COURSE : FUNDAMENTALS OF COMPUTER SCIENCE

LECTURE 9: "COMPUTER ALGORITHMS, TYPES AND PERFORMANCE"

Instructor:
PhD, Associate Professor
Tuyatsetseg Badarch

Алгоритм

Алгоритм гэдэг нь монгол хэлний **арга** гэдэг үгтэй төстэй утгатай грек үг юм. Өөрөөр хэлбэл тухайн асуудлыг шийдвэрлэх **зөв** аргын талаар ярьж байгаа хэрэг. Шаардлагатай үр дүнг гаргаж авах зорилгоор зохиож бичсэн, нэгэн утгатай биелэгдэж болох алхам-үйлдлүүдийн төгсгөлөг дарааллыг алхам алхмаар нь гүйцэтгэхэд биелэж төгсдөг бол энэ дарааллыг алгоритм гэнэ.

Алгоритм гэдэг ерөнхийдөө тодорхой нэг төлвөөс зорилгодоо хүрэх хүртлэх алхмуудын дараалалаар тодорхойлогдох юм.

Алгоритмын гол зорилго нь ямар нэг асуудлыг шийдэх/бодлого бодох хамгийн хялбар арга замыг олох, түүнд тохирсон үндсэн томъёоллыг тодорхойлоход оршино.

Судлах зүйл

- Алгоритмыг тодорхойлж, асуудлыг шийдвэрлэхэд ашиглах
- Алгоритмын гурван бүтцийг тодорхойлж ашиглах.
- UML диаграм ба псевдокод, тэдгээрийг алгоритмд хэрхэн ашигладаг талаар
- Үндсэн алгоритмууд болон тэдгээрийн хэрэглээ
- Хайлтын алгоритм
- Давталт ба рекурсив алгоритмуудыг ялгах

Алгоритмын чанарууд

a. Дискрет чанар: Алгоритм нь өмнөх алхмуудын утгаар тодорхой дүрмийн дагуу дараагийн утгыг олох тусдаа алхмуудаас бүтнэ.

b. Тодорхой байх чанар: алгоритмын үйлдэл бүр нь тус тусдаа утгатайгаар ойлгогддог гүйцэтгэгч нь тэдгээрийг ялгаж ойлгодог байх.

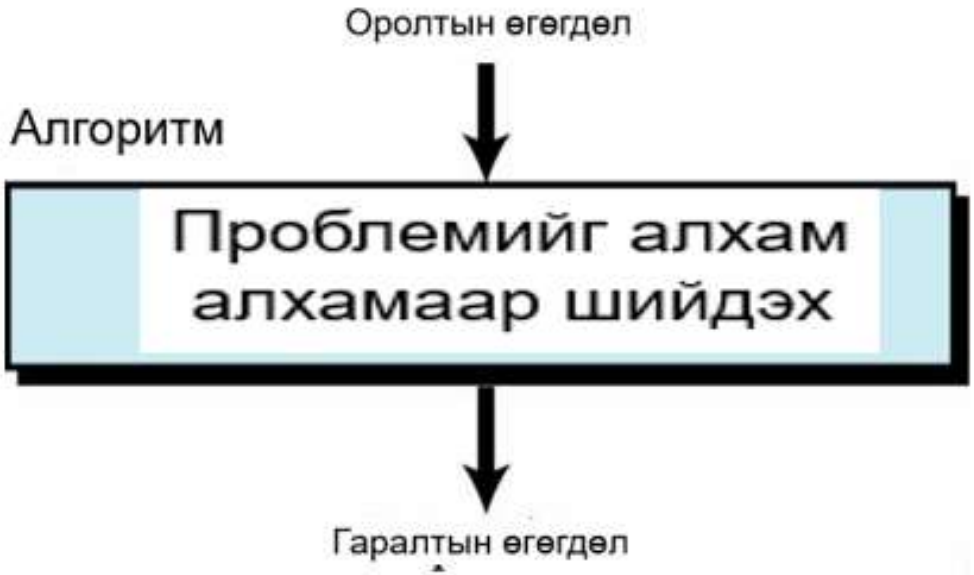
c. Төгсгөлөг байх чанар: Алгоритм нь тухайн өгөгдлөөс хамааран хэдэн ч алхамаар эцсийн зорилгодоо хүрч болох боловч алхам нь төгсгөлөг тооны байна.

d. Үр дүнтэй байх чанар: Алгоритмыг биелүүлхэд заавал тодорхой үр дүн гардаг байх ёстой

e. Нийтлэг чанар: Алгоритм нь тухайн нэг төрлийн бүх өгөгдөлд хүчинтэй байна.

Алгоритмын алхамчлал

Алгоритм: алхам алхмаар асуудлыг шийдвэрлэх эсвэл даалгавар гүйцэтгэх арга.



Зураг 1. Компьютерт хэрэглэгддэг алгоритмын албан бус тодорхойлолт

АЛГОРИТМЫН ИЛҮҮ АЛБАН ЁСНЫ ТОДОРХОЙЛОЛТ

Одоо алгоритмын тухай ойлголтыг дээрх тодорхойлолтуудаас дүгнэж илүү албан ёсны тодорхойлолтыг авч үзье. Алгоритм: Хоёрдмол утгагүй эрэмбэлэгдсэн багц үр дүнг бий болгож, хязгаарлагдмал хугацаанд дуусдаг алхмууд.

Алгоритм:

Үр дүнг гаргаж, хязгаарлагдмал хугацаанд дуусдаг тодорхой бус алхамуудын дараалалт багц юм.

Алгоритмын үндсэн үйлдлүүд

A. Зарлах үйлдэл : Тухайн алгоритмд ямар ямар хувьсагчууд ашиглахыг тодорхойлж зарлаж өгнө.

B. Утга оруулах үйлдэл : Алгоритмын эцсийн зорилго болох утгыг олход мэдээж эхлээд тодорхой хувьсагчуудын утгууд мэдэгдэж байх ёстой бөгөөд тэдгээрийг алгоритмд утга оруулах алхамыг ашиглан оруулж/тодорхойлж/ өгнө.

Алгоритмын үндсэн үйлдлүүд

Утга олгох үйлдэл : Алгоритмд аливаа үйлдлийг хийгээд үр дүнг нь хувьсагчид хадгалдаг. Тухайн хувьсагчид утга олгох үйлдлийг хэлнэ.

D. Хэвлэх үйлдэл : Хэрэгтэй хувьсагчийн утга, тайлбар, алгоритмын үр дүнг харуулахад ашиглана.

E. Нөхцөл шалгах үйлдэл : Ямар нэг нөхцөл биелэх эсэхээс хамааран ямар нэг үйлдэл хийх эсэх, 2 үйлдлийн алийг хийхээ шийддэг тохиолдолд ашиглана.

Алгоритм гүйцэтгэл

Бид эерэг бүхэл тоонуудын жагсаалтаас хамгийн их бүхэл тоог олох алгоритмыг боловсруулах гэж оролдъё.

Алгоритм нь аливаа утгуудын жагсаалтаас хамгийн их бүхэл тоог олох ёстой (жишээлбэл 5, 1000, 10,000, 1,000,000).

Алгоритм нь ерөнхий байх ёстой бөгөөд бүхэл тооны тоо ширхэгээс нь хамаарахгүй.
Энэ асуудлыг шийдэхийн тулд бидэнд зөн совингийн арга хэрэгтэй. Эхлээд цөөн тооны бүхэл тоо (жишээ нь, таван тоо) ашиглах харин дараа нь шийдлийг дурын тооны бүхэл тоо хүртэл сунгах замаар хамгийн ихийг олох (FINDLARGEST)

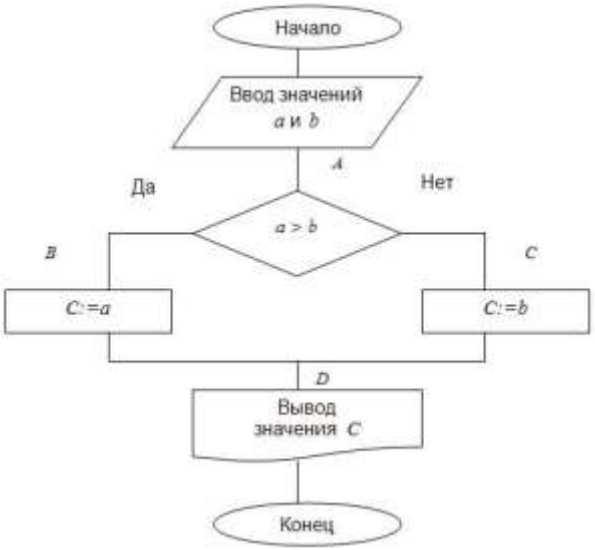
Алгоритм гүйцэтгэл

Бид эерэг бүхэл тоонуудын жагсаалтаас хамгийн их бүхэл тоог олох алгоритмыг боловсруулах гэж оролдъё.

Энэ асуудлыг шийдэхийн тулд бидэнд энгийн сэтгэлгээний арга хэрэгтэй. Эхлээд цөөн тооны бүхэл тоо (жишээ нь, таван тоо) ашиглах харин дараа нь шийдлийг дурын тооны бүхэл тоо хүртэл сунгах замаар хамгийн ихийг олох (FindLargest)

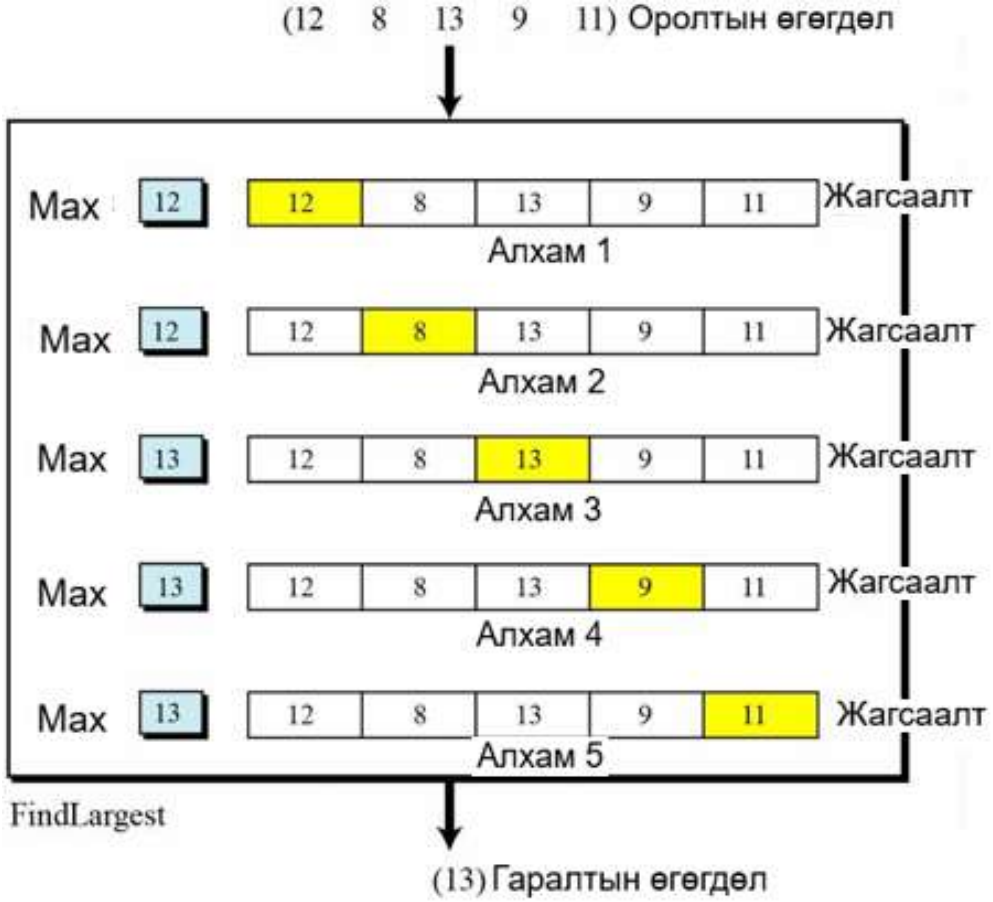
Алгоритм нь аливаа утгуудын жагсаалтаас хамгийн их бүхэл тоог олох ёстой (жишээлбэл 5, 1000, 10,000, 1,000,000).

Алгоритм нь ерөнхий байх ёстой бөгөөд бүхэл тооны тоо ширхэгээс нь хамаарахгүй.



Алгоритм гүйцэтгэл

Зураг 2-т Алгоритм нь таван бүхэл тоонуудын жагсаалтыг оролт болгон хүлээн авч хамгийн их бүхэл тоог гаралтын утгаар авна.

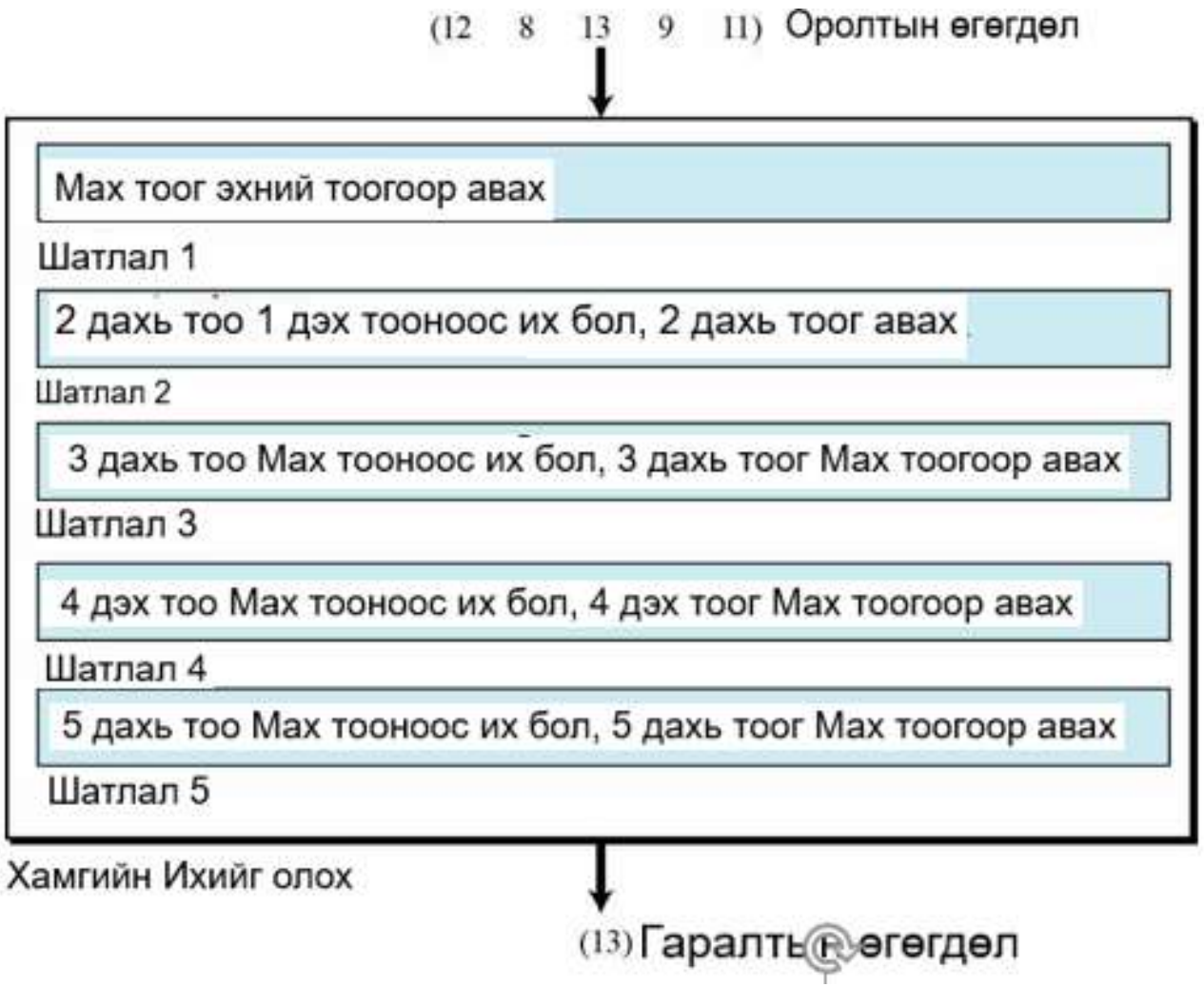


Гаралтын өгөгдөл

Алгоритм эхнээс нь шалгаж эхэлнэ.
Алхам 1. 12 IS THE LARGEST
Алхам 2. 12 >8
Алхам 3. 13 > 12
Алхам 4. 13 > 9
Алхам 5. 13>11

Зураг 2. Алгоритмын гүйцэтгэл

Алгоритм гүйцэтгэл



Алгоритм эхнээс нь шалгаж эхэлнэ.
Алхам 1. 12 IS THE LARGEST
Алхам 2. 12 >8
Алхам 3. 13 > 12
Алхам 4. 13 > 9
Алхам 5. 13>11

Зураг 3 FindLargest алгоритм дахь үйлдлүүдийг тодорхойлох

Алгоритмыг сайжруулах - Refinement

Энэхүү алгоритмын зарчмыг хүлээн зөвшөөрөгдөхүйц байхын тулд боловсронгуй болгох шаардлагатай. Дээрх алгоритмын зарчмын хувьд дараах асуудлууд бүрхэг байна:

1. Эхний алхам алхмуудын хувьд өөр байна.

2. 2-оос 5-р алхам дахь хэллэг нь ижил биш байна.

Алгоритмыг сайжруулах - Refinement

Тухайлбал, Бид 2-оос 5- дугаар алхам дахь хэллэгийг "Хэрэв одоогийн бүхэл тоо Хамгийн их тооноос их бол, Хамгийн их нь одоогийн бүхэл тоо байх болно гэж" гэж өөрчлөх боломжтой. Эхний алхам нь бусад алхмуудаас ялгаатай байгаагийн шалтгаан нь Largest тоогоор эхлээгүй нөхцөлтэй байгаатай холбоотой юм. Хэрэв бид Largest-ийг −∞ (хасах хязгааргүй) болгон эхлүүлбэл эхний алхам нь бусад алхмуудтай ижил байж болох тул бид шинэ алхам нэмж, бүхэл тоо боловсруулахаас өмнө үүнийг хийх ёстой гэдгийг үзүүлэхийн тулд 0 алхам гэж нэрлэнэ.

Зураг 4 **FindLargest сайжруулсан хувилбар**

Үүсгэлт

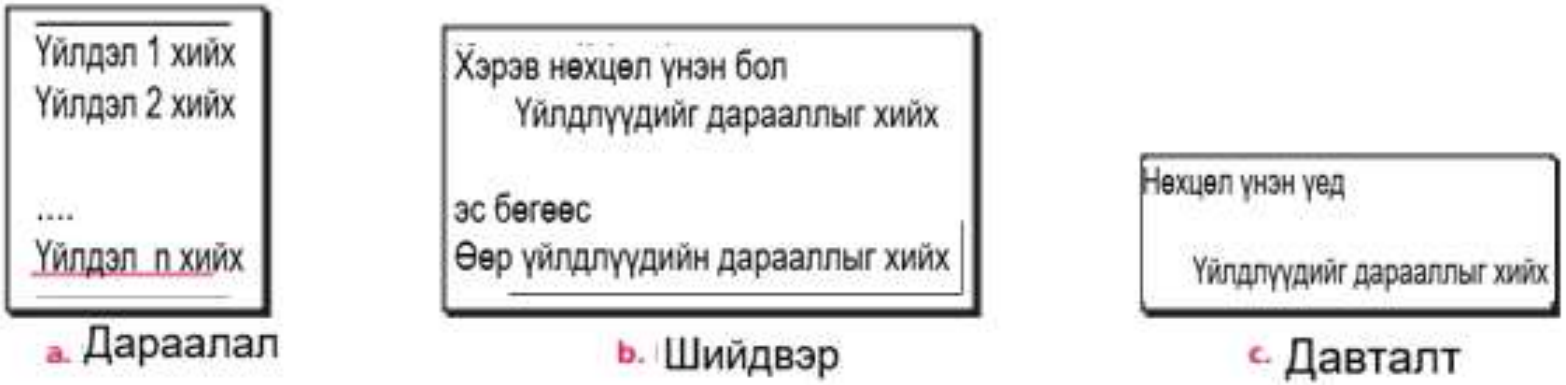
Бид n нь 1000, 1,000,000 ба түүнээс дээш байж болох n эерэг бүхэл тооноос хамгийн их утгатайг нь олох .
Зураг-ийг дагаж алхам бүрийг давтаж болно. Гэхдээ хэрэв бид алгоритмыг програм болгон өөрчилвөл, дараа нь бид n алхамын үйлдлүүдийг бичих хэрэгтэй болно. Үүнийг хийх илүү сайн арга бий. Бид компьютерт алхамуудыг n удаа давтана.

Зураг 5 **Үүсгэлт: FindLargest**

Гурвалсан бүтэц

КУ-ны эрдэмтэд бүтэцлэгдсэн програм буюу алгоритмын гурвалсан бүтцийг тодорхойлсон – Гол санаа нь зөвхөн дараах гурван аргыг хослуулах арга.

1. **sequence -** дараалал
2. **decision** (selection) – шийдвэрийн сонголт
3. **repetition** - давталт



Зураг 6. Гурвалсан бүтэц

Гурвалсан бүтцийн зарчим

Алгоритм нь энгийн заавар –instruction эсвэл бусад хоёр бүтцээс аль нэг нь байж болох зааврууудын дараалал юм.

Decision

Зарим асуудлыг зөвхөн энгийн зааварчилгааны дарааллаар шийдэх боломжгүй. Иймд нөхцөл байдлыг шалгах хэрэгтэй. Туршилтын үр дүн үнэн бол бид дараалсан зааврыг дагана: Хэрэв худал бол бид зааврын өөр дарааллыг дагадаг. Үүнийг шийдвэр (сонголт) бүтэц гэж нэрлэдэг. (***if-else***)

Repetition

Зарим асуудалд ижил дараалал бүхий зааврууд давтагдсан байх ёстой ба үүнийг давталт loop гогцоо бүтээх замаар зохицуулдаг.

Алгоритмын ангилал

Алгоритмын үйлдлүүдийн биелэх дараалал, тооноос хамааран шугаман, салаалсан, давталттай гэж 3 **ангилна**.

f. ***Шугаман*** : Алгоритмын бүх үйлдлүүд зөвхөн нэг удаа биелэдэг.

g. ***Салаалсан*** : Ямар нэгэн хувьсагчийн утгаас хамаарч 2 үйлдлийн аль нэгийг хийдэг.

h. ***Давталттай*** : Өгөгдлийн утгаас хамааран нэг үйлдэл нэг болон түүнээс олон удаа давтагддаг алгоритмыг хэлнэ.

АЛГОРИТМЫН ИЛЭРХИЙЛЭЛ UML

Unified Modeling Language нь алгоритмын зураг дүрслэл юм.
Алгоритмын бүх нарийн ширийн зүйлийг нууж, алгоритм хэрхэн эхнээсээ дуустал урсах “том дүр зургийг" харуулахыг оролддог.
Энд бид зөвхөн гурван бүтцийг UML ашиглан хэрхэн төлөөлдөгийг харуулав (Зураг).

Загварчлалын нэгдсэн хэл (Unified Modeling Language цаашид UML) бол системийн түгээмэл хэрэглэдэг визуал загварчлалын хэл юм. UML -ийг ихэвчлэн өргөтгөх боломжтой объект хандлагат програм хангамжийн системийг загварчлахад өргөн ашигладаг.

Алгоритмыг дүрслэх

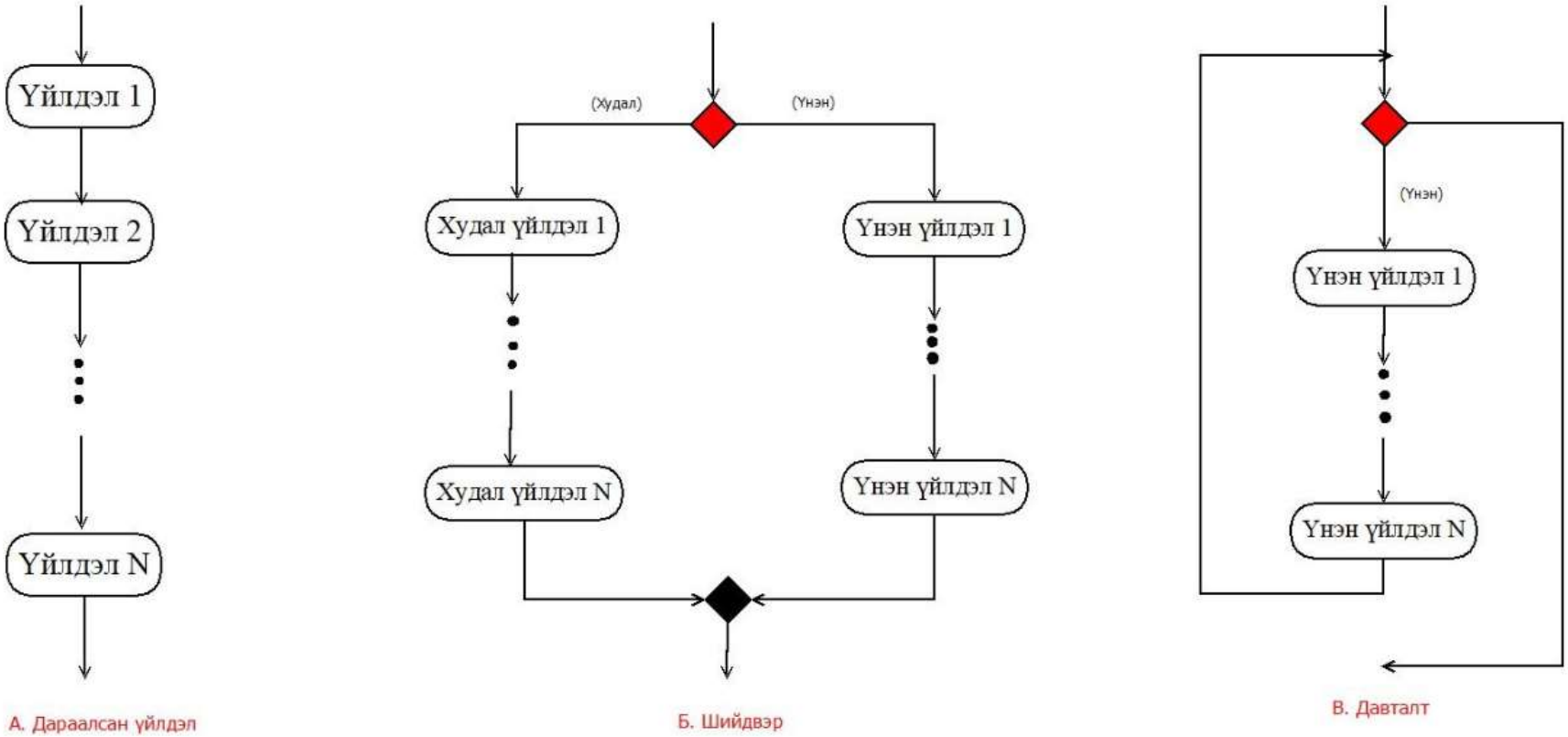
Алгоритмыг дараах хэлбэрээр дүрсэлдэг.

a. ***Ердийн харилцааны хэлээр***: Хүмүүсийн ердийн харилцааны хэл ашиглан алгоритмын алхмуудыг тайлбарлан бичдэг. /Дээр зам гатлах алгоритмыг энэ хэлбэрээр дүрслэв./

b. ***Блок схемээр***: Геометрийн дүрсүүд алгоритмын өөр өөр үйлдлийг илэрхийлэх бөгөөд тэдгээр дүрсүүд дотор тухайн дүрсэд харгалзах үйлдлийг математикийн томъёоллоор бичиж өгдөг. /Үүнийг цаашид дэлгэрэнгүй тайлбарлана./

c. ***Програмчлалын хэлээр***: Програмчлалын ямар нэг хэл ашиглан дүрслэхийг хэлнэ.

АЛГОРИТМЫН ИЛЭРХИЙЛЭЛ



Зураг 7. **Гурвалсан бүтцийг идэрхийлэх UML**

АЛГОРИТМЫН Pseudocode

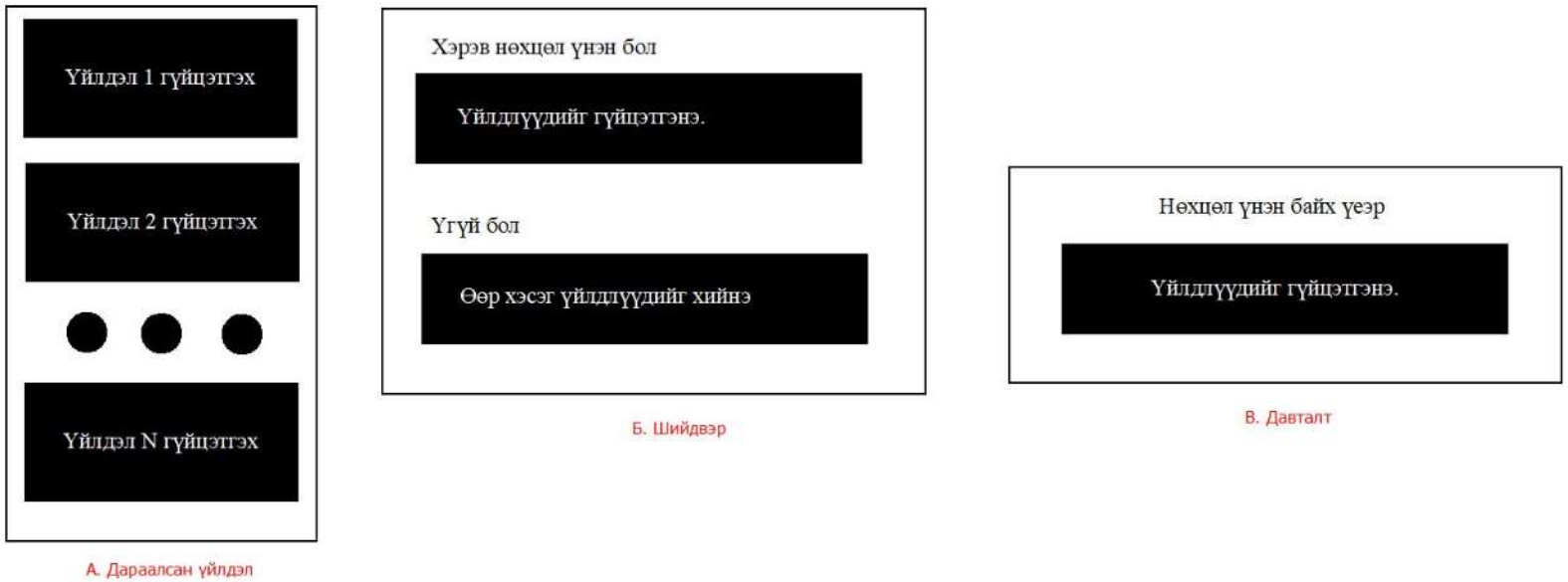
Псевдокод нь алгоритмын англи хэлтэй төстэй дүрслэл юм.

Псевдокодын стандарт байдаггүй—Зарим хүмүүс маш

их нарийн ширийн зүйлийг ашигладаг бол зарим нь бага ашигладаг. Зарим нь англи хэлтэй ойролцоо код ашигладаг бол зарим нь Паскал програмчлалын хэл зэрэг синтакс ашигладаг.

Энд псевдокодоор гурвалсан бүтцийг хэрхэн илэрхийлж болохыг харуулав (Зураг).

АЛГОРИТМЫН Pseudocode



Зураг 8. Pseudocode

ЖИШЭЭ АЛГОРИТМ

Нийлбэр олох

```
Алгоритм: Хоёр тооны нийлбэр (эхнийх, дараагийнх)

Зорилго:  Хоёр бүхэл тооны нийлбэр олох

Урьдчилан: Өгөгдсөн: хоёр бүхэл тоо(эхнийх ба дараагийнх)

Дараа нь: Байхгүй

Буцаах:   Нийлбэрийн утга

{
      sum ← эхнийх + дараагийнх
    return   sum
}
```

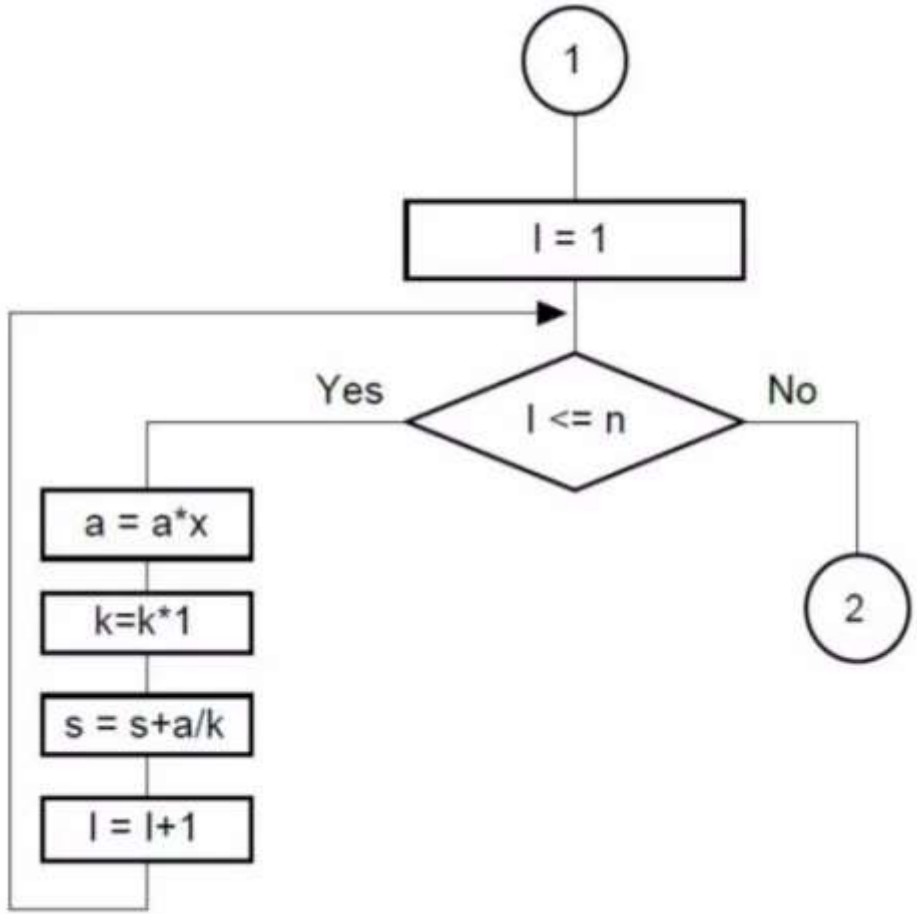
Жишээ 9. Хоёр бүхэл тооны нийлбэрийг олох алгоритмыг псевдокодоор бичих.

ЖИШЭЭ АЛГОРИТМ

```
Алгоритм:  Тэнцэх/Тэнцэхгүй (оноо)

Зорилго:   Онооноос хамааран тэнцсэн/тэнцээгүй дүн гаргана

Урьдчилан: Өгөгдсөн: дүн болгох оноо

Дараа нь: Байхгүй

Буцаах:    Дүн

{
      if    (score >= 70)
            grade ← "тэнцсэн"
      else
            grade ← "тэнцээгүй"
      return grade

}
```

Жишээ 10. Тоон дүнг тэнцсэн/үгүй гэж өөрчлөх алгоритмыг бичнэ үү

ЖИШЭЭ АЛГОРИТМ



Жишээ 11. Тоон дүнг тэнцсэн/үгүй гэж өөрчлөх алгоритмыг бичнэ үү

АЛГОРИТМЫН ИЛҮҮ АЛБАН ЁСНЫ ТОДОРХОЙЛОЛТ

1. **Well-Defined :** Алгоритм нь нарийн тодорхойлогдсон, дараалсан заавар байх ёстой.

2. **Unambiguous steps**: Алгоритм дахь алхам бүрийг тодорхой бөгөөд хоёрдмол утгагүй тодорхойлсон байх ёстой.

Хэрэв нэг алхамд хоёр бүхэл тоог хооронд нэмэх гэж байгаа тохиолдолд бид "бүхэл тоо" болон "нэмэх" үйлдлийг хоёуланг нь тодорхойлох ёстой: жишээ нь ижил тэмдгийг нэг алхамд тоог нэмэхийн тулд ашиглаад өөр алхамд үржүүлэх үйлдэлд ашиглаж болохгүй.

АЛГОРИТМЫН ИЛҮҮ АЛБАН ЁСНЫ ТОДОРХОЙЛОЛТ

3. Produce a result : Алгоритм үр дүнг заавал гаргах ёстой, эс бөгөөсс энэ нь үр ашиггүй болно. Үр дүн нь нөхцөл дуудах, алгоритм руу буцаж ирсэн өгөгдөл эсвэл бусад үр дүн (жишээлбэл, хэвлэх) байж болно.

4. Terminate in a finite time: Алгоритм заавар дуусгавар болох ёстой (halt). Хэрэв тийм биш (infinite loop) бол тус алгоритм зохиогдоогүй байна гэж үзнэ.

ҮНДСЭН АЛГОРИТМУУД

Компьютерийн шинжлэх ухаанд хэд хэдэн алгоритмууд түгээмэл хэрэгдэгддэг учраас тэдгээр алгоримтуудыг ҮНДСЭН алгоритмууд гэдэг.

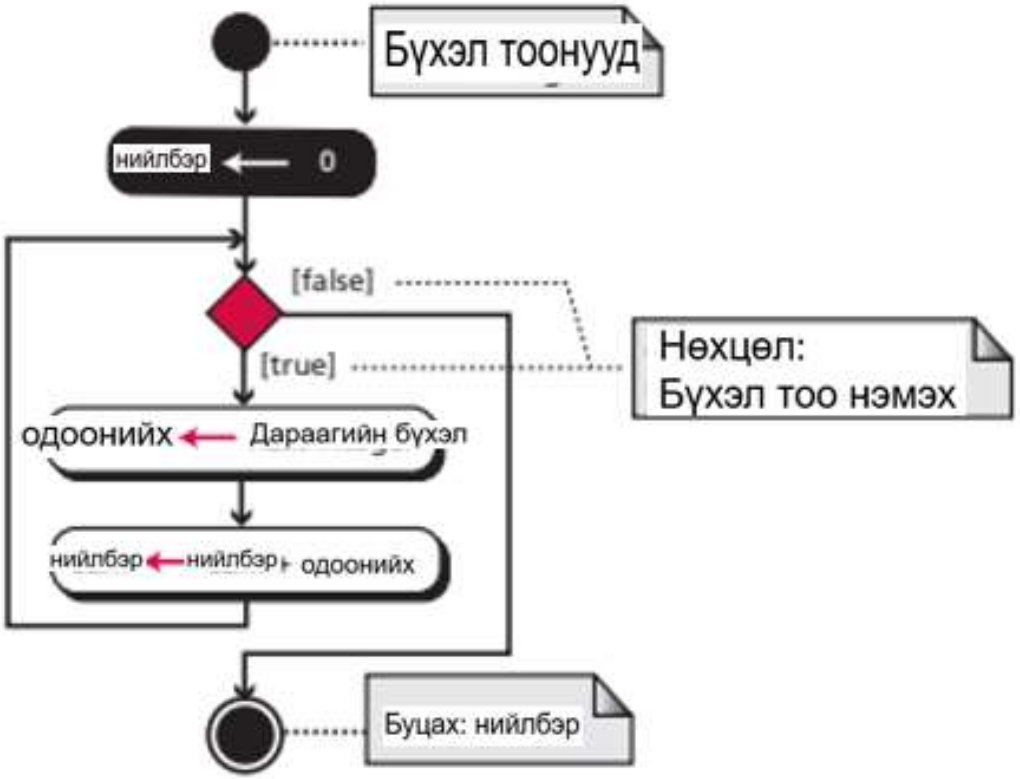
НИЙЛБЭРИЙН АЛГОРИТМ

Компьютерийн шинжлэх ухаанд түгээмэл хэрэглэгддэг алгоритмуудын нэг бол нийлбэр алгоритм юм. Бид хоёр, гурван бүхэл тоог маш амархан нэмэх боломжтой, гэхдээ яаж олон тооны бүхэл тоог нэмэх вэ? Шийдэл нь маш энгийн: бид нэмэх операторыг давталтад ашигладаг (Зураг 8.9).

Нийлбэрийн алгоритм нь гурван логик хэсэгтэй:

1. Эхэнд нь нийлбэрийг эхлүүлэх.
2. Давталт бүрт нийлбэр дээр шинэ бүхэл тоо нэмдэг /давталт/.
3. Гогцооноос /loop/ гарсны дараа үр дүнг буцаах.

ҮНДСЭН АЛГОРИТМУУД



1. Эхэнд нь нийлбэрийг эхлүүлэх.
2. Давталт бүрт нийлбэр дээр шинэ бүхэл тоо нэмдэг /давталт/.
3. Гогцооноос /loop/ гарсны дараа үр дүнг буцаах.

Зураг 12. Нийлбэрийн алгоритм

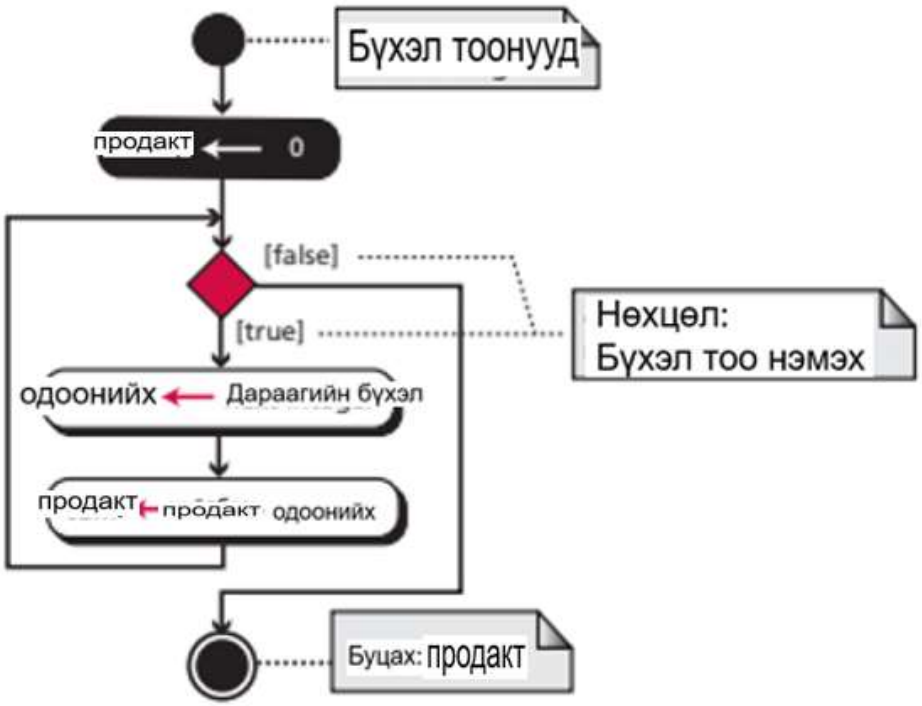
ҮНДСЭН АЛГОРИТМУУД

Бид хоёр, гурван бүхэл тоог маш амархан үржүүлэх боломжтой, гэхдээ яаж олон тооны бүхэл тоог үржүүлэх вэ? Шийдэл нь маш энгийн: бид үржүүлэх операторыг давталтад ашигладаг (Зураг 8.9).

Үржүүлгийн алгоритм нь гурван логик хэсэгтэй:

1. Эхэнд нь үржүүлгийг эхлүүлэх.
2. Давталт бүрт үржүүлгийг шинэ бүхэл тоогоор үржүүлнэ /давталт/.
3. Гогцооноос /loop/ гарсны дараа үр дүнг буцаах.

ҮНДСЭН АЛГОРИТМУУД



Зураг 13. Үржүүлэх алгоритм

1. Эхэнд нь үржүүлгийг эхлүүлэх.
2. Давталт бүрт үржүүлгийг шинэ бүхэл тоогоор үржүүлнэ /давталт/.
3. Гогцооноос /loop/ гарсны дараа үр дүнг буцаах.

Хамгийн бага, хамгийн их тоог олох

Энэ бүлгийн эхэнд бүхэл тоонуудын жагсаалтаас хамгийн ихийг нь олох алгоритмыг бид ярилцсан. Үүний зорилго нь хоёр бүхэл тооноос ихийг олох шийдвэрийн бүтцийг бичих явдал байв. Хэрэв бид энэ бүтцийг гогцоонд /loop/ оруулбал бүхий л бүхэл тоонуудын жагсаалтаас хамгийн ихийг нь олох боломжтой.

Бүхэл тоонуудын жагсаалтаас хамгийн бага бүхэл тоог олох нь ижил төстэй бөгөөд хоёр ялгаа байна.

1. Хоёр бүхэл тооноос бага хэсгийг олохын тулд шийдвэрийн бүтцийг ашиглана.
2. Маш бага биш харин маш их бүхэл тоогоор эхлүүлнэ.

Эрэмбэлэх АЛГОРИТМ /Sort/

Компьютерийн шинжлэх ухаанд хамгийн түгээмэл хэрэглэгддэг програмуудын нэг бол эрэмбэлэх бөгөөд энэ нь өгөгдлийг утгын дагуу байрлуулах процесс юм.

Энэ хэсэгт бид гурван ангилах алгоритмыг танилцуулж байна:

/selection sort/

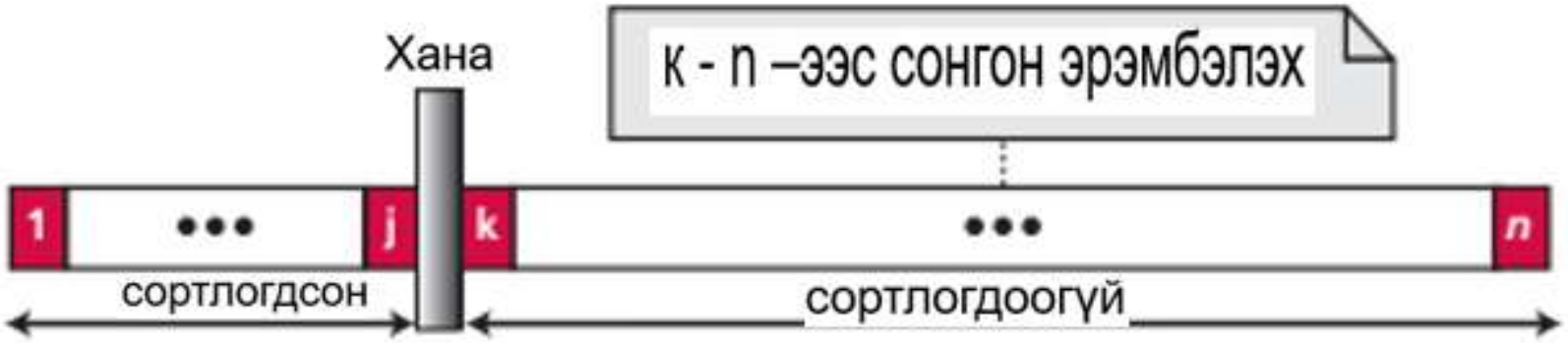
/bubble sort/

/insertion sort/

Эдгээр гурван эрэмбэлэх алгоритмууд нь өнөө үед компьютерийн шинжлэх ухаанд хэрэглэгдэж буй хурдан ялгах алгоритмуудын үндэс суурь болжээ.

Сонгон эрэмбэлэх АЛГОРИТМ /Sort/

Сонгон ангилахдаа эрэмбэлэгдэх жагсаалтыг хоёр дэд жагсаалтад хуваадаг. Эрэмблэгдээгүй дэд жагсаалтаас хамгийн бага элементийг олж, эрэмблэгдээгүй дэд жагсаалтын эхэнд байгаа элементтэй солино. Сонгох, солих бүрийн дараа хоёр дэд жагсаалтын хоорондох төсөөллийн хана нь нэг элементийг урагшлуулдаг.



Зураг 14. Сонгон эрэмблэлт

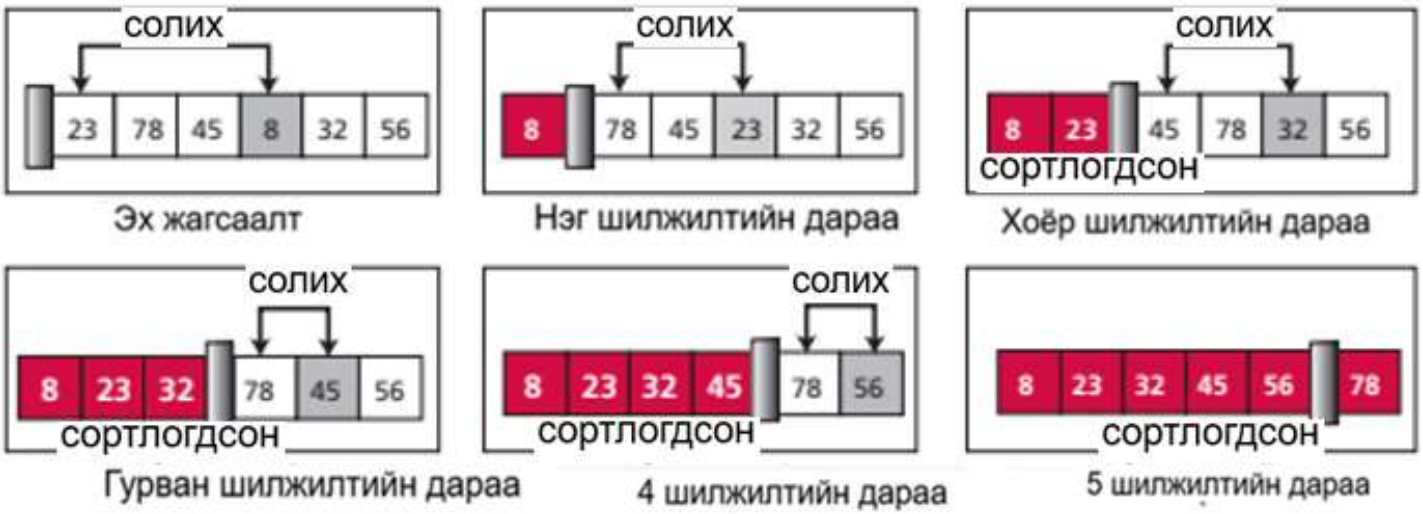
Эрэмбэлэх АЛГОРИТМ /Sort/

Сонгон эрэмбэлэх арга нь массивын хамгийн бага элементийг олж хамгийн эхний элементтэй сольж, дараа нь удаах хамгийн бага элементийг олж хоёрдох элементтэй солих байдлаар массивын бүх элмент эрэмбэлэгдэж дуустал гүйцэтгэгдэх арга юм.

Хэрэв буурахаар эрэмбэлэх бол хамгийн их элементийг сонгож солино гэсэн үг. Энэ арга нь маш амархан, эсрэгээрээ хурдны хувьд тийм ч сайн арга биш. Ямар ч төрлийн бага өгөгдөлтэй массивын

Эрэмбэлэх АЛГОРИТМ /Sort/

Сонгон эрэмбэлэхэд жагсаалтыг хоёр дэд жагсаалтад хуваадаг. Эрэмблэгдээгүй дэд жагсаалтаас хамгийн бага элементийг олж, эрэмблэгдээгүй дэд жагсаалтын эхэнд байгаа элементтэй солино /8 then change the location. 23 then change the location …/. Сонгох, солих бүрийн дараа хоёр дэд жагсаалтын хоорондох төсөөллийн хана нь нэг элементийг урагшлуулдаг.



Зураг 15. Сонголтыг эрэмбэлэх жишээ

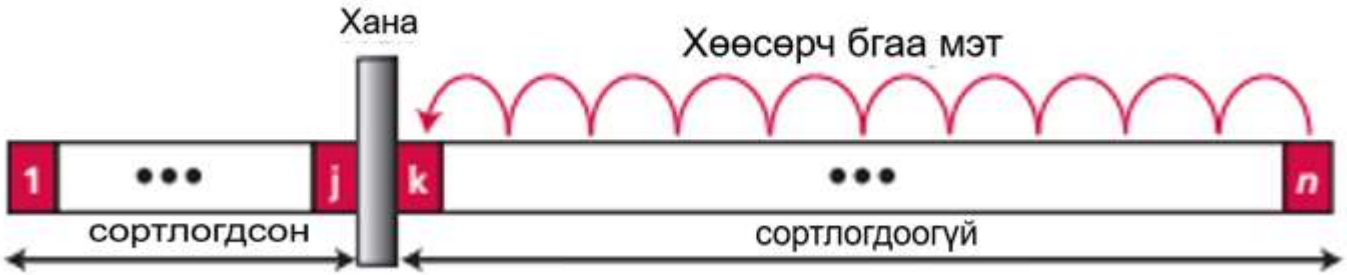
BubbleАЛГОРИТМ /Sort/

Бөмбөлөг эрэмбэлэх аргад эрэмбэлэгдэх жагсаалтыг мөн хоёр дэд жагсаалтад хуваадаг. Хамгийн бага элементийг эрэмбэлэгдээгүй дэд жагсаалтаас хөөсөрч эрэмбэлсэн дэд жагсаалтад шилжүүлнэ. Хамгийн бага элементийг эрэмбэлэгдсэн жагсаалтад шилжүүлсний дараа хана нэг элементийг урагшлуулна.

Энэ аргыг бөмбөлөгөөс гадна хөөсрүүлэн эрэмбэлэх арга ч гэж нэрлэдэг. Массивын элементүүдийг эхнээс нь зэрэгцээ байгаа хоёр элементийг жишиж өмнөх буюу индексээрээ бага элемент нь хойно байгаа буюу индексээрээ их элементээс их байвал (бууруухаар эрэмбэлэх тохиолдолд бага байвал) энэ хоёр элемэнтийн байрыг солих замаар массивын төгсгөл хүртэл үргэлжлүүлнэ. Тухайлбал, a[i] > a[i+1] бол энэ хоёрын байрыг солино гэсэн үг. Эсрэг тохиолдолд хэвээр үлдээнэ. Ингэсэнээр хамгийн их элемент массивын хамгийн сүүлд байрлана. Дараа нь үлдсэн n-1 элементэд мөн энэ үйлдлийг давтаж, нэг элемент үлдтэл (n==1 болтол) хийнэ.

BubbleАЛГОРИТМ /Sort/

Зэрэгцээ байгаа хоёр элементийг жишиж өмнөх буюу индексээрээ бага элемент нь хойно байгаа буюу индексээрээ их элементээс их байвал (бууруулхаар эрэмбэлэх тохиолдолд бага байвал) энэ хоёр элемэнтийн байрыг солих замаар массивын төгсгөл хүртэл үргэлжлүүлнэ. Тухайлбал, a[j] > a[j+1] бол энэ хоёрын байрыг солино гэсэн үг. Эсрэг тохиолдолд хэвээр үлдээнэ. Ингэсэнээр хамгийн их элемент массивын хамгийн сүүлд байрлана. Дараа нь үлдсэн n-1 элементэд мөн энэ үйлдлийг давтаж, нэг элемент үлдтэл (n==1 болтол) хийнэ.



Зураг 16. Bubble эрэмбэлэх жишээ

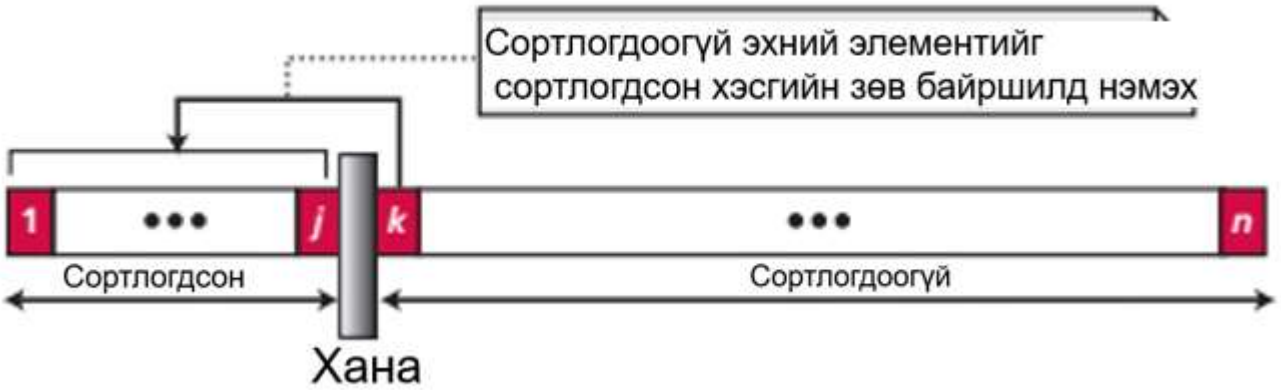
BubbleАЛГОРИТМ /Sort/

Зураг 17. bubble sort-ийн жишээ

Хамгийн бага элементийг эрэмбэлэгдсэн жагсаалтад шилжүүлсний дараа хана нэг элементийг урагшлуулна. a[j] > a[j+1] бол энэ хоёрын байрыг солино гэсэн үг. Эсрэг тохиолдолд хэвээр үлдээнэ. Ингэсэнээр хамгийн их элемент массивын хамгийн сүүлд байрлана. Дараа нь үлдсэн n-1 элементэд мөн энэ үйлдлийг давтаж, нэг элемент үлдтэл (n==1 болтол) хийнэ. Энэ эрэмбэлэх аргад эрэмбэлэгдэх жагсаалтыг мөн хоёр дэд жагсаалтад хуваадаг. Хамгийн бага элементийг эрэмбэлэгдээгүй дэд жагсаалтаас хөөсөрч эрэмбэлсэн дэд жагсаалтад шилжүүлнэ.

Insertion sort АЛГОРИТМ /Sort/

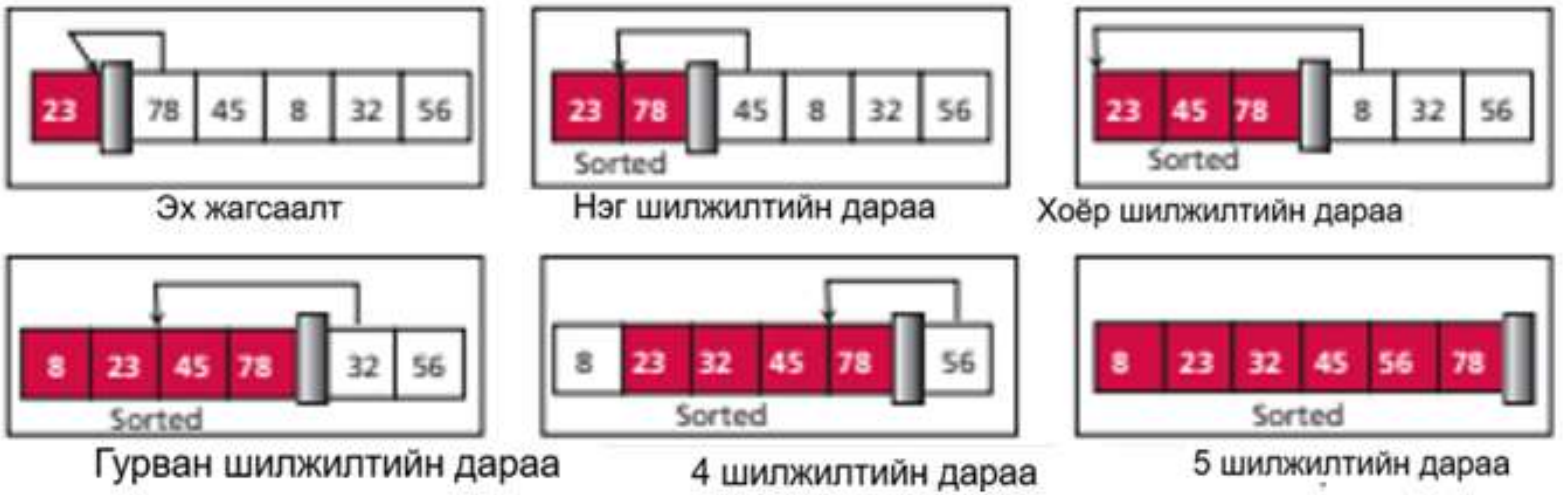
Оруулах эрэмбийн алгоритм нь эрэмблэх хамгийн түгнээмэл аргуудын нэг бөгөөд үүнийг картын тоглогчид ихэвчлэн ашигладаг. Тоглогчийн авсан карт бүрийг тодорхой дарааллыг хадгалахын тулд картынхаа гарт зохих газарт нь байрлуулдаг.



Зураг 18. **Insertion sort-ийн жишээ**

Insertion sort АЛГОРИТМ /Sort/

Тоглогчийн авсан карт бүрийг тодорхой дарааллыг хадгалахын тулд картынхаа гарт зохих газарт нь байрлуулдаг.



Зураг 19. **Insertion sort-ийн жишээ**

Searching АЛГОРИТМ

Түгээмэл алгоритмуудын нэг болох хайлтын алгоритм нь жагсаалтаас зорилтот байршлыг олох үйл явц юм.

Жагсаалтын хувьд хайлт хийх нь утга өгсөн гэсэн үг бөгөөд тухайн утгыг агуулсан жагсаалтын эхний элементийн байршлыг олохыг хүсч байна.

Жагсаалтыг хоёр үндсэн хайлт байдаг: дараалсан хайлт ба хоёртын хайлт. Дараалсан хайлтыг ямар ч жагсаалтад байгаа зүйлийг олоход ашиглаж болох бол хоёртын хайлт нь жагсаалтыг эхлээд эрэмбэлэхийг шаарддаг .

Sequential search АЛГОРИТМ

Ерөнхийдөө бид энэ аргыг зөвхөн жижиг жагсаалт эсвэл тэр бүр хайгаад байдаггүй жагсаалтад л ашигладаг. Бусад тохиолдолд хамгийн сайн арга бол жагсаалтыг эхлээд эрэмбэлээд дараа нь хэлэлцсэн хоёртын хайлтыг ашиглан хайх явдал юм.

Хаялт хийх жагсаалтыг захиалаагүй тохиолдолд дараалсан хайлтыг ашигладаг.

Дараалсан хайлтаар бид зорилгоо жагсаалтын эхнээс хайж эхэлнэ. Бид зорилгоо олох эсвэл жагсаалтын төгсгөлд хүрэх хүртэл үргэлжлүүлнэ.

Түгээмэл хайлтын АЛГОРИТМУУД

Зураг 20. **Insertion sort-ийн жишээ**

Хоёртын хайлт АЛГОРИТМ

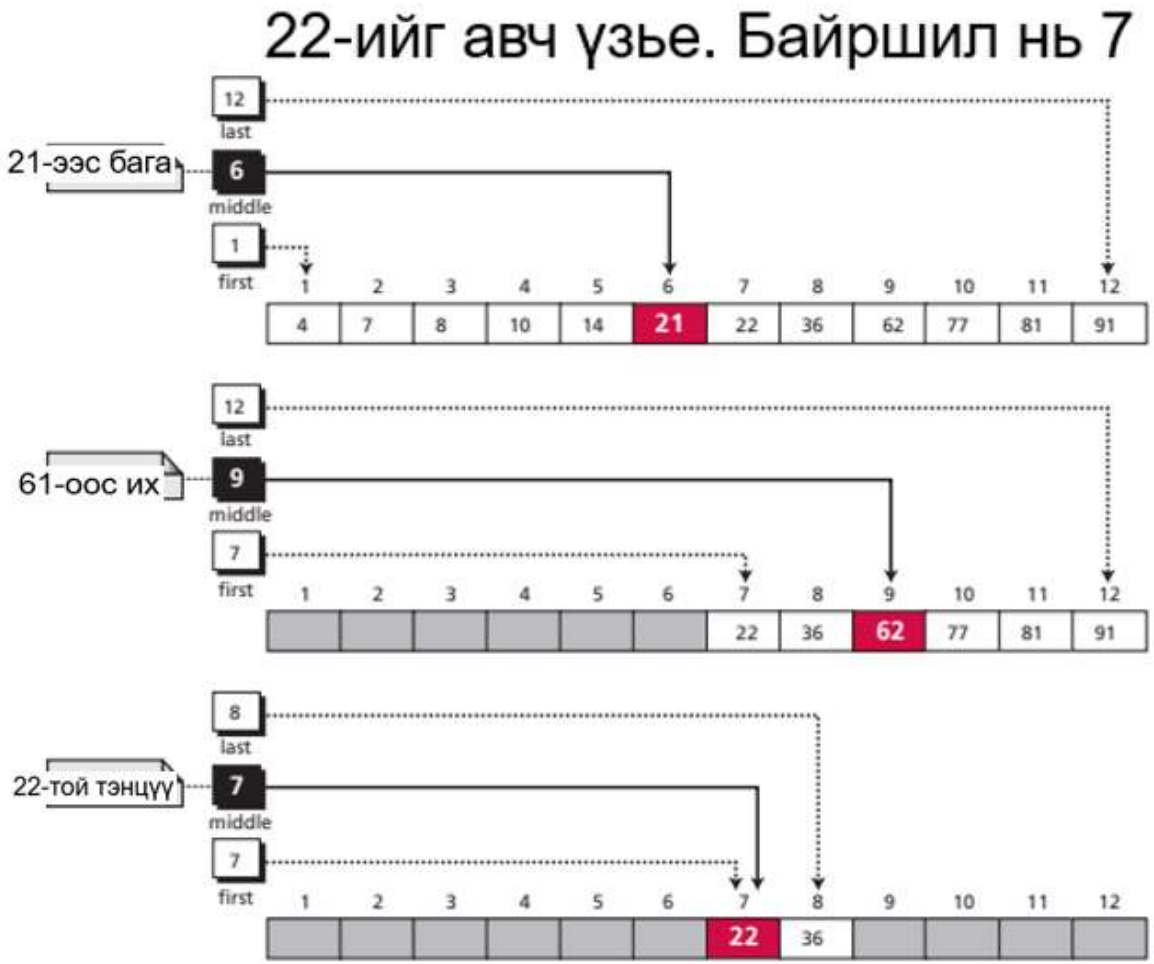
Хэрэв жагсаалтыг эрэмбэлсэн бол бид хоёртын хайлт хэмээх илүү үр дүнтэй алгоритмыг ашиглаж болно.

Хоёртын хайлт нь жагсаалтын дунд байгаа элемент дэх өгөгдлийг туршиж эхэлнэ. Энэ нь зорилтот жагсаалтын эхний хагаст эсвэл хоёрдугаар хагаст байгаа эсэхийг тодорхойлдог .

Хэрэв энэ нь эхний хагаст байгаа бол хоёр дахь хагасыг цаашид шалгах шаардлагагүй болно. Хэрэв энэ нь хоёрдугаар хагаст байгаа бол эхний хагасыг цаашид шалгах шаардлагагүй болно. Өөрөөр хэлбэл, бид жагсаалтын талыг цаашид хэлэлцэхээс хасах болно.

Энэ төрлийн хайлтын алгоритм нь эрэмбэлэгдсэн массив дахь тодорхой утгын байрлалыг олоход хэрэглэгддэг. Хоёртын хайлтын алгоритм нь divide and conquer зарчмаар ажилладаг бөгөөд ажиллахад илүү хурдан байдаг тул хайлтын хамгийн сайн алгоритм гэж тооцогддог

Хоёртын хайлтын АЛГОРИТМ



Зураг 21. Хоёртын хайлтын алгоритмын жишээ

Бидэнд өсөхөөр эрэмбэлэгдсэн N урттай дараалал байгаа гэж бодъё. Мөн Q ширхэг хүсэлт өгөгдөх ба хүсэлт бүрд нэг тоо байх ба энэ дараалалд тэр тоо байгаа эсэхэд хариулах юм.

SUBALGORITHMS ДЭД АЛГОРИТМУУД

Алгоритмчлалын гурвалсан бүтэц нь аливаа шийдвэрлэх боломжтой алгоритмийг бий болгох боломжийг бидэнд олгодог.

Гэхдээ бүтэцлэгдсэн програмчлалын зарчмууд нь алгоритмыг дэд алгоритм гэж нэрлэгддэг жижиг нэгжүүдэд хуваахыг шаарддаг. Дэд алгоритм бүрийг эргээд жижиг дэд алгоритмуудад хуваадаг. Сайн жишээ бол Зураг дахь сонголтыг эрэмбэлэх алгоритм юм .

РЕКУРСИВ НӨХЦӨЛ

Алгоритмчлалын аргаар аливаа асуудлын шийдэл хийх хоёр үндсэн хандлага байдаг.
Нэг. Давталт /iteration/
Хоёр. Өөрөө өөрийгээ дуудах рекурсив нөхцөл /recursion/

РЕКУРСИВ НӨХЦӨЛ

Энгийн жишээг судлахын тулд факториалын тооцоог анхаарч үзээрэй. Бүхэл тооны факториал бол 1-ээс бүхэл тоон хүртэлх салшгүй утгуудын үржвэр юм. Тодорхойлолт нь давталттай (Зураг 8.22). Тодорхойлолт нь алгоритмыг өөрөө агуулаагүй тохиолдолд алгоритм нь давталттай байдаг.

$$\text{факториал}\,(n) = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n-1) \times (n-2) \cdots 3 \times 2 \times 1 & \text{if } n > 0 \end{cases}$$

Зураг 22. Факториал

РЕКУРСИВ НӨХЦӨЛ

Энгийн жишээг судлахын тулд факториалын тооцоог анхаарч үзээрэй.

Зураг 23. Факториалын тооцоо

АШИГЛАСАН МАТЕРИАЛ

Foundations of Computer Science, Behrouz A. Forouzan, Fourth Edition, Cengage Learning EMEA, 2018

Tuyatsetseg Badarch, "Data communications and computer networking" , third edition, 2014.

Pressman, R. Software Engineering: A Practitioner’s Approach, Нью-Йорк: McGraw-Hill,
2005 он

АНХААРАЛ ХАНДУУЛСАНД БАЯРЛАЛАА