ШИНЖЛЭХ УХААН ТЕХНОЛОГИЙН ИХ СУРГУУЛЬ
Мэдээлэл, Холбооны Технологийн Сургууль

F.CS213 Биоалгоритм

Motif Discovery Algorithms

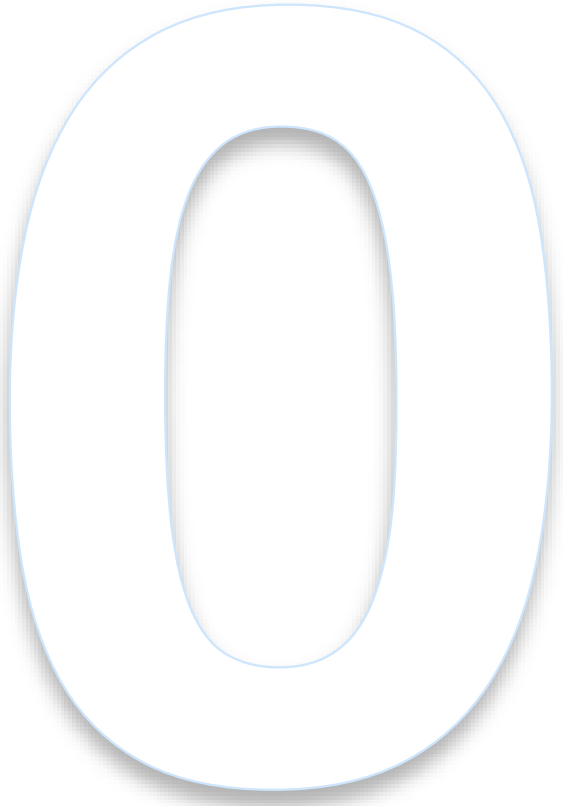
Мотиф шинжилгээний алгоритмууд

Лекц 9

Багш ***Ганбаатарын ГАНБАТ.*** *Өрөө* ***#304,*** *ganbatg@must.edu.mn,* ***99999467***

Лекцийн агуулга

- Удиртгал
- Детерминистик мотиф
    - Илрүүлэх алгоритмын хэрэгжүүлэлт
- Brute-Force алгоритмууд: Бүрэн (Exhaustive) хайлт
- Branch-and-Bound алгоритмууд
- Хьюристик алгоритмууд



Удиртгал

Биологийн дарааллын олборлолтын хүрээнд *мотиф (motif)* гэдэг нэр томьёо нь олон дараалал дээр илрэх *non-trivial* дараалал-паттернийг ойлгоно.

- Мотифийн non-trivial паттерн
  - Тэмдэгтүүдийнхээ тархалтаар ялгарах хамгийн бага урттай тэмдэгтүүдийн хослол
  - Гол нь рекуррент байх, ө.х шинжилсэн дарааллуудын хэд хэддээр нь илэрсэн байх ёстой.
- *Геномын элементүүд (Биологийн тусгай үзүүлэлтийг бүрдүүлдэг эсвэл Нэг зохицуулалтын удирдлага доор байдаг генүүд)*-рүү холбоотойгоор дарааллын олонлогийн хувьд мотиф оршин байх боломжтой.
  - *ДНХ дарааландд*, генийн промотер мужид мотиф үүсч болно. Генийн галиглалт (transcription)-ийг зохицуулах үүрэгтэй дан уураг эсвэл уургийн комплекс руу бэхлэгдэх хэсгүүд байгааг илтгэнэ.
  - *Уургийн дарааландд,* мотиф нь хуримталсан (conserved) домэйн оршин байгааг илтгэж болно. Туслах хэсэг *(substrate)* эсвэл бусад молекулуудад зориулсан энзимийн бэхлэгдэх зэрэг уураг дээрх биологийн тусгай үүрэг гүйцэтгэдэг хэсгүүд
- *Deterministic:* Оролтын дарааллууд дээр байгаа эсэхийг нь тодруулахаар сайжруулсан регуляр илэрхийллийн синтаксаар ерөнхийдөө танигддаг.
- *Probabilistic:* Мотиф илрэлүүдийн үндсэн хувиралуудыг илрүүлдэг байх *сул (loose)* загвараар танигддаг.
  - Дарааллын өгөгдсөн сегментийг загварт авч үзэн мотифын нэг хэсэг байх магадлалыг гарган авна.

* *Дараалал паттерн (sequence pattern)* ба *мотиф* гэсэн нэр томьёонуудыг ижил утгатайгаар авч үзнэ.

Детерминистик мотиф

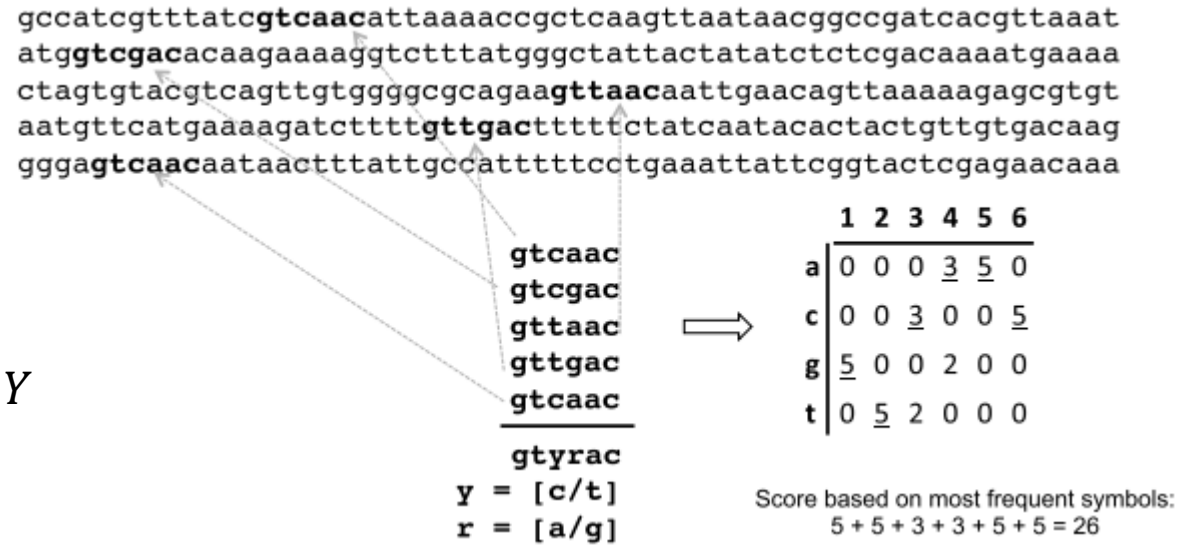
- $\Sigma_{DNA} = \{A, C, G, T\}$, $\Sigma_{Protein} = \{A, C, D, E, F, G, H, I, K, L, M, N, P, Q, R, S, T, V, W, Y\}$.
  - Энд мотиф хайлтын алгориутууд нь дараах онцлогуудын хувьд оновчлогдсэн байх шаардлагатай
  - *Үүнд: цагаан толгойн хэмжээ, дарааллын ерөнхий урт, суурь тэмдэгтийн тархалт.*
- Жнь, $S = abcdef$ дараалал байг.
  - *sub-string* bcd : Тэмдэгтүүдийн уртаас шалтгаалж мэдээлэл өөрчлөгдөггүй байх үргэлжилсэн хослол.
  - *sub-sequence* $acdf$: тогтмол бус тэмдэгтүүдэд зориулсан хувьсах урттай өөр хэсгүүдийг агуулдаг.
    - Илүү комплекс, илүү ерөнхий тохиолдолд
    - зарим тэмдэгтийг хасч, үүссэн хэсгүүдийг залгасан шинэ дараалал.
- Хэрэв $|S|$ нь S-ийн урт, $S = xyz, |x| \geq 0, |z| \geq 0$ бол x нь ***угтвар (prefix)***, z нь ***дагавар (suffix)*** байна.
  - S нь y-ийн *superstring* буюу *supersequence* болно.
- Ерөнхийдөө мотиф M нь $\Sigma' = \Sigma \cap E$ өргөтгөсөн цагаан толгойгоор үүсэх хоосон биш тэмдэгт мөр байна.
  - E нь Нэгээс олон хоосон-зай/тэмдэгтийг илэрхийлсэн байрлалын тэмдэглэгээ
  - Жнь IUPAC нуклеотидын код.
- Мотиф $\mathbf{M}$-ийн Σ' цагаан толгойн тэмдэгтүүдийг орлуулсны үр дүнд үүсэх боломжтой бүх регуляр илэрхийллийн олонлогийг ***паттерны хэл (pattern language)*** гээд $L(M)$ гэж тэмдэглэнэ.
  - S нь $L(M)$-д тохиолдох y тэмдэгт мөрийг агуулж байвал M мотиф нь S дараалалтай *match* эсвэл *hit* боллоо гэнэ.

ДЕТЕРМИНИСТИК МОТИФ

Паттерны хувирал

- $EcoRI$ restriction enzyme (Бүлэг 5): $GAATTC$ цуваа паттерн олж, "G" ба "A"-ын дундуур ДНХ-г тасладаг.
  - Давтагдахгүй тэмдэгт мөрөөр илэрхийлэгдсэн маш тодорхой паттерн.
  - Гэхдээ ихэнх тохиолдолд, дээрх холболтын дараалал бага зэрэг өөрчлөгддөг.
  - Нэгээс олон дарааллыг хамруулахын тулд илүү уян паттернийг шаарддаг.
  - Σ_{DNA} цагаан толгой

- *Type II restriction enzyme:* ДНХ-г 6 нуклеотидын урттай тусгай дарааллаар тасладаг. $5'$-ээс $3'$ чиглэлд
  - GT-ийн ард T эсвэл C, A эсвэл G-н ард AC
  - Хувирсан мотифтой байгаа ч давтагдсан байдал нь өндөр чанартай хэвээр байна.
  - Мотифын 2 ба 3 байрлалын C, T ба A, G ийг Y ба R тэмдэглэгээр авч үзэж болно
  - Σ'_{DNA} өргөтгөсөн цагаан толгой



- *Transcription factor (TFs)*: Нэлээд хувирсан дарааллаар холбогддог тул мотиф нь илүү комплекс байдаг.
  - Энэхүү хувиралт нь *de novo* мотифыг илрүүлэх хүндрэлийг маш их нэмэгдүүлдэг.

ДЕТЕРМИНИСТИК МОТИФ

Мотифын дүрслэл

- Оролтын дарааллууд дээр бүх тохиолдлын зэрэгцүүлэлтийг цуглуулах.
  - Эхлэлийн байрлалыг цуглуулах бөгөөд дараа нь харгалзах тэмдэгт мөрүүдийг сэргээнэ.
  - Энэ аргаар бүх мэдээллийг хадгалж, мотифын олон янз байдлыг олж авдаг.
  - *Сул тал:* Мотифын илрэлүүд болох олон тэмдэгт мөрийг хадгалах шаардлагатай байдаг.
- Өргөтгөсөн цагаан толгойн давуу талыг ашиглаж, регуляр илэрхийллийн синтаксаар мотифыг таних.
  - Үүнийг ***дундаж (consensus)*** мотиф гэж нэрлэдэг.
  - Энэ нь илүү авсаархан, ойлгомжтой мотифыг дүрслэх боломжийг олгодог
  - *Сул тал:* Мотифын тохиолдол бүр хэдэн удаа тохиолдох талаарх мэдээлэлийг алддаг.
- Мотифын байрлал бүр дээрх тэмдэгтүүдийн давтамжийн профайл/матрцыг байгуулах.
  - Зөвхөн цагаан толгойн хэмжээ болон мотифын уртаас хамаарах ба мотифын тооноос хамааралгүй.
  - *Сул тал:* Тэмдэгтүүдийн дарааллыг хадгалахаа больсонтой холбоотой юм. (Дараа үзнэ)

Мотиф илрүүлэх асуудал

Мотифын бүтцийн нэг чухал хэмжигдэхүүн нь *давтагдах байдал (recurrence)*. Үүнийг *давтамж (frequency)* эсвэл *тулгуур (support)* гэнэ.

- Ороолтын дарааллын багц D-ын хувьд дараах тохиолдлуудыг хэмжиж болно:
  - Мотиф M илэрсэн дарааллын тоо (дараалал бүр нэг удаа тоологдоно)
  - Мотиф M илэрсэн нийт тоо (дараалал бүрт нэгээс олон).
  - *Мотиф давтамж (frequent motif)* : Мотифийн давтамж нь тодорхой босгоос их буюу тэнцүү.
- Мотифын байрлал бүр дээрх хамгийн олон тэмдэгтийн давтамжийг харгалзсан онооны хэмжүүрийг тооцоолж профайлыг дүрслэнэ.

$$Score(M) = \sum_{i=1}^{L} max_{k \in \Sigma} count(k, i)$$

- *Мотифыг илрүүлэх асуудал (Motif discovery problem)*-ыг формалт тодорхойлолт:

Оролт:
- Σ цагаан толгойн $D = \{S_1, S_2, \dots, S_t\}$ дарааллын олонлог,
- Мотифын урт L,
- Онооны хамгийн бага утга σ (optional):

Гаралт:
- Өгөгдсөн σ-ээс их буюу тэнцүү оноотой D дээрх бүх мотиф.

ДЕТЕРМИНИСТИК МОТИФ Хэрэгжүүлэлт

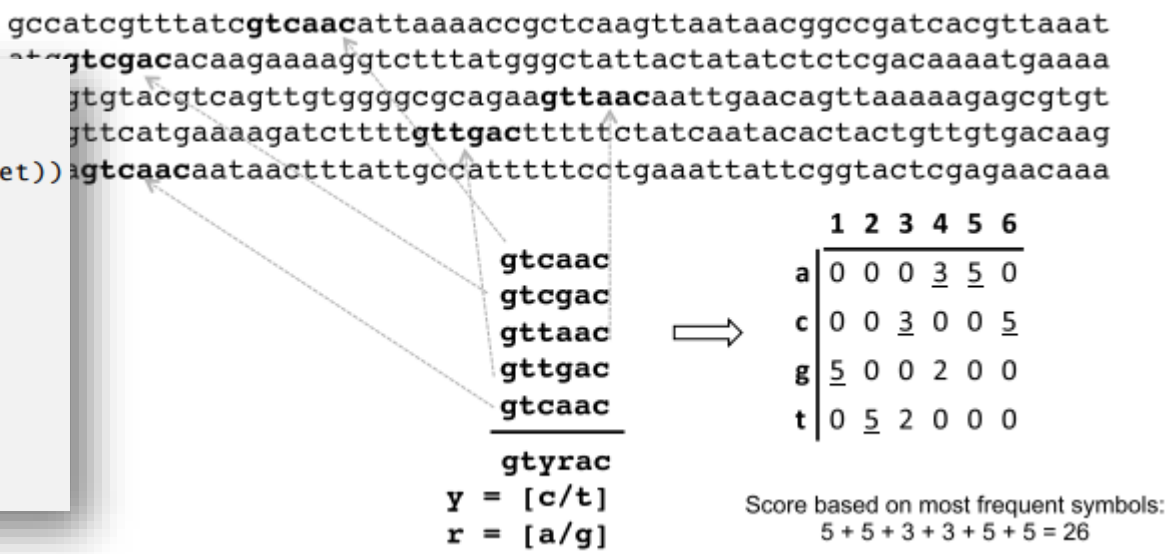
```
class DeterministicMotifFinding:
    """ Class for deterministic motif finding. """

    def __init__(self, size = 8, seqs = None):
        self.motif_size = size
        if (seqs != None):
            self.seqs = seqs
            self.alphabet = seqs[0].alphabet()
        else:
            self.seqs = []

    def __len__ (self):
        return len(self.seqs)

    def __getitem__(self, n):
        return self.seqs[n]

    def seq_size (self, i):
        return len(self.seqs[i])

    def read_file(self, fic, t):
        for s in open(fic, "r"):
            self.seqs.append(MySeq(s.strip().upper(),t))
        self.alphabet = self.seqs[0].alphabet()
```

DeterministicMotifFinding класс

- Гишүүн өгөгдлүүд
  - **motif _size**: мотифын урт
  - **seqs**: оролтын дарааллуудыг агуулсан вектор.
- Гол гишүүн функцүүд
  - **read_file** өгөгдсөн файлаас оролтын дарааллуудыг уншина
  - **create_motif_from_indexes:** профайл матрицыг байгуулна
  - **score :** нийлбэр дээр суурилсан онооны функцийг хэрэгжүүлдэг.
  - **score_multiplicative :** үржвэр дээр суурилсан онооны функцийг хэрэгжүүлдэг
- Эдгээр функцууд нь Мотиф илрүүлэх асуудал аргачлалд зориулсан тохиромжтой мотиф дүрслэлийг өгдөг.

Профайл матрицын байгуулалт

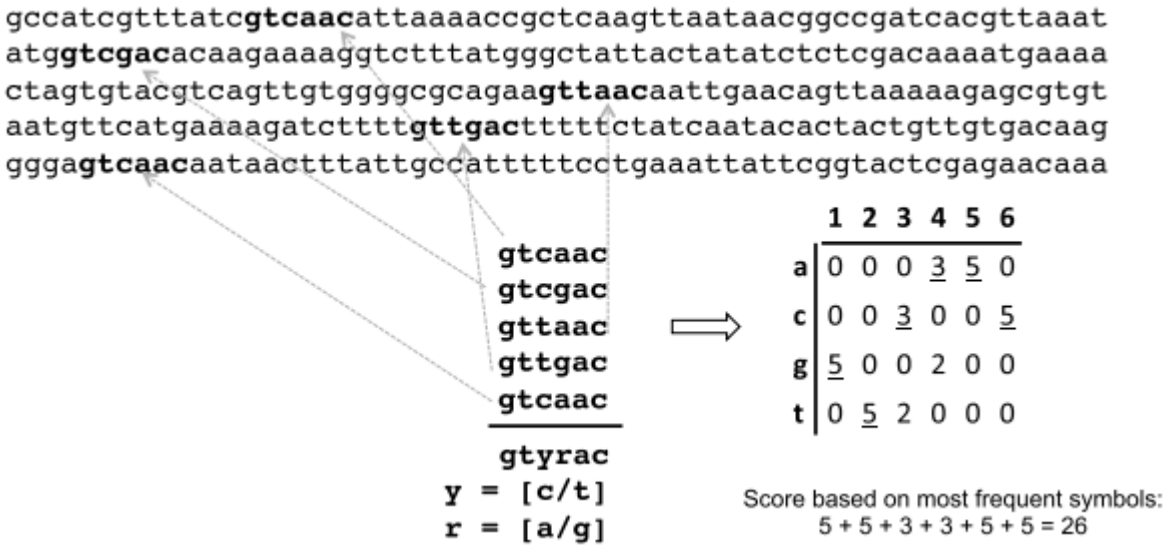
```
def create_motif_from_indexes(self, indexes):
    pseqs = []
    res = [[0]*self.motif_size for i in range(len(self.alphabet))
]
    for i,ind in enumerate(indexes):
        subseq = self.seqs[i][ind:(ind+self.motif_size)]
        for i in range(self.motif_size):
            for k in range(len(self.alphabet)):
                if subseq[i] == self.alphabet[k]:
                    res[k][i] = res[k][i] + 1
    return res
```



- Зурагт үзүүлснээр жагсаасан мотиф илрэлийн багцаас мотифын урттай тэнцүү тооны багана, цагаан толгойн хэмжээтэй тэнцүү тооны мөр бүхий матрицыг байгуулна
  - Элемент бүр нь тухайн байрлал дахь тэмдэгтийн давтамж..
- create_motif_from_indexes
  - Оролтын дарааллуудын дагуу мотифын индексүүдийг параметрээр авна.
  - Хувьсагч **res** нь хоёр хэмжээст матриц байна.
  - Бүх индексүүд дээр харгалзах дарааллыг авахын тулд давталт хийнэ.
  - Вектор дахь индекс болон харглзах утгыг буцаадаг **enumerate** built-in функцийг ашиглаж байна
  - Дотоод давталтын тусламжтайгаар матрицын харгалзах нүдний утгыг нэмэгдүүлнэ.

Онооны функцүүд

```
def score(self, s):
    score = 0
    mat = self.create_motif_from_indexes(s)
    for j in range(len(mat[0])):
        maxcol = mat[0][j]
        for i in range(1, len(mat)):
            if mat[i][j] > maxcol:
                maxcol = mat[i][j]
        score += maxcol
    return score

def score_multiplicative(self, s):
    score = 1.0
    mat = self.create_motif_from_indexes(s)
    for j in range(len(mat[0])):
        maxcol = mat[0][j]
        for  i in range(1, len(mat)):
            if mat[i][j] > maxcol:
                maxcol = mat[i][j]
        score *= maxcol
    return score
```



- score
  - мотифын бүх байрлалыг давтаж, байрлал бүрийн хувьд мотифын онооны нэмэгдэх хамгийн их утгыг тодорхойлдог.
- score_multiplicative
  - Өмнөхтэй ижил оноог тооцдог боловч онооын нийлбэрийн оронд мотивийн байрлал бүрийн хамгийн их утгыг үржүүлнэ.

Brute-Force алгоритмууд: Бүрэн (Exhaustive) хайлт

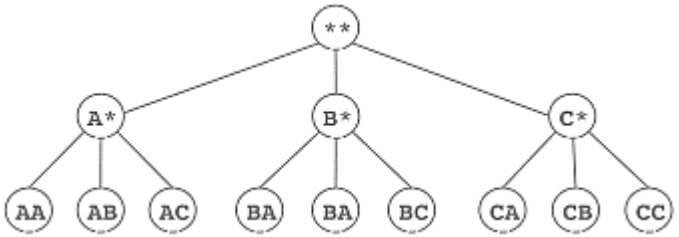
```
def next_solution (self, s):
    next_sol= [0]*len(s)
    pos = len(s) - 1
    while pos >=0 and s[pos] == self.seq_size(pos) - self.
        pos -= 1
    if (pos < 0):
        next_sol = None
    else:
        for i in range(pos):
            next_sol[i] = s[i]
        next_sol[pos] = s[pos]+1;
        for i in range(pos+1, len(s)):
            next_sol[i] = 0
    return next_sol
def exhaustive_search(self):
    best_score = -1
    res = []
    s = [0]* len(self.seqs)
    while (s!= None):
        sc = self.score(s)
        if (sc > best_score):
            best_score = sc
            res = s
        s = self.next_solution(s)
    return res
```

- Өмнөх алгортмын хульд дараалал бүр дэх мотифын хамгийн сайн байрлал бүхий s векторыг олох шийдлийг хэрэгжүүлэх хэрэгтэй.
  - *Оролт*: t – дараалал (урт n); L - мотифын урт; ба σ – хамгийн бага оноо (optional).
  - *Гаралт:* $Score(s, D)$ хамгийн их байх D дээрх M мотифын $s = (s_1, s_2, \dots, s_t)$ гэсэн анхны байрлалаар дараалсан вектор.
- Хамгийн сайн байрлалуудын exhaustive хайлт дээр үндэслэн шийдлийг гаргаж чадна.
  - $Score(s, D)$-г хамгийн их байлгах зорилгоор анхны байрлалын бүх боломжит векторуудыг уншиж, оноог тооцоолно.
  - Хамгийн сайн мотиф - хамгийн сайн онооны байрлалын вектор.
  - Мотифын профайл болон consensus дарааллыг гаргаж болно.
- exhaustive_search : Боломжит шийдэл бүрийн хувьд, анхны байрлалын вектор, хамгийн өндөр онооны шийдийг буртгэдэг.
- next_solution : D дахь оролтын дарааллын байрлалын бүх боломжит $n - L + 1$ давталтыг байгуулна

Branch-and-Bound алгоритмууд

Exhaustive арга боломжит шийд хүрээ нь $(n - L + 1)^t$.

Branch-and-Bound алгоритм нь боломжийг тоочдог боловч зарим шийдийг *урьдчилан харах (lookahead)* ухаалаг механизмд суурилан хасдаг.



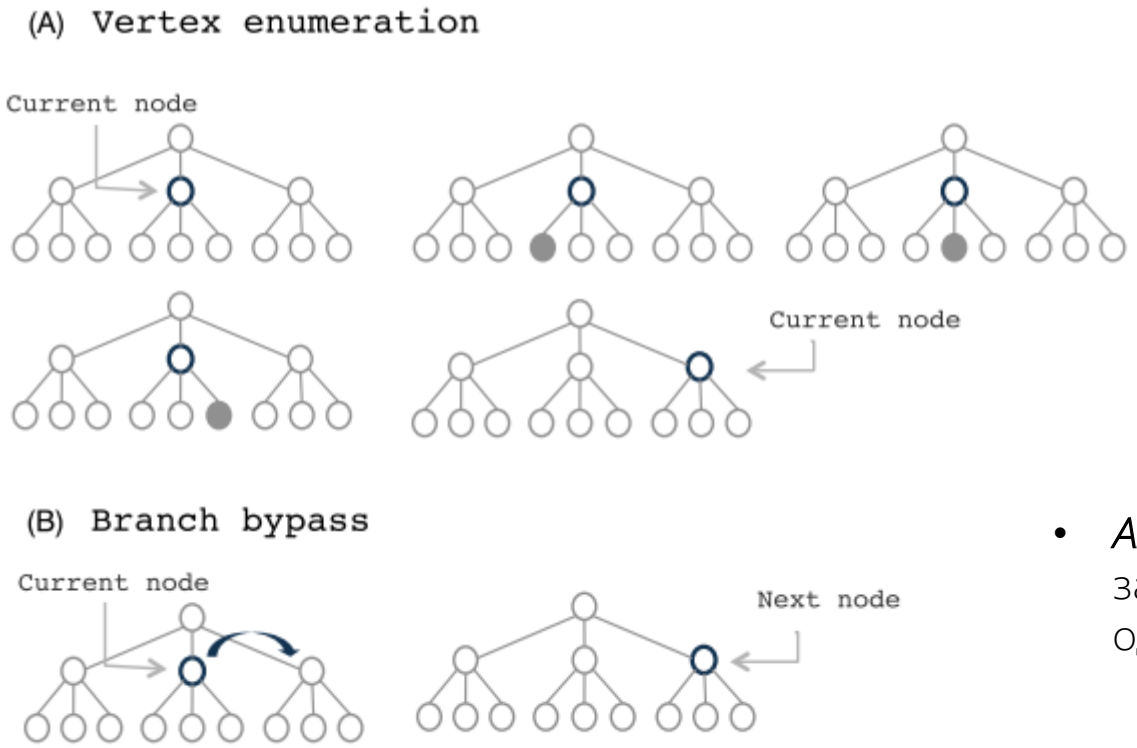
Мотифын модны бүтэц.

Цагаан толгой: $\Sigma = \{A, B, C\}$
Мотифын урт: 2
Боломжууд: AA, AB, AC, BA, BB ...
Нийт: $\Sigma^L = 3^2 = 9$

- Хайлтын боломжуудыг дэд багц болгон хуваах боломжтой болно.
- Модны навч бүр нь боломж, зангилаа нь хэсэгчилсэн дэд багц байна.
- Модны бүх навчийг тойроход бүрэн тооллогод болно.
- Модны мөчрүүдээр дамжих замаар шийдийн боломжуудыг үр дүнтэй тооцохыг эрмэлздэг.
- Хэсэгчилсэн шийдүүд нь модны зангилаан дээрх шийдтэй хамааралтай.
- Тиймээс мөчир бүр нь доорх навчнуудын зорилгын функцтэй холбоотой мэдээллийг агуулна.
- Тиймээс зангилаа эсвэл мочир бүрт тухайн дэд багцын зорилгын функцийн онооны дээд, доод хязгаарыг тооцоолж болно.
- Алгоритм нь одоогийнхоос илүү сайн шийдлийг гаргах боломжтой бол модны мөчир доторх хайлтыг өргөжүүлнэ.
- Үгүй бол тухайн дэд багцийн бүх боломжуудыг хасна.
- Мөчир доторх зарим боломжийг хасахыг хайлтын боломжуудыг ***майрах*** ***(pruning)*** гэж нэрлэдэг.

BRANCH-AND-BOUND Бүх боломжуудыг тоолох, алгасах

- Мотиф илрүүлэх асуудалд branch and bound алгоритмыг тохируулахын тулд
  - Модыг тойрох болон боломжуудын зорилгын функцийн утгын хязгаарыг бодох функцийг хэрэгжүүлнэ.
  - Зорилгын функцийн хувьд өмнө нь танилцуулсан Score функцтэй тохирно.



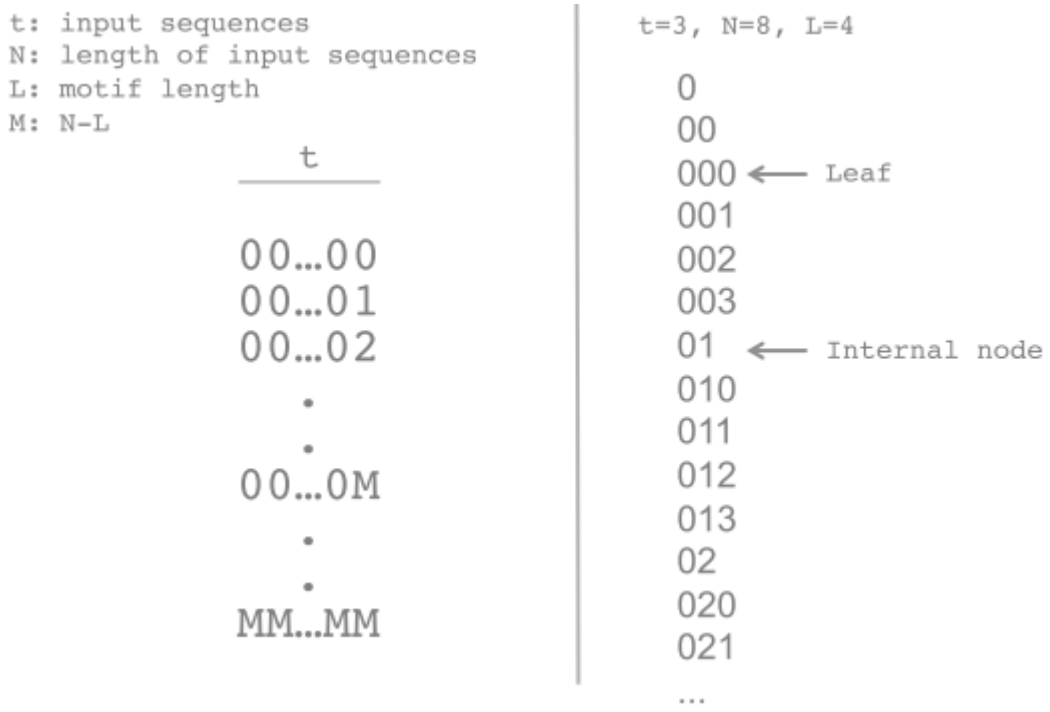
- Хайлтын боломжуудыг тойрохын тулд
  - Модны холбогдох навчнууд дээр очиж боломжуудыг тоолох боломжийг олгодог байх хэрэгтэй.
  - Дараагийн навч руу очихыг зааж өгөх функц шаардлагатай.
  - Хэрэв тойролтын одоогийн цэг нь зангилаа бол функц нь түүний дэд багцын эхний навч руу чиглүүлэх ёстой.
  - Хэрэв одоогийн цэг нь навч бол тухайн дэд багцын арын навц руу шилжинэ.
  - Хэрэв энэ навч нь тухайн дэд багцын сүүлийнх бол одоогийн түвшний дараагийн зангилаа руу чиглүүлэх ёстой.

- *Алгасах (Bypass) үйлдэл:* Хязгаарлах үйлдэл нь тухайн зангилаанаас салаалсан дэд модны дэд багцыг орхиж, одоогийн түвшний дараагийн зангилаа руу шилжинэ.
  - Дэд багцыг аласах эсэхээ шийдэхийн тулд бид дэд багцын дээд хязгаарыг тооцоолох хэрэгтэй.
  - Энэ нь дэд багцад байгаа боломжууд нь одоогийнхоос сайжирч чадах шийдийг агуулж байгаа эсэхийг шийднэ.

BRANCH-AND-BOUND

Мотифын тооллогын боломж.

- Мотифыг илрүүлэх асуудлын хувьд $s = (s_1, s_2, \ldots, s_t)$ мотифын хамгийн сайн анхны байрлалыг сонгох замаар *Score* функцийг хамгийн их байлгахыг зорьдог.
- Тиймээс, мотиф дүрслэлийн хайлтын хамрах хүрээнд биш байрлал дүрслэлийн хүрээнд ажиллана.
- N урттай t оролтын дараалал ба мотивийн урт L-ийн хувьд бид t оролтын дарааллын 0-ээс $M = N - L$ индекс хүртэлх эхлэх байрлал бүрийг хайж болно.
- Зурагт модны навч хэлбэрээр дүрслэгдсэн бүх боломжит шийдийн дүрслэлийг үзүүлэв.
- Навч бүрийг t урттай вектороор илэрхийлсэн.
- i түвшинд харглзах зангилаа нь i урттай вектор байна.



Хьюристик алгоритмууд

- Өмнө үзсэн хоёр алгоритмын хувьд тооцооллын хүндрэл нь оролтын дарааллын t тоо ба урт N, мотивийн L уртаас пропорционал/шууд хамааралтай.
  - Exhaustive тооцох нь $L * (N - L + 1)^t$ үйлдлийг шаарддаг.
  - нь i алгасах үйлдэлтэй branch болон bound аргад хайлт нь $L * (N - L + 1)^{t-i}$ боломжийг алгасдаг.
  - Аль ч тохиолдолд, дарааллын тоо $10 - 12$-аас дээш, урт нь зуу, мянган тэмдэгтэд хүрэхэд асуудал шийдэгдэх боломжгүй болно.
- *CONSENSUS алгоритм:*
  - Эхлээд хамгийн сайн оноотой s_1 ба s_2 анхдагч байрлалуудыг хайхын тулд зөвхөн эхний хоёр оролтын хувьд дарааллыг авч үзэхээс эхэлдэг.
  - Эдгээр нь хамгийн сайн хэсэгчилсэн үр дүнгүүд дээр суурилан нийт оноог гаргадаг.
  - Оролтын $i = 3, \dots, t$ дарааллын хувьд алгоритм нь дараалал бүрийг давтаж, оноог хамгийн их байлгах зорилгоор анхдагч байрлалыг сонгоно.
  - Өмнөх дарааллын байрлалууд $(s_1, s_2, \dots, s_{i-1})$ тогтмол байна.
- Энэ алгоритм нь тооцооллын үр ашгийг ихээхэн сайжруулдаг ч
  - Үр дүн нь эхний хоёр анхдагч байрлалаас ихээхэн хамааралтай
  - Эхний хоёр анхдагч оролтын дарааллын эрэмбээс хамаардаг.
  - Тиймээс алгоритмд оролтын дарааллыг өөр эрэмбээр оруулах.

АНХААРАЛ ТАВЬСАНД БАЯРЛАЛАА