Нэгдэл-Хайлтын Бодлого

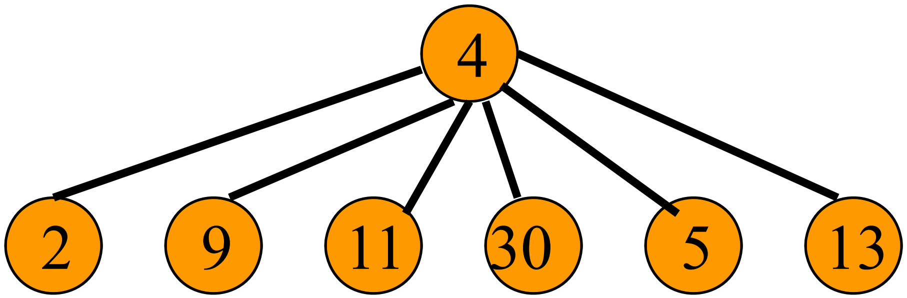
- n элементтэй {1, 2, …, n} олонлог байна
- Анхандаа элемент бүр олонлог.
  - {1}, {2}, …, {n}
- Олонлогт нэгдэл, хайлтын үйлдлийг гүцэтгэнэ.
- Нэгдэлээр хоёр олонлогийг нийлүүлж нэг болгоно.
  - n –ийн элемент бүр ямагт нэг олонлог.
- Хайлтын үйлдэл тодорхой элементийг олонлогт байгаа эсэхийг тогтооно.

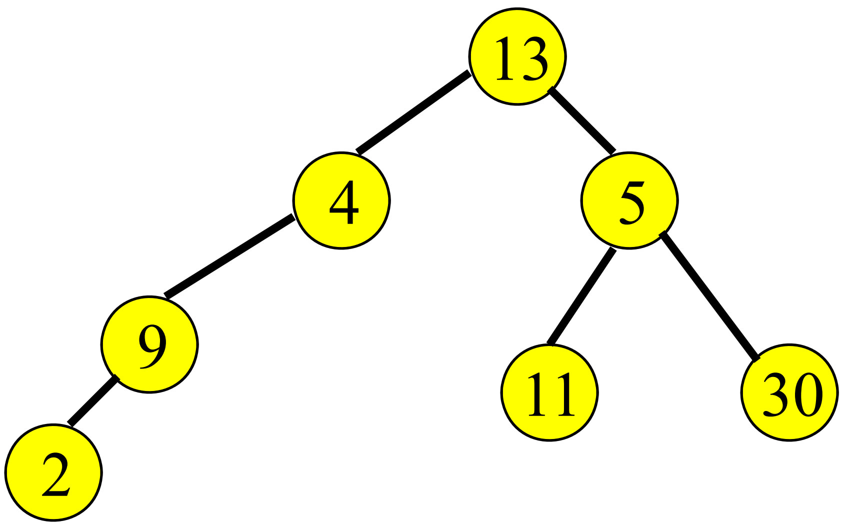
Массив ба Холбоосыг ашиглах

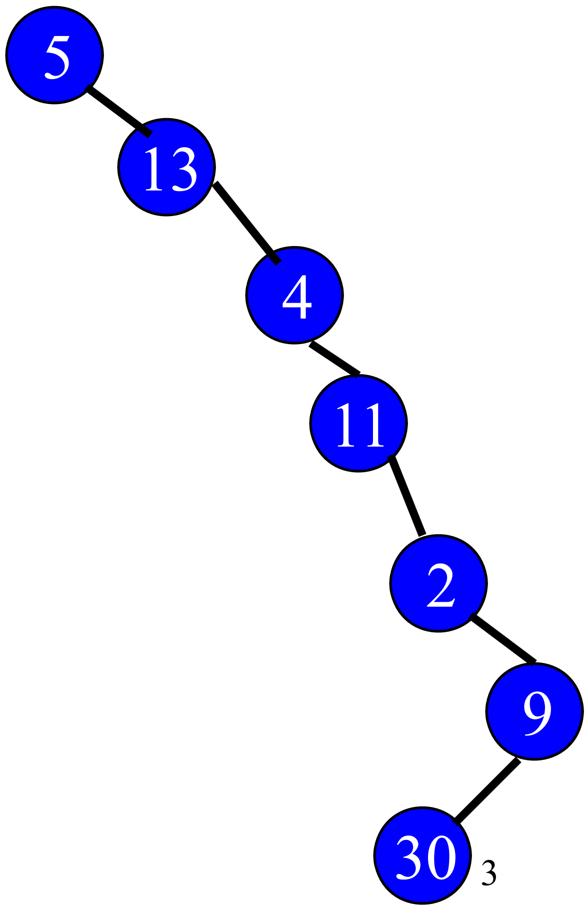
- Сурах бичгийн 7.7 –д массив болон холбоос ашигласан бодлогын шийдэл байгаа.
- 7.7 –н шийдлийн хамгийн сайн хугацаа нь $O(n + u \log u + f)$, үүнд u , f хийгдэх нэгдэл, хайлтын үйлдлийн тоо.
- Мод (хоёртын биш) ашиглаж олонлогийг дүрслэхэд, зарцуулах хугацаа $O(n + f)$ байна. (дор хаяж $n/2$ нэгдэх үйлдэл хийгдсэн гэж үзвэл).

Олонлогийг мод болгох

- S = {2, 4, 5, 9, 11, 13, 30}
- Зарим боломжит модны дүрслэл:

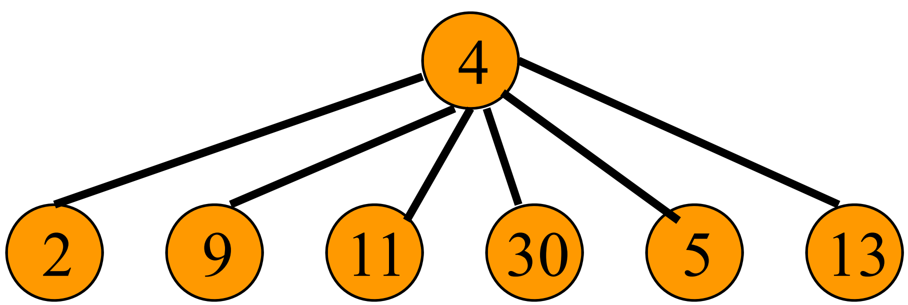






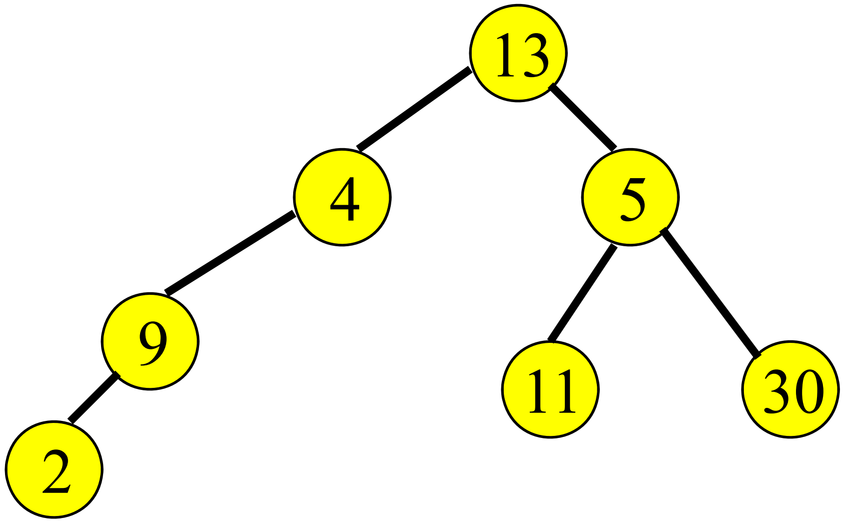
Хайлтын үйлдлийн үр дүн

- find(i) элемент i -г агуулсан олонлогийг тогтооно
- Нэгдэл-хайлтын бодлого ашигласан ихэнх хэрэглээнд хэрэглэгч олонлогийг таних үйлдэл хийдэггүй.
- find(i) , find(j) үйлдлүүдийн шаардлага бол хэрвээ i , j элементүүд нэг олнглогт байгаа бол адилхан утга буцаах явдал.



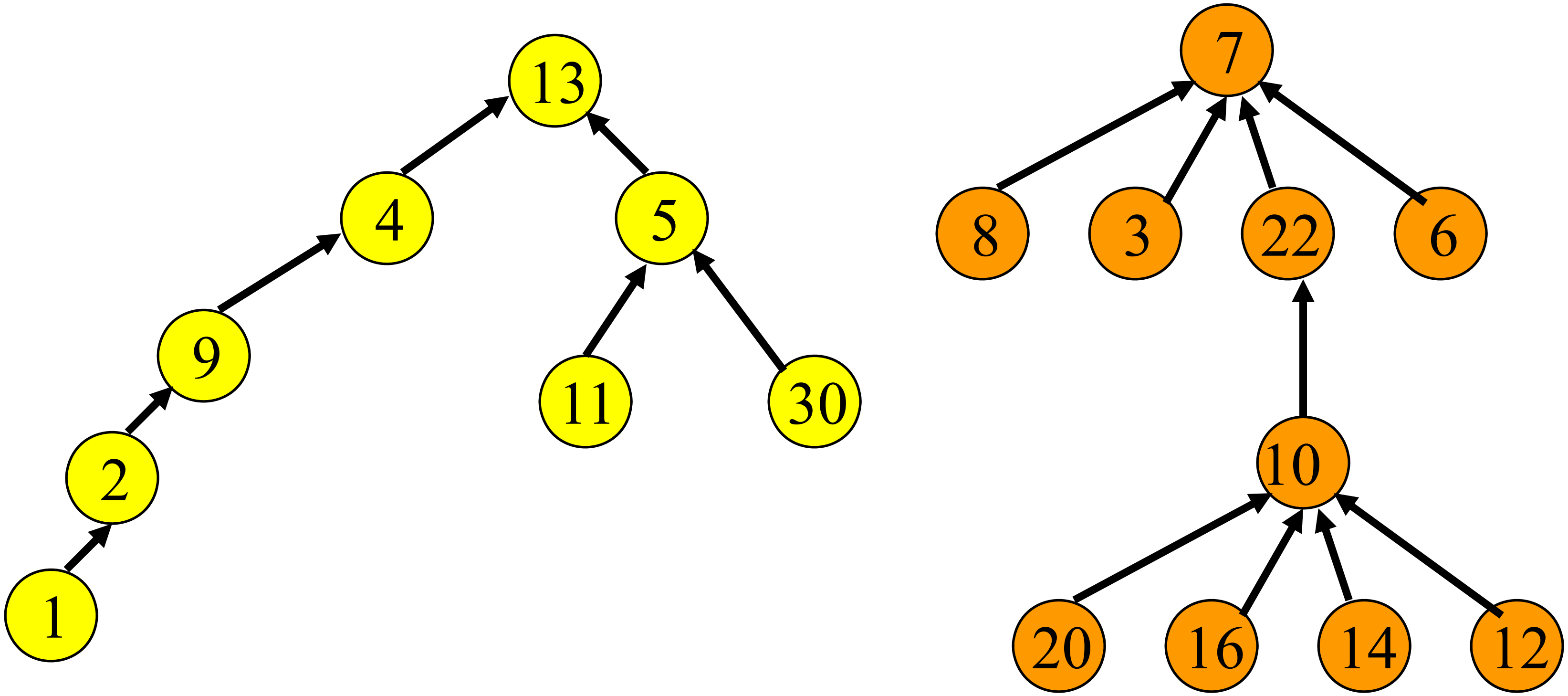
find(i) модны үдэст байгаа элементийг буцаах болно.

find(i) үйлдлийн стратеги



- Элемент i –г дүрсэлсэн зангилаанаас эхлээд модны үндэс хүртэл өгсөнө.
- Үндэсний элементийг буцаана.
- Модоор өгсөхөд зангилаа бүрт дээд түвшний заагч байх ёстой.

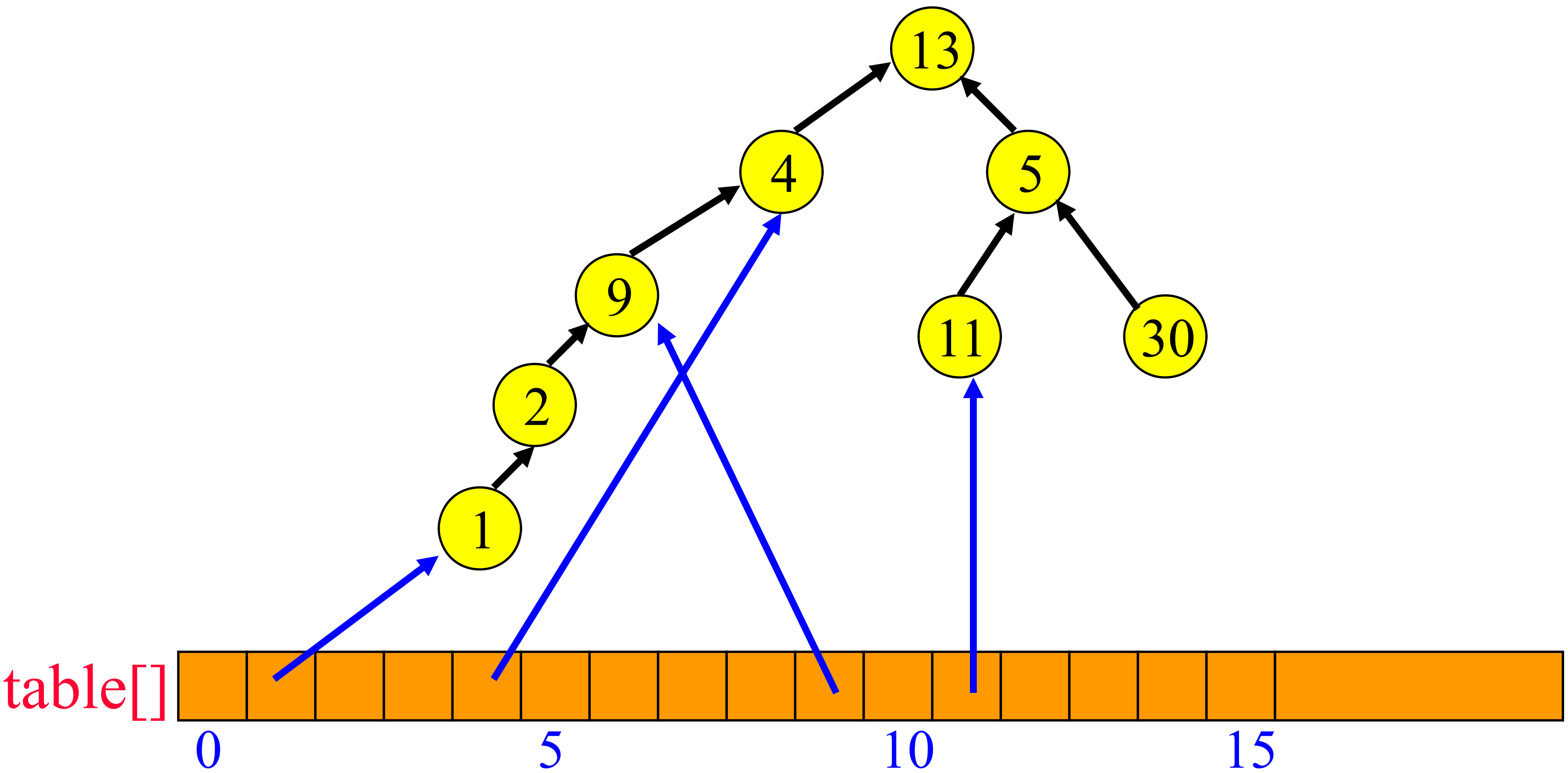
“Эцэг” заагчтай моднууд



Боломжит зангилааны бүтэц

- Зангилаа бүр хоёр талбартай: element , parent.
  - table[] массивыг ашиглахдаа table[i] нь i элементтэй зангилаанд хүрэх заагч байна
  - find(i) үйлдлийг хийхийн тулд, table[i] –ийн өгсөн зангилаанаас эхлээд, parent талбаруудаар явсаар энэ талбар нь null зангилаанд хүрнэ.
  - Үндэс болсон зангилааны элементийг буцаана.

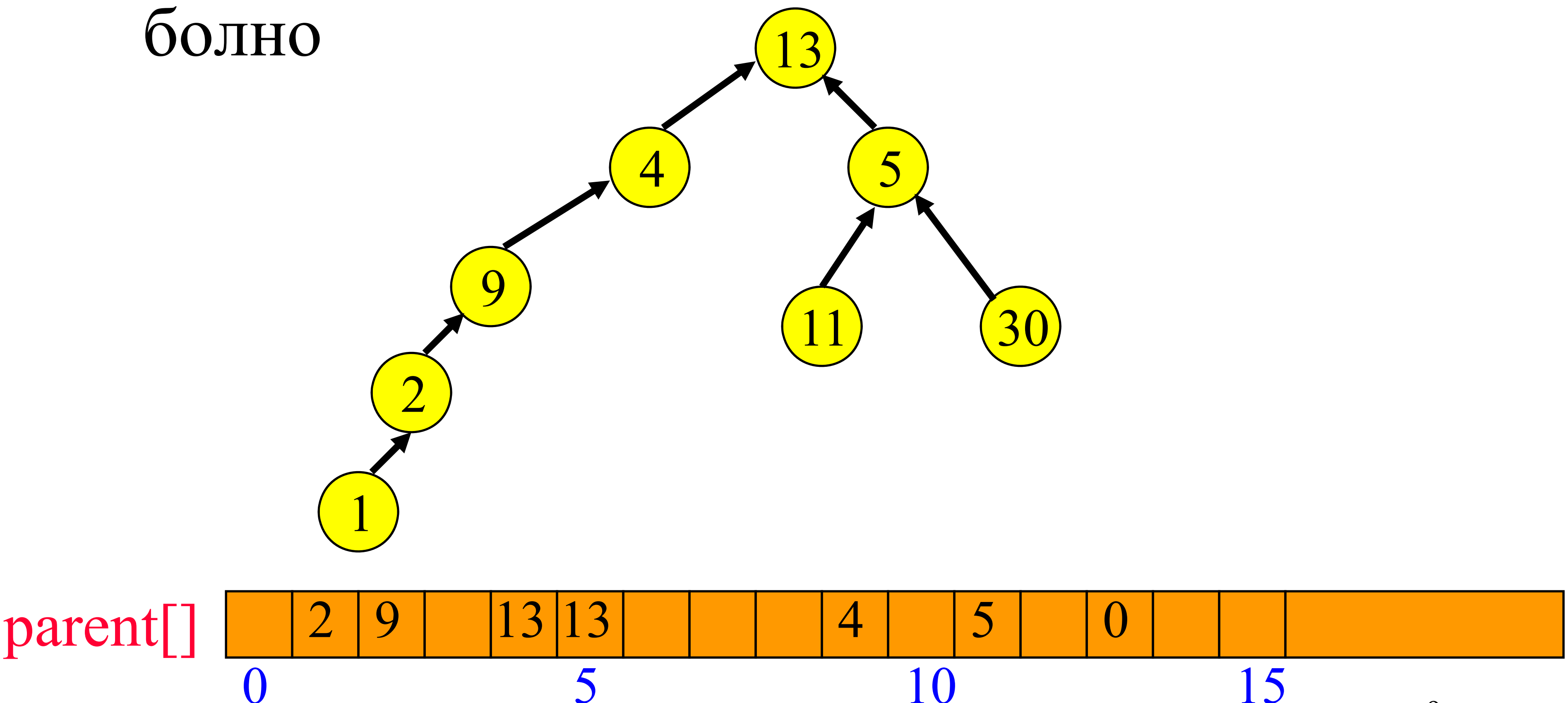
Жишээ



(table –ийн зарим нэг утгыг харууллаа.)

Сайн дүрслэл

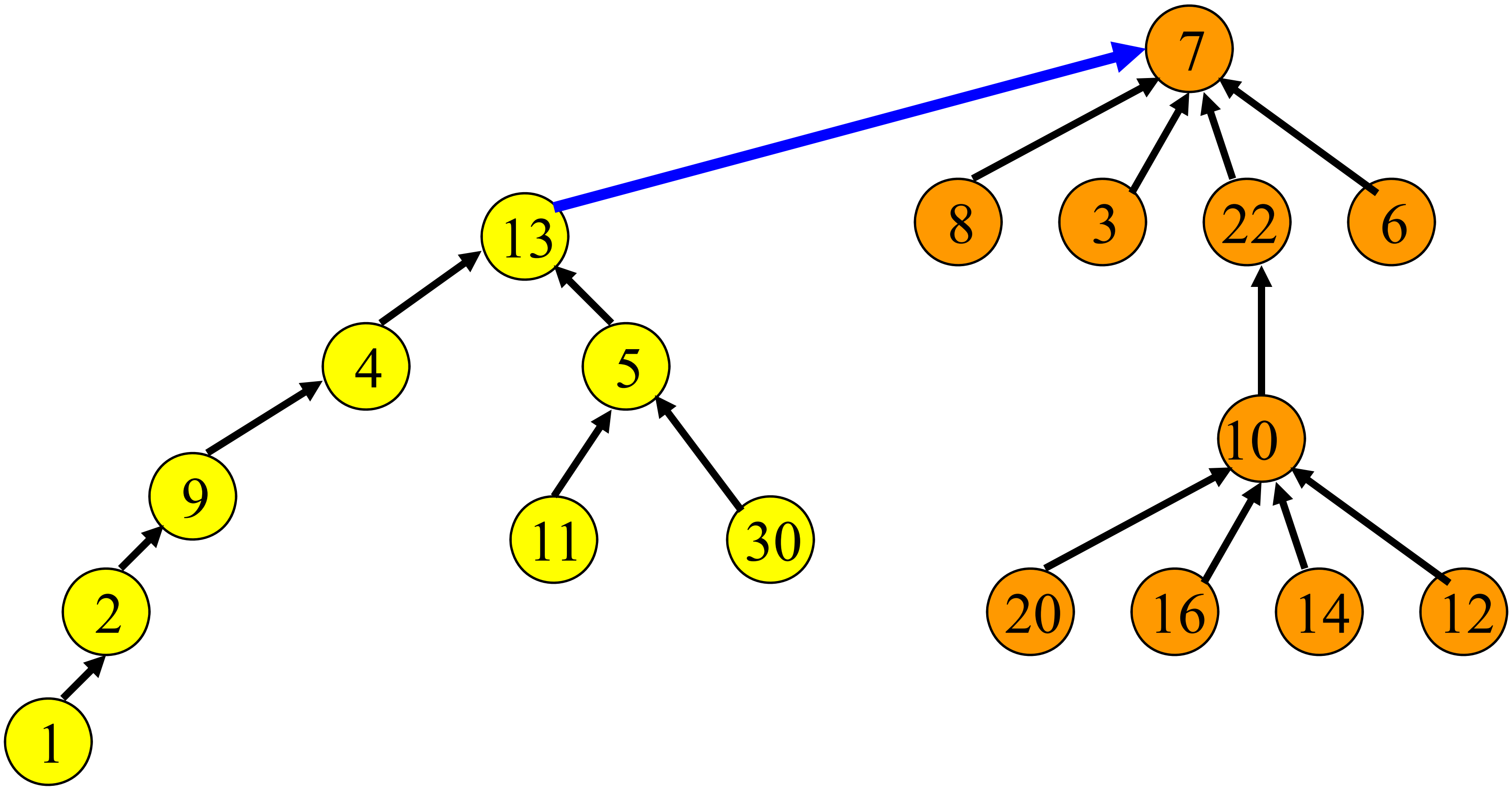
- Бүхэл массив parent[] –г ашиглахдаа parent[i] нь i элементийн эцэг элемент болно



Union үйлдэл

- union(i,j)
  - i , j бол өөр моднуудын үндэс, i != j.
- Моднуудыг нэгтгэхдээ нэг модыг нь нөгөөгийн дэд мод болгоно.
  - parent[j] = i

Union жишээ



- union(7,13)

Find арга

```
public int find(int theElement)
{
    while (parent[theElement] != 0)
      theElement = parent[theElement];  // дээш явах
    return theElement;
}
```

Union арга

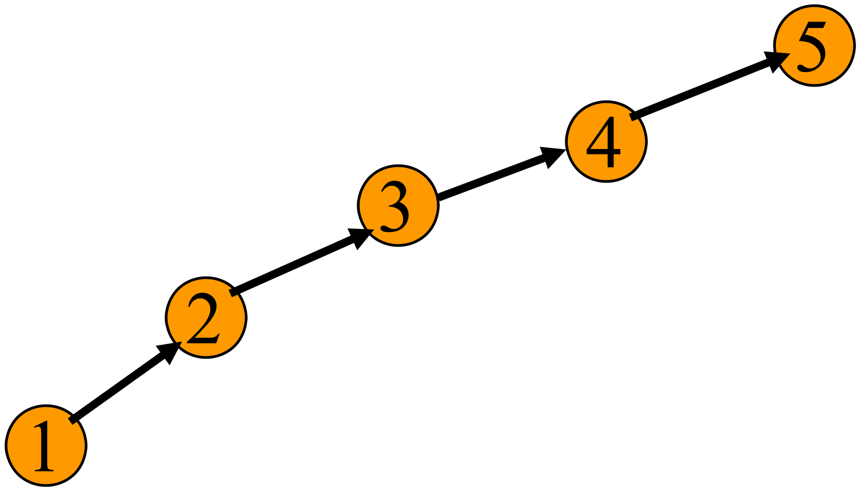
```
public void union(int rootA, int rootB)
    {parent[rootB] = rootA;}
```

union() үйлдлийн хугацаа

- O(1)

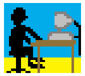
find() үйлдлийн хугацаа

- Модны өндөр түүний элементийн тоотой тэнцүү байж болно.
  - union(2,1), union(3,2), union(4,3), union(5,4)…

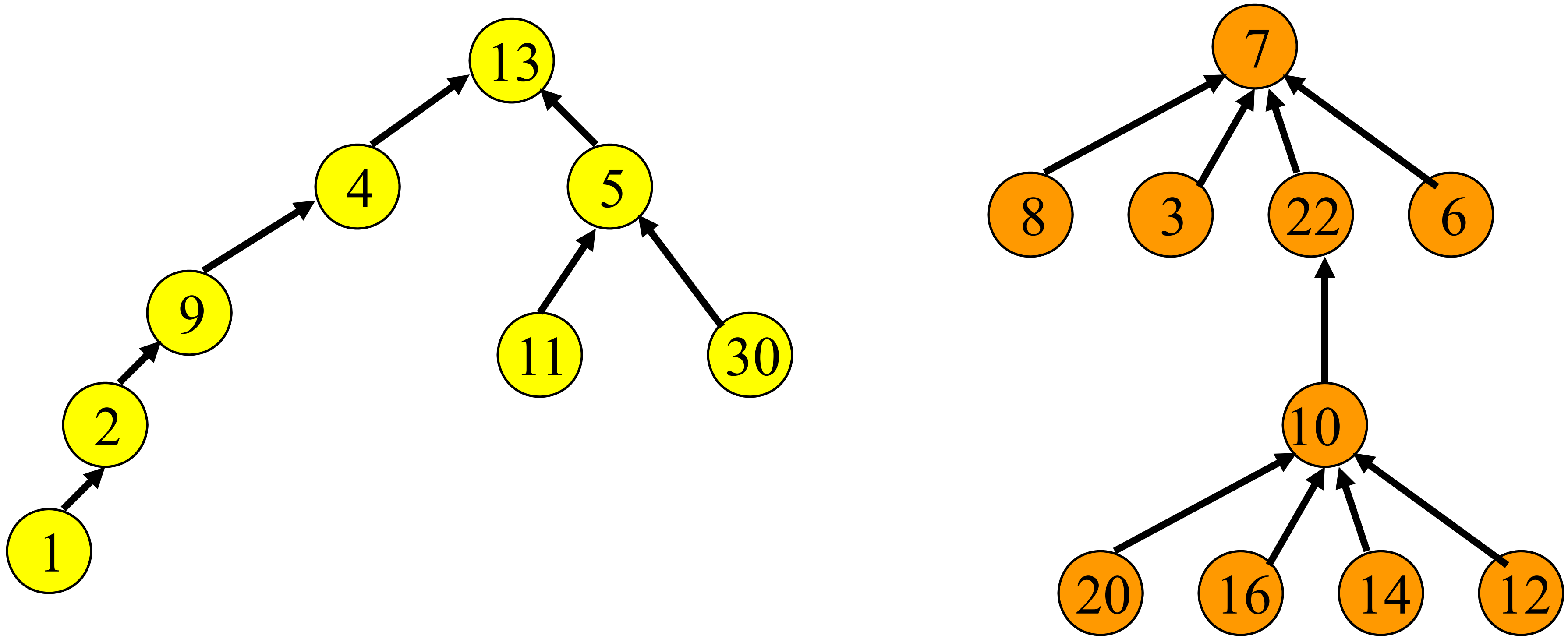


Тэгвэл O(u).

u нэгдэл , f хайлт үйлдэл

- $O(u + uf) = O(uf)$
- Бүх i –г идэвхижүүлэх $parent[i] = 0$ хугацаа $O(n)$.
- Нийт хугацаа $O(n + uf)$.
- 7.7 шийдлээс муу байна!
- Дахиад зурцгаая. 

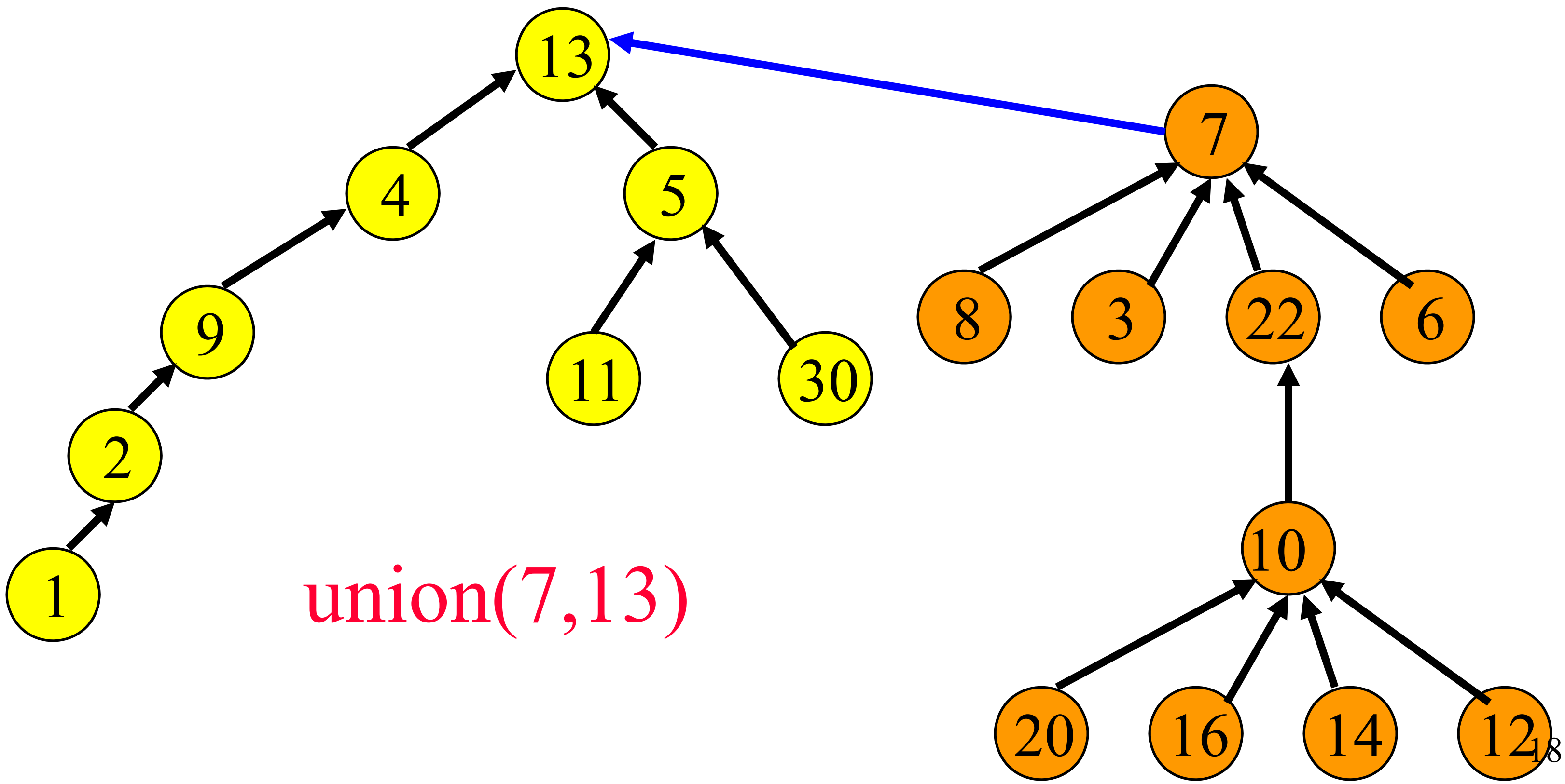
Ухаалаг Union стратеги



- union(7,13)
- Аль мод нь дэд мод болох вэ?

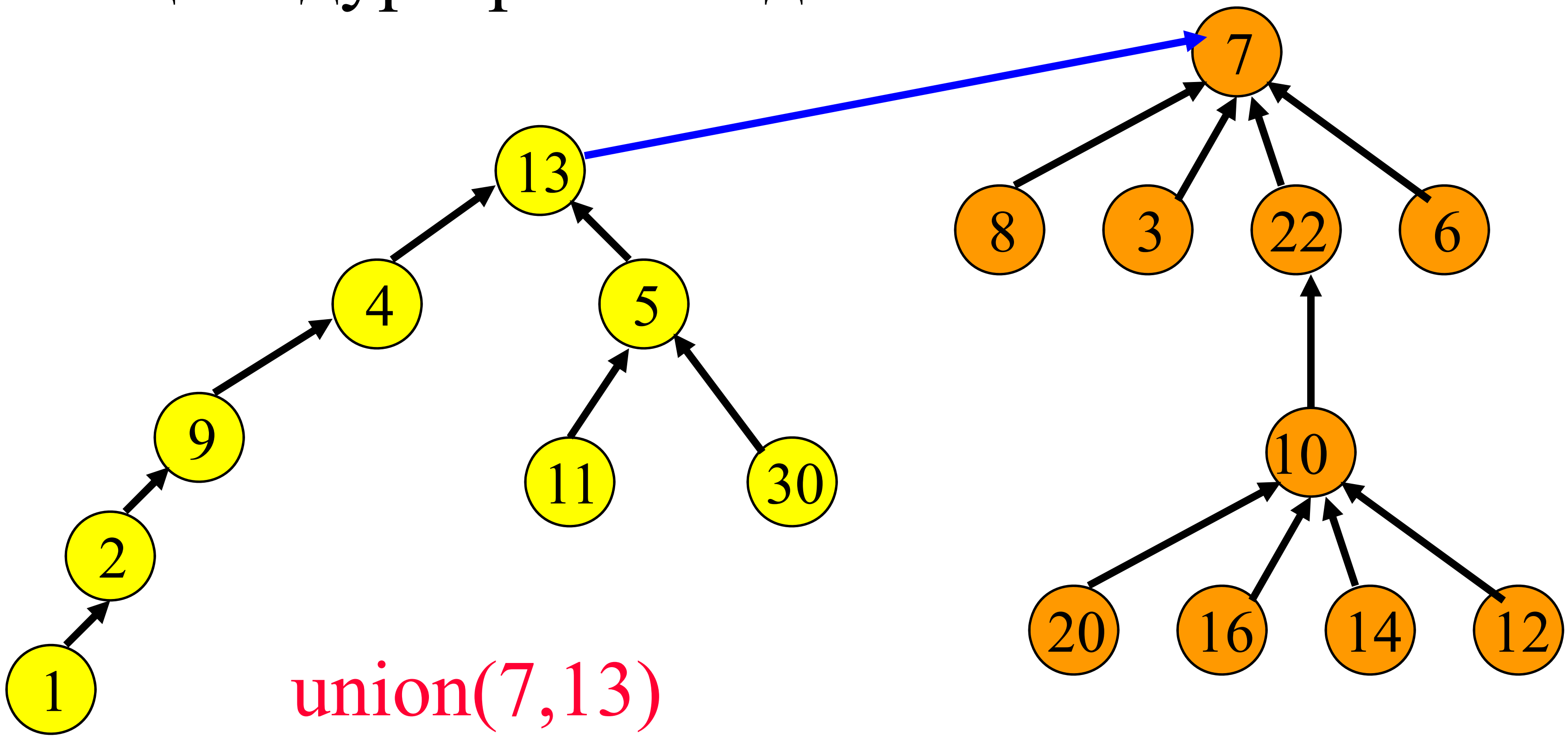
Өндрийн дүрэм

- Бага өндөртэй нь дэд мод болно.
- Хайнцааг дураараа шийднэ.



Жингийн дүрэм

- Цөөн элементэй нь дэд мод болно.
- Хайнцааг дураараа шийднэ.



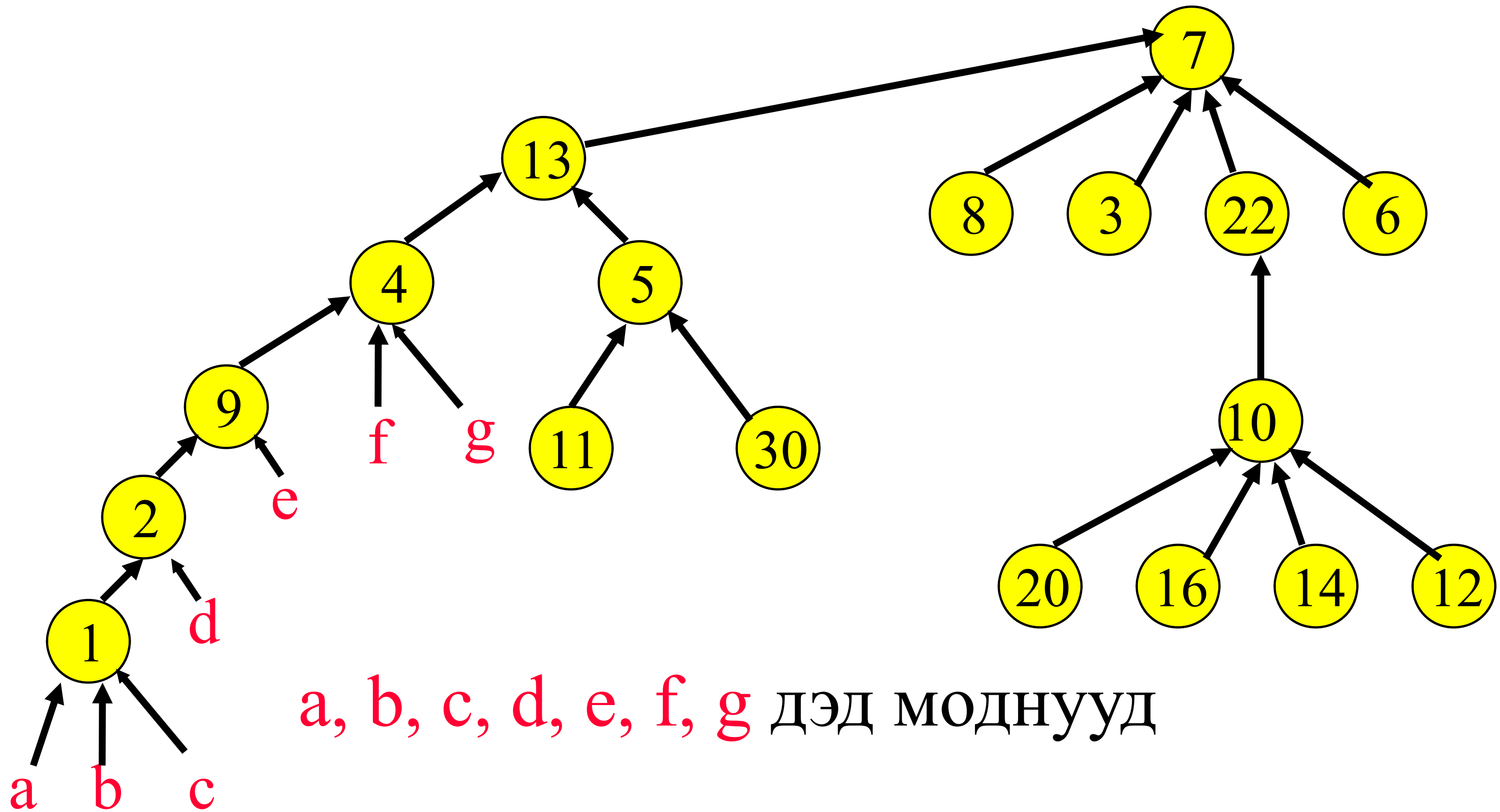
Хэрэгжүүлэлт

- Модны үндэс бүрт өндөр, элементийн тооны аль нэгийг бичнэ
- Хэрвээ union –д өндрийн дүрэм ашигласан бол, зөвхөн ижил өндөртэй моднуудыг нэгтгэхэд өндөр өснө.
- Хэрвээ union –д жингийн дүрмийг ашигласан бол, шинэ модны жин нь хоёр модны жингийн нийлбэр байна.

Модны өндөр

- Ганц элементтэй модноос эхэлж өндрийн, эсхүл жингийн дүрмээр нэгдлүүдийг гүйцэтгэсэн гэж үзье.
- p элементтэй модны өндөр $\text{floor}(\log_2 p) + 1$.
- Баталгааг сурах бичгээс харна уу.

Find аргыг тордож, сайжруулья

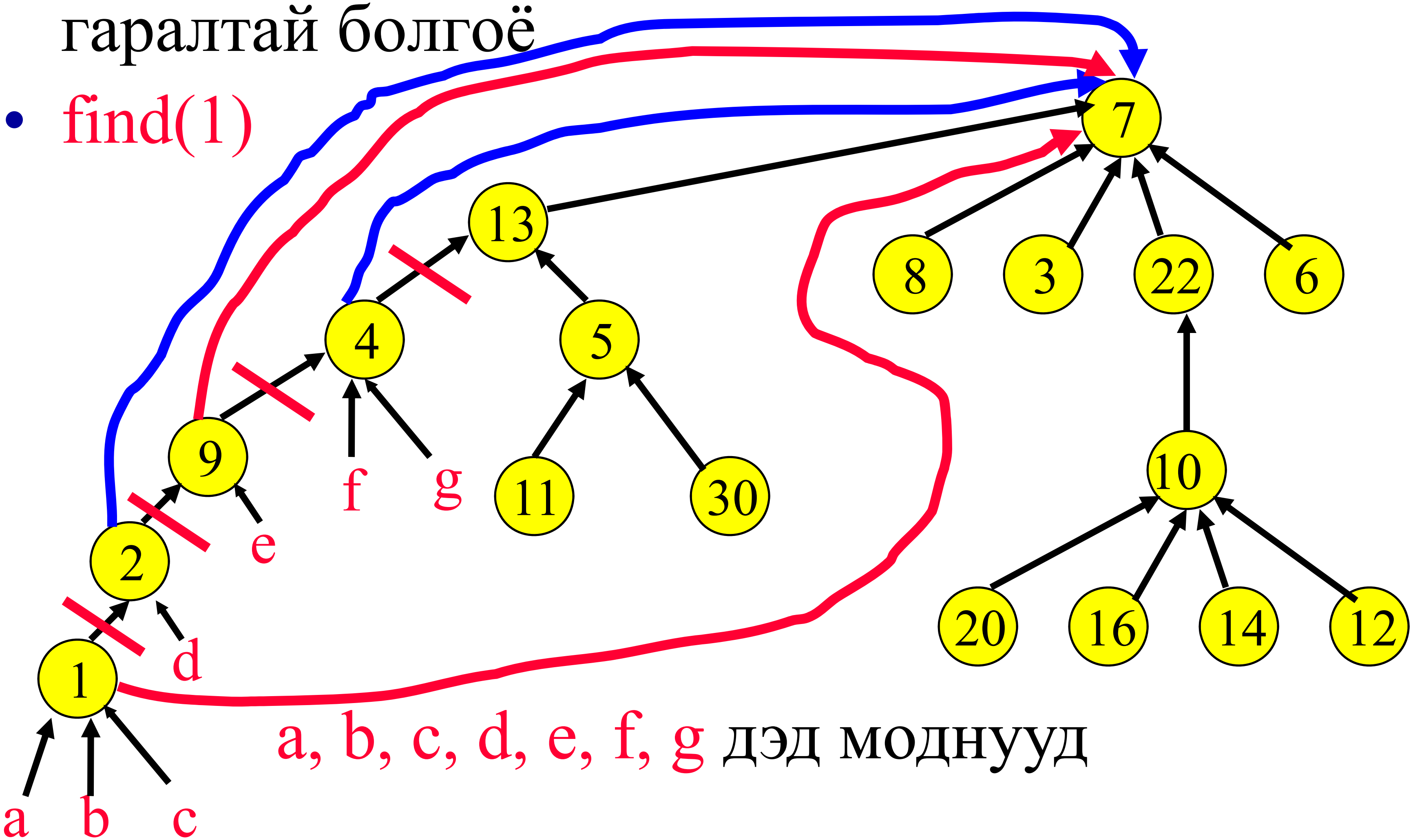


a, b, c, d, e, f, g дэд моднууд

- find(1)
- Хожмын хайлтыг хялбаршуулах нэмэлт ажил хийе

Замыг нягтруулах

- Хайлтын зам дээрх бүх зангилааг үндэснээс гаралтай болгоё
- find(1)

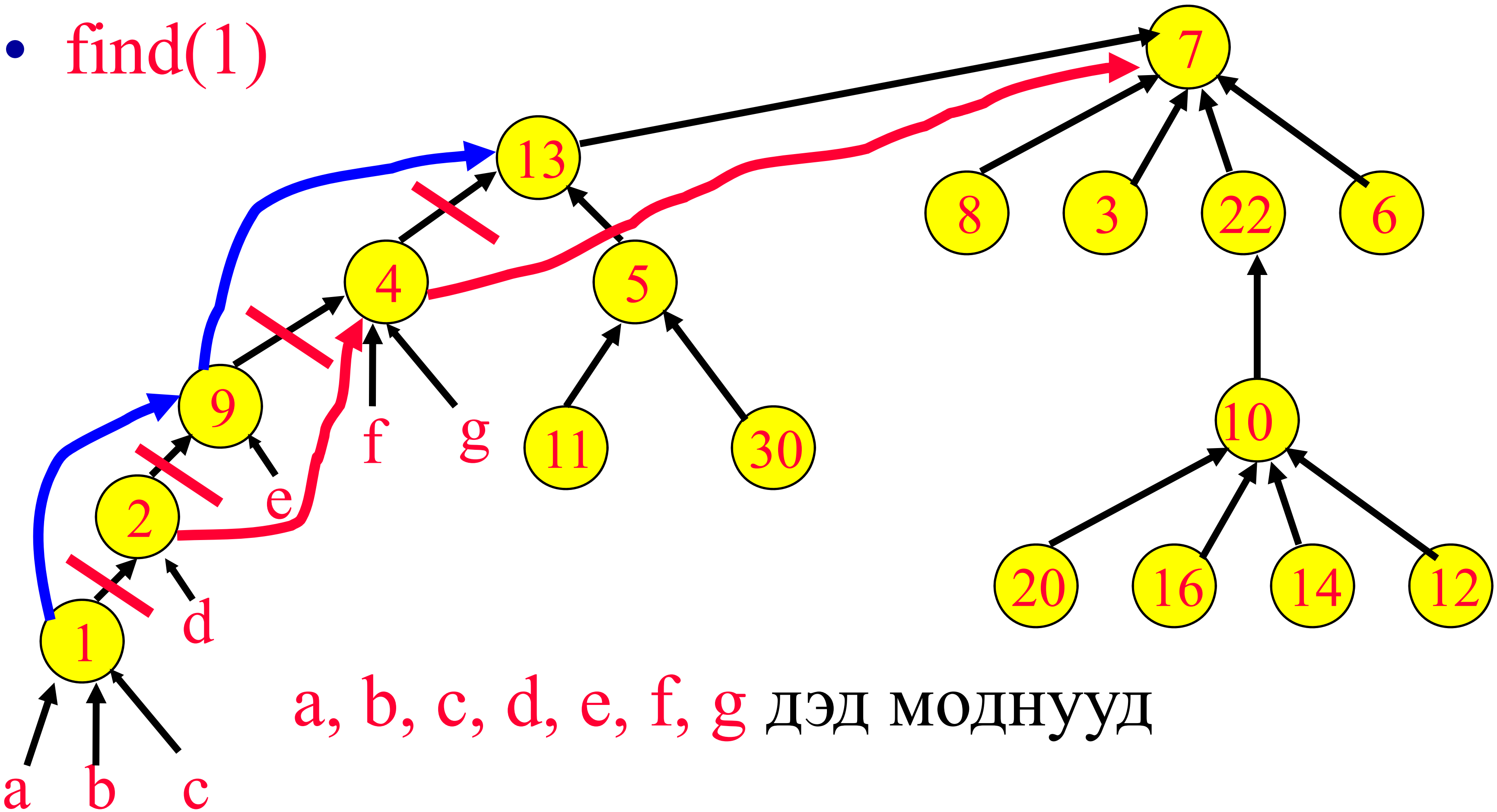


a, b, c, d, e, f, g дэд моднууд

Хоёроор дээшиллээ.

Замыг хуваах

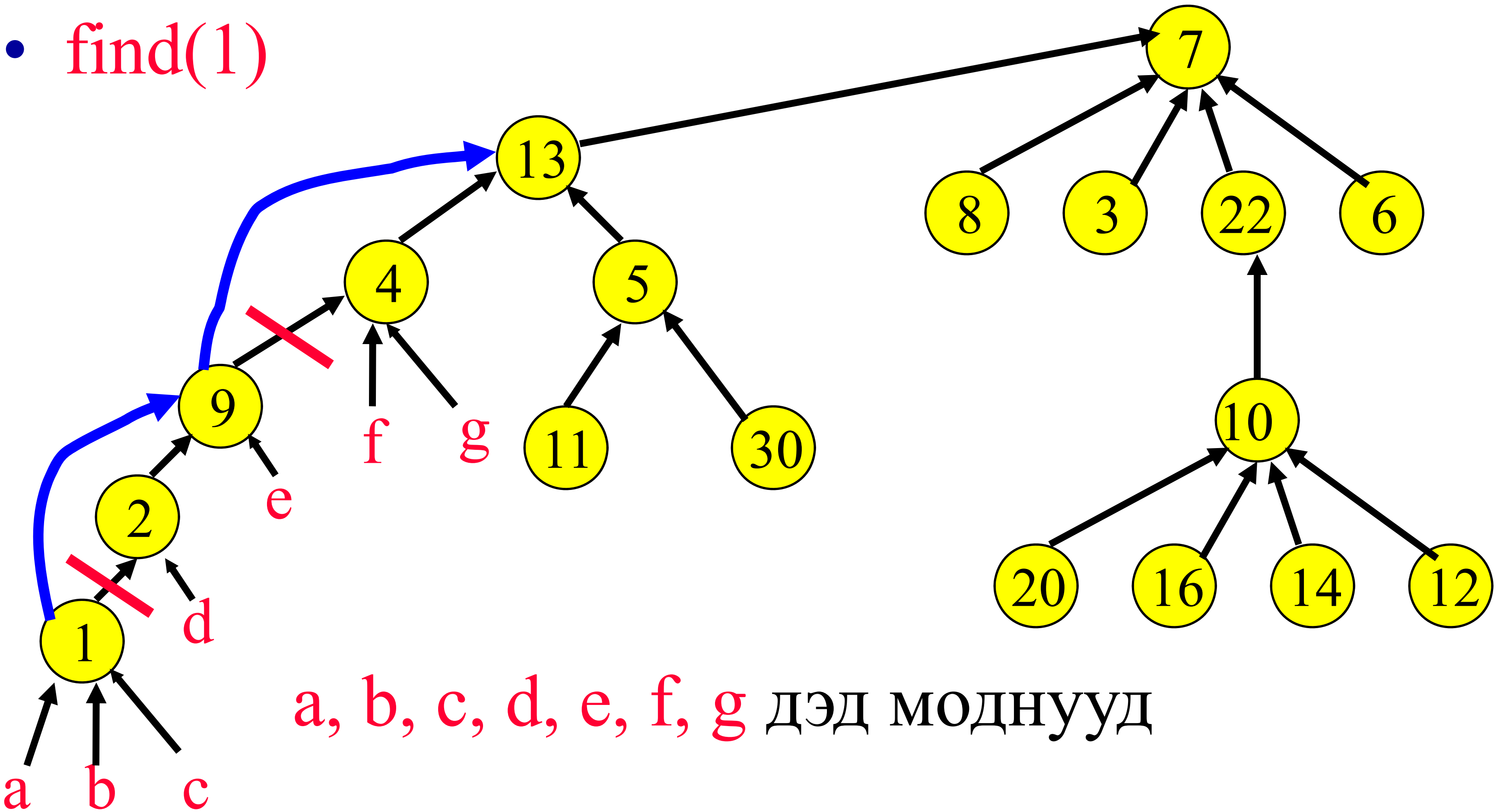
- Хайлтын замын зангилаануудыг өвөг эцэг рүү заах
- find(1)



Зөвхөн нэгээр дээшиллээ

Замыг таллах

- Хайлтын замын бусад зангилаануудын эцэг заагчийг өвөг эцэг заагч болгон өөрчилье.
- find(1)



a, b, c, d, e, f, g дэд моднууд

Өөрчлөлт ихэнх заагчийг таллана.

Хугацааны шинжилгээ

- Акерманы функц.
  - $A(i,j) = 2^j$, $i = 1$ ба $j >= 1$
  - $A(i,j) = A(i-1,2)$, $i >= 2$ ба $j = 1$
  - $A(i,j) = A(i-1,A(i,j-1))$, $i, j >= 2$
- Акерманы урвуу функц.
  - $alpha(p,q) = \min\{z>=1 \mid A(z, p/q) > \log_2 q\}$, $p >= q >= 1$

Хугацааны шинжилгээ

- i болон j өсөхийн хирээр Акерманы функц маш хурдэн өсдөг.
  - $A(2,4) = 2^{65,536}$
- Урвуу функц удаан өсдөг.
  - $alpha(p,q) < 5$ -аас $q = 2^{A(4,1)}$
  - $A(4,1) = A(2,16) >>>> A(2,4)$
- Нэгдэл-хайлтын бодлогын шинжилгээнд, q нь элементийн тоо n; $p = n + f$; ба $u >= n/2$.
- Бүх практик зорилгод, $alpha(p,q) < 5$.

Хугацааны шинжилгээ

Теорем 12.2 [Tarjan ба Van Leeuwen]

$T(f,u)$ –г хоорондоо хутгалдсан f хайлт, u нэгдэл үйлдлийн цувааг гүйцэтгэхэд шаардлагатай максимум хугацаа гэж үзье. Гэхдээ $u >= n/2$. Тэгвэл

$$a*(n + f*\mathrm{alpha}(f+n, n)) \le T(f,u) \le b*(n + f*\mathrm{alpha}(f+n, n))$$

үүнд a , b тогтмолууд.

Энэ хязгаарлалтыг ганцаардлагдсан олонлогуудаас эхлээд өндрийн, эсхүл жингийн нэгдэл, хайлтын зам шахах дурын аргыг ашиглах үед хэрэглэнэ.