



Алгоритмын шинжилгээ ба зохиомж

F.CSM301 Намар

Д. Батмөнх

ШҮТИС, МХТС

2025 оны 11-р сарын 19

Хичээлийн удирдамж

Хичээлийн удирдамж

Ашиглах сурх бичиг:

- ▶ Алгоритмын зохиомж ба шинжилгээ (гурав дахь хэвлэл),
Томас Кормен, Чарльз Лейзерсон, Роналд Ривест,
Клиффорд Штайн
- ▶ Introduction to Algorithms (fourth Edition) by Thomas
H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford
Stein

Хичээлийн удирдамж

Ашиглах цахим хуудас:

- ▶ LeetCode
- ▶ Codeforces

Хичээлийн удирдамж

Хичээлийн үнэлгээ:

Ирц	5 оноо	Лаборатори	
Идэвх	5 оноо	Linux орчин бэлтгэх	
Бие даалт 1	10 оноо	09/25–10/17	V–VIII
Бие даалт 2	10 оноо	11/13–12/05	XI–XIV
Явцын сорил 1	10 оноо	10/13	VIII
Явцын сорил 2	10 оноо	11/24	XIII
Лаборатори	20 оноо	16 даалгавар	
Шалгалт	30 оноо	12/22	XVII

Хичээлийн удирдамж

Бусад үнэлгээ:

Дүн гуйвал	-5 оноо
Дүн гуйлгавал	-8 оноо

Хичээлийн удирдамж

Бие даалтын үед холбогдох суваг:

<https://meet.jit.si/FCSM301-2025>

Хичээлийн удирдамж

Даалгавар гүйцэтгэх хэл:

- ▶ C++
- ▶ Java
- ▶ Python

Хичээлийн удирдамж

Онлайн платформ ашиглахгүй тохиолдолд туршилтын өгөгдөл бэлтгэж, unit testing framework ашиглана:

- ▶ C++ бол Catch2
- ▶ Java бол JUnit
- ▶ Python бол unittest

Хичээлийн удирдамж

Бие даалтын ажлын гүйцэтгэлийг үнэлэх:

- ▶ Баримт бичиг бэлтгэсэн байдал
- ▶ Үгийн алдаа
- ▶ Агуулгыг өөрийн үгээр найруулан бичсэн эсэх
- ▶ Судлагдахууныг хир гүнзгий ойлгосон эсэх

Тооцоололд гүйцэтгэх Алгоритмын Үүрэг

Алгоритм

Алгоритм

Алгоритм—оролтын утгыг тогтоосон хугацаанд боловсруулан гаргах нарийвчлан боловсруулсан тооцооллын алхмуудын дараалал.

Алгоритм

Эрэмбэлэх бодлогын ерөнхий тохиолдол

Оролт n ширхэг тоон дараалал $\langle a_1, a_2, \dots, a_n \rangle$

Гаралт Оролтын өгөгдлийн $a'_1 \leq a'_2 \leq \dots \leq a'_n$ байх
сэлгэлт $\langle a'_1, a'_2, \dots, a'_n \rangle$

Алгоритм

Тухайн тохиолдол

Оролт $\langle 31; 41; 59; 26; 41; 58 \rangle$

Гаралт $\langle 26; 31; 41; 41; 58; 59 \rangle$

Алгоритм

Ямар төрлийн бодлого алгоритмаар бодогдох вэ?

- ▶ Ерөнхий шийдийн ажиллах хугацаа нь олон гишүүнт буюу полином хугацаанд ажилладаг:
 - ▶ Эрэмбэлэлт
 - ▶ Хайлт
 - ▶ Богино зам
 - ▶ Фиbonаччийн дараалал
 - ▶ Энгийн тоон алгоритм

Алгоритм

Ямар төрлийн бодлого алгоритмаар бодогдох вэ?

- ▶ Биологийн бодлогууд
- ▶ Интернет хайлт
- ▶ Цахим худалдаа хийх
- ▶ Бараа түгээх
- ▶ Онош тогтоох
- ▶ Θгөгдөл шахах

Алгоритм

Харин ямар төрлийн бодлогыг алгоритмчлах боломжгүй вэ?

- ▶ Математикийн шийдэгдээгүй асуудал
- ▶ Таамаглал
- ▶ Гоозүй шаардсан асуудал

Алгоритм

Ямар төрлийн бодлогыг алгоритмчлах боломжгүй вэ?

- ▶ NP-Complete (NP + NP-Hard) problem (ерөнхий тохиолдол):
 - ▶ Knapsack Problem: Тодорхой багтаамжтай уутанд тодорхой үнэ бүхий зүйлсийг багтаах
 - ▶ Hamiltonian Cycle: Графын бүх оройг нэг удаа дайрах зам байгаа эсэх
 - ▶ Travelling Salesman Problem: Хамгийн богино замаар бүх хотыг нэг удаа дайрах боломжтой эсэх

Алгоритм

Ямар төрлийн бодлогыг алгоритмчлах боломжгүй вэ?

- ▶ NP-Hard problem (ерөнхий тохиолдол)
 - ▶ Зогсолтын бодлого (Halting problem): өгөгдсөн программ болон оролтод тухайн программ ажиллах эсэхийг шийдэх ерөнхий алгоритм олдохгүй
 - ▶ Traveling Salesman Problem
 - ▶ Knapsack Problem
 - ▶ Hamiltonian Cycle
 - ▶ Subset Sum: Тооны олонлог нь тодорхой нийлбэртэй тэнцэх

Алгоритм

Өгөгдлийн бүтэц:

- ▶ Өгөгдлийг санах ойд хуваарилна
- ▶ Улмаар тэдгээр өгөгдөл хандаж, тооцоолол хийж, өөрчилнө

Алгоритм—технологийн хувьд

Алгоритм

Техникийн хувьд:

- ▶ Компьютерын хүчин чадал өссөөр байгаа ч
- ▶ Гүйцэтгэх хурд болон хугацаа хязгаартай
- ▶ Тооцоолох хугацаа болон санах ойг зэрэг сайжруулах боломжгүй
- ▶ Аль нэгийг зайлшгүй сонгох шаардлагатай

Алгоритм

Бүтээмж (efficiency):

- ▶ 1 бодлогыг олон аргаар (алгоритмаар) бодоход зарцуулах нөөц (санах ой, гүйцэтгэлийн хугацаа) харилцан адилгүй
- ▶ Их өгөгдлийн хувьд энэ ялгаа улам бүр ихэснэ

Алгоритм

Жишээлбэл,

- ▶ insertion sort: $c_1 n^2$
- ▶ merge sort: $c_2 n \lg n$

Алгоритм

n хувьсагчийн багахан утгад:

- ▶ insertion sort: $n = 1,000$
- ▶ merge sort: $\lg 1,000 \approx 10$

Өсгөвөл

- ▶ insertion sort: $n = 1,000,000$
- ▶ merge sort: $\lg 1,000,000 \approx 20$

Алгоритм

Өөр нэгэн жишээ: 10 сая өгөгдлийг дараах байдлаар эрэмбэлье

- ▶ insertion sort: туршлагатай программ зохиогч доод төвшний хэл ашиглана
- ▶ merge sort: дундаж төвшний программ зохиогч өндөр төвшний хэл ашиглана

Алгоритм

Нийт зарцуулах хугацаа:

- ▶ insertion sort: 20,000 секунд (5.5 цаг)
- ▶ merge sort: 1,163 секунд (20 минут)

Үдиртгал

Зөөх эрэмбэлэлт

Зөөх эрэмбэлэлт

Өмнө зөөх эрэмбэлэлтийн аргыг (insertion sort) дурдсан:

Оролт: n тооны дараалал $\langle a_1, a_2, \dots, a_n \rangle$

Гаралт: Оролтын өгөгдлийн $a'_1 \leq a'_2 \leq \dots \leq a'_n$ байх сэлгэлт
 $\langle a'_1, a'_2, \dots, a'_n \rangle$

Зөөх эрэмбэлэлт

Insertion-Sort(A, n)

```
1  for  $i = 2$  to  $n$ 
2       $key = A[i]$ 
3       $j = i - 1$ 
4      while  $j > 0$  and  $A[j] > key$ 
5           $A[j + 1] = A[j]$ 
6           $j = j - 1$ 
7       $A[j + 1] = key$ 
```

Зөөх эрэмбэлэлт

(a)

1	2	3	4	5	6
5	2	4	6	1	3



(b)

1	2	3	4	5	6
2	5	4	6	1	3



(c)

1	2	3	4	5	6
2	4	5	6	1	3



(d)

1	2	3	4	5	6
2	4	5	6	1	3



(e)

1	2	3	4	5	6
1	2	4	5	6	3



(f)

1	2	3	4	5	6
1	2	3	4	5	6

Зөөх эрэмбэлэлт

Хийсвэр кодын дүрэм:

- ▶ Догол мөрөөр бүлэг кодыг тэмдэглэнэ.
- ▶ Давталтыг тэмдэглэхдээ **while**, **for** болон **repeat-until**,
- ▶ Нөхцөл тэмдэглэхдээ **if-else**, **elseif** ашиглана
- ▶ Тайлбарыг **//** түлхүүр ашиглаж бичнэ.
- ▶ Гадаад хувьсагчийг **global** хэмээн зааж өгнө.

Зөөх эрэмбэлэлт

- ▶ Жагсаалтын элементүүдийн товъёгийг дөрвөлжин хаалтад тэмдэглэх бөгөөд энэ нь 1 гэсэн утгаас эхэлнэ.
- ▶ Дэд жагсаалтыг (subarray) : тэмдэг ашиглан бичнэ.
- ▶ Объектын атрибуутуудыг дуудахдаа цэг ашиглана.

Зөөх эрэмбэлэлт

- ▶ Функцэд утга дамжуулах үед дамжуулсан утгын хожмын өөрчлөлт функцэд нөлөөлөхгүй
- ▶ Харин объект дамжуулсан бол дамжуулсан объектын заагчийг хуулбарлах боловч объектын атрибутуудыг хуулбарлахгүй.

Зөөх эрэмбэлэлт

- ▶ **return** нь олон өгөгдлийг, шинээр объект үүсгэж багцлахгүйгээр шууд буцаадаг.
- ▶ **and** болон **or** нь богино холбоо (short circuiting) бөгөөд *x and y* гэвэл эхэлж *x* үнэн байж удаах үнэлэгдэнэ.
- ▶ **error** түлхүүр үгээр функцэд алдаа гарах тохиолдлыг бүртгэхэд ашиглана.

Алгоритмын шинжилгээ

Алгоритмын шинжилгээ

Алгоритмын шинжилгээг гүйцэтгэх машин:

- ▶ RAM (random-access machine): ердийн 1 процессор бүхий зэрэгцээ бус ердийн горимоор ажилладаг
- ▶ Θгөгдлийн төрөл: integer, floating point болон character

Алгоритмын шинжилгээ

n тооны оролтын хэмжээ бүхий ажиллах хугацааг $T(n)$ хэмээн тэмдэглэнэ гэвэл:

Insertion-Sort(A, n)

```
1  for i = 2 to n
2      key = A[i]
3      j = i - 1
4      while j > 0 and A[j] > key
5          A[j + 1] = A[j]
6          j = j - 1
7      A[j + 1] = key
```

c_1	n
c_2	$n - 1$
c_3	$n - 1$
c_4	$\sum_{i=2}^n t_i$
c_5	$\sum_{i=2}^n (t_i - 1)$
c_6	$\sum_{i=2}^n (t_i - 1)$
c_7	$n - 1$

Алгоритмын шинжилгээ

Дээрх алгоритмын хувьд гүйцэтгэлийн хугацаа нь:

$$\begin{aligned} T(n) &= c_1 n + c_2(n - 1) + c_3(n - 1) + c_4 \sum_{i=2}^n t_i + \\ &+ c_5 \sum_{i=2}^n (t_i - 1) + c_6 \sum_{i=2}^n (t_i - 1) + c_7(n - 1) \end{aligned}$$

Алгоритмын шинжилгээ

Хамгийн бага хугацаа (best-case running time) буюу
эрэмбэлэгдсэн тохиолдол:

$$\begin{aligned}T(n) &= c_1n + c_2(n - 1) + c_3(n - 1) + c_4(n - 1) + c_7(n - 1) = \\&= (c_1 + c_2 + c_3 + c_4 + c_7)n - (c_2 + c_3 + c_4 + c_7)\end{aligned}$$

буюу $an + b$ гэсэн шугаман функц гарна.

Алгоритмын шинжилгээ

Хамгийн их хугацаа (worst case) буюу урвуу эрэмбэлэгдсэн тохиолдол:

$$\begin{aligned} T(n) &= c_1 n + c_2(n - 1) + c_3(n - 1) + c_4 \left(\frac{n(n + 1)}{2} - 1 \right) + \\ &\quad + c_5 \left(\frac{n(n + 1)}{2} - 1 \right) + c_6 \left(\frac{n(n + 1)}{2} - 1 \right) + c_7(n - 1) = \\ &= \left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2} \right) n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7 \right) n - \\ &\quad - (c_2 + c_3 + c_4 + c_7) \end{aligned}$$

буюу $an^2 + bn + c$ гэсэн квадрат функц байна.

Алгоритмын шинжилгээ

Өмнөх тохиолдолд доорх нийлбэрүүдийг ашигласан болно:

$$\sum_{i=2}^n i = \left(\sum_{i=1}^n i \right) - 1 = \frac{n(n+1)}{2} - 1,$$

$$\sum_{i=2}^n (i-1) = \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2}$$

Алгоритмын шинжилгээ

Алгоритмын шинжилгээний хувьд ихэвчлэн хамгийн их хугацааг авч үздэг. Учир нь:

- ▶ Хамгийн их хугацааг тогтоосноор алгоритмын ажиллах хязгаар баталгаажна
- ▶ Зарим программын хувьд гүйцэтгэлийн хурд ихэвчлэн удаан байдаг
- ▶ Дундаж хугацаа нь мөн хамгийн их хугацааны адилаар түгээмэл ашиглагддаг

Алгоритмын шинжилгээ

Өсөлтийн хурд—rate of growth буюу order of growth:

- ▶ Өсөх хурдыг грек тета үсгээр (Θ) тэмдэглэнэ
- ▶ Өмнөх алгоритмын хувьд ажиллах хамгийн их хугацаа нь $\Theta(n^2)$
- ▶ Хамгийн бага хугацаа нь $\Theta(n)$

Алгоритмын зохиомж

Алгоритмын зохиомж

Алгоритмын зохиомж:

- ▶ Зөөх эрэмбэлэлтэд өсгөх арга ашиглагдсан
- ▶ Эрэмбэлэх бодлогыг хэсэгчлэх аргаар бодъё

Алгоритмын зохиомж

Хэсэгчлэх Бодлогыг ижил төрлийн дэд бодлого болгон задална

Боловсруулах Дэд бодлогуудыг дахин дуудагдах (recursively) байдлаар бодно

Нэгтгэх Ийнхүү хэсэгчлэн задалж боловсруулсан үр дүнгээ нэгтгэнэ

Алгоритмын зохиомж

Merge(A, p, q, r)

```
1   $n_L = q - p + 1$ 
2   $n_R = r - q$ 
3   $L[0 : n_L - 1]$  6а  $R[0 : n_R - 1]$  нь жагсаалт
4  for  $i = 0$  to  $n_L - 1$ 
5     $L[i] = A[p + i]$ 
6  for  $j = 0$  to  $n_R - 1$ 
7     $R[j] = A[q + j + 1]$ 
8   $i = 0$ 
9   $j = 0$ 
10  $k = p$ 
```

...

```
11 while  $i < n_L$  and  $j < n_R$ 
12   if  $L[i] \leq R[j]$ 
13      $A[k] = L[i]$ 
14   else  $A[k] = R[j]$ 
15      $j = j + 1$ 
16    $k = k + 1$ 
17 while  $i < n_L$ 
18    $A[k] = L[i]$ 
19    $i = i + 1$ 
20    $k = k + 1$ 
21 while  $j < n_R$ 
22    $A[k] = R[j]$ 
23    $j = j + 1$ 
24    $k = k + 1$ 
```

Алгоритмын зохиомж

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	2	4	6	7	1	2	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(a)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	6	7	1	2	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(c)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	2	3	1	2	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(e)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	2	3	4	5	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	4	
	2	4	6	7		1	2	3	5		
	i					j					

(g)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	4	6	7	1	2	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(b)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	2	2	7	1	2	3	5	...
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(d)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	2	3	4	2	3	5	...	
	k										

L	0	1	2	3	R	0	1	2	3	3	
	2	4	6	7		1	2	3	5		
	i					j					

(f)

A	8	9	10	11	12	13	14	15	16	17	\dots
	...	1	2	2	3	4	5	6	7	...	
	k										

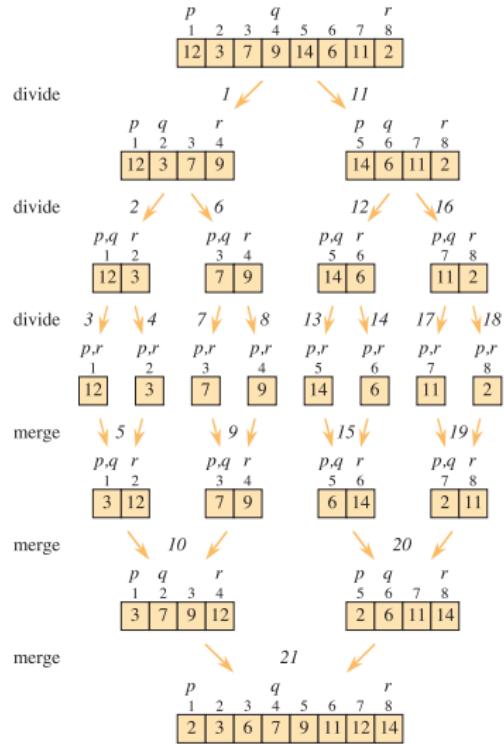
L	0	1	2	3	R	0	1	2	3	4	
	2	4	6	7		1	2	3	5		
	i					j					

(h)

Алгоритмын зохиомж

```
Merge-Sort( $A, p, r$ )  
1  if  $p \geq r$   
2      return  
3   $q = \lfloor (p + r)/2 \rfloor$   
4  Merge-Sort( $A, p, q$ )  
5  Merge-Sort( $A, q + 1, r$ )  
6  Merge( $A, p, q, r$ )
```

Алгоритмын зохиомж



Алгоритмын зохиомж

Алгоритмын шинжилгээ:

Хэсэгчлэх Энэ үйлдэл зөвхөн хуваалт хийх тул $D(n) = \Theta(1)$

Боловсруулах 2 хэсэгт хуваах тул $2T(n/2)$ хугацаа зарцуулна

Нэгтгэх n элементийг нэгтгэх тул $C(n) = \Theta(n)$ болно

Алгоритмын зохиомж

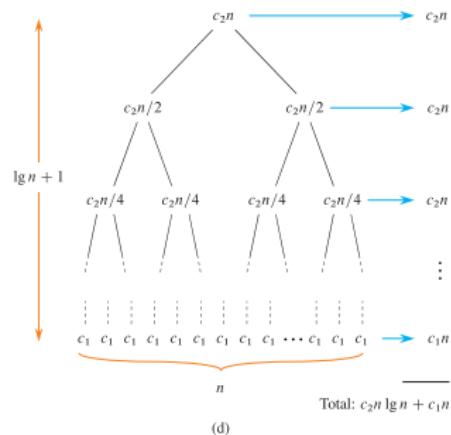
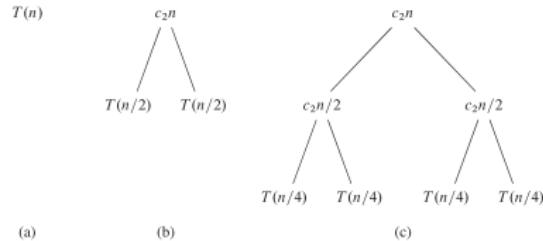
Алгоритмын шинжилгээ:

$$T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n) \quad (1)$$

Энд $\lg n$ гэдэг нь $\log_2 n$. Ийнхүү

$$\Theta(n \lg n) < \Theta(n^2)$$

Алгоритмын зохиомж



Зураг 1: Нэгтгэн эрэмбэлэх алгоритмын гүйцэтгэлийн шинжилгээ

Ажиллах хугацааг тодорхойлох

Ажиллах хугацааг тодорхойлох

O, Ω болон Θ тэмдэглэгээ

O , Ω болон Θ тэмдэглэгээ

$$\left(\frac{c_4}{2} + \frac{c_5}{2} + \frac{c_6}{2} \right) n^2 + \left(c_1 + c_2 + c_3 + \frac{c_4}{2} - \frac{c_5}{2} - \frac{c_6}{2} + c_7 \right) n - (c_2 + c_3 + c_4 + c_7)$$

Ажиллах хугацааг тодорхойлох

Дээд хязгаарын тэмдэглэгээ

O тэмдэглэгээ

Функцийн дээд хязгаарын тэмдэглэгээ: O

- ▶ $7n^3 + 100n^2 - 20n + 6$ функцийн хувьд $7n^3$ нь өндөр зэрэгтэй гишүүн
- ▶ Иймд ажиллах хугацаа нь $O(n^3)$

O тэмдэглэгээ

Тэгвэл $7n^3 + 100n^2 - 20n + 6$ функцийн хувьд өсөлтийн дээд хязгаар нь:

- ▶ $O(n^4)$
- ▶ $O(n^5)$
- ▶ $O(n^6)$

Буюу ерөнхий тохиолдолд:

- ▶ $O(n^c)$ энд $c \geq 3$

Ажиллах хугацааг тодорхойлох

Доод хязгаарын тэмдэглэгээ

Ω тэмдэглэгээ

Функцийн доод хязгаарын тэмдэглэгээ: Ω

- ▶ $7n^3 + 100n^2 - 20n + 6$ функцийн хувьд $7n^3$ нь өндөр зэрэгтэй гишүүн
- ▶ Иймд энэ функц дор хаяж n^3 хурдтай өснө

Ω тэмдэглэгээ

Тэгвэл дээрх функцийн хувьд:

- ▶ $\Omega(n^3)$ нь доод хязгаар болно гэвэл
- ▶ $\Omega(n^2)$ нь мөн доод хязгаар
- ▶ $\Omega(n)$ нь мөн доод хязгаар болно

Буюу ерөнхий тохиолдолд:

- ▶ $\Omega(n^c)$ энд $c \leq 3$

Ажиллах хугацааг тодорхойлох

Зааглагдсан хязгаарын тэмдэглэгээ

Θ тэмдэглэгээ

Зааглагдсан функцийн хязгаарын тэмдэглэгээ: Θ

- ▶ Дээд хязгаар нь $O(f(n))$
- ▶ Доод хязгаар нь $\Omega(f(n))$

Жишээ: зөөх эрэмбэлэлт

Insertion-Sort(A, n)

```
1  for  $i = 2$  to  $n$ 
2       $key = A[i]$ 
3       $j = i - 1$ 
4      while  $j > 0$  and  $A[j] > key$ 
5           $A[j + 1] = A[j]$ 
6           $j = j - 1$ 
7       $A[j + 1] = key$ 
```

Жишээ: зөөх эрэмбэлэлт

Зөөх эрэмбэлэлтийн оролт урвуугаар эрэмбэлэгдсэн (worst-case) тохиолдолд:

- ▶ Ажиллагааны доод хязгаар нь: $\Omega(n^2)$
 - ▶ Ажиллагааны дээд хязгаар нь: $O(n^2)$
- бол
- ▶ Ажиллагааны дундаж хязгаар нь: $\Theta(n^2)$

Жишээ: зөөх эрэмбэлэлт

$A[1:n/3]$	$A[n/3 + 1:2n/3]$	$A[2n/3 + 1:n]$
each of the $n/3$ largest values moves	through each of these $n/3$ positions	to somewhere in these $n/3$ positions

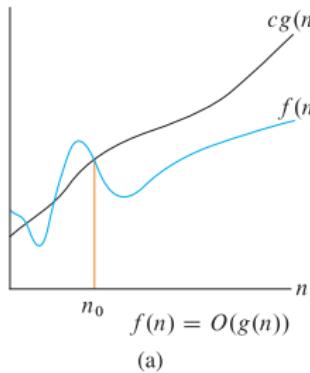


The diagram consists of two curved arrows. One arrow originates from the bottom of the first column and points to the bottom of the third column. Another arrow originates from the bottom of the second column and also points to the bottom of the third column.

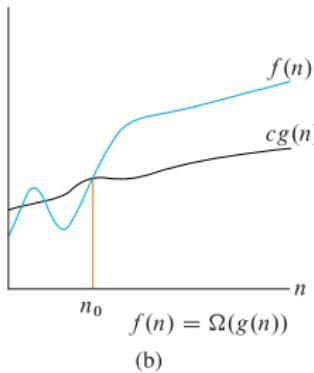
Ажиллах хугацааг тодорхойлох

Хязгаарын тэмдэглэгээ

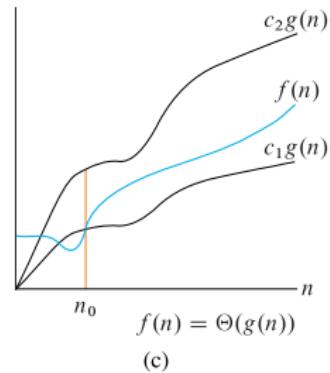
Хязгаарын тэмдэглэгээ



(a) $f(n) = O(g(n))$



(b) $f(n) = \Omega(g(n))$



(c) $f(n) = \Theta(g(n))$

- ▶ O нь дөхөлтийн дээд хязгаар (asymptotic upper bound)
- ▶ Ω нь дөхөлтийн доод хязгаар (asymptotic lower bound)
- ▶ Θ нь дөхөлтийн зааглагдсан хязгаар (asymptotically tight bounds)

O тэмдэглэгээ

$O(g(n)) = \{f(n) : n \geq n_0$ байх бүх n тооны хувьд $0 \leq f(n) \leq cg(n)$
нөхцөлийг хангах c , n_0 эерэг тогтмол тоо олдоно}

O тэмдэглэгээ

$$4n^2 + 100n + 500 = O(n^2)$$
$$4n^2 + 100n + 500 \leq cn^2$$

$$n^3 - 100n^2 = O(n^2)$$
$$n^3 - 100n^2 \leq cn^2$$

Ω тэмдэглэгээ

$\Omega(g(n)) = \{f(n) : n \geq n_0$ байх бүх n тооны хувьд $0 \leq cg(n) \leq f(n)$
нөхцөлийг хангах c , n_0 эерэг тогтмол тоо олдоно}

Ω тэмдэглэгээ

$$4n^2 + 100n + 500 = \Omega(n^2)$$

$$n^2/100 - 100n - 500 = \Omega(n^2)$$

Θ тэмдэглэгээ

$\Theta(g(n)) = \{f(n) : 0 \leq c_1g(n) \leq f(n) \leq c_2g(n)$ байх бүх n тооны хувьд $0 \leq cg(n) \leq f(n)$ нөхцөлийг хангах c_1, c_2 болон n_0 эерэг тогтмол тоо олдоно}

Теорем

Теорем 4.1

$f(n)$ болон $g(n)$ функцуудийн хувьд $f(n) = \Theta(g(n))$ байх гарцаагүй бөгөөд хүрэлцээтэй нөхцөл нь $f(n) = O(g(n))$ ба $f(n) = \Omega(g(n))$.

Хязгаарын тэмдэглэгээ ба ажиллах хугацаа

Алгоритмын ажиллах хугацааг тодорхойлохдоо хязгаарын тэмдэглэгээг зөв ашигласан эсэхийг нягтлах хэрэгтэй!

Хязгаарын тэмдэглэгээ ба ажиллах хугацаа

Нэгтгэх эрэмбэлэлт:

- ▶ Ямар ч тохиолдолд ажиллах хугацаа нь $\Theta(n \lg n)$ тул
- ▶ $\Theta(n \lg n)$ хугацаанд ажилладаг гэнэ

Хязгаарын тэмдэглэгээ ба ажиллах хугацаа

Ажиллах хугацааны хязгаар нь ихэвчлэн дараах 2 төрлийн функц байна:

- ▶ олон гишүүнт: $n, n^{1/2}$
- ▶ логарифм: $n \lg^2 n, n^2 \lg n$

Зарим тохиолдолд илтгэгч (exponential) функц байдаг:

- ▶ $\lg \lg n, \lg^* n$

Хязгаарын тэмдэглэгээ тэнцэтгэл болон тэнцэтгэлбишид

Хязгаарын тэмдэглэгээ нь тэнцэтгэлийн баруун талд дангаар:

$$4n^2 + 100n + 500 = O(n^2)$$

$$4n^2 + 100n + 500 \in O(n^2)$$

Хязгаарын тэмдэглэгээ нь томьёон дунд:

$$2n^2 + 3n + 1 = 2n^2 + \Theta(n)$$

$$2n^2 + 3n + 1 = 2n^2 + f(n)$$

Үүнд: $f(n) \in \Theta(n)$ ба $f(n) = 3n + 1$ болно.

Хязгаарын тэмдэглэгээ тэнцэтгэл болон тэнцэтгэлбишид

Илэрхийлэлд орсон олон тооны тодорхойгүй функц бүрийг хязгаарын тэмдэглэгээгээр илэрхийлж болно.

$$\sum_{i=1}^n O(i) \tag{2}$$

$$O(1) + O(2) + \cdots + O(n) \tag{3}$$

Хязгаарын тэмдэглэгээ тэнцэтгэл болон тэнцэтгэлбишид

Тэнцэтгэлийн зүүн талд орсон хязгаарын тэмдэглэгээ:

$$2n^2 + \Theta(n) = \Theta(n^2)$$

Дүрэм 4.1

Тэнцэтгэлийн зүүн талд орсон тодорхойгүй функцийг хэрхэн сонгосноос үл хамааран тэнцэтгэлийг үнэн байлгах тодорхойгүй функцийг тэнцэтгэлийн баруун талд сонгож болно.

Хязгаарын тэмдэглэгээ тэнцэтгэл болон тэнцэтгэлбишид

Дээрх хамаарлыг угсруулан ашиглаж болно:

$$\begin{aligned}2n^2 + 3n + 1 &= 2n^2 + \Theta(n) \\&= \Theta(n^2).\end{aligned}$$

Хязгаарын тэмдэглэгээний буруу хэрэглээ

Буруу хэрэглээ:

$$n < 3 \text{ үед } T(n) = O(1) \quad (4)$$

$$n < 3 \text{ үед } T(n) = \Theta(1) \quad (5)$$

Хязгаарын тэмдэглэгээний буруу хэрэглээ

Хязгаар нь тодорхойгүй бол тухайн тохиолдолд хязгаарыг таамагладаг.

О ТЭМДЭГЛЭГЭЭ

$o(g(n)) = \{f(n) : c > 0$ байх дурын эерэг тооны хувьд
 $n \geq n_0$ үед $0 \leq f(n) < cg(n)$ нөхцөлийг хангах
 $n_0 > 0$ эерэг тогтмол тоо олдоно}

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$$

$$n^{1.9999} = o(n^2), \quad n = o(n^2), \quad 2n = o(n^2), \quad 2n^2 \neq o(n^2)$$

ω ТЭМДЭГЛЭЛ

$\omega(g(n)) = \{f(n) : c > 0$ байх дурын эерэг тооны хувьд
 $n \geq n_0$ үед $0 \leq cg(n) < f(n)$ нөхцөлийг хангах
 $n_0 > 0$ эерэг тогтмол тоо олдоно}

$$\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$$

$$n^2/2 = \omega(n), \quad n^2/2 \neq \omega(n^2)$$

Функцийн харьцуулалт

Дамжих чанар:

$$f(n) = \Theta(g(n)) \text{ ба } g(n) = \Theta(h(n)) \text{ бол } f(n) = \Theta(h(n)),$$

$$f(n) = O(g(n)) \text{ ба } g(n) = O(h(n)) \text{ бол } f(n) = O(h(n)),$$

$$f(n) = \Omega(g(n)) \text{ ба } g(n) = \Omega(h(n)) \text{ бол } f(n) = \Omega(h(n)),$$

$$f(n) = o(g(n)) \text{ ба } g(n) = o(h(n)) \text{ бол } f(n) = o(h(n)),$$

$$f(n) = \omega(g(n)) \text{ ба } g(n) = \omega(h(n)) \text{ бол } f(n) = \omega(h(n))$$

Функцийн харьцуулалт

Эгэх чанар:

$$f(n) = \Theta(f(n)),$$

$$f(n) = O(f(n)),$$

$$f(n) = \Omega(f(n))$$

Функцийн харьцуулалт

Тэгш хэмт чанар:

$$f(n) = \Theta(g(n))$$
 байх гарцаагүй бөгөөд хүрэлцээтэй нөхцөл нь
$$g(n) = \Theta(f(n))$$

Функцийн харьцуулалт

Тэгш хэмт урвуу чанар:

$f(n) = O(g(n))$ байх гарцаагүй бөгөөд хүрэлцээтэй нөхцөл нь
 $g(n) = \Omega(f(n))$

$f(n) = o(g(n))$ байх гарцаагүй бөгөөд хүрэлцээтэй нөхцөл нь
 $g(n) = \omega(f(n)).$

Функцийн харьцуулалт

Харьцуулалт:

$f(n) = O(g(n))$	нь	$a \leq b,$
$f(n) = \Omega(g(n))$	нь	$a \geq b,$
$f(n) = \Theta(g(n))$	нь	$a = b,$
$f(n) = o(g(n))$	нь	$a < b,$
$f(n) = \omega(g(n))$	нь	$a > b.$

Функцийн харьцуулалт

Дүрэм 4.2 (Гурвалын хууль)

Дурын бодит a ба b тоонуудын хувьд дараах нөхцөлүүдээс нэг нь зайлшгүй биелнэ: $a < b$, $a = b$, $a > b$.

Ажиллах хугацааг тодорхойлох

Үндсэн тэмдэглэгээ ба түгээмэл
функцууд

Нэгэн хэвийн чанар

Хэрэв $f(n)$ функц нь:

- ▶ $m \leq n$ үед $f(m) \leq f(n)$ бол нэгэн хэвийн өснө,
- ▶ $m \leq n$ үед $f(m) \geq f(n)$ бол нэгэн хэвийн буурна,
- ▶ $m < n$ үед $f(m) < f(n)$ бол эрс өснө,
- ▶ $m < n$ үед $f(m) > f(n)$ бол эрс буурна.

Доод болон дээд бүхэл тоо

Дурын бодит x тооны хувьд

- ▶ x -ээс бага буюу тэнцүү байх хамгийн их бүхэл тоо: $[x]$
- ▶ x -ээс их буюу тэнцүү байх хамгийн бага бүхэл тоо: $\lceil x \rceil$

Доод болон дээд бүхэл тоон чанар

Дурын бүхэл n тооны хувьд

$$\lfloor n \rfloor = n = \lceil n \rceil \quad (6)$$

Бүх бодит x тооны хувьд

$$x - 1 < \lfloor x \rfloor \leq x \leq \lceil x \rceil < x + 1, \quad (7)$$

$$-\lfloor x \rfloor = \lceil -x \rceil, \quad (8)$$

$$-\lceil x \rceil = \lfloor -x \rfloor. \quad (9)$$

Доод болон дээд бүхэл тоо

Дурын бодит тоо $x \geq 0$ болон $a, b > 0$ байх бүхэл тооны хувьд

$$\left\lceil \frac{\lceil x/a \rceil}{b} \right\rceil = \left\lceil \frac{x}{ab} \right\rceil, \quad (10)$$

$$\left\lfloor \frac{\lfloor x/a \rfloor}{b} \right\rfloor = \left\lfloor \frac{x}{ab} \right\rfloor, \quad (11)$$

$$\left\lceil \frac{a}{b} \right\rceil \leq \frac{a + (b - 1)}{b}, \quad (12)$$

$$\left\lfloor \frac{a}{b} \right\rfloor \geq \frac{a - (b - 1)}{b}. \quad (13)$$

Доод болон дээд бүхэл тоо

Дурын бүхэл n болон бодит x тооны хувьд

$$\lfloor n + x \rfloor = n + \lfloor x \rfloor, \quad (14)$$

$$\lceil n + x \rceil = n + \lceil x \rceil. \quad (15)$$

Үлдэгдлийн арифметик

Дурын бүхэл a тоо болон дурын эерэг бүхэл n тооны хувьд $a \bmod n$ нь a/n ногдворын үлдэгдэл болно:

$$a \bmod n = a - n \lfloor a/n \rfloor. \quad (16)$$

Эндээс, a нь сөрөг үед ч дараах тэнцэтгэл биш мөрдөнө:

$$0 \leq a \bmod n < n. \quad (17)$$

Үлдэгдлийн арифметик

- ▶ $(a \bmod n) = (b \bmod n)$ бол $a = b \pmod n$
- ▶ n нь $b - a$ тооны хуваагч байх гарцаагүй бөгөөд хүрэлцээтэй нөхцөл нь $a = b \pmod n$
- ▶ a нь n тооны хувьд үлдэгдлээрээ b тоотой тэнцүү биш бол $a \neq b \pmod n$

Олон гишүүнт

Өгөгдсөн сөрөг биш d бүхэл тооны хувьд d зэрэгтэй n олон гишүүнт $p(n)$ функц нь:

$$p(n) = \sum_{i=0}^d a_i n^i.$$

Үүнд, a_0, a_1, \dots, a_d тогтмолууд нь олон гишүүнтийн үржүүлэгч ба $a_d \neq 0$ байна.

Олон гишүүнт

Олон гишүүнтийн хязгаар эерэг байх зайлшгүй бөгөөд хүрэлцээтэй нөхцөл нь $a_d > 0$ байх явдал.

Олон гишүүнт

Хязгаар нь d зэрэг бүхий эерэг $p(n)$ олон гишүүнтийн хувьд

$$p(n) = \Theta(n^d)$$

Олон гишүүнт

n^a функц нь:

- ▶ $a \geq 0$ дурын бодит тогтмолын хувьд нэг хэвийн өсөлттэй,
- ▶ $a \leq 0$ дурын бодит тогтмолын хувьд нэг хэвийн бууралттай.

Олон гишүүнт

k тогтмолын хувьд $f(n) = O(n^k)$ бол $f(n)$ функцийг олон гишүүнтээр зааглагдсан гэдэг.

Илтгэгч функц

Бүх $a > 0$, m ба n бодит тооны хувьд дараах адилтгал үнэн:

$$a^0 = 1,$$

$$a^1 = a,$$

$$a^{-1} = 1/a,$$

$$(a^m)^n = a^{mn},$$

$$(a^m)^n = (a^n)^m,$$

$$a^m a^n = a^{m+n}.$$

Илтгэгч функц

Бүх n болон $a \geq 1$ байх бодит тоонуудын хувьд a^n функц нь n илтгэгчээр нэг хэвийн өсөлттэй байна.

Илтгэгч функц

Олон гишүүнт болон илтгэгч функцийн өсөлтийн хурдны харьцаа $a > 1$ ба b тооны хувьд

$$\lim_{n \rightarrow \infty} \frac{n^b}{a^n}$$

гэдгээс

$$n^b = o(a^n) \tag{18}$$

Илтгэгч функц

Натурал логарифм функц:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots = \sum_{i=0}^{\infty} \frac{x^i}{i!}$$

гэдгээс дараах тэнцэтгэл биш гарна:

$$1 + x \leq e^x \quad (19)$$

Энэ нь $|x| \leq 1$ үед

$$1 + x \leq e^x \leq 1 + x + x^2 \quad (20)$$

болно.

Илтгэгч функц

$x \rightarrow 0$ үед e^x -г $1 + x$ -ээр ойролцоолбол

$$e^x = 1 + x + \Theta(x^2).$$

Дурын x тооны хувьд дараах томъёо биелнэ:

$$\lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n = e^x \quad (21)$$

Логарифм

Тэмдэглэгээнүүд:

$$\lg n = \log_2 n$$

хоёртын логарифм

$$\ln n = \log_e n$$

натурал логарифм

$$\lg^k n = (\lg n)^k$$

зэрэг дэвшүүлэх

$$\lg \lg n = \lg(\lg n)$$

давхар логарифм

Логарифм

Томъёоны хувьд хаалтгүй тохиолдолд логарифм функц нь зөвхөн дараагийнхаа илэрхийлэлдээ хамаарна:

$$\begin{aligned} \lg n + 1 &\quad \text{гэдэг нь} \quad (\lg n) + 1 \\ &\quad \text{харин} \quad \lg(n + 1) \quad \text{биш} \end{aligned}$$

Логарифм

Дурын $b > 1$ тогтмолын хувьд $\log_b n$ функц нь:

- ▶ $n \leq 0$ үед тодорхойгүй,
- ▶ $n > 0$ үед эрс өсөлттэй,
- ▶ $0 < n < 1$ үед сөрөг,
- ▶ $n > 1$ үед эерэг,
- ▶ $n = 1$ үед 0

байна.

Логарифм

Дурын бодит $a > 0$, $b > 0$, $c > 0$ болон n тоонуудын хувьд

$$a = b^{\log_b a} \quad (22)$$

$$\log_c(ab) = \log_c a + \log_c b \quad (23)$$

$$\log_b a^n = n \log_b a$$

$$\log_b a = \frac{\log_c a}{\log_c b} \quad (24)$$

$$\log_b(1/a) = -\log_b a \quad (25)$$

$$\log_b a = \frac{1}{\log_a b}$$

$$a^{\log_b c} = c^{\log_b a} \quad (26)$$

Дээрх томъёонуудын хувьд логарифмын суурь нь нэгтэй тэнцэхгүй хэмээн үзнэ.

Логарифм

Тогтмол үржүүлэгч:

$$\log_a n = \frac{1}{\log_b a} \cdot \log_b n$$

$\log_a n$ ба $\log_b n$ нь тогтмол $\frac{1}{\log_b a}$ тоогоор ялгаатай.

$$O(\log_2 n) = O(\log_{10} n) = O(\lg n)$$

O тэмдэглэгээний хувьд тогтмол тоог үл хэрэгсдэг.

Логарифм

$|x| < 1$ үед $\ln(1 + x)$ томъёоны хувьд дараах цуваа хүчинтэй:

$$\ln(1 + x) = x - \frac{x^2}{2} + \frac{x^3}{3} - \frac{x^4}{4} + \frac{x^5}{5} - \dots \quad (27)$$

$x > -1$ үед дараах тэнцэтгэлбиш хүчинтэй:

$$\frac{x}{1 + x} \leq \ln(1 + x) \leq x \quad (28)$$

Энд, $x = 0$ үед тэнцэлд хүрнэ.

Логарифм

Хэрэв k тогтмолын хувьд $f(n) = O(\lg^k n)$ бол $f(n)$ функцийг зэрэгт логарифмоор хязгаарлагдсан гэдэг.

Логарифм

Илтгэгч функц болон олон гишүүнт функцийн өсөлтийн хамаарлыг ольё ((18)-р тэгшитгэлд n -г $\lg n$, a -г 2^a орлуулъя):

$$n^b = o(a^n)$$

$$(\lg n)^b = o((2^a)^{\lg n}) = o(2^{a \lg n}) = o(2^{\lg n^a}) = o((2^{\lg n})^a) = o(n^a)$$

болно. Ийнхүү

$$\lg^b n = o(n^a) \tag{29}$$

дурын эерэг олон гишүүнт функц нь зэрэгт логарифм функцээс хурдан өснө хэмээн дүгнэж болно.

Факториал

$n \geq 0$ байх бүхэл n тооны дараалсан үржвэр буюу факториал:

$$n! = \begin{cases} 1, & \text{хэрэв } n = 0 \text{ бол,} \\ n \cdot (n - 1)!, & \text{хэрэв } n > 0 \text{ бол.} \end{cases}$$

Өөрөөр хэлбэл,

$$n! = 1 \cdot 2 \cdot 3 \dots n.$$

Факториал

n дараалсан үржвэрийн хувьд дарааллын гишүүн бүр нь n тооноос хэтрэхгүй тул түүний барагцаалсан дээд хязгаар нь $n! \leq n^n$ байна.

Стирлингийн барагцаалал¹:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n \left(1 + \Theta\left(\frac{1}{n}\right)\right) \quad (30)$$

¹Stirling's approximation

Факториал

Стирлингийн томьёогоор дараах тэгшитгэл үнэн:

$$n! = o(n^n) \quad (31)$$

$$n! = \omega(2^n) \quad (32)$$

$$\lg(n!) = \Theta(n \lg n) \quad (33)$$

$n \geq 1$ үед Стирлингийн томьёо нь:

$$n! = \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\alpha_n} \quad (34)$$

болно. Үүнд,

$$\frac{1}{12n + 1} < \alpha_n < \frac{1}{12n}.$$

Функцийн давталт

$f(n)$ функция нь i удаа давтагдсан бол:

$$f^{(i)}(n) = \begin{cases} n, & \text{хэрэв } i = 0 \text{ бол,} \\ f(f^{(i-1)}(n)), & \text{хэрэв } i > 0 \text{ бол.} \end{cases} \quad (35)$$

Үүнд, i нь эерэг бүхэл тоо.

Жишээлбэл, $f(n) = 2n$ бол $f^{(i)}(n) = 2^i n$ байна.

Давталттай логарифм функц

Давталттай логарифм функц:

$$\lg^* n = \min\{i \geq 0 : \lg^{(i)} n \leq 1\}.$$

Жишээлбэл, $\lg^{(3)} n = \lg(\lg(\lg n))$

Давталттай логарифм функц

Давталттай логарифм функцийн өсөлт маш удаан байдаг.

Жишээ нь:

$$\lg^* 2 = 1,$$

$$\lg^* 4 = 2,$$

$$\lg^* 16 = 3,$$

$$\lg^* 65536 = 4,$$

$$\lg^*(2^{65536}) = 5.$$

Ертөнц дээрх нийт атомын тоо нь 10^{80} гэвэл 2^{65536} нь түүнээс харьцаангуй их гэдгийг харуулъя. $a^b = 10^{\log_{10} a^b} = 10^{b \log_{10} a}$ гэдгийг ашиглавал:

$$2^{65536} = 10^{65536 / \lg 10} = 10^{65536 / 3.3} \approx 10^{19,728} > 10^{80}$$

болно.

Давталттай логарифм функц

$$\lg^*(2^{65536}) = 5:$$

1. $\lg^{(0)} n = 2^{65536} > 1$
2. $\lg^{(1)} n = \lg(2^{65536}) = 65536 > 1$
3. $\lg^{(2)} n = \lg(65536) = 16 > 1$
4. $\lg^{(3)} n = \lg(16) = 4 > 1$
5. $\lg^{(4)} n = \lg(4) = 2 > 1$
6. $\lg^{(5)} n = \lg(2) = 1 \leq 1$

Фибоначчиин тоо

$i \geq 0$ байх F_i функцийг Фибоначчиин тоо гэвэл:

$$F_i = \begin{cases} 0, & \text{хэрэв } i = 0 \text{ бол,} \\ 1, & \text{хэрэв } i = 1 \text{ бол,} \\ F_{i-1} + F_{i-2}, & \text{хэрэв } i \geq 2 \text{ бол.} \end{cases} \quad (36)$$

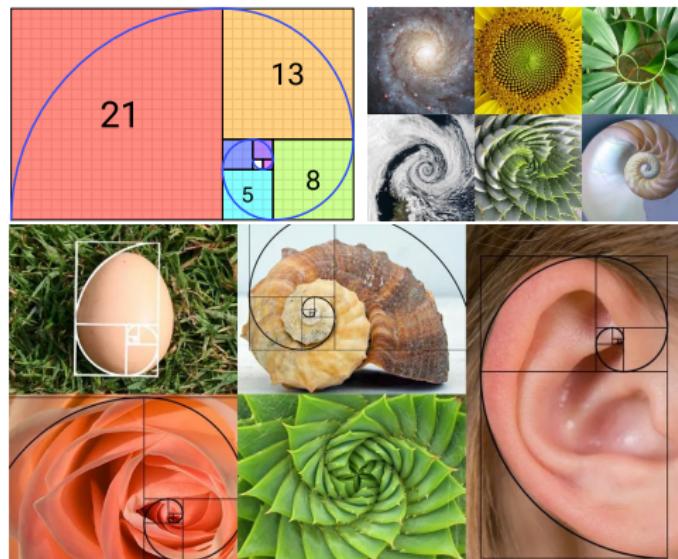
Эхний 2 тохиолдлоос бусад үед Фибоначчиийн тоо нь өмнөх хоёр тооны нийлбэртэй тэнцүү дараалал үүсгэнэ:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, \dots$$

Фибоначчиин тоо

Фибоначчиин дарааллын харьцаа нь алтан харьцаа уруу дөхдөг:

$$\lim_{n \rightarrow \infty} \frac{F_{n-1}}{F_{n-2}} = \phi$$



Зураг 2: Алтан харьцаа нь байгалийн зүй тогтолд түгээмэл

Фибоначчиин тоо

Алтан харьцаа нь квадрат тэгшитгэлээр тодорхойлогддог:

$$x^2 = x + 1$$

Түүний утга нь:

$$\phi = \frac{1 + \sqrt{5}}{2} = 1.61803\dots \quad (37)$$

Хосмог утга нь:

$$\hat{\phi} = \frac{1 - \sqrt{5}}{2} = -.61803\dots \quad (38)$$

Фибоначчиин тоо

Тэгвэл Фибоначчиийн i -р тоо нь:

$$F_i = \frac{\phi^i - \hat{\phi}^i}{\sqrt{5}}$$

төмьёогоор тодорхойлогддог гэвэл $|\hat{\phi}| < 1$ тул

$$\frac{|\hat{\phi}^i|}{\sqrt{5}} < \frac{1}{\sqrt{5}} < \frac{1}{2}$$

гэдгээс

$$F_i = \left\lfloor \frac{\phi^i}{\sqrt{5}} + \frac{1}{2} \right\rfloor \quad (39)$$

болно.

Хэсэгчлэх

Хэсэгчлэх



Зураг 3: Жон фон Нейман

Хэсэгчлэх

Хуваах байдлаар бодлогыг дэд бодлого буюу тухайн тохиолдол болгон хуваана

Бодох дэд бодлогыг дахин дуудагдаж бодно

Нэгтгэх байдлаар дэд бодлогын шийдүүдийг эх бодлогын шийд болгоно

Дахин дуудагдах тэгшитгэл

- ▶ Дахин дуудагдах тэгшитгэлийн (recurrence) функцийн хувьсагчийн утга нь хуваан хорогдох байдлаар дахин тодорхойлогддог
- ▶ Хэсэгчлэх аргыг ашиглаж буй алгоритмын ажиллах хугацааг математик аргаар тооцоолоходоо өөрийгөө дахин дуудагдах тэгшитгэлийг ашигладаг

Дахин дуудагдах тэгшитгэл

Зөв тодорхойлогдсон Ядаж нэг функц дахин дуудагдах
нөхцөлийг хангана

Буруу тодорхойлогдсон Нэг ч функц дахин дуудагдах
нөхцөлийг хангахгүй

Дахин дуудагдах тэгшитгэл

$$T(n) = \begin{cases} 1, & \text{хэрэв } n = 1, \\ 2T\left(\frac{n}{2}\right) + n, & \text{хэрэв } n > 1. \end{cases}$$

Энд,

- ▶ $T(1) = 1$ нь суурь нөхцөл
- ▶ $T(n) = 2T(n/2) + n$ нь дахин дуудагдах нөхцөл
- ▶ Зөв тодорхойлогдсон

Дахин дуудагдах тэгшитгэл

$$T(n) = T(n) + 1$$

- ▶ суурь нөхцөл тодорхойлогдоогүй
- ▶ дахин дуудагдахаа оролтын хэмжээг хэсэгчлэн багасгахгүй байна
- ▶ Буруу тодорхойлогдсон

Дахин дуудагдах тэгшитгэл

Хэрэв $n_0 > 0$ тогтмолын хувьд дараах хоёр чанар:

1. $n < n_0$ байх бүх n -ийн хувьд $T(n) = \Theta(1)$
 2. $n \geq n_0$ байх бүх n -ийн хувьд төгсгөлөг давталтын дараа суурь нөхцөлдөө хүрдэг
- биелж байвал $T(n)$ давталтыг алгоритмчлагдсан байна гэдэг.

Дахин дуудагдах тэгшитгэл

Дүрэм:

Хэрэв дахин дуудагдах тэгшитгэл нь суурь нөхцөлгүй бол түүнийг алгоритмчлагдсан байна гэж таамаглана.

Жишээлбэл:

$$T(n) = 2T(n/2) + \Theta(n)$$

- ▶ $T(1) = \Theta(1)$ нь суурь нөхцөл
- ▶ $2T(n/2)$ нь дахин дуудагдах нөхцөл
- ▶ Зөв тодорхойлогдсон

Дахин дуудагдах тэгшитгэл

Жишээлбэл:

$$T(n) \leq 2T(n/2) + \Theta(n) \text{ хувьд } O \text{ тэмдэглэгээгээр дээд хязгаар}$$
$$T(n) \geq 2T(n/2) + \Theta(n) \text{ хувьд } \Omega \text{ тэмдэглэгээгээр доод хязгаар}$$

Хэсэгчлэх алгоритм болон дахин дуудагдах тэгшитгэл

Хэсэгчлэх аргаар матрицыг үржүүлэх:

$$T(n) = 8T(n/2) + \Theta(1)$$

шийд нь $T(n) = \Theta(n^3)$

Штрассены алгоритм:

$$T(n) = 7T(n/2) + \Theta(n^2)$$

шийд нь $T(n) = \Theta(n^{\lg 7}) = O(n^{2.81})$

Хэсэгчлэх алгоритм болон дахин дуудагдах тэгшитгэл

Бодлогыг тогтмол бус хэмжээтэйгээр хэсэгчлэх:

$$T(n) = T(n/3) + T(2n/3) + \Theta(n)$$

бол шийд нь $T(n) = \Theta(n \lg n)$.

$$T(n) = T(n/5) + T(7n/10) + \Theta(n)$$

бол шийд нь $T(n) = \Theta(n)$.

$$T(n) = T(n - 1) + \Theta(1)$$

бол шийд нь $T(n) = \Theta(n)$ болно.

Дахин дуудагдах тэгшитгэлийг бодох

Дахин дуудагдах тэгшитгэлийг бодох аргууд:

Орлуулах арга Нэгтгэн дүгнэх арга ашиглана

Дахин дуудагдах модны арга Мод хэлбэрээр загварчилна

Ерөнхий арга $T(n) = aT(n/b) + f(n)$ томьёог ашиглана

Дөрвөлжин матрицуудыг үржүүлэх

Дөрвөлжин матрицуудыг үржүүлэх

Матриц үржүүлэх:

$$c_{ij} = \sum_{k=1}^n a_{ik} \cdot b_{kj} \quad (40)$$

$C = C + A \cdot B$ үржвэрийг хадгалахын тулд n^2 хэмжээтэй C матрицыг 0 утгаар дүүргэхэд $\Theta(n^2)$ хугацаа зарцуулна.

Matrix-Multiply(A, B, C, n)

```
1  for i = 1 to n
2      for j = 1 to n
3          for k = 1 to n
4               $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$ 
```

Алгоритмын ажиллах хугацаа нь, гуравласан давталт (n хүртэл) хийсэн тул $\Theta(n^3)$.

Дөрвөлжин матрицуудыг үржүүлэх

$n \times n$ хэмжээтэй A , B , C матрицуудыг $n/2 \times n/2$ хэмжээтэй дэд матрицад хуваана:

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, \quad C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} \quad (41)$$

Улмаар матриц хооронд дараах байдлаар үржих үйлдэл хийнэ:

$$\begin{aligned} \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} \\ &= \begin{pmatrix} A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\ A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22} \end{pmatrix} \quad (42) \end{aligned}$$

Дөрвөлжин матрицуудыг үржүүлэх

Өөрөөр хэлбэл, үржвэрийг дараах байдлаар харгалзуулна:

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}, \quad (43)$$

$$C_{12} = A_{11}B_{12} + A_{12}B_{22}, \quad (44)$$

$$C_{21} = A_{21}B_{11} + A_{22}B_{21}, \quad (45)$$

$$C_{22} = A_{21}B_{12} + A_{22}B_{22}. \quad (46)$$

Дөрвөлжин матрицуудыг үржүүлэх

Matrix-Multiply-Recursive(A, B, C, n)

```
1  if  $n == 1$ 
2       $c_{11} = c_{11} + a_{11} \cdot b_{11}$ 
3      return
4  partition  $A, B$  and  $C$  into  $n/2 \times n/2$  submatrices
     $A_{11}, A_{12}, A_{21}; B_{11}, B_{12}, B_{21}, B_{22};$ 
    and  $C_{11}, C_{12}, C_{21}, C_{22}$ 
5  Matrix-Multiply-Recursive( $A_{11}, B_{11}, C_{11}, n/2$ )
6  Matrix-Multiply-Recursive( $A_{11}, B_{12}, C_{12}, n/2$ )
7  Matrix-Multiply-Recursive( $A_{21}, B_{11}, C_{21}, n/2$ )
8  Matrix-Multiply-Recursive( $A_{21}, B_{12}, C_{22}, n/2$ )
9  Matrix-Multiply-Recursive( $A_{12}, B_{21}, C_{11}, n/2$ )
10 Matrix-Multiply-Recursive( $A_{12}, B_{22}, C_{12}, n/2$ )
11 Matrix-Multiply-Recursive( $A_{22}, B_{21}, C_{21}, n/2$ )
12 Matrix-Multiply-Recursive( $A_{22}, B_{22}, C_{22}, n/2$ )
```

Дөрвөлжин матрицуудыг үржүүлэх

Матрицыг дахин дуудагдах аргаар үржүүлэх тэгшитгэлийн ажиллах хугацаа:

$$T(n) = 8T(n/2) + \Theta(1) \quad (47)$$

Шийд нь $T(n) = \Theta(n^3)$ болохыг хожим «Ерөнхий арга» ашиглан тогтоох болно.

Шийдийн ажиллах хугацааны өсөлт нь нэгтгэх эрэмбэлэлтийнхээс маш хурдан өснө:

$$\Theta(n^3) > \Theta(n \lg n)$$

Матриц үржүүлэх Штрассены алгоритм

Матриц үргүүлэх Штрассены алгоритм



Зураг 4: Фолькер Штрассен

Матриц үржүүлэх Штассены алгоритм

Штассены алгоритмын ажиллах хугацаа:

$$T(n) = 7T(n/2) + \Theta(n^2) \quad (48)$$

$\Theta(n^{\lg 7}) = O(n^{2.81})$ буюу $o(n^3)$.

Ажиллах зарчмыг хялбаршуулан төсөөлбөл:

$$\begin{aligned} x^2 - y^2 &= x^2 - xy + xy - y^2 \\ &= x(x - y) + y(x - y) = (x + y)(x - y) \end{aligned}$$

Матриц үржүүлэх Штассены алгоритм

$n/2 \times n/2$ хэмжээтэй 10 матрицад нэмэх хасах үйлдлийг хийхэд зарцуулах хугацаа: $\Theta(n^2)$.

$$S_1 = B_{12} - B_{22},$$

$$S_2 = A_{11} + A_{12},$$

$$S_3 = A_{21} + A_{22},$$

$$S_4 = B_{21} - B_{11},$$

$$S_5 = A_{11} + A_{22},$$

$$S_6 = B_{11} + B_{22},$$

$$S_7 = A_{12} - A_{22},$$

$$S_8 = B_{21} + B_{22},$$

$$S_9 = A_{11} - A_{21},$$

$$S_{10} = B_{11} + B_{12}.$$

Матриц үржүүлэх Штассены алгоритм

$$P_1 = A_{11} \cdot S_1 \quad (= A_{11} \cdot B_{12} - A_{11} \cdot B_{22}),$$

$$P_2 = S_2 \cdot B_{22} \quad (= A_{11} \cdot B_{22} + A_{12} \cdot B_{22}),$$

$$P_3 = S_3 \cdot B_{11} \quad (= A_{21} \cdot B_{11} + A_{22} \cdot B_{11}),$$

$$P_4 = A_{22} \cdot S_4 \quad (= A_{22} \cdot B_{21} - A_{22} \cdot B_{11}),$$

$$P_5 = S_5 \cdot S_6 \quad (= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22}),$$

$$P_6 = S_7 \cdot S_8 \quad (= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22}),$$

$$P_7 = S_9 \cdot S_{10} \quad (= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12}).$$

4-р алхамд, 3-р алхамд гарган авсан P_i матрицыг ашиглаад $n/2 \times n/2$ хэмжээтэй C гэсэн 4 матрицыг тооцоолно:

$$C_{11} = C_{11} + P_5 + P_4 - P_2 + P_6.$$

Матриц үржүүлэх Штассены алгоритм

$$\begin{array}{r} A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \\ - A_{22} \cdot B_{11} \quad \quad \quad + A_{22} \cdot B_{21} \\ - A_{11} \cdot B_{22} \quad \quad \quad - A_{12} \cdot B_{22} \\ - A_{22} \cdot B_{22} - A_{22} \cdot B_{21} + A_{12} \cdot B_{22} + A_{12} \cdot B_{21} \end{array}$$

$$A_{11} \cdot B_{11} \quad \quad \quad + A_{12} \cdot B_{21}$$

Матриц үржүүлэх Штассены алгоритмын ажиллах хугацаа:

$$T(n) = \Theta(n^{\lg 7}) = o(n^3)$$

Дахин дуудагдах тэгшитгэлийг
орлуулах аргаар бодох

Дахин дуудагдах тэгшитгэлийг орлуулах аргаар бодох

Орлуулах арга:

1. Тогтмол хувьсагч ашиглан шийдийн тэгшитгэлийн хэлбэрийг таамаглах
2. Математик нэгтгэн дүгнэх арга ашиглан шийд зөв болохыг баталж, тогтмолуудыг бодож гаргана

Жишээлбэл, дахин дуудагдах тэгшитгэлийн дээд хязгаарыг дараах байдлаар тодорхойлъё:

$$T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n). \quad (49)$$

Одоо дээрх тэгшитгэлийн дээд хязгаарыг $T(n) = O(n \lg n)$ хэмээн таамаглаж, түүнийгээ орлуулах аргаар баталъя.

Дахин дуудагдах тэгшитгэлийг орлуулах аргаар бодох

$n \geq n_0$ байх бүх n -ийн хувьд

$$T(n) \leq cn \lg n$$

байх нэгтгэн дүгнэх таамаглал дэвшиүүлье. Энд, $c > 0$ ба $n_0 > 0$ байх тогтмолуудыг хожим тогтооно.

Хэрэв $n \geq 2n_0$ бол $T(\lfloor n/2 \rfloor) \leq c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$ болно. (49)
томъёонд орлуулга хийвэл

$$\begin{aligned} T(n) &\leq 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + \Theta(n) \\ &\leq 2(c(n/2) \lg(n/2)) + \Theta(n) \\ &= cn \lg(n/2) + \Theta(n) \\ &= cn \lg n - cn \lg 2 + \Theta(n) \\ &= cn \lg n - cn + \Theta(n) \\ &\leq cn \lg n \end{aligned}$$

Дахин дуудагдах тэгшитгэлийг орлуулах аргаар бодох

Суурь нөхцөлд (base case) баталъя. Θөрөөр хэлбэл,
 $n_0 \leq n < 2n_0$ үед

$$T(n) \leq cn \lg n$$

гэж баталъя.

$$n_0 = 2 \text{ үед } T(2) \leq c < (2 \lg 2)c$$

$$n_0 = 3 \text{ үед } T(3) \leq c < (3 \lg 3)c$$

Дахин дуудагдах тэгшитгэлийг орлуулах аргаар бодох

Зөв таамаглал дэвшүүлэх арга №1:

$$T(n) = 2T(n/2 + 17) + \Theta(n)$$

тэгшитгэлийн хязгаарыг $T(n) = O(n \lg n)$ хэмээн таамаглана.

Зөв таамаглал дэвшүүлэх арга №2:

Доод хязгаар: $T(n) = \Omega(n)$,

Дээд хязгаар: $T(n) = O(n^2)$

Улмаар хязгааруудыг дундажлавал $T(n) = \Theta(n \lg n)$ болно.

Дахин дуудагдах тэгшитгэлийг орлуулах аргаар бодох

Алдаанаас зайлсхийх

Хязгаарын тэмдэглэгээг буруу ашиглах:

$$\begin{aligned}T(n) &\leq 2 \cdot O(\lfloor n/2 \rfloor) + \Theta(n) \\&= 2 \cdot O(n) + \Theta(n) \\&= O(n)\end{aligned}$$

Тогтмолыг буруу хялбарчлах:

$$\begin{aligned}T(n) &\leq 2(c\lfloor n/2 \rfloor) + \Theta(n) \\&= cn + \Theta(n) = O(n)\end{aligned}$$

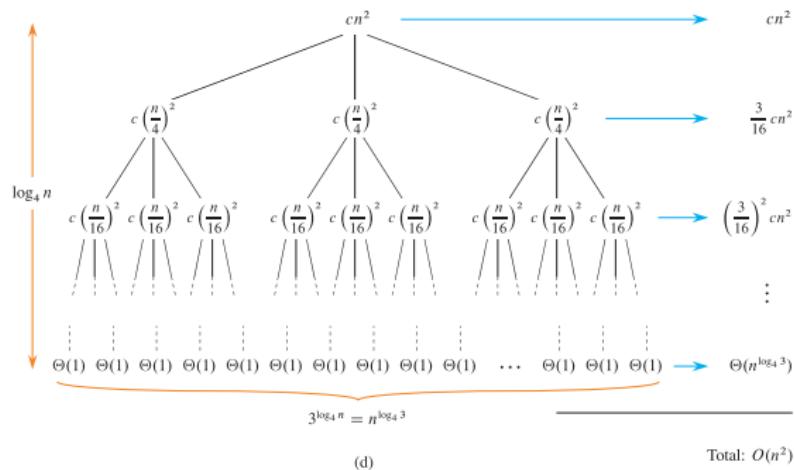
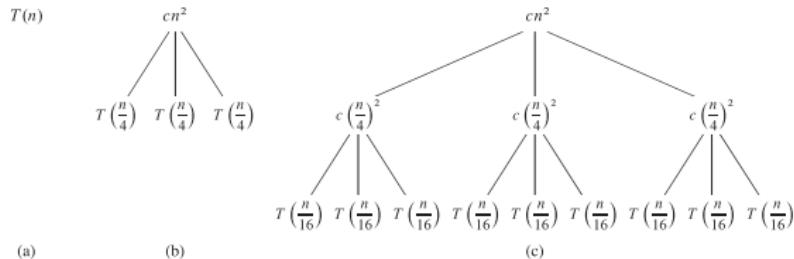
Дахин дуудагдах тэгшитгэл бодох давталтын модны арга

Тэгшитгэлийн шийдийн дээд хязгаарыг таамаглах:

$$T(n) = 3T(n/4) + \Theta(n^2) \quad (50)$$

$T(n) = 3T(n/4) + cn^2$ тэгшитгэлийн давталтын модыг дүрсэлье. Үүнд, $c > 0$.

Дахин дуудагдах тэгшитгэл бодох давталтын модны арга



Дахин дуудагдах тэгшитгэл бодох давталтын модны арга

Давталтын модны нийт өртөг:

$$\begin{aligned}T(n) &= cn^2 + \frac{3}{16}cn^2 + \left(\frac{3}{16}\right)^2 cn^2 + \cdots + \left(\frac{3}{16}\right)^{\log_4 n} cn^2 + \Theta(n^{\log_4 3}) \\&= \sum_{i=0}^{\log_4 n} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\&< \sum_{i=0}^{\infty} \left(\frac{3}{16}\right)^i cn^2 + \Theta(n^{\log_4 3}) \\&= \frac{1}{1 - (3/16)} cn^2 + \Theta(n^{\log_4 3}) \\&= \frac{16}{13} cn^2 + \Theta(n^{\log_4 3}) \\&= O(n^2)\end{aligned}$$

Дахин дуудагдах тэгшитгэл бодох давталтын модны арга

$T(n) = O(n^2)$ нь дээд хязгаар болохыг баталъя:

$$\begin{aligned} T(n) &\leq 3T(n/4) + cn^2 \\ &\leq 3d(n/4)^2 + cn^2 \\ &= \frac{3}{16}dn^2 + cn^2 \\ &\leq dn^2 \end{aligned}$$

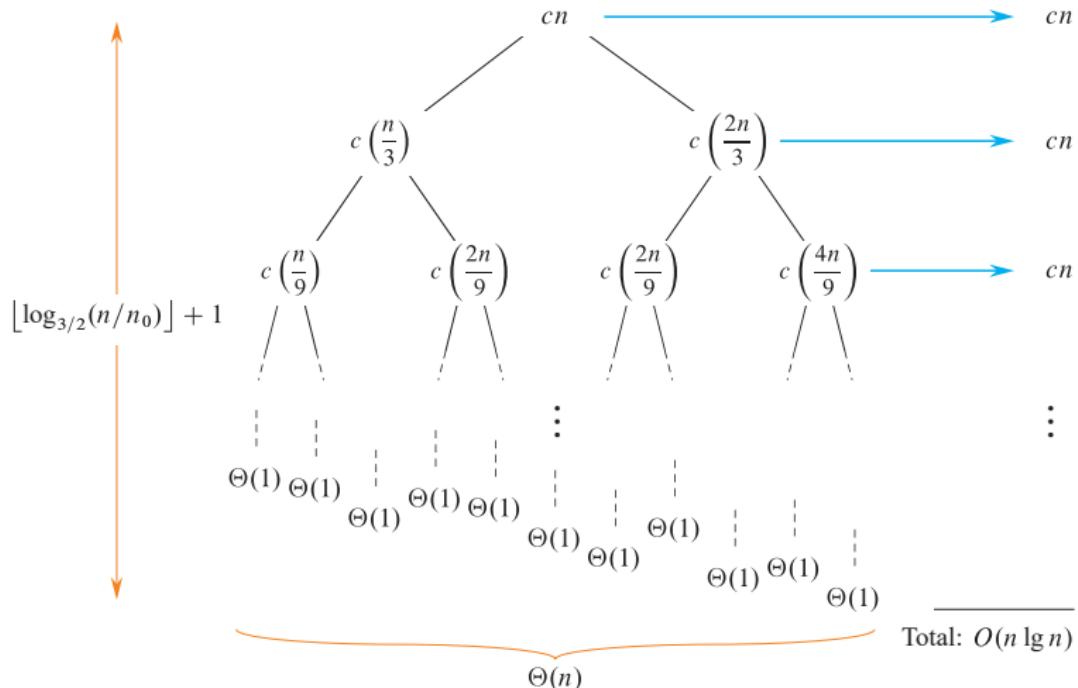
Дахин дуудагдах тэгшитгэл бодох давталтын модны арга

Ердийн бус тэгшитгэл:

$$T(n) = T(n/3) + T(2n/3) + \Theta(n) \quad (51)$$

Дээрх, дахин дуудагдах тэгшитгэл нь тэнцвэргүй мод үүсгэнэ.

Дахин дуудагдах тэгшитгэл бодох давталтын модны арга



Дахин дуудагдах тэгшитгэл бодох давталтын модны арга

Нийт навчны тоо:

$$L(n) = \begin{cases} 1, & \text{хэрэв } n < n_0 \text{ бол} \\ L(n/3) + L(2n/3) & \text{хэрэв } n \geq n_0 \text{ бол} \end{cases} \quad (52)$$

$T(n) = \Theta(n \lg n)$ гэдгийг орлуулах аргаар баталъя. $L(n) = O(n)$ гэж таамаглаад $d > 0$ тогтмолын хувьд $L(n) \leq dn$ гэсэн нэгдсэн дүгнэлт хийвэл:

$$\begin{aligned} L(n) &= L(n/3) + L(2n/3) \\ &\leq dn/3 + 2(dn)/3 \\ &\leq dn. \end{aligned}$$

Дахин дуудагдах тэгшитгэл бодох
ерөнхий арга

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Дахин дуудагдах өрөнхий тэгшитгэл:

$$T(n) = aT(n/b) + f(n)$$

энд $a > 0$ ба $b > 1$ байх тогтмолууд. $f(n)$ нь чиглүүлэгч функц.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Теорем 5.1 (Ерөнхий теорем)

$a > 0$ ба $b > 1$ байх тогтмолууд болон эерэг бодит тоогоор тодорхойлогдсон $f(n)$ чиглүүлэгч функцүүд өгөгдсөн байг. Тэгвэл $n \in \mathbb{N}$ байх $T(n)$ дахин дуудагдах тэгшитгэлийг

$$T(n) = aT(n/b) + f(n) \quad (53)$$

хэмээн тодорхойлъё. Үүнд, $aT(n/b)$ нь угтаа $a = a' + a''$ нөхцөлийг хангах зарим $a' \geq 0$ ба $a'' \geq 0$ тогтмолуудын хувьд $a'T(\lfloor n/b \rfloor) + a''T(\lceil n/b \rceil)$ байна.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Теорем 5.2 (...үргэлжлэл)

Тэгвэл $T(n)$ тэгшитгэлийн хязгаар нь дараах байдлаар тодорхойлогдоно:

1. Хэрэв $f(n) = O(n^{\log_b a - \epsilon})$ байх $\epsilon > 0$ тогтмол оршин байвал $T(n) = \Theta(n^{\log_b a})$ байна.
2. Хэрэв $f(n) = \Theta(n^{\log_b a} \lg^k n)$ байх $k \geq 0$ тогтмол оршин байвал $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$ байна.
3. Хэрэв $f(n) = \Omega(n^{\log_b a + \epsilon})$ байх $\epsilon > 0$ тогтмол оршин байгаад $f(n)$ функц нь $c < 1$ тогтмол болон дурын p тооны хувьд $af(n/b) \leq cf(n)$ тогтмол нөхцөлийг хангаж байвал $T(n) = \Theta(f(n))$ байна.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Дээрх гурван тохиолдолд $f(n)$ функцийг $n^{\log_b a}$ урсгал функцтэй харьцуулна:

- ▶ хэрэв урсгал функцийн хязгаарын өсөлт нь чиглүүлэгч функцээс их бол 1-р тохиолдлыг ашиглана
- ▶ хэрэв хоёр функцийн өсөлт ойролцоо бол 2-р тохиолдлыг ашиглана
- ▶ хэрэв чиглүүлэгч функцийн хязгаарын өсөлт нь урсгал функцийн өсөлтөөс их бол 3-р тохиолдлыг ашиглана

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

1-р тохиолдолд урсгал функцийн хязгаарын өсөлт нь чиглүүлэгч функцийнхээс их төдийгүй, олон гишүүнт функцээр хурдан өснө.

Өөрөөр хэлбэл, урсгал $n^{\log_b a}$ функц нь чиглүүлэгч функцээс ядаж $\Theta(n^\epsilon)$ үржүүлэхүүнээр их байх ёстой гэсэн үг. Ерөнхий теоремоор $T(n) = \Theta(n^{\log_b a})$ байна.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

2-р тохиолдолд урсгал болон чиглүүлэгч функцүүдийн өсөлтийн хязгаар ойролцоо байна.

Ерөнхий теоремоор $f(n)$ функцэд $\lg n$ үржүүлэхүүнийг нэмэх бөгөөд шийд нь $T(n) = \Theta(n^{\log_b a} \lg^k n)$ болно.

Бодит байдал дээр, $k = 0$ үед чиглүүлэгч болон урсгал функцүүдийн хязгаарын өсөлт ижил байгаад, шийд нь $T(n) = \Theta(n^{\log_b a} \lg n)$ байна.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

З-р тохиолдолд чиглүүлэгч функц нь урсгал функцээс хязгаарын өсөлт ихтэй байхаас гадна, олон гишүүнт функцийн өсөлт нь их байна.

Чиглүүлэгч функц нь $af(n/b) \leq cf(n)$ нөхцөлийг хангах ёстай.

Ерөнхий теоремоор шийд нь $T(n) = \Theta(f(n))$ байна.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №1:

$$T(n) = 9T(n/3) + n$$

$a = 9$, $b = 3$ гэдгээс $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$.

Дурын $\epsilon \leq 1$ тогтмолын хувьд $f(n) = n = O(n^{2-\epsilon})$ тул өрөнхий теоремын 1-р нөхцөлийг ашиглавал $T(n) = \Theta(n^2)$.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №2:

$$T(n) = T(2n/3) + 1$$

$a = 1, b = 3/2$ гэдгээс $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1$.

$f(n) = 1 = \Theta(n^{\log_b a} \lg^0 n) = \Theta(1)$ тул 2-р тохиолдлыг ашиглана.

Ийнхүү шийд нь $T(n) = \Theta(\lg n)$.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №3:

$$T(n) = 3T(n/4) + n \lg n$$

$$a = 3, \quad b = 4 \text{ гэдгээс } n^{\log_b a} = n^{\log_4 3} = O(n^{0.793}).$$

$f(n) = n \lg n = \Omega(n^{\log_4 3 + \epsilon})$, энд ϵ нь ойролцоогоор 0.2 үед $f(n)$ чиглүүлэгч функцийн хувьд 3-р нөхцөлийг ашиглай.

$c = 3/4$ үед дурын n тооны хувьд

$af(n/b) = 3(n/4) \lg(n/4) \leq (3/4)n \lg n = cf(n)$ болно. Ийнхүү 3-р нөхцөл ёсоор $T(n) = \Theta(n \lg n)$ болно.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №4:

$$T(n) = 2T(n/2) + n \lg n$$

$a = 2$, $b = 2$ тул $n^{\log_b a} = n^{\log_2 2} = n$.

$f(n) = n \lg n = \Theta(n^{\log_b a} \lg^1 n)$ тул 2-р нөхцөл ёсоор

$T(n) = \Theta(n \lg^2 n)$ болно.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №5:

$$T(n) = 2T(n/2) + \Theta(n)$$

$a = 2$, $b = 2$ тул урсгал функц нь $n^{\log_b a} = n^{\log_2 2} = n$.

$f(n) = \Theta(n)$ тул 2-р нөхцөлийг ашиглавал шийд нь

$T(n) = \Theta(n \lg n)$ болно.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №6:

$$T(n) = 8T(n/2) + \Theta(1)$$

$a = 8$, $b = 2$ тул урсгал функц нь $n^{\log_b a} = n^{\log_2 8} = n^3$ болно.

n^3 нь олон гишүүнтнийн өсөлтөөр $f(n) = \Theta(1)$ чиглүүлэгч функцээс их.

Учир нь 1-р нөхцөл ёсоор дурын $\epsilon < 3$ хувьд $f(n) = O(n^{3-\epsilon})$.

Ийнхүү $T(n) = \Theta(n^3)$.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ №7:

$$T(n) = 7T(n/2) + \Theta(n^2)$$

$a = 7$, $b = 2$ тул урсгал функц нь $n^{\log_b a} = n^{\lg 7}$.

$\lg 7 = 2.807355\dots$ гэдгийг ашиглаад, $\epsilon = 0.8$ хэмээн сонговол чиглүүлэгч функцийн хязгаар нь $f(n) = \Theta(n^2) = O(n^{\lg 7 - \epsilon})$ болно.

1-р нөхцөл ёсоор $T(n) = \Theta(n^{\lg 7})$ болно.

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Ерөнхий аргыг ашиглаж болохгүй тохиолдол:

- ▶ урсгал болон чиглүүлэгч функцүүдийг хязгаараар нь харьцуулах боломжгүй тохиолдол
- ▶ $f(n) = o(n^{\log_b a})$ үед урсгал функц нь чиглүүлэгч функцээс олон гишүүнт функцийн хязгаарын өсөлтөөр ихгүй ч 1 болон 2-р нөхцөлийн хооронд зааг гарч болно
- ▶ $f(n) = \omega(n^{\log_b a})$ үед чиглүүлэгч функц нь урсгал функцээс хязгаарын зэрэгт логарифм функцийн өсөлтөөрөө хурдан ч, олон гишүүнт функцийн өсөлтөөр хурдан биш үед 2 болон 3-р нөхцөлийн хооронд зааг үүсэж болно

Дахин дуудагдах тэгшитгэл бодох өрөнхий арга

Жишээ:

$$T(n) = 2T(n/2) + n/\lg n$$

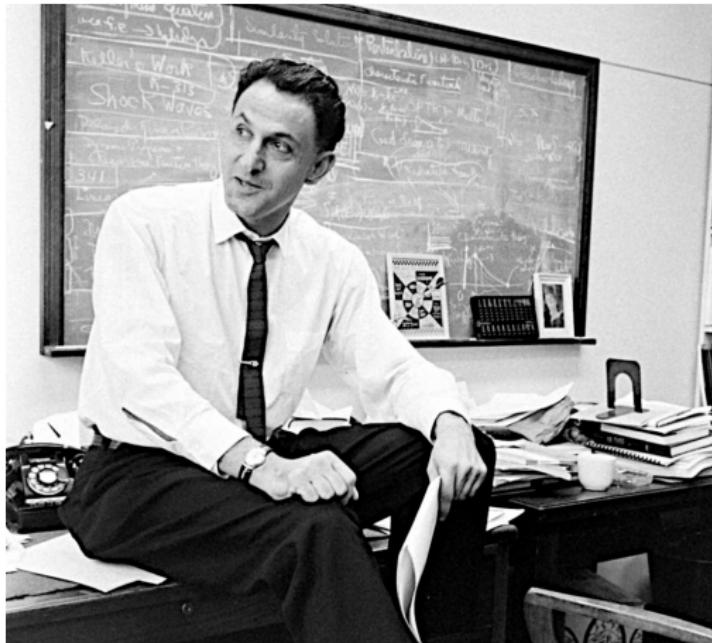
$a = 2$, $b = 2$ тул урсгал функц нь $n^{\log_b a} = n^{\log_2 2} = n^1 = n$ болно.

Чиглүүлэгч функц нь $n/\lg n = o(n)$ буюу хязгаарын өсөлтөөрөө урсгал n функцээс удаан.

$\epsilon > 0$ үед $n/\lg n = O(n^{\log_b a - \epsilon})$ тул 1-р нөхцөлийг хангах тогтмол олдохгүй. Мөн $k = -1$ үед $n/\lg n = \Theta(n^{\log_b a} \lg^k n)$ байх тул 2-р нөхцөл биелэгдэхгүй, учир нь k нь ээрэг байх ёстой.

Динамик программчал

Динамик программчлал



Зураг 5: Ричард Беллман

Динамик программчлал

Динамик программчлалын алгоритмыг хэрэгжүүлэхэд дараах 4 алхмыг мөрдөнө:

1. Оновчтой шийдийн бүтцийг тодорхойлох
2. Оновчтой шийдийн утгыг дахин дуудалт ашиглан тодорхойлох
3. Ихэвчлэн доороос дээшээ зарчмаар оновчтой шийдийн утгыг тооцоолох
4. Оновчтой шийдийг тооцоолсон мэдээллээс гарган авах

Ган савааг хуваах

Ган савааг хуваах

Ган савааг хуваах:

$$n = i_1 + i_2 + \cdots + i_k \quad 1 \leq k \leq n$$

$r_n = p_{i_1} + p_{i_2} + \cdots + p_{i_k}$ i_1, i_2, \dots, i_k урттай хэсэг болгох

Ган савааг хуваах

i үрт	1	2	3	4	5	6	7	8	9	10
p_i γнэ	1	5	8	9	10	17	17	20	24	30



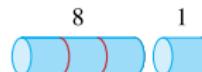
(a)



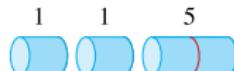
(b)



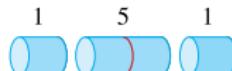
(c)



(d)



(e)



(f)



(g)



(h)

Ган савааг хуваах

i урт	1	2	3	4	5	6	7	8	9	10
p_i үнэ	1	5	8	9	10	17	17	20	24	30

$i = 1, 2, \dots, 10$ байх r_i өндөр орлогыг доорх байдлаар тооцно:

$$r_1 = 1 \quad \text{шийдийн дэд бодлого нь: } 1 = 1 \text{ (хуваахгүй)}$$

$$r_2 = 5 \quad \text{шийдийн дэд бодлого нь: } 2 = 2 \text{ (хуваахгүй)}$$

$$r_3 = 8 \quad \text{шийдийн дэд бодлого нь: } 3 = 3 \text{ (хуваахгүй)}$$

$$r_4 = 10 \quad \text{шийдийн дэд бодлого нь: } 4 = 2 + 2$$

$$r_5 = 13 \quad \text{шийдийн дэд бодлого нь: } 5 = 2 + 3$$

$$r_6 = 17 \quad \text{шийдийн дэд бодлого нь: } 6 = 6 \text{ (хуваахгүй)}$$

$$r_7 = 18 \quad \text{шийдийн дэд бодлого нь: } 7 = 1 + 6 \text{ эсвэл } 7 = 2 + 2 + 3$$

$$r_8 = 22 \quad \text{шийдийн дэд бодлого нь: } 8 = 2 + 6$$

$$r_9 = 25 \quad \text{шийдийн дэд бодлого нь: } 9 = 3 + 6$$

$$r_{10} = 30 \quad \text{шийдийн дэд бодлого нь: } 10 = 10 \text{ (хуваахгүй)}$$

Ган савааг хуваах

2 дэд бодлого болгон задлах:

$$r_n = \max\{p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \dots, r_{n-1} + r_1\} \quad n \geq 1 \quad (54)$$

Ган савааг хуваах

1 дэд бодлого болгон задлах:

$$r_n = \max\{p_i + r_{n-i} : 1 \leq i \leq n\}. \quad (55)$$

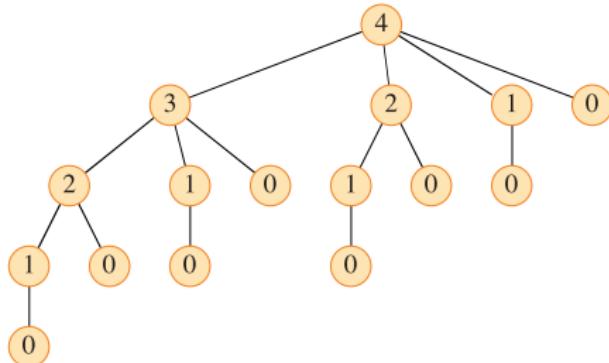
Ган савааг хуваах

(55) томьёог дээрээс доош дахин дуудагдах аргаар хэрэгжүүлбэл:

Cut-Rod(p, n)

```
1  if  $n == 0$ 
2      return 0
3   $q = -\infty$ 
4  for  $i = 1$  to  $n$ 
5       $q = \max\{q, p[i] + \text{Cut-Rod}(p, n - i)\}$ 
6  return  $q$ 
```

Ган савааг хуваах



$$T(n) = \begin{cases} 1, & \text{хэрэв } n = 0 \\ 1 + \sum_{j=0}^{n-1} T(j), & \text{хэрэв } n \geq 1 \end{cases}$$

Ажиллах хугацаа нь $T(n) = 2^n$.

Ган савааг хуваах

Динамик программчлалын арга нь олон гишүүнт функцийн хугацаанд ажилладаг бөгөөд бодлогыг 2 янзаар боддог:

- ▶ top-down with memoization
- ▶ bottom-up

Ган савааг хуваах

Memoized-Cut-Rod(p, n)

```
1 let  $i = 0$  be a new array
2 for  $i = 0$  to  $n$ 
3    $r[i] = -\infty$ 
4 return Memoized-Cut-Rod-Aux( $p, n, r$ )
```

Memoized-Cut-Rod-Aux(p, n, r)

```
1 if  $r[n] \geq 0$ 
2   return  $r[n]$ 
3 if  $n == 0$ 
4    $q = 0$ 
5 else  $q = -\infty$ 
6 for  $i = 1$  to  $n$ 
7    $q = \max\{q, p[i] + \text{Memoized-Cut-Rod-Aux}(p, n - i, r)\}$ 
8  $r[n] = q$ 
9 return  $q$ 
```

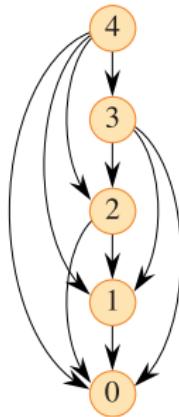
Ган савааг хуваах

Bottom-Up-Cut-Rod(p, n)

```
1 let  $r[0 : n]$  be a new array
2  $r[0] = 0$ 
3 for  $j = 1$  to  $n$ 
4      $q = -\infty$ 
5     for  $i = 1$  to  $j$ 
6          $q = \max\{q, p[i] + r[j - i]\}$ 
7          $r[j] = q$ 
8 return  $r[n]$ 
```

Дээрээс доош, доороос дээш аргачлалууд аль аль нь $\Theta(n^2)$ хугацаанд ажиллана.

Ган савааг хуваах



Зураг 6: Дэд бодлогуудын хоорондын хамаарлыг ойлгоход чиглэлт граф тусалдаг. Дэд бодлогын $G = (V, E)$ графын хэмжээ нь динамик программчлалын алгоритмын ажиллах хугацааг тодорхойлдог.

Ган савааг хуваах

Extended-Bottom-Up-Cut-Rod(p, n)

```
1 let  $r[0 : n]$  and  $s[1 : n]$  be new arrays
2  $r[0] = 0$ 
3 for  $j = 1$  to  $n$ 
4      $q = -\infty$ 
5     for  $i = 1$  to  $j$ 
6         if  $q < p[i] + r[j - i]$ 
7              $q = p[i] + r[j - i]$ 
8              $s[j] = i$ 
9      $r[j] = q$ 
10 return  $r$  and  $s$ 
```

Print-Cut-Rod-Solution(p, n)

```
1  $(r, s) = \text{Extended-Bottom-Up-Cut-Rod}(p, n)$ 
2 while  $n > 0$ 
3     print  $s[n]$ 
4      $n = n - s[n]$ 
```

Ган савааг хуваах

Савааг хуваах шийдэл:

i	0	1	2	3	4	5	6	7	8	9	10
$r[i]$	0	1	5	8	10	13	17	18	22	25	30
$s[i]$		1	2	3	2	2	6	1	2	3	10

Энд, $r[i]$ нь өндөр орлого, $s[i]$ нь хуваалтын эхний хэсэг.

Матрицуудын цуваа үржвэр

Матрицуудын цуваа үржвэр

n ширхэг цуваагаар (chain sequence) өгөгдсөн $\langle A_1, A_2, \dots, A_n \rangle$ матрицын үржвэрийг олох:

$$A_1 A_2 \cdots A_n. \quad (56)$$

Үүнд, матриц бүр нь дөрвөлжин байх албагүй.

Матрицуудын цуваа үржвэр

Хэрэв 4 ширхэг матриц $\langle A_1, A_2, A_3, A_4 \rangle$ цуваагаар өгөгдсөн гэвэл хаалтыг нийт 5 байрлалд ашиглаж болно:

$$(A_1(A_2(A_3A_4))),$$

$$(A_1((A_2A_3)A_4)),$$

$$((A_1A_2)(A_3A_4)),$$

$$((A_1(A_2A_3))A_4),$$

$$(((A_1A_2)A_3)A_4).$$

Хэрэв матрицуудын цуваа нь 3 бол:

$$((A_1A_2)A_3),$$

$$(A_1(A_2A_3)).$$

Матрицуудын цуваа үржвэр

Матрицуудын цувааг бүлэглэх боломж:

$$P(n) = \begin{cases} 1, & \text{хэрэв } n = 1 \text{ бол,} \\ \sum_{k=1}^{n-1} P(k)P(n-k), & \text{хэрэв } n \geq 2 \text{ бол.} \end{cases} \quad (57)$$

Нийт боломж нь Каталаны тоотой тэнцдэг:

$$P(n) = C_{n-1} = \frac{1}{n} \binom{2n-2}{n-1} = \frac{(2n-2)!}{n!(n-1)!}.$$

Гэвэл алгоритмын зарцуулах хугацаа нь $\Omega(4^n/n^{3/2}) \approx \Omega(2^n)$.

Матрицуудын цуваа үржвэр

Матрицуудын цувааг динамик программчлалын аргаар бүлэглэх:

1. Оновчтой шийдийн бүтцийг тодорхойлох
2. Оновчтой шийдийн утгыг дахин ашиглах
3. Оновчтой шийдийн утгыг тооцоолох
4. Тооцоолсон утгуудаас оновчтой шийдийг тодорхойлох

Матрицуудын цуваа үржвэр

Алхам 1: Оновчтой бүлэглэх бүтэц

- ▶ Матрицуудын цуваа үржвэр: $A_i A_{i+1} \cdots A_j = A_{i:j}$, ($i \leq j$)
- ▶ $A_{i:j}$ цуваа матрицын хамгийн бага зардал нь $A_{i:k}$ болон $A_{k+1:j}$ цуваа матрицуудын зардал болон тэдгээрийг үржүүлэх зардлын нийлбэртэй тэнцэнэ

Матрицуудын цуваа үржвэр

Алхам 2: Шийдийг дахин дуудагдах

$$m[i, j] = \begin{cases} 0, & i = j, \\ \min\{m[i, k] + m[k + 1, j] + p_{i-1}p_kp_j : i \leq k < j\}, & i < j. \end{cases} \quad (58)$$

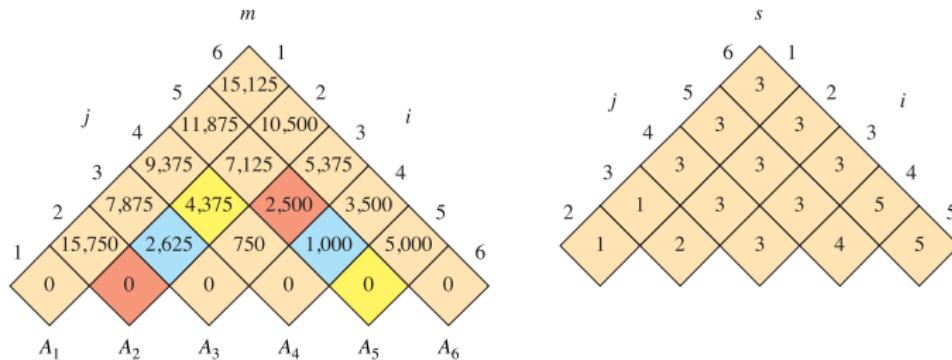
Матрицуудын цуваа үржвэр

Алхам 3: Оновчтой зардлыг тооцоолох

Matrix-Chain-Order(p, n)

```
1    $m[1 : n, 1 : n]$  ба  $s[1 : n - 1, 2 : n]$  нь шинэ хүснэгт байг
2   for  $i = 1$  to  $n$ 
3        $m[i, i] = 0$ 
4   for  $l = 2$  to  $n$ 
5       for  $i = 1$  to  $n - l + 1$ 
6            $j = i + l - 1$ 
7            $m[i, j] = \infty$ 
8           for  $k = i$  to  $j - 1$ 
9                $q = m[i, k] + m[k + 1, j] + p_{i-1}p_k p_j$ 
10              if  $q < m[i, j]$ 
11                   $m[i, j] = q$ 
12                   $s[i, j] = k$ 
13      return  $m$  ба  $s$ 
```

Матрицуудын цуваа үржвэр



Зураг 7: Матрицуудын цувааны дараалал

Матриц	\$A_1\$	\$A_2\$	\$A_3\$	\$A_4\$	\$A_5\$	\$A_6\$
Хэмжээс	\$30 \times 35\$	\$35 \times 15\$	\$15 \times 5\$	\$5 \times 10\$	\$10 \times 20\$	\$20 \times 25\$

$$m[2, 5] = \min \begin{cases} m[2, 2] + m[3, 5] + p_1 p_2 p_5 = 0 + 2500 + 35 \cdot 15 \cdot 20 = 13,000. \\ m[2, 3] + m[4, 5] + p_1 p_3 p_5 = 2625 + 1000 + 35 \cdot 5 \cdot 20 = 7,125. \\ m[2, 4] + m[5, 5] + p_1 p_4 p_5 = 4375 + 0 + 35 \cdot 10 \cdot 20 = 11,375. \end{cases}$$

$$= 7,125.$$

Матрицуудын цуваа үржвэр

Алхам 4: Оновчтой шийдийг тодорхойлох

Print-Optimal-Parens(s, i, j)

```
1  if  $i == j$ 
2      print "A";
3  else print "("
4      Print-Optimal-Parens( $s, i, s[i, j]$ )
5      Print-Optimal-Parens( $s, s[i, j] + 1, j$ )
6      print ")"
```

Print-Optimal-Parens($s, 1, 6$) $\Rightarrow ((A_1(A_2A_3))((A_4A_5)A_6))$

Динамик программчлал

Динамик программчлалын шинжүүр

Динамик программчлалын шинжүүр

Сэдвийн хүрээнд судлах асуудал:

- ▶ Оновчтой дэд бүтэц
- ▶ Давхцах дэд бодлогууд

Динамик программчлалын шинжүүр

Оновчтой дэд бүтцийг олох:

1. Бодлогын шийдэл нь сонголтод тулгуурласан эсэхийг шалгана
2. Улмаар өгөгдсөн бодлогын хувьд оновчтой шийдэлд хүргэх сонголт өгөгдсөн гэж үзнэ
3. Тухайн сонголтод ямар дэд бодлого хамаарах, тэдгээрийн хамрах мужийг хэрхэн зөв тодорхойлохыг шийднэ
4. Бодлогын оновчтой шийдэд ашиглагдсан дэд бодлогын шийдэл нь оновчтой байх ёстой гэдгийг орлуулах аргаар батална

Динамик программчлалын шинжүүр

Оновчтой дэд бүтэц нь бодлогын хувьд дараах хоёр төрлөөр тодорхойлогдоно:

1. Үндсэн бодлогын оновчтой шийдэлд хэдэн дэд бодлого ашиглагдаж байгаа
2. Оновчтой шийдийг байгуулахад оролцож байгаа дэд бодлогыг хэчнээн сонголтоор тодорхойлж байгаа.

Динамик программчлалын шинжүүр

Динамик программчлалын алгоритмын ажиллах хугацаа нь дараах хоёр хүчин зүйлээс хамаардаг:

1. дэд бодлогын тоо
2. дэд бодлого болгон дахь сонголтын тоо

Динамик программчлалын шинжүүр

Оновчтой дэд бүтэц нь бодлого бүрд үйлчлэх албагүй:

Хамгийн богино зам Ирмэгийн жин өгөгдөөгүй, мөчлөггүй
шулуун графын хувьд *и* оройгоос *и* орой хүртэлх
хамгийн цөөн ирмэгийн тоог олох

Хамгийн урт шулуун зам *и* оройгоос *и* орой хүртэлх хамгийн
урт шулуун замыг олох

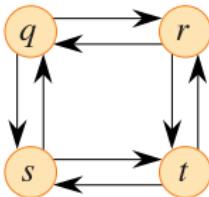
Динамик программчлалын шинжүүр

Ирмэгийн жин өгөгдөөгүй богино зам олох бодлого нь оновчтой дэд бүтэцтэй:

- ▶ $u \neq v$ үед u оройгоос v орой хүртэлх дурын p замд ω орой оршдог
- ▶ Тэгвэл $u \xrightarrow{p} v$ замыг $u \xrightarrow{p_1} \omega \xrightarrow{p_2} v$ хэмээн хувааж болно
- ▶ p замын ирмэгийн тоо нь p_1 болон p_2 замын ирмэгүүдийн тоотой тэнцэнэ
- ▶ Хэрэв p нь u оройгоос v орой хүртэлх богино зам бол p_1 нь u оройгоос ω орой хүртэлх богино зам байна

Динамик программчлалын шинжүүр

Ирмэгийн жин өгөгдөөгүй хамгийн урт зам олох бодлого нь оновчтой дэд бүтээцтэй юу?

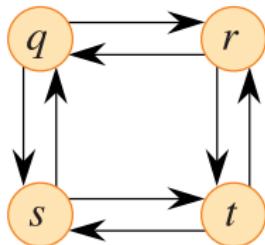


Зураг 8: Чиглэлт граф

$q \rightarrow r \rightarrow t$ нь q -ээс t хүртэлх хамгийн урт шулуун зам бол:

- ▶ $q \rightarrow r$ нь q оройгоос r орой хүртэлх хамгийн урт шулуун зам мөн үү
- ▶ $r \rightarrow t$ нь r оройгоос t орой хүртэлх хамгийн урт шулуун зам мөн үү

Динамик программчалын шинжүүр



- ▶ Урт зам олох, ирмэгийн жин өгөгдөөгүй бодлогын хувьд дэд бодлого нь хараат бүтэцтэй

Динамик программчлалын шинжүүр

Богино зам олох, ирмэгийн жин өгөгдөөгүй бодлогын хувьд дэд бодлогууд нь хоорондоо хамааралгүй, бие даасан бүтэцтэй.

Баталгаа:

- ▶ $u \xrightarrow{p_1} \omega$
- ▶ $\omega \xrightarrow{p_2} v$
- ▶ $p = u \xrightarrow{p_1} \omega \xrightarrow{p_2} v$

Эсрэг баталгаа:

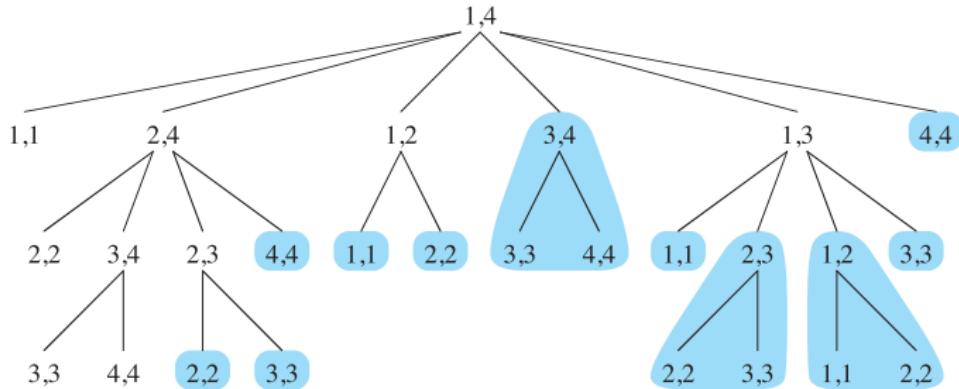
- ▶ $u \xrightarrow{p_{ux}} x \rightsquigarrow \omega$
- ▶ $\omega \rightsquigarrow x \xrightarrow{p_{xv}} v$
- ▶ $p' = u \xrightarrow{p_{ux}} x \xrightarrow{p_{xv}} v$

Динамик программчлалын шинжүүр

Оновчлолын бодлогод динамик программчлалын аргыг ашиглах нөхцөл:

- ▶ Дахин дуудагдах дэд бодлогуудын муж нь бага буюу дэд бодлогын тоо цөөн байх
- ▶ $m[3, 4]$ өртөг нь 4 удаа ашиглагдсан:
 $m[2, 4]$, $m[1, 4]$, $m[3, 5]$ $m[3, 6]$

Динамик программчалын шинжүүр



Зураг 9: Recursive-Matrix-Chain($p, 1, 4$) алгоритмын дахин дуудагдах модны цэнхэр өнгөөр тэмдэглэсэн мөчир болон зангилаануудын утга нь Memoized-Matrix-Chain функцээс дахин дуудагдаж байгаа болно.

Динамик программчалын шинжүүр

Recursive-Matrix-Chain(p, i, j)

```
1  if  $i == j$ 
2      return 0
3   $m[i, j] = \infty$ 
4  for  $k = i$  to  $j - 1$ 
5       $q = \text{Recursive-Matrix-Chain}(p, i, k)$ 
       +  $\text{Recursive-Matrix-Chain}(p, k + 1, j)$ 
       +  $p_{i-1} p_k p_j$ 
6      if  $q < m[i, j]$ 
7           $m[i, j] = q$ 
8  return  $m[i, j]$ 
```

Динамик программчлалын шинжүүр

$$T(n) \geq \begin{cases} 1, & \text{хэрэв } n = 1 \text{ бол,} \\ 1 + \sum_{k=1}^{n-1} (T(k) + T(n-k) + 1), & \text{хэрэв } n > 1 \text{ бол.} \end{cases}$$

$$T(n) \geq 2 \sum_{i=1}^{n-1} T(i) + n \quad (59)$$

Одоо орлуулах аргаар $\Omega(2^n)$ буюу $T(n) \geq 2^{n-1}$ болохыг нэгтгэн дүгнэх аргаар баталбал:

$$\begin{aligned} T(n) &\geq 2 \sum_{i=1}^{n-1} 2^{i-1} + n = 2 \sum_{j=0}^{n-2} 2^j + n \\ &= 2(2^{n-1} - 1) + n \quad (\text{геометр цувааны томьёогоор}) \\ &= 2^n - 2 + n \geq 2^{n-1} \end{aligned}$$

Динамик программчалын шинжүүр

Memoized-Matrix-Chain(p, n)

```
1   $m[1 : n, 1 : n]$  нь шинэ хүснэгт байг
2  for  $i = 1$  to  $n$ 
3      for  $j = i$  to  $n$ 
4           $m[i, j] = \infty$ 
5  return Lookup-Chain( $m, p, 1, n$ )
```

Lookup-Chain($m, p, 1, n$)

```
1  if  $m[i, j] < \infty$ 
2      return  $m[i, j]$ 
3  if  $i == j$ 
4       $m[i, j] = 0$ 
5  else for  $k = i$  to  $j - 1$ 
6       $q = \text{Lookup-Chain}(m, p, i, k)$ 
         +  $\text{Lookup-Chain}(m, p, k + 1, j) + p_{i-1} p_k p_j$ 
7      if  $q < m[i, j]$ 
8           $m[i, j] = q$ 
9  return  $m[i, j]$ 
```

Динамик программчлал

Хамгийн урт нийтлэг дэд дараалал

Хамгийн урт нийтлэг дэд дараалал

ДНХ дарааллын суурь (base):

- A Аденин
- C Цитозин
- G Гуанин
- T Тиамин

Жишээ дарааллуудын төсөө:

$S_1 = ACCGGTCGAGTGC GCG GAA AGCCGGCCGAA,$

$S_2 = GTCGTTCGGAATGCCGTTGCTCTGTAAA,$

$S_3 = GTCGT CGGAAGCCGGCCGAA$

Хамгийн урт нийтлэг дэд дараалал

Longest Common Subsequence (LCS):

- ▶ $X = \langle x_1, x_2, \dots, x_m \rangle$ дараалал өгөгдсөн
- ▶ Хэрэв $x_{i_j} = z_j$ ($j = 1, 2, \dots, k$) байх $\langle i_1, i_2, \dots, i_k \rangle$ өсөх дараалал олдвол
- ▶ $Z = \langle z_1, z_2, \dots, z_k \rangle$ нь X дарааллын дэд дараалал болно

Жишээ нь: $Z = \langle B, C, D, B \rangle$ нь $X = \langle A, B, C, B, D, A, B \rangle$ дарааллын хувьд $\langle 2, 3, 5, 7 \rangle$ дараалсан товъёгоор дэд дараалал болно.

Хамгийн урт нийтлэг дэд дараалал

Өгөгдсөн X болон Y дарааллын хувьд Z дараалал нь X болон Y дарааллын дэд дараалал бол Z дарааллыг X болон Y дарааллын **нийтлэг дэд дараалал** гэдэг.

$X = \langle A, B, C, B, D, A, B \rangle$ ба $Y = \langle B, D, C, A, B, A \rangle$ дарааллын хувьд $\langle B, C, A \rangle$ нь нийтлэг дэд дараалал болно.

Хамгийн урт нийтлэг дэд дараалал

Өгөгдсөн $X = \langle x_1, x_2, \dots, x_m \rangle$ дарааллын хувьд X дарааллын i дэх утварыг $i = 0, 1, \dots, m$ байх $X_i = \langle x_1, x_2, \dots, x_i \rangle$ хэмээн тодорхойлъё.

Жишээлбэл $X = \langle A, B, C, B, D, A, B \rangle$ бол $X_4 = \langle A, B, C, B \rangle$ ба X_0 нь хоосон дараалал гэсэн үг.

Хамгийн урт нийтлэг дэд дараалал

Теорем 6.1 (LCS бодлогын оновчтой дэд бүтэц)

$X = \langle x_1, x_2, \dots, x_m \rangle$ ба $Y = \langle y_1, y_2, \dots, y_n \rangle$ дарааллууд өгөгдсөн бөгөөд тэдгээрийн хамгийн урт нийтлэг дэд дараалал (LCS) нь $Z = \langle z_1, z_2, \dots, z_k \rangle$ байг. Тэгвэл:

1. Хэрэв $x_m = y_n$ бол $z_k = x_m = y_n$ байх ба Z_{k-1} нь X_{m-1} болон Y_{n-1} дарааллуудын LCS байна.
2. Хэрэв $x_m \neq y_n$ ба $z_k \neq x_m$ бол Z нь X_{m-1} болон Y дарааллуудын LCS байна.
3. Хэрэв $x_m \neq y_n$ ба $z_k \neq y_n$ бол Z нь X болон Y_{n-1} дарааллуудын LCS байна.

Хамгийн урт нийтлэг дэд дараалал

$$c[i, j] = \begin{cases} 0 & i = 0 \text{ буюу } j = 0, \\ c[i - 1, j - 1] + 1 & i, j > 0 \text{ ба } x_i = y_j, \\ \max\{c[i, j - 1], c[i - 1, j]\} & i, j > 0 \text{ ба } x_i \neq y_j. \end{cases} \quad (60)$$

Хамгийн урт нийтлэг дэд дараалал

LCS-Length(X, Y, m, n)

```
1   $b[1 : m, 1 : n]$  ба  $c[0 : m, 0 : n]$  нь хүснэгт
2  for  $i = 1$  to  $m$ 
3     $c[i, 0] = 0$ 
4  for  $j = 0$  to  $n$ 
5     $c[0, j] = 0$ 
6  for  $i = 1$  to  $m$ 
7    for  $j = 1$  to  $n$ 
8      if  $x_i == y_j$ 
9         $c[i, j] = c[i - 1, j - 1] + 1$ 
10        $b[i, j] = "↖"$ 
11     elseif  $c[i - 1, j] \geq c[i, j - 1]$ 
12        $c[i, j] = c[i - 1, j]$ 
13        $b[i, j] = "↑"$ 
14     else  $c[i, j] = c[i, j - 1]$ 
15        $b[i, j] = "←"$ 
16 return  $c$  ба  $b$ 
```

Print-LCS(b, X, i, j)

```
1  if  $i == 0$  буюу  $j == 0$ 
2    return
3  if  $b[i, j] == "↖"$ 
4    Print-LCS( $b, X, i - 1, j - 1$ )
5    print  $x_i$ 
6  elseif  $b[i, j] == "↑"$ 
7    Print-LCS( $b, X, i - 1, j$ )
8  else Print-LCS( $b, X, i, j - 1$ )
```

Хамгийн урт нийтлэг дэд дараалал

	j	0	1	2	3	4	5	6
i	y_j	B	D	C	A	B	A	
0	x_i	0	0	0	0	0	0	0
1	A	0	0	0	0	1	-1	1
2	B	0	1	-1	-1	1	2	-2
3	C	0	1	1	2	-2	2	2
4	B	0	1	1	2	2	3	-3
5	D	0	1	2	2	2	3	3
6	A	0	1	2	2	3	3	4
7	B	0	1	2	2	3	4	4

Зураг 10: LCS байгуулах

Хоёртын оновчтой хайлтын мод

Хоёртын оновчтой хайлтын мод

Хоёртын хайлтыг удаашруулж болох тохиолдлууд:

- ▶ өндөр давтамжтай үгс модны үндсээс хол гүнд байрлах
- ▶ цөөн давтамжтай үгс модны үндсэнд ойр байрлах
- ▶ зарим үгс хайлтын модонд үл багтаж

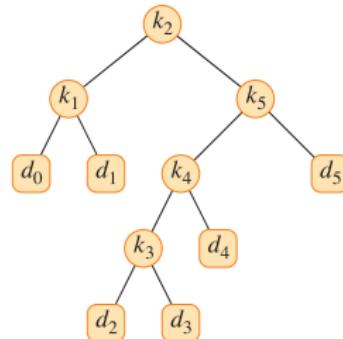
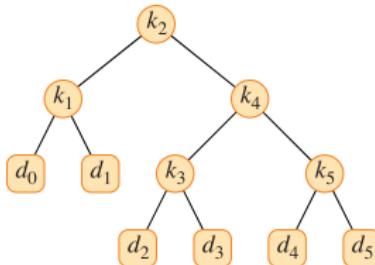
Хоёртын оновчтой хайлтын мод

i	0	1	2	3	4	5
p_i		0.15	0.10	0.05	0.10	0.20
q_i	0.05	0.10	0.05	0.05	0.05	0.10

$$\sum_{i=1}^n p_i + \sum_{i=0}^n q_i = 1. \quad (61)$$

$$\begin{aligned} E[T \text{ модны хайлтын өртөг}] &= \sum_{i=1}^n (\Gamma_H(T)(k_i) + 1) \cdot p_i + \sum_{i=0}^n (\Gamma_H(T)(d_i) + 1) \cdot q_i \\ &= 1 + \sum_{i=1}^n \Gamma_H(T)(k_i) \cdot p_i + \sum_{i=0}^n \Gamma_H(T)(d_i) \cdot q_i \quad (62) \end{aligned}$$

Хоёртын оновчтой хайлтын мод



node	depth	probability	contribution
k_1	1	0.15	0.30
k_2	0	0.10	0.10
k_3	2	0.05	0.15
k_4	1	0.10	0.20
k_5	2	0.20	0.60
d_0	2	0.05	0.15
d_1	2	0.10	0.30
d_2	3	0.05	0.20
d_3	3	0.05	0.20
d_4	3	0.05	0.20
d_5	3	0.10	0.40
Total			2.80

(a)

node	depth	probability	contribution
k_1	1	0.15	0.30
k_2	0	0.10	0.10
k_3	3	0.05	0.20
k_4	2	0.10	0.30
k_5	1	0.20	0.40
d_0	2	0.05	0.15
d_1	2	0.10	0.30
d_2	4	0.05	0.25
d_3	4	0.05	0.25
d_4	3	0.05	0.20
d_5	2	0.10	0.30
Total			2.75

(b)

Хоёртын оновчтой хайлтын мод

Алхам 1: Хоёртын оновчтой хайлтын модны бүтэц:

- ▶ Дурын хоёртын хайлтын дэд мод нь $1 \leq i \leq j \leq n$ байх k_i, \dots, k_j дараалсан түлхүүртэй, тэдгээр нь мөн хуурмаг түлхүүр d_{i-1}, \dots, d_j гэгдэх навч агуулдаг
- ▶ Хэрэв хоёртын хайлтын оновчтой мод T нь k_i, \dots, k_j түлхүүр бүхий T' дэд модтой бол T' дэд мод нь k_i, \dots, k_j ба d_{i-1}, \dots, d_j хуурмаг түлхүүр бүхий дэд бодлогын хувьд оновчтой байна

Хоёртын оновчтой хайлтын мод

Алхам 1: Хоёртын оновчтой хайлтын модны бүтэц:

- ▶ k_i, \dots, k_j түлхүүр бүхий дэд модны хувьд k_i түлхүүрийг үндсээр сонгосон гэж саная. Тэгвэл k_i түлхүүрийн зүүн дэд мод нь k_i, \dots, k_{i-1} буюу нэг ч түлхүүр агуулахгүй ч ганц d_{i-1} хуурмаг түлхүүр агуулна.

Хоёртын оновчтой хайлтын мод

Алхам 2: Дахин дуудагдах шийдэл:

- ▶ Хоёртын оновчтой хайлтын модны хайлтын өртөг: $e[1, n]$
- ▶ Хялбар тохиолдол нь $j = i - 1$ гэвэл хайлтын өртөг нь $e[i, i - 1] = q_{i-1}$

Хоёртын оновчтой хайлтын мод

Алхам 2: Дахин дуудагдах шийдэл:

$$w(i, j) = \sum_{l=i}^j p_l + \sum_{l=i-1}^{j-1} q_l. \quad (63)$$

$$e[i, j] = p_r + (e[i, r-1] + w(i, r-1)) + (e[r+1, j] + w(r+1, j))$$

$$w(i, j) = w(i, r-1) + p_r + w(r+1, j)$$

Дахин дуудагдах томъёо:

$$e[i, j] = e[i, r-1] + e[r+1, j] + w(i, j). \quad (64)$$

Хоёртын оновчтой хайлтын мод

Алхам 2: Дахин дуудагдах шийдэл: Хайлтын өртөг тооцох дахин дуудагдах тэгшитгэл

$$e[i, j] = \begin{cases} q_{i-1}, & j = i - 1, \\ \min\{e[i, r-1] + e[r+1, j] + w(i, j) : i \leq r \leq j\}, & i \leq j. \end{cases} \quad (65)$$

Хоёртын оновчтой хайлтын мод

Алхам 3: Хоёртын оновчтой хайлтын модны хайх өртгийг тооцоолох

- ▶ $e[i, j]$ хайлтын өртгийг $e[1 : n + 1, 0 : n]$ хүснэгтэд хадгална.
- ▶ Суурь нөхцөлийг тооцоолох магадлал $1 \leq i \leq n + 1$ үед
 $w[i, i - 1] = q_{i-1}$
- ▶ $j \geq i$ үед дараах томьёо хүчинтэй:

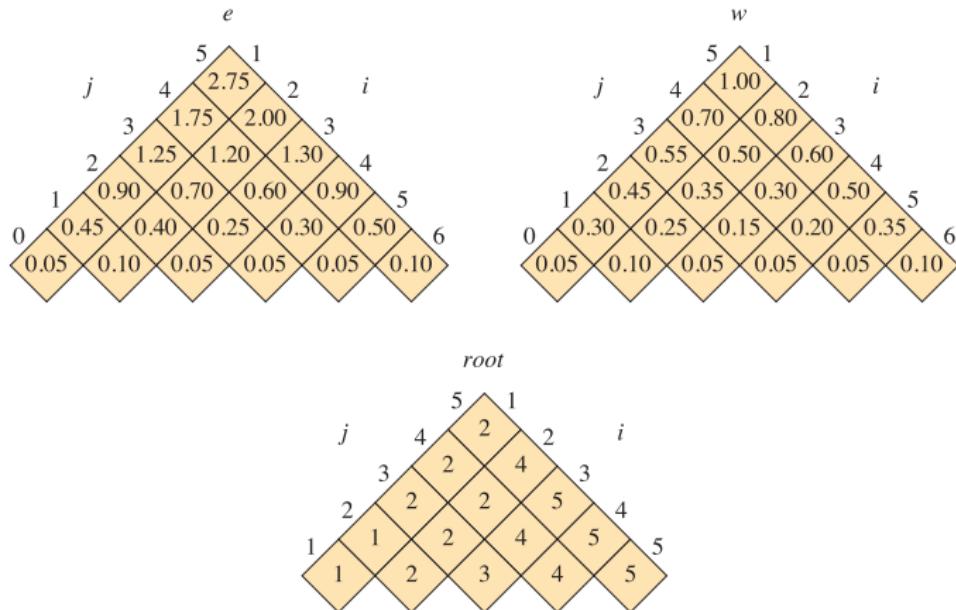
$$w[i, j] = w[i, j - 1] + p_j + q_j \quad (66)$$

Хоёртын оновчтой хайлтын мод

Optimal-BST(p, q, n)

```
1   $e[1 : n + 1, 0 : n]$ ,  $w[1 : n + 1, 0 : n]$  ба  $root[1 : n, 1 : n]$  нь хүснэгт
2  for  $i = 1$  to  $n + 1$ 
3       $e[i, i - 1] = q_{i-1}$ 
4       $w[i, i - 1] = q_{i-1}$ 
5  for  $l = 1$  to  $n$ 
6      for  $i = 1$  to  $n - l + 1$ 
7           $j = i + l - 1$ 
8           $e[i, j] = \infty$ 
9           $w[i, j] = w[i, j - 1] + p_j + q_j$ 
10         for  $r = i$  to  $j$ 
11              $t = e[i, r - 1] + e[r + 1, j] + w[i, j]$ 
12             if  $t < e[i, j]$ 
13                  $e[i, j] = t$ 
14                  $root[i, j] = r$ 
15 return  $e$  ба  $root$ 
```

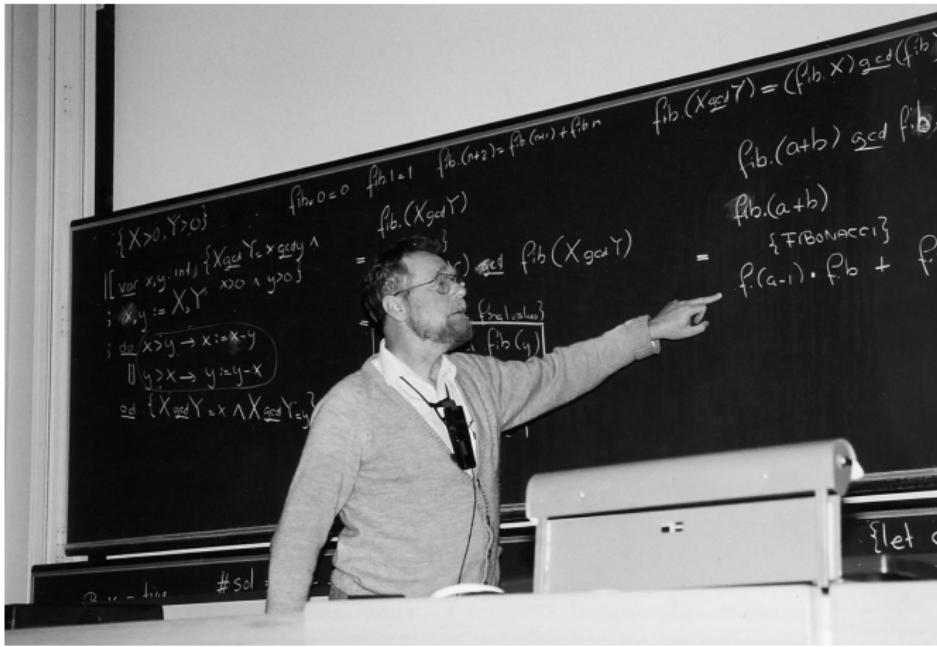
Хоёртын оновчтой хайлтын мод



Зураг 11: $e[i, j]$ хайлтын өртөг, $w[i, j]$ нийлбэр магадлал, $root[i, j]$ дэд модны үндэс түлхүүр

Хомхойлох алгоритм

Хомхойлох алгоритм



Зураг 12: Эдсгер Вибе Дейкстра

Ажиллагааг түүвэрлэх бодлого

Ажиллагааг түүвэрлэх бодлого

Тухайн агшинд өрөөнд нэг л үйл ажиллагаа явагдах бөгөөд өрөөг ашиглах нийт n тооны хурлын үйл ажиллагааны

$S = \{a_1, a_2, \dots, a_n\}$ олонлог өгөгдсөн.

- ▶ a_i ажиллагаа бүр s_i цагт эхэлж, f_i цагт дуусна, энд $0 \leq s_i < f_i < \infty$.
- ▶ Хэрэв a_i ажиллагаа сонгогдвол хугацааны хагас задгай завсарт явагдана.
- ▶ Хэрэв $[s_i, f_i)$ болон $[s_j, f_j)$ завсрууд нь давхцахгүй бол a_i болон a_j ажиллагааг нийцтэй гэнэ.

Ажиллагааны дуусах хугацаа бүр нэгэн эрэмбээр өснө:

$$f_1 \leq f_2 \leq f_3 \leq \cdots \leq f_{n-1} \leq f_n. \quad (67)$$

Ажиллагааг түүвэрлэх бодлого

i	1	2	3	4	5	6	7	8	9	10	11
s_i	1	3	0	5	3	5	6	7	8	2	12
f_i	4	5	6	7	9	9	10	11	12	14	16

Хүснэгт 1: Ажиллагааны $\{a_1, a_2, \dots, a_{11}\}$ олонлогийн хувьд a_i ажиллагаа нь s_i нь эхлэх хугацаа, f_i нь төгсөх хугацаа болно.

Ажиллагааг түүвэрлэх бодлого

Ажиллагааг түүвэрлэх бодлогын оновчтой дэд бүтэц

S_{ij} олонлог дахь харилцан нийцтэй ажиллагаануудын хамгийн том A_{ij} олонлогт a_k ажиллагаа харьяалагддаг гээд a_k ажиллагааг оновчтой шийддэг оруулбал дараах хоёр дэд бодлого үүснэ:

- ▶ S_{ik} олонлогоос харилцан нийцтэй олонлог олох
- ▶ S_{kj} олонлогоос харилцан нийцтэй олонлог олох
- ▶ $A_{ik} = A_{ij} \cap S_{ik}$ ба $A_{kj} = A_{ij} \cap S_{kj}$ бол $A_{ij} = A_{ik} \cup \{a_k\} \cup A_{kj}$
- ▶ S_{ij} олонлогт хамаарах харилцан нийцтэй A_{ij} олонлогийн хамгийн их хэмжээ нь $|A_{ij}| = |A_{ik}| + |A_{kj}| + 1$

Ажиллагааг түүвэрлэх бодлого

Хэрэв S_{kj} олонлогт, харилцан нийцтэй $|A'_{kj}| > |A_{kj}|$ байх A'_{kj} олонлог олдвол S_{ij} дэд бодлогын шийдийг A_{kj} бус A'_{kj} гэж үзнэ гэвэл

$$|A_{ik}| + |A'_{kj}| + 1 > |A_{ik}| + |A_{kj}| + 1 = |A_{ij}|$$

болно.

Ийнхүү ерөнхий тохиолдолд:

$$c[i, j] = \begin{cases} 0, & \text{хэрэв } S_{ij} = \emptyset \text{ бол,} \\ \max\{c[i, k] + c[k, j] + 1 : a_k \in S_{ij}\} & \text{хэрэв } S_{ij} \neq \emptyset \text{ бол.} \end{cases} \quad (68)$$

Ажиллагааг түүвэрлэх бодлого

Хомхойлон түүвэрлэх

Түүвэрлэх ажиллагаануудын дотроос нэг нь түрүүлж дуусах нь гарцаагүй тул S олонлогийн хувьд ажиллагааг түүвэрлэхдээ, эх үүсвэрийг бусад олон ажиллагаанд хүртээмжтэйгээр хуваарилахыг эрмэлзэж, түрүүлж дуусахыг нь сонгоно.

Ажиллагааг түүвэрлэх бодлого

Хомхойлон түүвэрлэх

- ▶ Хомхой сонголт хиймэгц шийдвэл зохих ганц л бодлого үлдэнэ: a_1 дууссаны дараах ажиллагаануудыг олох.
- ▶ a_k ажиллагаа дууссаны дараа эхлэх ажиллагаануудын олонлог $S_k = \{a_i \in S : s_i \geq f_k\}$ байг. Хэрэв хомхойлох сонголтоор a_1 ажиллагааг сонгоход, шийдвэл зохих дэд бодлого нь S_1 болно.
- ▶ Оновчтой дэд бүтцээр, хэрэв a_1 нь оновчтой шийд бол бодлогын оновчтой шийд нь, a_1 ажиллагаа болон S_1 дэд бодлогын оновчтой шийдэд хамаарах бүх ажиллагааны нийлбэр байна.

Ажиллагааг түүвэрлэх бодлого

Теорем 7.1

Дурын хоосон биш S_k дэд бодлогын хувьд a_m нь S_k олонлогийн хамгийн түрүүнд дуусах ажиллагаа байг. Тэгвэл a_m нь, S_k олонлогийн харилцан нийцтэй ажиллагаануудын хамгийн том дэд олонлогт багтана.

Ажиллагааг түүвэрлэх бодлого

Хомхойлох алгоритмд дээрээс доош аргачлал зохимжтой:

- ▶ Эхлээд сонголт хийнэ,
- ▶ Дараа нь дэд бодлогыг бодно

Ажиллагааг түүвэрлэх бодлого

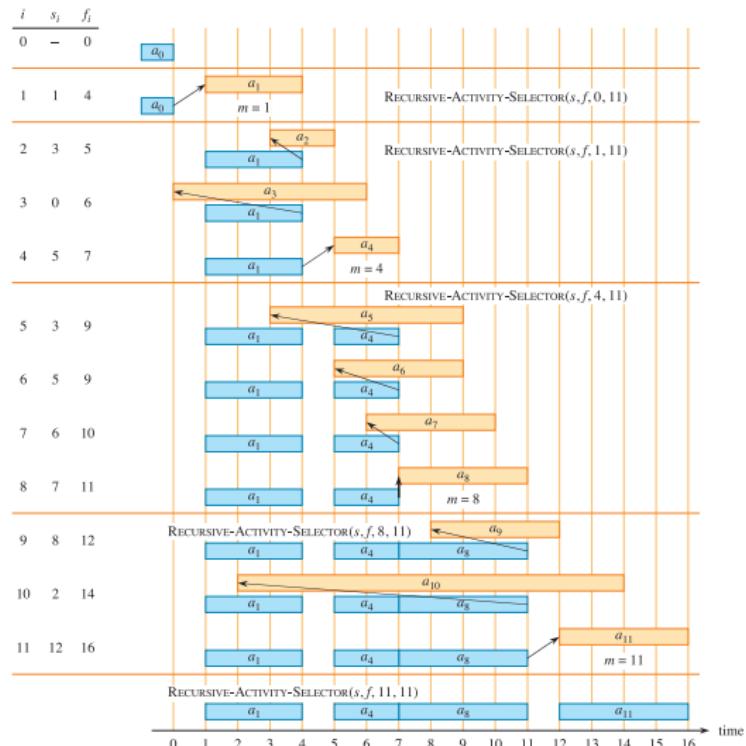
Дахин дуудагдах хомхойлох алгоритм

s болон f нь ажиллагааны эхлэх болон дуусах хугацаа, k товъёг нь S_k дэд бодлогын дугаар, n нь бодлогын хэмжээ болно.

Recursive-Activity-Selector(s, f, k, n)

```
1    $m = k + 1$ 
2   while  $m \leq n$  and  $s[m] < f[k]$ 
3        $m = m + 1$ 
4   if  $m \leq n$ 
5       return  $\{a_m\} \cup$  Recursive-Activity-Selector( $s, f, m, n$ )
6   else return  $\emptyset$ 
```

Ажиллагааг түүвэрлэх бодлого



Зураг 13: Ажиллагааг түүвэрлэх дахин дуудагдах алхмууд

Ажиллагааг түүвэрлэх бодлого

Давталтат хомхойлох алгоритм

k товъёг нь A олонлогт хамгийн сүүлд нэмэгдсэн a_k ажиллагааны утга.

Greedy-Activity-Selector(s, f, n)

```
1    $A = \{a_1\}$ 
2    $k = 1$ 
3   for  $m = 2$  to  $n$ 
4       if  $s[m] \geq f[k]$ 
5            $A = A \cup \{a_m\}$ 
6            $k = m$ 
7   return  $A$ 
```

Ажиллагааг түүвэрлэх бодлого

Ажиллагаанууд нь дуусах хугацаагаараа өсөх дарааллаар эрэмбэлэгдсэн тул A олонлогийн дурын ажиллагааны хувьд хамгийн сүүлийн дуусах хугацаа нь ямагт f_k байна:

$$f_k = \max\{f_i : a_i \in A\}. \quad (69)$$

Хомхойлох алгоритм

Хомхойлох алгоритмын шинжүүр

Хомхойлох алгоритмын шинжүүр

Өмнөх бүлэгт хомхойлох алгоритмыг дараах зарчмаар хөгжүүлсэн:

- ▶ Бодлогын дэд бүтцийг тодорхойлох
- ▶ Дахин дуудагдах шийдлийг боловсруулах
- ▶ Хэрэв хомхой сонголт хийсэн бол, эцэст нь ганц дэд бодлого үлдэнэ гэдгийг батална
- ▶ Хомхой сонголт хийх нь бодлогын оновчтой шийдэд сөрөг нөлөөгүй гэдгийг батлах
- ▶ Хомхойлох төлөвлөгөөг хэрэгжүүлэх дахин дуудагдах алгоритмыг хөгжүүлэх
- ▶ Дахин дуудагдах алгоритмыг давталтат алгоритм уруу хөрвүүлэх

Хомхойлох алгоритмын шинжүүр

Хомхойлох алгоритмыг хомхойлох зарчимд тулгуурлан шийдэх:

- ▶ Оновчлолын бодлогыг, сонголт хийсний дараа ганц дэд бодлого үлдэх байдлаар шийднэ
- ▶ Бодлогын оновчтой шийдэл нь ямагт хомхой сонголт гэдгийг батална
- ▶ Дэд бодлого бүрд хомхой сонголт хийх замаар бодлогын оновчтой шийдийг гарган авч болдгийг батална

Хомхойлох алгоритмын шинжүүр

Хомхой сонголтын шинж чанар

- ▶ Дэд бодлогуудаас хомхой буюу оновчтой сонголт хийх замаар бодлогын оновчтой шийдийг байгуулж болно.
- ▶ Хомхойлох алгоритм нь дэд бодлогуудыг бодохоос өмнө эхний сонголтоо хийдэг бол динамик программчлалын алгоритм нь эхний сонголтоо хийхээс өмнө дэд бодлогуудаа бодно.
- ▶ Динамик программчлалын алгоритм нь доороос дээш чиглэлд ажиллах боломжтой бол хомхойлох алгоритм нь ихэвчлэн дээрээс доош чиглэлд ажилладаг.
- ▶ Оролтын өгөгдлийг урьдчилан боловсруулах буюу эрэмбэлэх байдлаар хомхойлох алгоритмын ажиллагааг илүү хурдан, үр ашигтай болгож болно.

Хомхойлох алгоритмын шинжүүр

Оновчтой дэд бүтэц

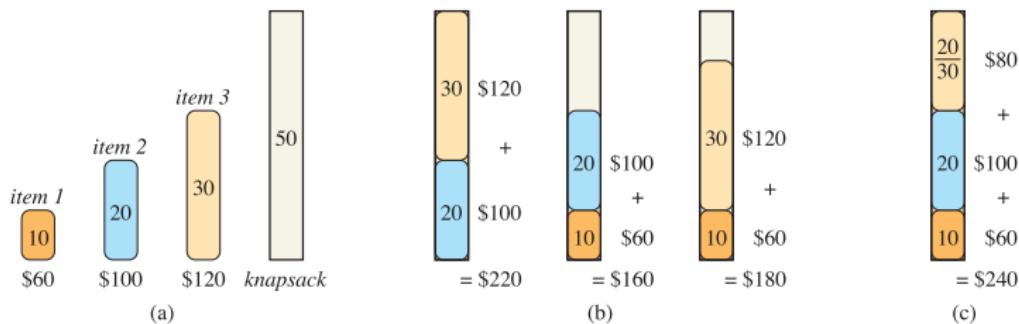
- ▶ Хэрэв бодлогын оновчтой шийдэл нь дэд бодлогын оновчтой шийдэлд агуулагдаж байвал бодлогыг оновчтой дэд бүтэцтэй гэж үзнэ
- ▶ Хомхойлох алгоритмын хувьд дээрх зарчмыг, дэд бодлогын оновчтой шийдийг өмнөх хомхой сонголттой нэгтгэж, бодлогын оновчтой шийдийг гарган авах байдлаар хэрэгжүүлдэг
- ▶ Үүнийг математик нэгтгэн дүгнэх аргачлалыг дэд бодлогод ашиглах замаар баталж болно

Хомхойлох алгоритмын шинжүүр

Хомхойлох болон динамик программчлалын ялгаа

Дараах 2 төрлийн үүргэвчтэй бодлого бий:

- ▶ 0-1 буюу бүхэл
- ▶ бутархай



Зураг 14: (a) барааны жин, нэгжийн үнэ, үүргэвчний багтаамж. (b) хоёр ба гурав дахь барааны оновчтой сонголт. (c) бутархай бодлогын оновчтой сонголт

Хаффманы код

Хаффманы код



Зураг 15: Дэвид Хаффман

Хаффманы код

	a	b	c	d	e	f
Давтамж (мянгаар)	45	13	12	16	9	5
Тогтмол урттай код	000	001	010	011	100	101
Хувьсах урттай код	0	101	100	111	1101	1100

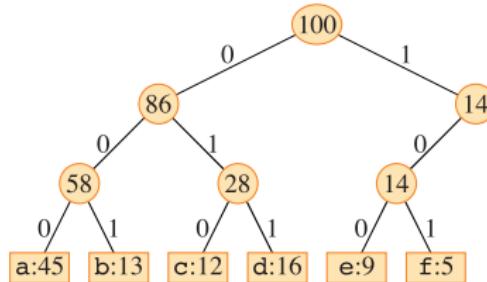
Хүснэгт 2: 100,000 тэмдэгт өгөгдөл нь a-f хүртэл нийт 6 тэмдэгтийг агуулах бөгөөд тэдгээрийн давтамж

Хувьсах урттай кодлолтын нийт хэмжээ:

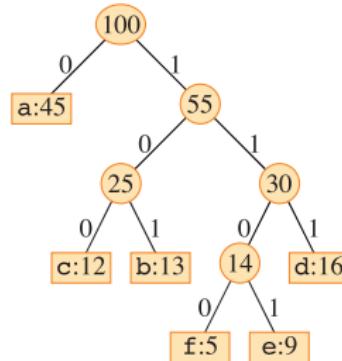
$$(45 \cdot 1 + 13 \cdot 3 + 12 \cdot 3 + 16 \cdot 3 + 9 \cdot 4 + 5 \cdot 4) \cdot 1000 = 224,000$$

Хаффманы код

Үгтваргүй код (prefix-free codes)



(a)



(b)

Зураг 16: (a) тогтмол урттай кодод харгалзах мод. (b) үгтваргүй кодыг дүрслэх хувьсах урттай кодод харгалзах мод.

Хаффманы код

Үгтвартгүй код

T модны зардал:

$$B(T) = \sum_{c \in C} c.freq \cdot d_T(c) \quad (70)$$

Хаффманы код

Хаффманы кодыг байгуулах

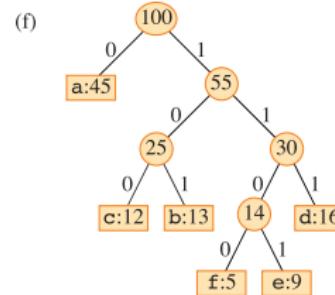
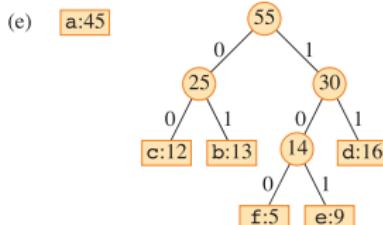
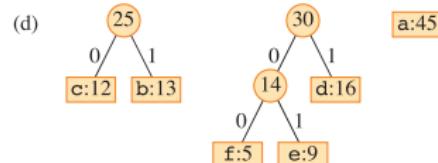
Huffman(C)

```
1    $n = |C|$ 
2    $Q = C$ 
3   for  $i = 1$  to  $n - 1$ 
4        $z$  шинэ зангилаа үүсгэх
5        $x = \text{Extract-Min}(Q)$ 
6        $y = \text{Extract-Min}(Q)$ 
7        $z.left = x$ 
8        $z.right = y$ 
9        $z.freq = x.freq + y.freq$ 
10       $\text{Insert}(Q, z)$ 
11  return  $\text{Extract-Min}(Q)$ 
```

Хаффманы код

(a) **f:5 e:9 c:12 b:13 d:16 a:45**

(b) **c:12 b:13 d:16 a:45**



Зураг 17: Хаффманы алгоритмаар мод байгуулах

Орчны нөөцлөлт Offline caching

Орчны нөөцлөлт

b_i үүрийн хүсэлт нь дараах гурван тохиолдлыг хангана:

1. b_i нь өмнөх хүсэлтийн улмаас нөөцөд байвал нөөц үл өөрчлөгднө
2. b_i нь нөөцөд байхгүй бөгөөд нөөц нь хараахан дүүрээгүй бол b_i үүрийг нөөцөд бүртгэж, нөөцийн үүрийн тоог нэмнэ
3. b_i нь нөөцөд ороогүй бөгөөд нөөц дүүрсэн бол нөөц дэх зарим үүрийг чөлөөлж, b_i нь нөөцөд нэмэгдэнэ.

Орчны нөөцлөлт

Орчны нөөцлөлтийн оновчтой дэд бүтэц

Орчны нөөцлөлт нь оновчтой дэд бүтэцтэй болохыг харуулахын тулд b_i, b_{i+1}, \dots, b_n үүрийн хүсэлтийг боловсруулах (C, i) дэд бодлогыг дараах байдлаар тодорхойлъё:

- ▶ C нь $|C| \leq k$ байх үүрийн дэд олонлог
- ▶ (C, i) дэд бодлогын оновчтой шийд нь нөөцөд багтаагүй үүрийн тоог багасгах явдал.

Орчны нөөцлөлт

Орчны нөөцлөлтийн оновчтой дэд бүтэц

$R_{C,i}$ нь b_i үүрийн хүсэлтийг боловсруулсны дараах C олонлогийн бүрдэл. Тэгвэл:

- ▶ Нөөцөд өөрчлөлт ороогүй бол $R_{C,i} = \{C\}$
- ▶ Нөөцөд байхгүй бол b_i үүрэнд хүсэлт нэмэх бол
 $R_{C,i} = \{C \cup \{b_i\}\}$
- ▶ Нөөц дүүрэн үед b_i үүрэнд хүсэлт нэмэх бол
 $R_{C,i} = \{(C - \{x\}) \cup \{b_i\} : x \in C\}$

Орчны нөөцлөлт

Орчны нөөцлөлтийн оновчтой дэд бүтэц

$$miss(C, i) = \begin{cases} 0, & i = n \text{ ба } b_n \in C \\ 1, & i = n \text{ ба } b_n \notin C \\ miss(C, i + 1), & i < n \text{ ба } b_i \in C \\ 1 + \min\{miss(C', i + 1) : C' \in R_{C,i}\}, & i < n \text{ ба } b_i \notin C \end{cases}$$