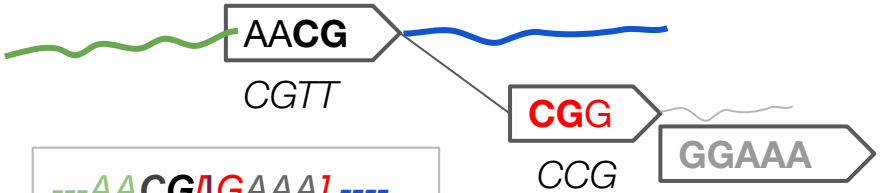
UST-Tip-Compress by Absorption

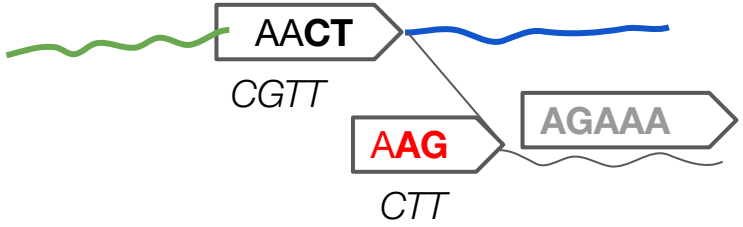
Nov 18, 2019



---AA**CG**[|GAAA] ----

---AA**CG**----
CGGAAA

1 -> 0



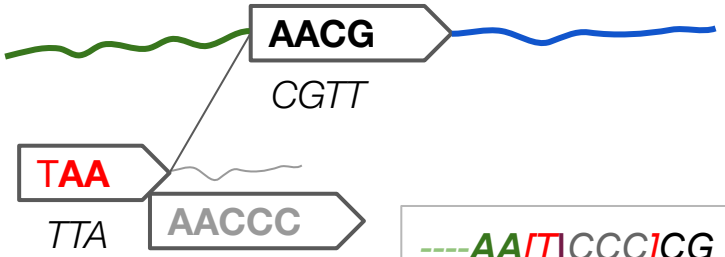
k = 3,
2 len
overlap

---AACT[A~AAA] ----

1 -> 1

---AA**CT**----
AAGAAA

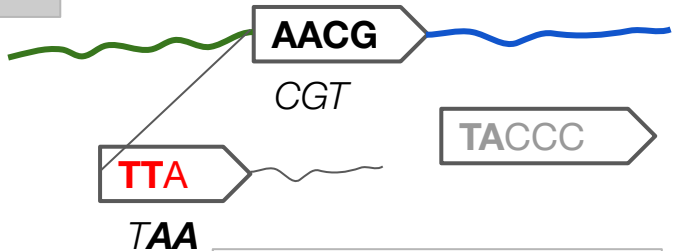
0 -> 1



----**AA**[T|CCC]CG ----

-----**AACG**-----
TAACCC

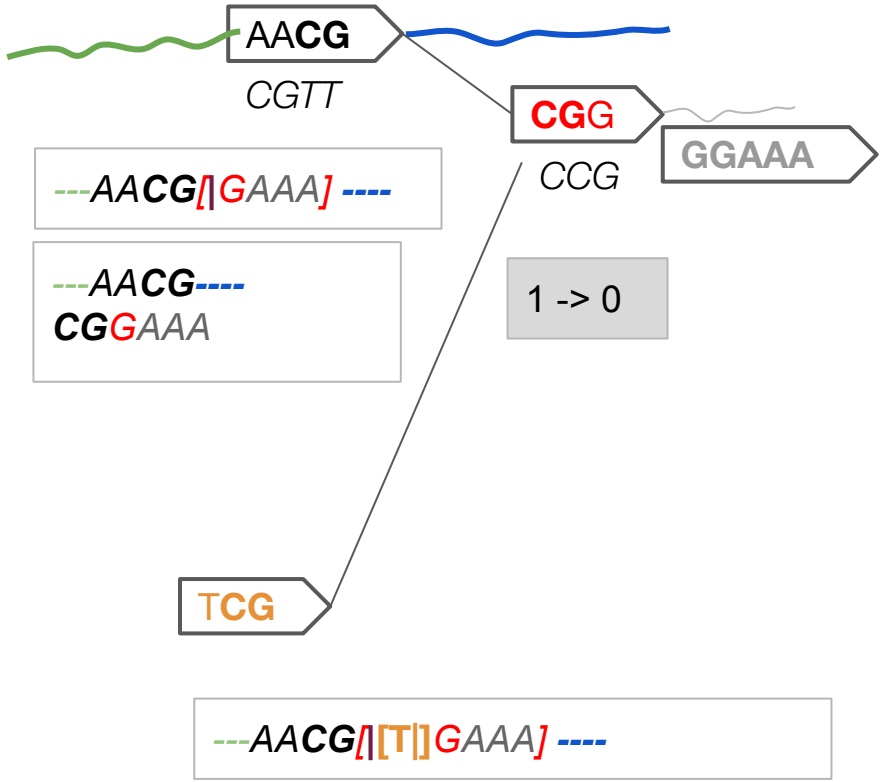
0 -> 0



-----**AA**[~ACCC]CG ----

-----AA**CG**-----
CGGAAA

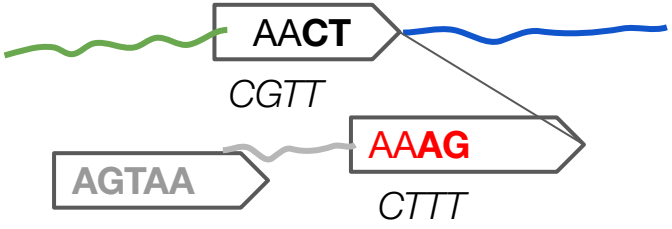
Nested compression



---AA**CG**[AAA|G] ----

---AA**CG**----
AAA**CG**G

1 -> 0



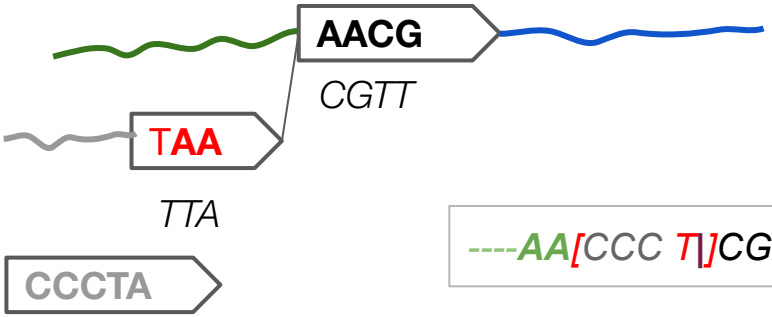
k = 3,
2 len
overlap

---AA**CT**[AGTAA~] ----

1 -> 1

---AA**CT**----
AGTAA**G**

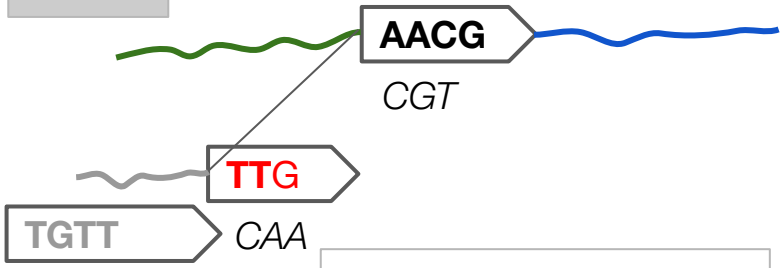
0 -> 1



----AA[CCC T]CG ----

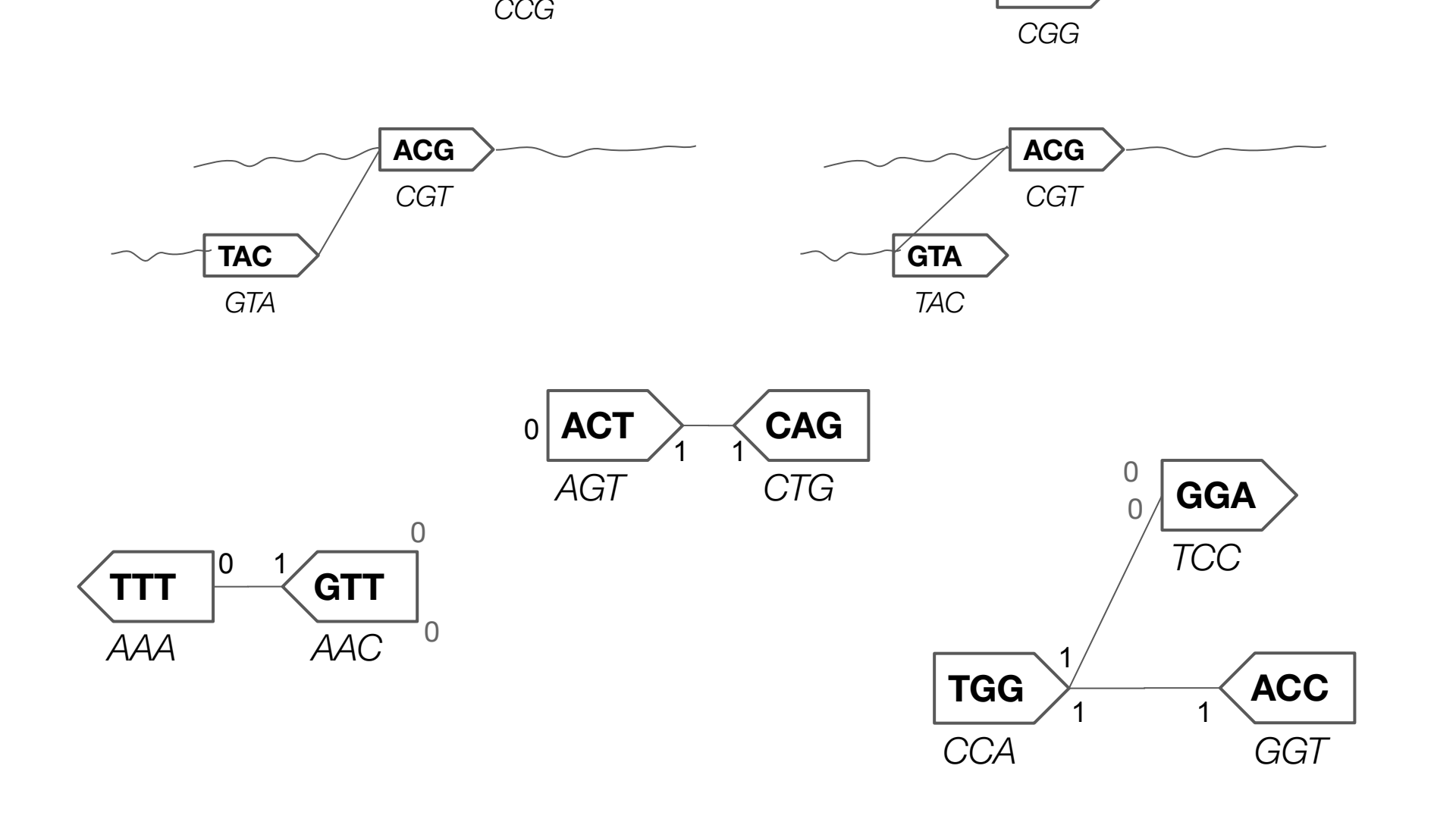
-----**AA**CG-----
CCCTAA

0 -> 0

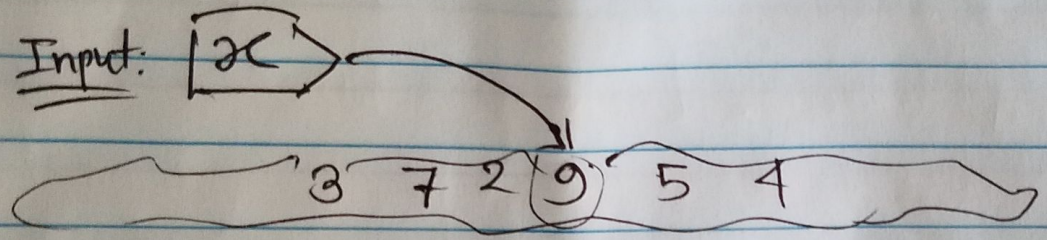


----AA[TG~G]CG ----

-----AA**CG**-----
TGTTG

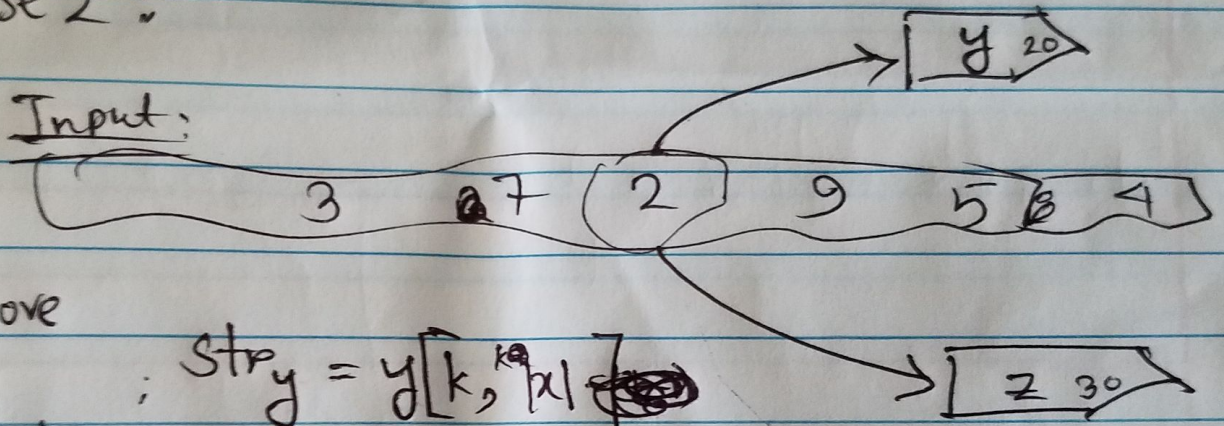

CCG
CGG
ACG
CGT
TAC
GTA
ACG
CGT
GTA
TAC
0
ACT
1
1
CAG
AGT
CTG
0
0
GGA
TCC
1
TGG
1
1
ACC
CCA
GGT
0
0
TTT
1
GTT
0
AAA
AAC

Case 1:

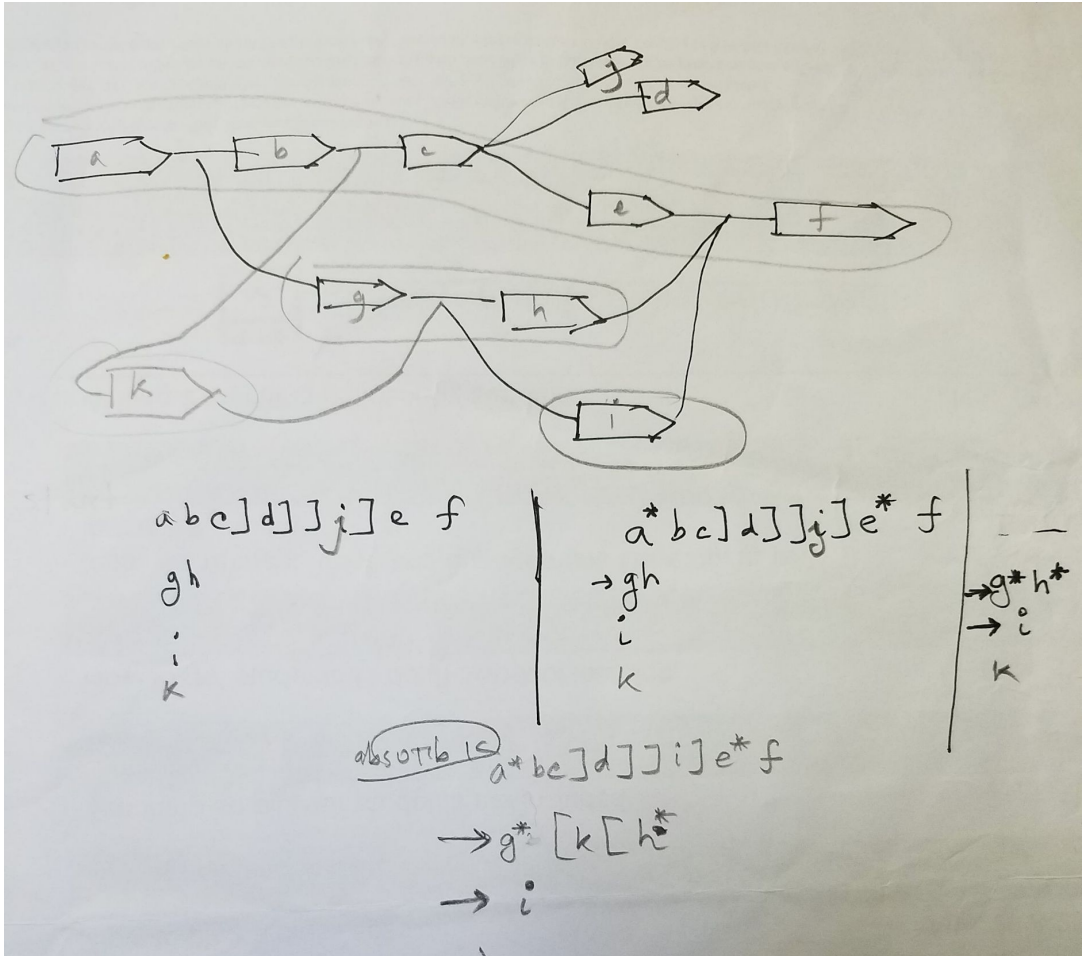


$str_x = x[1, |x|-(k-1)]$ remove last (k-1)

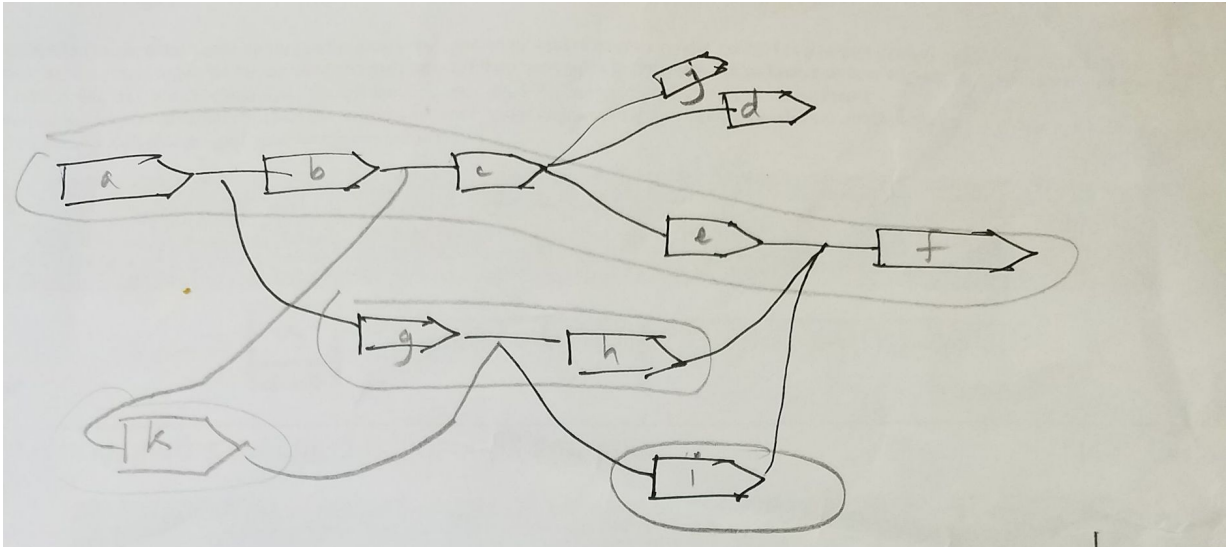
Case 2 :



output: 3 7 2 9] 20 str_y]] 30 str_z] 5 4



1. Both-end-absorption:
    a. Pick a walk which has both start and end connected to same walk id, absorb it with **



Initial steps:

Remove tips.
Run UST.
Absorb tips.

1. abc]d]]j] e f
2. gh
3. i
4. k

Pick walks with one vertices and absorb them.

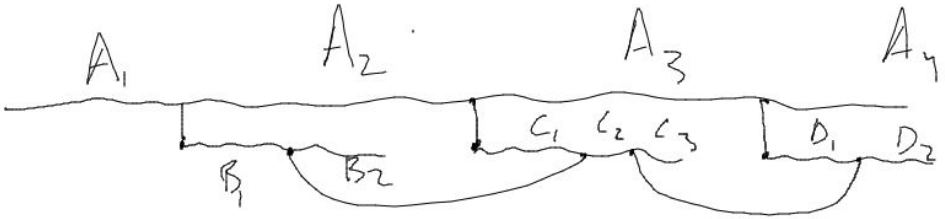
1. abc]d]]j] e f
2. g[i[[k[h

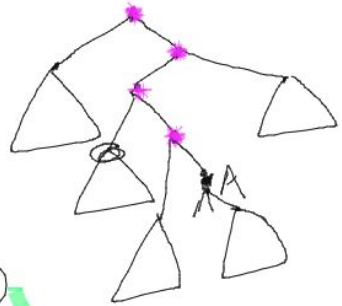
Add all walks to list.

Arbitrarily pick a walk,
If it can be absorbed, absorb it.
Else, cross it off our list.

Repeat until list is empty

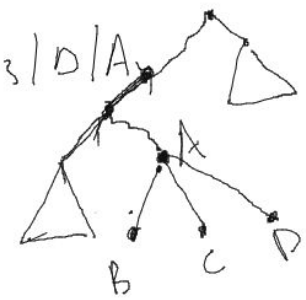
1. a [g[i[[k[h[bc]d]]j] e f

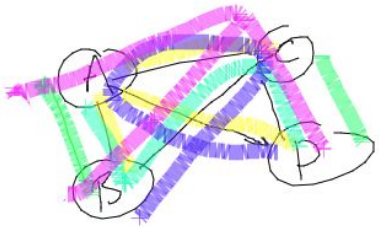




$A_1 | B_1 B_2 | A_2 | A_3 | D | A_4$

$A_1 | B_1 | C | D$





UST → Greedy absorption → one string T

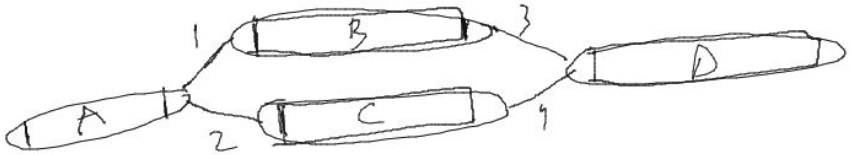
low prior. 1. Order of absorption → effects decompression mem.

2. Claim: $L_{SA}(T) = n + (k-1)(\#\text{ of c.c.}) + 2|UST|$
   a. regardless of absorption order.

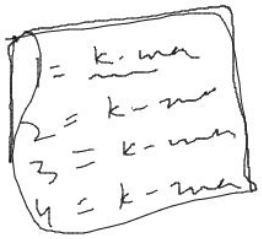
low prior. 3. What if we interleave UST / absorption.

4. Non-nested both-end absorption

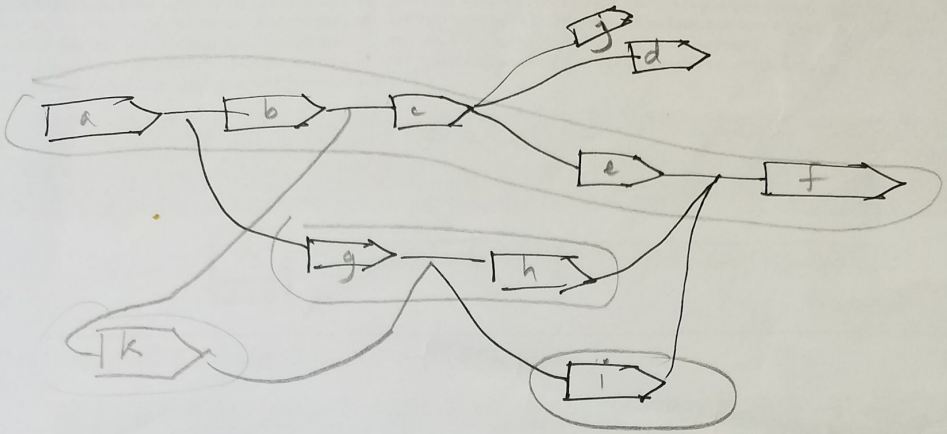
1. Write the theory.
2. implement.
3. Write down the alg. w/ mathem. precision.



int(A):1
1.int(B):3
2.int(C):4
3.int(D)



log n



a b c] d]] j] e f

g h

i

k

a* b c] d]] j] e* f

→ g h

i

k

→ g* h*

→ i

k

absorbs a* b e] d]] i] e* f

→ g* [k [h*

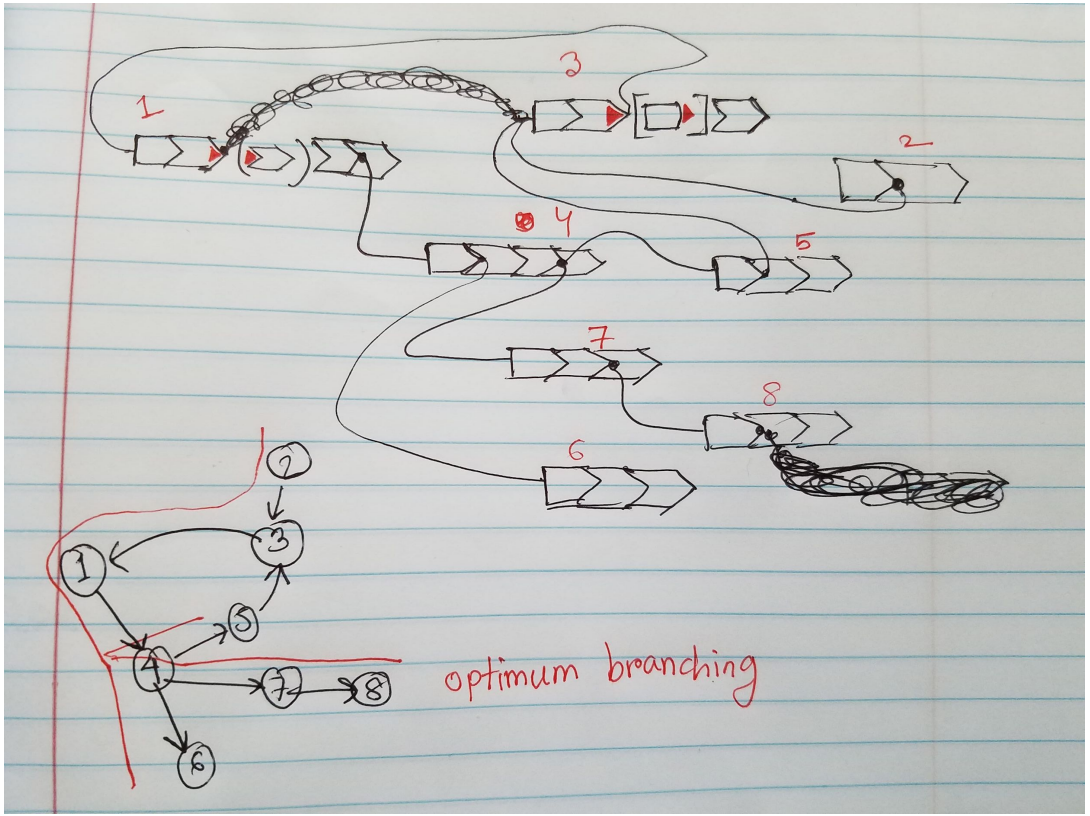
→ i

Branching

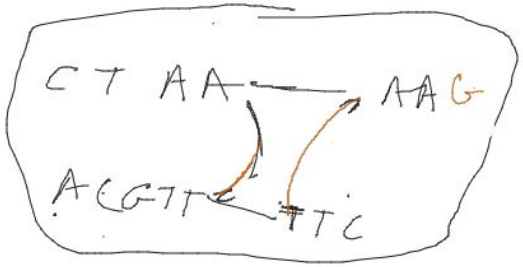
A branching $B \subseteq E$ of $G = (V, E)$ is an acyclic subset of edges in which the indegree of each vertex is at most 1.

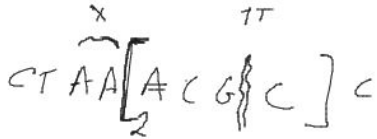
Lemma 1. *If there exists one optimum branching (tree) for the graph G_A, then the total number of characters by absorption algorithm is, $c_{absorbed} \leq c_{ust} - (|W| - 1)(k - 4)$. If there are $|T|$ such branchings, then $c_{absorbed} \leq c_{ust} - (|W| - |T|)(k - 4)$*

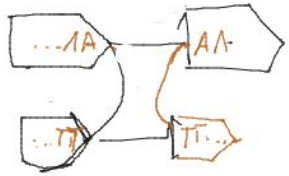
An instance of optimum branching

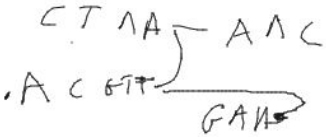


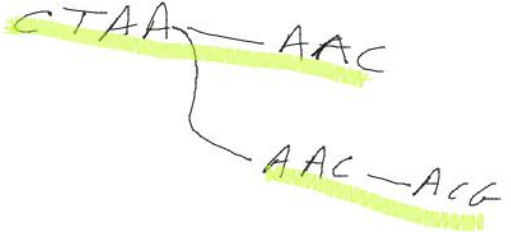
- Finding Optimum Branchings (or minimum spanning tree in a directed graph)
- O(E log V) algorithm exists



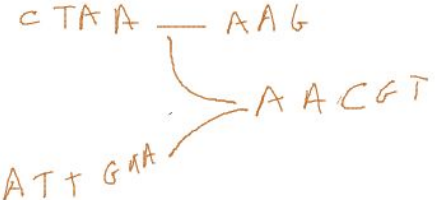






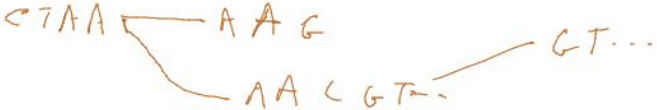


CTAAC
GTTC



TGAX

CTAA — AAG



Find non overlapping interval

1. Sort the intervals based on increasing order of starting position.
2. Push the first interval on to a stack.
3. For each interval do the following
   a. If the current interval does not overlap with the stack top, push it.
   b. If the current interval overlaps with stack top and ending time of current interval is more than that of stack top, update stack top with the ending time of current interval.
4. At the end stack contains the merged intervals.

Extra Slide