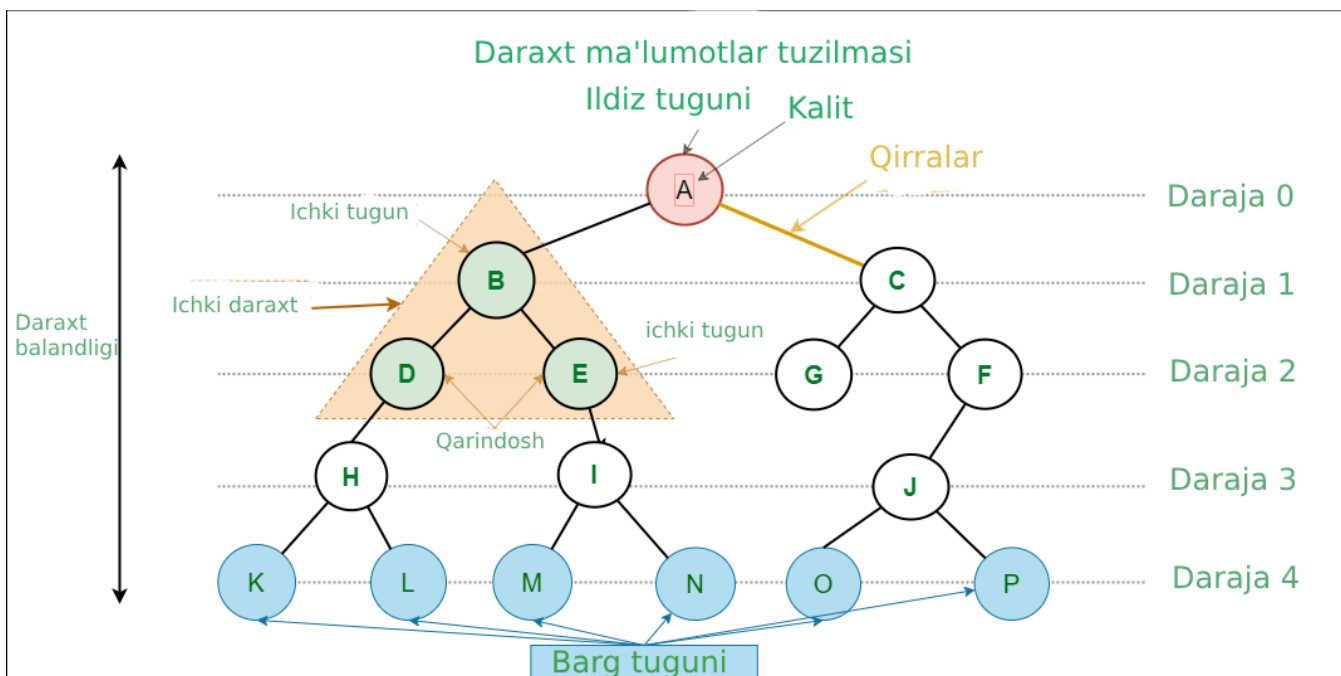


## **I BOB      Daraxtlar ma'lumotlar tuzilmasi**

### **1. Daraxtlar ma'lumotlar tuzilmasiga kirish**

Daraxtga o'xshash bo'lgan bu ma'lumotlar tuzilishi - bu ma'lumotlarni navigatsiya va qidirish uchun qulay tarzda taqdim etish va tartibga solish uchun ishlatiladigan ierarxik tuzilma. Bu qirralar bilan bog'langan tugunlar to'plami va tugunlar o'rtasida ierarxik munosabatlarga ega bo'lgan ma'lumotlar tuzilmasi. Daraxtning eng yuqori tugunlari ildiz tugunlari deb ataladi va uning ostidagi avlodlar tuguni deb ataladi. Har bir tugun bir nechta bola tugunlariga ega bo'lishi mumkin va bu bola tugunlari ham o'z bola tugunlariga ega bo'lib, rekursiv tuzilmani hosil qilishi mumkin.

Ushbu ma'lumotlar tuzilishi ma'lumotlarni yanada samarali foydalanish uchun kompyuterda tartibga solish va saqlashning maxsus usuli hisoblanadi. U qirralar bilan bog'langan Markaziy tugun, strukturaviy tugunlar va pastki tugunlardan iborat. Shuni ham aytishimiz mumkinki, daraxtga o'xshash ma'lumotlar strukturasi bir-biriga bog'langan ildizlar, novdalar va barglar mavjud.



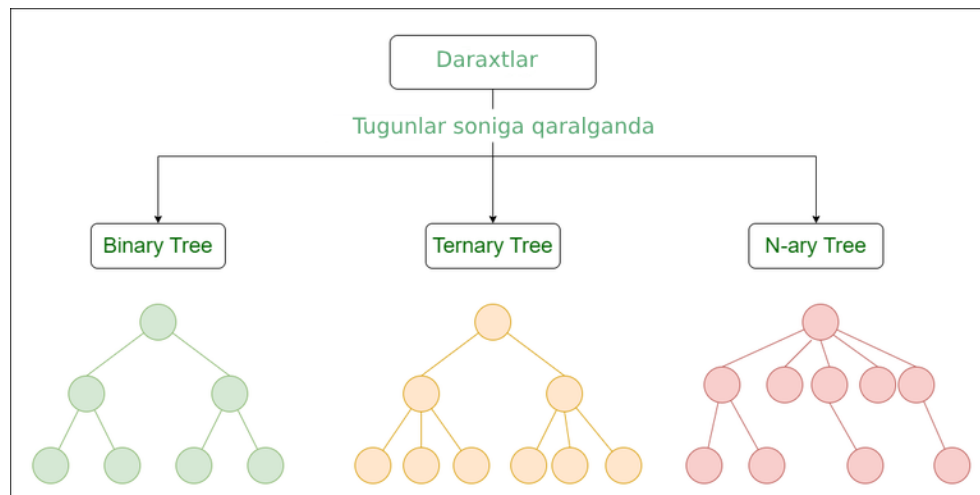
Rasm- 1 Daraxt ma'lumotlar tuzilmasi

### 1.1 Daraxtlarning xususiyatlari:

- *Qirralar soni*: ikkita tugun orasidagi bog'lanish sifatida aniqlash mumkin. Agar daraxtda  $N$  tugunlari bo'lsa, unda u  $(N-1)$  qirralarga ega bo'ladi.
- *Tugun chuqurligi*: tugun chuqurligi ildizdan shu tugungacha bo'lgan yo'l uzunligi sifatida aniqlanadi. Har bir qirra yo'lga 1 birlik uzunlik qo'shadi. Shunday qilib, uni daraxt ildizidan tugungacha bo'lgan yo'l qirralar soni sifatida ham aniqlash mumkin.
- *Tugun balandligi*: tugunning balandligini tugundan daraxtning oxirgi tugunigacha bo'lgan eng uzun yo'lning uzunligi sifatida aniqlash mumkin.
- *Tugun darajasi*: ushbu tugunga biriktirilgan pastki daraxtlarning umumiy soni tugun darajasi deb ataladi. Daraxt darajasi-bu daraxtdagi barcha tugunlar orasidagi tugunning maksimal darajasi.

### 1.2 Daraxt turlari:

- Ikkilik daraxt (Binary tree): ikkilik daraxtda har bir tugun unga bog'langan maksimal ikkita avlod (bola) ega bo'lishi mumkin. Ikkilik daraxtlarning ba'zi keng tarqalgan turlariga to'liq ikkilik daraxtlar, muvozanatli ikkilik daraxtlar va degeneratsiyalangan yoki patologik ikkilik daraxtlar kiradi.
- Uchlik daraxti (ternary tree): uchlik daraxti-bu ikkilik daraxtga o'xshash ma'lumotlar tuzilishi bo'lib, unda har bir tugunda uchta bola tugunidan ko'p bo'lmagan, odatda "chap", "o'rta" va "o'ng" deb farqlanadi.
- N-Ary daraxti yoki ko'p qirrali daraxt: ko'p qirrali daraxtlar tugunlar to'plamidir, bu erda har bir tugun yozuvlar va uning avlodlariga havolalar ro'yxatidan iborat ma'lumotlar tuzilishi (takroriy havolalar qabul qilinishi mumkin emas). Bog'langan ro'yxatdan farqli o'laroq, har bir tugun bir nechta tugunlarning manzillarini saqlaydi.



*Rasm- 2 Daraxt turlari*

### 1.3 Daraxt ma'lumotlari tuzilishining asosiy operatsiyasi:

- Create-ma'lumotlar tarkibida daraxt yaratish.
- Insert - ma'lumotlarni daraxtga kiritadi.
- Search-mavjud yoki yo'qligini tekshirish uchun daraxtdagi ba'zi ma'lumotlarni qidiradi.

### *Traversal:*

- Oldindan buyurtma berish - ma'lumotlar tuzilmasida oldindan buyurtma berish tartibida daraxt bo'ylab harakatlanishni amalga oshiradi.
- Tartibda aylanib o'tish - daraxt bo'ylab tartibda harakatlanishni amalga oshiradi.
- Buyurtmadan keyin o'tish - buyurtmadan keyin tartibda daraxt bo'ylab harakatlanishni amalga oshiradi.

```
- // Tepadagi amallarni c++ dasturlash tilida
  implementatsiyasi
- #include <bits/stdc++.h>
- using namespace std;
- // x va y orasiga qirralar qo'shish funktsiyasi
- void addEdge(int x, int y, vector<vector<int> >& adj)
- {
-     adj[x].push_back(y);
-     adj[y].push_back(x);
- }
- // ichki tugunni chiqaruvchi tugun
- void printParents(int node, vector<vector<int> >& adj,
-                 int parent)
- {
-     //hozirgi tugun ildiz tuguni
-     if (parent == 0)
-         cout << node << "->Ildiz tuguni" << endl;
-     else
-         cout << node << "->" << parent << endl;
-     // DFS dan foydalanish
-     for (auto cur : adj[node])
-         if (cur != parent)
-             printParents(cur, adj, node);
- }
- // avlodlarni chiqaruvchi funktsiya
- void printChildren(int Root, vector<vector<int> >& adj)
- {
-     // Navbat BFS ga
```

```

-   queue<int> q;
-   // Ildizni qo'shish
-   q.push(Root);
-   // massivga tashrif buyirish va tugunlarni eslab qalish
-   int vis[adj.size()] = { 0 };
-   // BFS
-   while (!q.empty()) {
-       int node = q.front();
-       q.pop();
-       vis[node] = 1;
-       cout << node << "-> ";
-       for (auto cur : adj[node])
-           if (vis[cur] == 0) {
-               cout << cur << " ";
-               q.push(cur);
-           }
-       cout << endl;
-   }
- }
- // Barglarni chiqaruvchi funktsiya
- void printLeafNodes(int Root, vector<vector<int> >& adj)
- {
-     // Leaf nodes have only one edge and are not the root
-     // Barg tugunlarining faqat bir qirralar ildizi
-     for (int i = 1; i < adj.size(); i++)
-         if (adj[i].size() == 1 && i != Root)
-             cout << i << " ";
-     cout << endl;
- }
- //har bir tugunlarni darajasini chiqarish funktsiyasi
- void printDegrees(int Root, vector<vector<int> >& adj)
- {
-     for (int i = 1; i < adj.size(); i++) {
-         cout << i << ": ";
-         // Root has no parent, thus, its degree is equal
to
-         //Ildizda ajdot yo'q,
-         // qirlarni bir biriga bog'lash

```

```

-         if (i == Root)
-             cout << adj[i].size() << endl;
-         else
-             cout << adj[i].size() - 1 << endl;
-     }
- }
- // Bosh funktsiya
- int main()
- {
-     // Tugunlar soni
-     int N = 7, Root = 1;
-     // Ro'yxatlarni daraxtka qo'shish
-     vector<vector<int>> > adj(N + 1, vector<int>());
-     // daraxtni yaratish
-     addEdge(1, 2, adj);
-     addEdge(1, 3, adj);
-     addEdge(1, 4, adj);
-     addEdge(2, 5, adj);
-     addEdge(2, 6, adj);
-     addEdge(4, 7, adj);
-     // Tugunlarning ajdotlarini chiqarish
-     cout << "Tugunlarning ajdotlari:" << endl;
-     printParents(Root, adj, 0);
-
-     // avlodlarni chiqarish
-     cout << "Tugunlarning avlodlari: " << endl;
-     printChildren(Root, adj);
-
-     // Daraxtning barglarini chiqarish
-     cout << "Daraxt barglarini chiqarish: " << endl;
-     printLeafNodes(Root, adj);
-
-     // Har bir tugunlarni chiqarish
-     cout << "Tugunlarning darajasi: " << endl;
-     printDegrees(Root, adj);
-
-     return 0;
- }

```

|                    |  |   |
|--------------------|--|---|
| - Dastur natijasi: |  |   |
| -                  | Tugunlarning ajdotlari:<br>1->Ildiz tuguni<br>2->1<br>5->2<br>6->2<br>3->1<br>4->1<br>7->4 | Tugunlarning avlodlari:<br>1-> 2 3 4<br>2-> 5 6  <br>3-><br>4-> 7<br>5-><br>6-><br>7->                                  |
| -                  | -  | Daraxt barglarini chiqaris<br>3 5 6 7<br>Tugunlarning darajasi:<br>1: 3<br>2: 2<br>3: 0<br>4: 1<br>5: 0<br>6: 0<br>7: 0 |

## 1.4 Daraxt ma'lumotlar tuzilmasining qo'llanishi, afzallikgi va kamchiligi

### Daraxtga ma'lumotlar tuzilmasini qo'llash:

- *Fayl tizimi:* bu samarali navigatsiya va fayllarni tartibga solishni ta'minlaydi.
- *Ma'lumotlarni siqish:* Huffman kodlash-bu ikkilik daraxtni qurishni o'z ichiga olgan mashhur ma'lumotlarni siqish usuli bo'lib, unda barglar belgilar va ularning paydo bo'lish chastotasini ifodalaydi. Olingan daraxt ma'lumotlarni kerakli xotira hajmini minimallashtiradigan tarzda kodlash uchun ishlatiladi.
- *Kompilyator dizayni:* kompilyatorni loyihalashda sintaksis daraxti dastur tuzilishini ifodalash uchun ishlatiladi.
- *Ma'lumotlar bazasini indeksatsiya qilish:* b-daraxtlar va boshqa daraxt tuzilmalari ma'lumotlar bazasini indeksatsiya qilishda ma'lumotlarni samarali qidirish va olish uchun ishlatiladi.

### Daraxtning afzalliklari:

- *Samarali qidirish:* daraxtlar ma'lumotlarni qidirish va olish uchun ayniqsa samarali. Daraxtda qidirishning vaqt murakkabligi odatda  $O(\log n)$ , bu juda katta ma'lumotlar to'plamlari uchun ham juda tez ekanligini anglatadi.
- *Moslashuvchan o'lcham:* daraxtlar qo'shilgan yoki olib tashlangan tugunlar soniga qarab dinamik ravishda o'sishi yoki qisqarishi mumkin.

Bu ularni vaqt o'tishi bilan ma'lumotlar hajmi o'zgarishi mumkin bo'lgan ilovalar uchun ayniqsa foydali qiladi.

- *O'tish oson:* daraxtni kesib o'tish oddiy operatsiya bo'lib, uni dastur talablariga qarab bir necha xil usulda bajarish mumkin. Bu daraxt tuzilishidan ma'lumotlarni olish va qayta ishlashni osonlashtiradi.
- *Saqlash oson:* daraxtlarni parvarish qilish oson, chunki ular tugunlar orasidagi qat'iy ierarxiya va munosabatlarni ta'minlaydi. Bu daraxt tuzilishining qolgan qismiga ta'sir qilmasdan tugunlarni qo'shish, olib tashlash yoki o'zgartirishni osonlashtiradi.

**Daraxtning kamchiliklari:**

- *Xotira ustki qismi:* daraxtlar saqlash uchun katta hajmdagi xotirani talab qilishi mumkin, ayniqsa ular juda katta bo'lsa. Bu xotira resurslari cheklangan ilovalar uchun muammo bo'lishi mumkin.
- *Muvozanatsiz daraxtlar:* agar daraxt muvozanatli bo'lmasa, bu qidiruv vaqtining notekisligiga olib kelishi mumkin. Bu tezlik juda muhim bo'lgan dasturlarda muammo bo'lishi mumkin.
- *Murakkablik:* daraxtlar murakkab ma'lumotlar tuzilmalari bo'lishi mumkin va ularni to'g'ri tushunish va amalga oshirish qiyin bo'lishi mumkin. Bu ular bilan tanish bo'lmagan ishlab chiquvchilar uchun muammo bo'lishi mumkin.
- *Cheklangan moslashuvchanlik:* daraxtlar hajmi va tuzilishi jihatidan moslashuvchan bo'lsa-da, ular Xesh jadvallari kabi boshqa ma'lumotlar tuzilmalari kabi moslashuvchan emas. Ma'lumotlar hajmi tez-tez o'zgarishi mumkin bo'lgan dasturlarda bu muammo bo'lishi mumkin.
- *Muayyan operatsiyalar uchun samarasiz:* daraxtlar ma'lumotlarni qidirish va olish uchun juda samarali bo'lsa-da, ular saralash yoki guruhlash kabi boshqa operatsiyalar uchun unchalik samarali emas. Ushbu turdagi



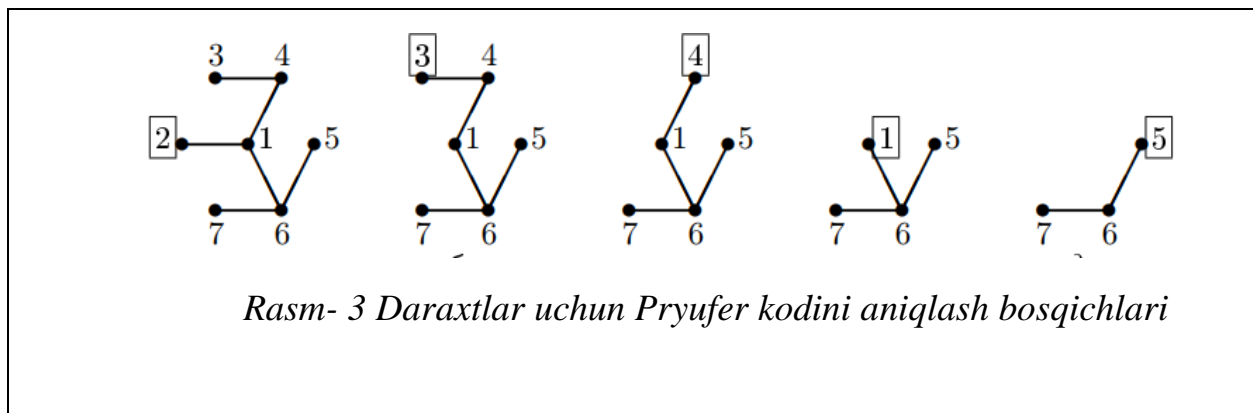
operatsiyalar uchun boshqa ma'lumotlar tuzilmalari ko'proq mos kelishi mumkin.

- *Tez kiritish va o'chirish*: daraxtga tugunlarni kiritish va o'chirish  $O(\log n)$  vaqtida amalga oshirilishi mumkin, ya'ni bu juda katta daraxtlar uchun ham juda tezdir.

### 1.5 Pryufer Kodi

Pryufer kodi  $[1, n]$  kesmadagi  $n-2$  butun sonlar ketma-ketligi yordamida  $n$  uchlari bilan belgilangan daraxtlarni birma-bir kodlash usuli. Ya'ni, Pryufer kodi - bu to'liq graf va raqamlar ketma-ketligining barcha daraxtlari orasidagi biyeksiyasidir. Daraxtlarni kodlashning ushbu usuli nemis matematiki Xaynts Pryufer tomonidan 1918-yilda taklif qilingan.  $n$  ta uchlari bilan berilgan daraxt uchun Pryufer kodini qurish algoritmini ko'rib chiqaylik. Kirishda qirralarning ro'yxati berilgan. Eng kichik raqamga ega bo'lgan daraxtning bargi tanlanadi, keyin u daraxtdan olib tashlanadi va bu barg bilan bog'langan uchlarning soni Pryufer kodiga qo'shiladi. Ushbu protsedura  $n - 2$  marta takrorlanadi. Oxir-oqibat, daraxtda faqat 2 ta uch qoladi va algoritmi shu yerda tugaydi. Qolgan ikkita uchning raqamlari kodga yozilmaydi.

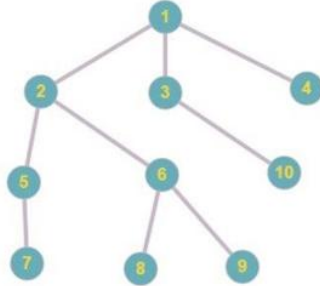
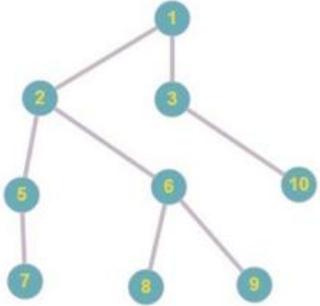
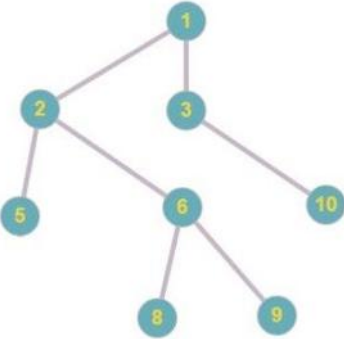
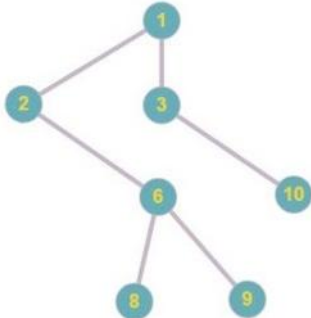
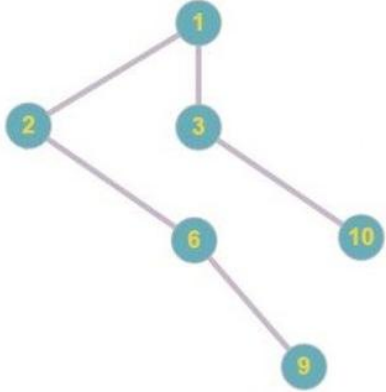
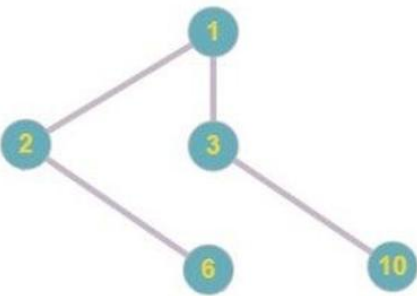
Shunday qilib, ma'lum bir daraxt uchun Pryufer kodi  $n - 2$  ta raqamlar ketma-ketligi bo'lib, bu yerda har bir raqam eng kichik barg bilan bog'langan uchning soni - bu segmentdagi raqam  $[1, n]$ . Pryufer kodini aniqlash:

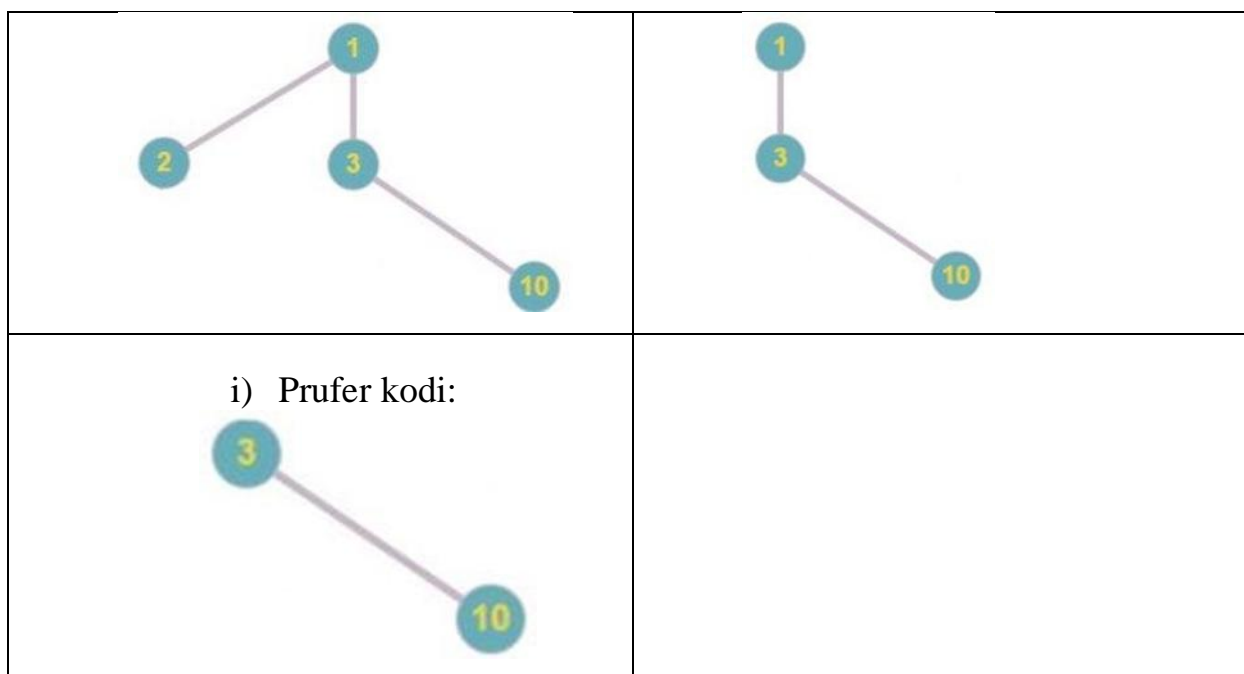


Kodni olish algoritmi quyidagicha. Daraxt tugunlari 1 dan n gacha bo'lgan raqamlar bilan belgilangan (raqamlangan) bo'lsin. Biz eng kichik sonli 1-darajali uchni topamiz va kodga unga qo'shni bo'lgan uchning sonini kiritamiz, shundan so'ng topilgan uchni (qirradi bilan birga) o'chirib tashlaymiz. Olingan graf osti bilan biz xuddi shu amalni bajaramiz, uni faqat bitta qirra qolguncha takrorlaymiz. Kodni yaratish jarayoni 49-rasmda keltirilgan. Keyingi bosqichda o'chirilgan uchning soni ramkaga kiritilgan. Berilgan grafda (49-rasm, a) birinchi darajali uchlar orasida minimal son 2-uchda joylashgan. U 1-uchga qo'shni. Shuning uchun Pryufer kodining birinchi raqami 1. 2-uchni olib tashlash natijasida biz b-rasmda ko'rsatilgan grafni olamiz. Ushbu grafda darajasi birga teng bo'lgan uchlar orasidagi minimal son 3 ga teng, shuning uchun kodning ikkinchi raqami 4. Shaklda ko'rsatilgan graflarga mos keladigan yana uchta takrorlashni bajargandan so'ng, c, d, e-rasmlardagi bitta qirradan iborat daraxtni olamiz {7; 6}. Jarayon tugallandi.

Qabul qilingan qadamlarning natijalari jadvalda keltirilgan. Oxirgi qatorida kerakli kod mavjud - 14166.

|                                  |  |       |                    |       |       |       |
|----------------------------------|--|-------|--------------------|-------|-------|-------|
| Qadam                            |  | 1     | 2                  | 3     | 4     | 5     |
| 49-rasm                          |  | a     | b                  | c     | d     | e     |
| Minimal raqam                    |  | 2     | 3                  | 4     | 1     | 5     |
| O'chirilgan qirra                |  | {1;2} | {4;3}              | {1;4} | {6;1} | {6;5} |
| Pryufer kodi                     |  | 1     | 4                  | 1     | 6     | 6     |
| a) Dastlabki daraxning berilishi |  |       | b) Pryufer kodi: 1 |       |       |       |

|  |   |
|--|---|
|                                   |                                     |
| <p>c) Prufer kodi: 1,5</p>        | <p>d) Prufer kodi: 1,5,2</p>       |
| <p>e) Prufer kodi: 1,5,2,6</p>  | <p>f) Prufer kodi: 1,5,2,6,6</p>  |
| <p>g) Prufer kodi: 1,5,2,6,6,2</p>   | <p>h) Prufer kodi: 1,5,2,6,6,2,1</p>  |



**Pryufer kodi orqali daraxtni tiklash.** Pryufer kodi bilan ifodalangan daraxtlarni hosil qilish algoritmi qirralarning tegishli ro'yxatini olishga imkon beradi. Antikodni Pryufer kodiga kiritilmagan uchlarning sonining ortib boruvchi ketma-ketligi deylik. Ko'rib chiqilgan misol uchun antikod 2357 ga teng. Daraxt ketma-ket qirralarni qo'shib quriladi. Keyingi qo'shilgan chekka, birinchisidan boshlab, vertikal juftlik bilan hosil bo'ladi, ularning raqamlari kod satrida va antikod satrida birinchi bo'ladi. Shundan so'ng, ishlatilgan satr elementlari chiziladi. Agar kod satridan chiqib ketgan raqam undagi qolgan elementlar qatoriga kiritilmagan bo'lsa, uning tartibini buzmasdan antikod qatoriga qo'shilishi kerak. Ta'riflangan harakatlar kod va antikod satrlarining "qoldiqlari" bilan ularning birinchisining barcha elementlari o'chirilguncha takrorlanadi. Bunday holda, antikod chizig'ida hosil qilingan ro'yxatga qo'shiladigan so'nggi chekkani belgilaydigan ikkita element bo'ladi, natijada biz Pryufer kodi bilan belgilangan daraxtga mos keladigan  $n - 1$  qirralarning ro'yxatini olamiz.

**Misol.** Masalan, 1-misolda berilgan 14166 kodi yordamida daraxtni tiklaylik. Yuqorida 1-misolda ko'rsatilgandek mos keladigan antikod 2357 ni tashkil qiladi.

Shuning uchun daraxtning birinchi qirrasi {1; 2}. 1 va 2-ni kesib o‘tib, biz kod satrida 4166 va antikod satrida 357 olamiz. Keyingi takrorlashda {4; 3} juftligini kesib tashlaymiz va qatorga antikod 4 ni kiritamiz va hokazo. Takrorlashlar ketma-ketligi jadvalda keltirilgan.

|                |       |        |             |                  |             |             |   |
|----------------|-------|--------|-------------|------------------|-------------|-------------|---|
| Qadam          | 1     | 2      | 3           | 4                | 5           | 6           |   |
| Kod satri      | 4     | 4      | 4           | 6                | 6           |             |   |
| Antikod satri  | 2     | 3<br>3 | 5<br>5<br>4 | 7<br>7<br>5<br>4 | 7<br>5<br>5 | 7<br>7<br>6 | 7 |
| Qirra qo‘shish | {1;2} | {4;3}  | {1;4}       | {6;1}            | {6;5}       | {6;7 }      |   |

Qirralarning ro‘yxatini tahlil qilib, asl daraxt olinganligiga ishonch hosil qilamiz. E’tibor bering, qirralarning tartibi avvalgi jadvaldagi kabi.

**Misol.** Pryufer kodini yaratish vazifasining oldida kodlangan daraxtni tiklash vazifasi ham mavjud. Daraxtlarni qayta qurish algoritmini quyidagi shartlar bilan ko‘rib chiqamiz: kirish sifatida Pryufer kodini ifodalovchi raqamlar (uchlar) ketma-ketligi, natijada daraxt qirralarining ro‘yxati bo‘ladi.

Kod hal qilish algoritmini batafsil ko‘rib chiqamiz. Koddan tashqari, bizga grafning barcha uchlari ro‘yxati kerak. Biz bilamizki, Pryufer kodi  $n-2$  ta uchlardan iborat, bu yerda  $n$  - grafdagi uchlar soni. Ya’ni kodlangan daraxtdagi uchlar sonini kodning kattaligi bo‘yicha aniqlashimiz mumkin. Natijada, algoritmnining boshida bizda Pryuferning  $n - 2$  o‘lchamdagi kodlari va grafdagi barcha uchlar qatori mavjud:  $[1 \dots n]$ . Keyin quyidagi protsedura  $n - 2$  marta takrorlanadi: Pryufer kodini o‘z ichiga olgan massivning birinchi elementi olinadi va kod bilan massivda bo‘lmagan eng kichik uchni qidirish daraxt uchlari bilan massivda amalga oshiriladi.

Topilgan uch va Pryufer kodi bilan massivning joriy elementi daraxtning qirrasini tashkil qiladi. Ushbu uchlar tegishli massivlardan olib tashlanadi va yuqoridagi protsedura kodli qator elementlari tugamaguncha takrorlanadi. Algoritm oxirida graf uchlari bilan massivda ikkita uch qoladi; ular daraxtning so‘nggi uchini tashkil qiladi. Natijada biz grafning kodlangan barcha qirralarining ro‘yxatini olamiz.

2- misolda olingan Pryufer kodi yordamida daraxtni tiklaylik.

Birinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo‘lmagan minimal uch 4 ga teng

Qirralar ro‘yxati: 1 4

Ikkinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo‘lmagan minimal uch 7 ga teng

Qirralarning ro‘yxati: 1 4, 5 7

Uchinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo‘lmagan minimal tepalik 5 ga teng

Qirralarning ro‘yxati: 1 4, 5 7, 2 5

To‘rtinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal tepalik 8 ga teng  
Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8

Beshinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal vertex 9 ga teng  
Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8, 6 9

Oltinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal vertex 6 ga teng  
Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8, 6 9, 2 6

Yettinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal tepalik 2 ga teng  
Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8, 6 9, 2 6, 1 2

Sakkizinchi qadam

Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal tepalik 1 ga teng  
Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8, 6 9, 2 6, 1 2, 3 1

Algoritmni yakunlash Pryufer kodi: 1 5 2 6 6 2 1 3

Daraxtlar uchlari massivi: 1 2 3 4 5 6 7 8 9 10

Pryufer kodida mavjud bo'lmagan minimal tepalik 1 ga teng

Qirralarning ro'yxati: 1 4, 5 7, 2 5, 6 8, 6 9, 2 6, 1 2, 3 1, 3 10