# DARAXTSIMON TUZILMALAR

Ishdan maqsad: Talabalar daraxtsimon tuzilmalar, binar daraxtlarni e’lon qilish, uning ustida amallar bajarish algoritmlarini tadqiq qilishlari va o‟rganishlari kerak, bu algoritmlarning dasturiy realizatsiyasini amalga oshirish ko‟nikmasiga ega bo‟lishlari kerak.

Qo‟yilgan masala: Har bir talaba topshiriq varianti olib, undagi masalaning qo‟yilishiga mos binar daraxtlarni tadqiq qilishga oid dasturni ishlab chiqishlari kerak.

Ish tartibi:

* + - Tajriba ishi nazariy Ma’lumotlarini o‟rganish;
    - Berilgan topshiriqning algoritmini ishlab chiqish;
    - C++ dasturlash muhitida dasturni yaratish;
    - Natijalarni tekshirish;
    - Hisobotni tayyorlash va topshirish.

# Daraxt ko‟rinishidagi Ma’lumotlar tuzilmasi haqida umumiy tushunchalar.

Uzellar (elementlar) va ularning munosabatlaridan iborat elementlar to‟plamining ierarxik tuzilmasiga daraxtsimon Ma’lumotlar tuzilmasi deyiladi.

**Daraxt** – bu shunday chiziqsiz bog‟langan Ma’lumotlar tuzilmasiki, u quyidagi belgilari bilan tavsiflanadi:

* daraxtda shunday bitta element borki, unga boshqa elementlardan murojaat yo‟q. Bu element daraxt ildizi deyiladi;
* daraxtda ixtiyoriy element chekli sondagi ko‟rsatkichlar yordamida boshqa tugunlarga murojaat qilishi mumkin;
* daraxtning har bir elementi faqatgina o‟zidan oldingi kelgan bitta element bilan bog‟langan.

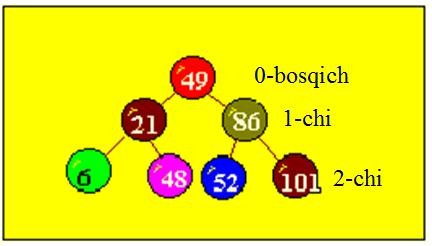
# Binar daraxtlarni qurish

Binar daraxtda har bir tugun-elementdan ko‟pi bilan 2 ta shox chiqadi. Daraxtlarni xotirada tasvirlashda uning ildizini ko‟rsatuvchi ko‟rsatkich berilishi kerak. Daraxtlarni kompyuter xotirasida tasvirlanishiga ko‟ra har bir element (binar daraxt tuguni) to‟rtta maydonga ega yozuv shaklida bo‟ladi, ya’ni kalit maydon, informatsion maydon, ushbu elementni o‟ngida va chapida joylashgan elementlarning xotiradagi adreslari saqlanadigan maydonlar.

Shuni esda tutish lozimki, daraxt hosil qilinayotganda, otaga nisbatan chap tomondagi o‟g‟il qiymati kichik kalitga, o‟ng tomondagi o‟g‟il esa katta qiymatli kalitga ega bo‟ladi. Har safar daraxtga yangi element kelib qo‟shilayotganda u avvalambor daraxt ildizi bilan solishtiriladi. Agar element ildiz kalit qiymatidan kichik bo‟lsa, uning chap shoxiga, aks holda o‟ng shoxiga o‟tiladi. Agar o‟tib ketilgan shoxda tugun mavjud bo‟lsa, ushbu tugun bilan ham solishtirish amalga oshiriladi, aks holda, ya’ni u shoxda tugun mavjud bo‟lmasa, bu element shu tugunga joylashtiriladi.

Masalan, daraxt tugunlari quyidagi qiymatlarga ega 6, 21, 48, 49, 52, 86, 101.

U holda binar daraxt ko‟rinishi quyidagi 4.1-rasmdagidek bo‟ladi:



4.1-rasm. Binar daraxt ko‟rinishi

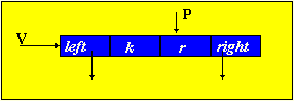
Natijada, o‟ng va chap qism daraxtlari bir xil bosqichli tartiblangan binar daraxt hosil qildik. Agar daraxtning o‟ng va chap qism daraxtlari bosqichlarining farqi birdan kichik bo‟lsa, bunday daraxt ideal muvozanatlangan daraxt deyiladi. Yuqorida hosil qilgan binar daraxtimiz ideal muvozanatlangan daraxtga misol

bo‟ladi. Daraxtni muvozanatlash algoritmini sal keyinroq ko‟rib chiqamiz. Undan oldin binar daraxtni yaratish algoritmini o‟rganamiz.

* + 1. **Algoritm**

**Binar daraxt yaratish funksiyasi**

Binar daraxtni hosil qilish uchun kompyuter xotirasida elementlar quyidagi 4.2-rasmdagidek toifada bo‟lishi lozim.



4.2-rasm. Binar daraxt elementining tuzilishi

p – yangi element ko‟rsatkichi

next, last – ishchi ko‟rsatkichlar, ya’ni joriy elementdan keyingi va oldingi elementlar ko‟rsatkichlari

r=rec – element haqidagi birorta Ma’lumot yoziladigan maydon k=key – elementning unikal kalit maydoni

left=NULL – joriy elementning chap tomonida joylashgan element adresi right=NULL – joriy elementning o‟ng tomonida joylashgan element adresi.

Dastlab yangi element hosil qilinayotganda bu ikkala maydonning qiymati 0 ga teng bo‟ladi.

tree – daraxt ildizi ko‟rsatkichi n – daraxtdagi elementlar soni

Boshida birinchi kalit qiymat va yozuv maydoni Ma’lumotlari kiritiladi, element hosil qilinadi va u daraxt ildiziga joylashadi, ya’ni tree ga o‟zlashtiriladi. Har bir hosil qilingan yangi elementning left va right maydonlari qiymati 0 ga tenglashtiriladi. Chunki bu element daraxtga terminal tugun sifatida joylashtiriladi, hali uning farzand tugunlari mavjud emas. Qolgan elementlar ham shu kabi hosil

qilinib, kerakli joyga joylashtiriladi. Ya’ni kalit qiymati ildiz kalit qiymatidan kichik bo‟lgan elementlar chap shoxga, katta elementlar o‟ng tomonga joylashtiriladi. Bunda agar yangi element birorta elementning u yoki bu tomoniga joylashishi kerak bo‟lsa, mos ravishda left yoki right maydonlarga yangi element adresi yozib qo‟yiladi.

Binar daraxtni hosil qilishda har bir element yuqorida ko‟rsatilgan toifada bo‟lishi kerak. Lekin hozir biz o‟zlashtirish osonroq va tushunarli bo‟lishi uchun key va rec maydonlarni bitta qilib info maydon deb ishlatamiz.

|  |  |  |
| --- | --- | --- |
| **left** | **info** | **right** |

4.3-rasm. Binar daraxt elementining tuzilishi

Ushbu toifada element hosil qilish uchun oldin bu toifani yaratib olishimiz kerak. Uni turli usullar bilan amalga oshirish mumkin. Masalan, ***node*** nomli yangi toifa yaratamiz:

*class* ***node****{*

*public:*

*int info; node \*left; node \*right;*

*};*

Endi yuqoridagi belgilashlarda keltirilgan ko‟rsatkichlarni shu toifada yaratib olamiz.

*node \*tree=NULL; node \*next=NULL;*

*int n,key; cout<<"n=";cin>>n;*

Nechta element (n) kiritilishini aniqlab oldik va endi har bir element qiymatini kiritib, binar daraxt tuzishni boshlaymiz.

*for(int i=0;i<n;i++){*

*node \*p=new node; node \*last=new node; cin>>key;*

*p->info=key;*

*p->left=NULL; p->right=NULL;*

*if(i==0){ tree=p; next=tree;sontinue;} next=tree;*

*while(1){ last=next;*

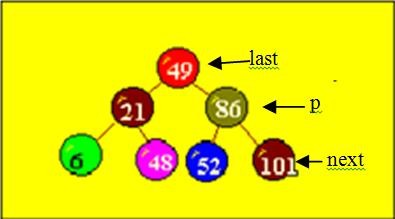
*if(p->info<next->info) next=next->left; else next=next->right; if(next==NULL) break;*

*}*

*if(p->info<last->info) last->left=p; else last->right=p;*

*}*

Bu yerda *p* hali aytganimizdek, kiritilgan kalitga mos hosil qilingan yangi element ko‟rsatkichi, *next* yangi element joylashishi kerak bo‟lgan joyga olib boradigan shox adresi ko‟rsatkichi, ya’ni u har doim *p* dan bitta qadam oldinda yuradi, *last* esa ko‟rilayotgan element kimning avlodi ekanligini bildiradi, ya’ni u har doim *p* dan bir qadam orqada yuradi (4.4-rasm).

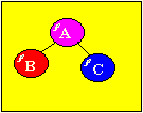


4.4-rasm. Binar daraxt elementlarini belgilash

Shunday qilib binar daraxtini ham yaratib oldik. Endigi masala uni ekranda tasvirlash kerak, ya’ni u ko‟rikdan o‟tkaziladi yoki vizuallashtirsa ham bo‟ladi.

* + 1. **Daraxt “ko‟rigi” funksiyalari**

4.5-rasmdagidek binar daraxt berilgan bo‟lsin:



4.5-rasm. 3 ta elemetdan iborat binar daraxt

Binar daraxtlari ko‟rigini uchta tamoyili mavjud. Ularni berilgan daraxt misolida ko‟rib chiqaylik:

1. Yuqoridan pastga ko‟rik (daraxt ildizini qism daraxtlarga nisbatan oldinroq ko‟rikdan o‟tkaziladi): A, B, C ;
2. Chapdan o‟ngga: B, A, C ;
3. Quyidan yuqoriga (ildiz qism daraxtlardan keyin ko‟riladi): B, C, A .

Daraxt ko‟rigi ko‟pincha ikkinchi usul bilan, ya’ni tugunlarga kirish ularning kalit qiymatlarini o‟sish tartibida amalga oshiriladi.

* + 1. **Daraxt ko**‟**rigining rekursiv funksiyalari**

1. *int pretrave(node \*tree){ if(tree!=NULL) {int a=0,b=0;*

*if(tree->left!=NULL) a=tree->left->info; if(tree->right!=NULL) b=tree->right->info;*

*cout<<tree->info<<" - chapida "<<a<<" - o’ngida "<<b<<" \n";*

*pretrave(tree->left); pretrave(tree->right);*

*}*

*return 0;*

*};*

1. *int intrave(node \*tree){ if(tree!=NULL) {*

*intrave(tree->left); cout<<tree->info; intrave(tree->right);*

*}*

*return 0;*

*};*

1. *int postrave(node \*tree){ if(tree!=NULL) {*

*postrave(tree->left); postrave(tree->right); cout<<tree->info;*

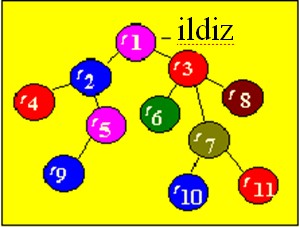
*}*

*return 0;*

*};*

Daraxtning har bir tuguni 4.6-rasmdagidek oraliq (2, 3, 5, 7 elementlar) yoki

terminal (daraxt “barg”i) (4, 9, 10, 11, 8, 6 elementlar) bo‟lishi mumkin.



4.6-rasm. Daraxtsimon tuzilma

1. Agar tugunning otasi yo‟q bo‟lsa, bu tugun ildiz hisoblanadi. Buni aniqlash uchun dastur kodini keltiramiz. Dasturda p izlanayotgan tugun.

*if(p==tree) cout<<”bu tugun ildiz ekan”; else cout<<”bu tugun ildiz emas”;*

1. Biz izlayotgan element daraxtda oraliq tugun ekanligini tekshirish uchun uning yoki o‟ng shoxi, yoki chap shoxi, yoki ikkalasiyam mavjudligini tekshirish kerak. Agar ikkala shoxi NULL dan farqli bo‟lsa, bu 2 ta farzandga ega oraliq tugun hisoblanadi, yoki ikkalasidan bittasi NULL ga teng bo‟lsa, bu tugun 1 ta farzandga ega oraliq tugun hisoblanadi. Berilgan p element daraxtning oraliq tugun ekanligini aniqlash dastur kodini keltiramiz.

*if(p!=tree){*

*if((p->left!=NULL)&&(p->right!=NULL)) cout<<”bu tugun 2 ta farzandga ega oraliq tugun”;*

*else if((p->left!=NULL)||(p->right!=NULL) cout<<”bu 1 ta farzandga ega oraliq tugun”;*

*} else cout<<”bu tugun oraliq tugun emas”;*

1. Biz izlayotgan tugun terminal tugunligini tekshirishni ko‟rib chiqamiz. Agar tugunning har ikkala shoxi NULL ga teng bo‟lsa, bu ***terminal tugun*** hisoblanadi. Dastur kodini keltiramiz.

*if((p->left==NULL)&&(p->right==NULL)) cout<<”bu tugun terminal tugun”;*

*else cout<<”bu terminal tugun emas”;*

**Ishni bajarishga namuna**

Topshiriq variantlariga o‟xshash bitta misolning algoritmi va to‟liq dasturini ko‟rib chiqaylik.

Misol: berilgan binar daraxtdan ko‟rsatilgan *key* kalitga mos tugunni o‟chirish dasturini tuzing.

**Topshiriqlar**

1. Binar daraxtning tugunlari sonini aniqlashning algoritmi va dasturini keltiring.
2. Binar daraxtda berilgan tugungacha bo‟lgan masofani aniqlashning algoritmi va dasturini keltiring.
3. Bo‟sh bo‟lmagan binar daraxtning eng katta va eng kichik kalitli tugunlarini aniqlashning algoritmi va dasturini keltiring.
4. T1 va T2 binar daraxtlar tengligini tekshiruvchi dastur tuzing. (Daraxtlar teng deyiladi, agar ikkala daraxt mos uchlarining yozuv va kalitlari o‟zaro teng bo‟lsa).
5. Binar daraxtni o‟ngdan chapga va chapdan o‟ngga ko‟rik o‟tkazish dasturi va algoritmini keltiring.
6. Daraxt tugunlari haqiqiy sonlar bo‟lsin. Yozuvi (a,b) oraliqqa tegishli bo‟lmagan daraxt tugunlarini o‟chiruvchi dastur tuzing.
7. Daraxt tugunlari haqiqiy sonlar bo‟lsin. Yozuvi (a,b) oraliqqa tegishli bo‟lgan daraxt tugunlarini o‟chiruvchi dastur tuzing.
8. Berilgan binar daraxtdan kalit qiymatlari kamayish tartibida joylashgan bir bog‟lamli ro‟yhat hosil qilish algoritmi va dasturini keltiring.
9. Bo‟sh bo‟lmagan binar daraxtning eng katta va eng kichik kalitli tugunlarini o‟rta arifmetigiga teng kalitli tugunni berilgan daraxtga qo‟yish algoritmi va dasturini keltiring.
10. Berilgan binar daraxtda kalit qiymati ildizning kalit qiymatiga eng yaqin bo‟lgan tugun kaliti va yozuvini chop etish algoritmi va dasturini keltiring.
11. Berilgan binar daraxtda kalit qiymati ildizning kalit qiymatiga eng uzoq bo‟lgan tugun kaliti va yozuvini chop etish algoritmi va dasturini keltiring.
12. Butun sonlardan iborat binar daraxtning toq qiymatli tugunlaridan yangi muvozanatlangan daraxt hosil qiling.
13. Berilgan binar daraxt muvozanatlanganmi yoki yo‟qligini tekshiring.
14. Berilgan muvozanatlangan binar daraxtdan qaysi tugunlar o‟chirilsa, uning muvozanatlanganligi buzilmasligini ko‟rsatish dasturini tuzing.
15. Berilgan ro‟yhat binar daraxt bo‟la oladimi, yo‟qmi, shuni aniqlash dasturini keltiring.

Har bir talaba 2 tadan misol ishlashi kerak!