasidagi barcha bog‘lanishlarni tekshirish kerak

====

Faoliyatni faqat o‘ng tugunlar o‘rtasida bajaradi  
++++

Floyd-Warshall algoritmining vaqt murakkabligi qanday?

====

O(n³), bunda n - tugunlar soni

====

O(n²)

====

O(n)

====

O(nlogn)  
++++

Deykstra algoritmida eng qisqa yo‘lni aniqlashda qanday yondashuv ishlatiladi?

====

Greedy (ochko‘rlik) yondashuvi

====

Dinamik dasturlash yondashuvi

====

Rekursiv yondashuv

====

Binar qidiruv yondashuvi  
++++

Ford-Bellman algoritmining ishlash printsipi qanday?

====

Tugunlar orasidagi eng qisqa yo‘llarni kiritilgan og‘irliklarga qarab hisoblaydi

====

Foydalanuvchi tomonidan kiritilgan yo‘llar bo‘yicha faqat eng qisqa yo‘lni topadi

====

Har bir tugun uchun faqat o‘ng farzandni tekshiradi

====

Har doim maksimal og‘irlikdagi yo‘llarni tanlaydi  
++++

Deykstra algoritmi qaysi holatlarda ishlatilmaydi?

====

Manfiy og‘irliklarga ega graflarda ishlamaydi

====

Faoliyatni faqat to‘g‘ri yo‘llarda amalga oshiradi

====

Faoliyatni faqat bitta tugun orqali amalga oshiradi

====

Foydalanuvchi tomonidan kiritilgan maxsus yo‘llarga qarab ishlaydi  
++++

Floyd-Warshall algoritmining afzalligi nima?

====

Barcha juft tugunlar orasidagi eng qisqa yo‘lni topish imkoniyatini beradi

====

Faoliyatni faqat bitta tugunda amalga oshiradi

====

Foydalanuvchi tomonidan kiritilgan maxsus yo‘llarga asoslanadi

====

Faoliyatni faqat o‘ng farzandlar o‘rtasida bajaradi  
++++